European Union Aviation Safety Agency

This project has received funding from the European Union's Horizon Europe Programme

RESEARCH PROJECT EASA.2021.C38, MLEAP

FINAL REPORT

Machine Learning Application Approval

An Agency of the European Union

Disclaimer

DELIVERABLE NUMBER AND TITLE:	MLEAP final report
CONTRACT NUMBER:	EASA.2021.C38, MLEAP
CONTRACTOR / AUTHOR:	Airbus Protect, LNE, Numalis
IPR OWNER:	European Union Aviation Safety Agency
DISTRIBUTION:	Public
CITING THIS REPORT:	MLEAP Consortium, EASA Research – Machine Learning Application Approval (MLEAP) final report, May 2024

```
@techreport
{
        MLEAP final report,
         author     = "MLEAP Consortium",
         title         = " EASA Research – Machine Learning Application Approval (MLEAP) final report ",
         institution = "European Union Aviation Safety Agency",
         year        = "2024",
         type       = " Horizon Europe research and innovation programme report",
         month    = "5",
}
```

APPROVED BY:	AUTHORS	REVIEWER	MANAGING DEPARTMENT
Olivier GALIBERT LNE [illegible]	Thiziri BELKACEM Arnault IOUALALEN Swen RIBEIRO Noémie RODRIGUEZ Jean-Baptiste ROUFFET Jérémy Bascans Quentin Signé	MLEAP consortium	*Project Manager: Michel Kaczmarek* *Quality Manager : Bernard Beaudouin* M Kaczmarek

DATE: 28 May 2024

EXECUTIVE SUMMARY

Context

Artificial Intelligence (AI) is becoming ubiquitous, and many industrial domains, including aeronautics, aim to harness its promises to improve their performance. The most spectacular progress of contemporary AI comes from Machine Learning (ML) and Deep Learning (DL). These technologies extract and learn behavioural patterns for a given task from corresponding data. This latter comprises a set of samples of the operational context of the target domain and application. However, that same learning process can make it harder for systems, including those modules, to be trusted in critical situations. Hence, more adequate approaches need to be developed to build that trust.

In the aeronautics domain, the European Union Aviation Safety Agency (EASA) published its Artificial Intelligence Roadmap in February 2020, followed by the first primary deliverable, a Concept Paper, *'First usable guidance for level 1 machine learning applications'* in December 2021. This latter has been recently updated to EASA Artificial Intelligence Concept Paper Issue 2, published in March 2024, to cover level 2 AI applications. It refines the guidance for Level 1 AI applications and extends the exploration of several concepts, such as *learning assurance*, *explainability* and *ethics-based assessment*. This new issue provides comprehensive guidance for developing and deploying Level 2 AI-based systems, which concerns human-AI teaming applications, including cooperation and collaboration actions where AI systems automatically make decisions under human oversight.

These different versions of the EASA AI concept paper lay the basis of EASA guidance for ML application approval. They set several areas where further research is necessary to identify efficient and practicable means of compliance within a defined *AI trustworthiness* objective. Hence, the framework of learning assurance, namely the *W-shaped learning process*, has been updated. This process serves as a reference for the Machine Learning Application Approval (MLEAP) project, which is initiated to provide a set of recommendations for methods and tools to achieve the different requirements allocated to the ML components of the system.

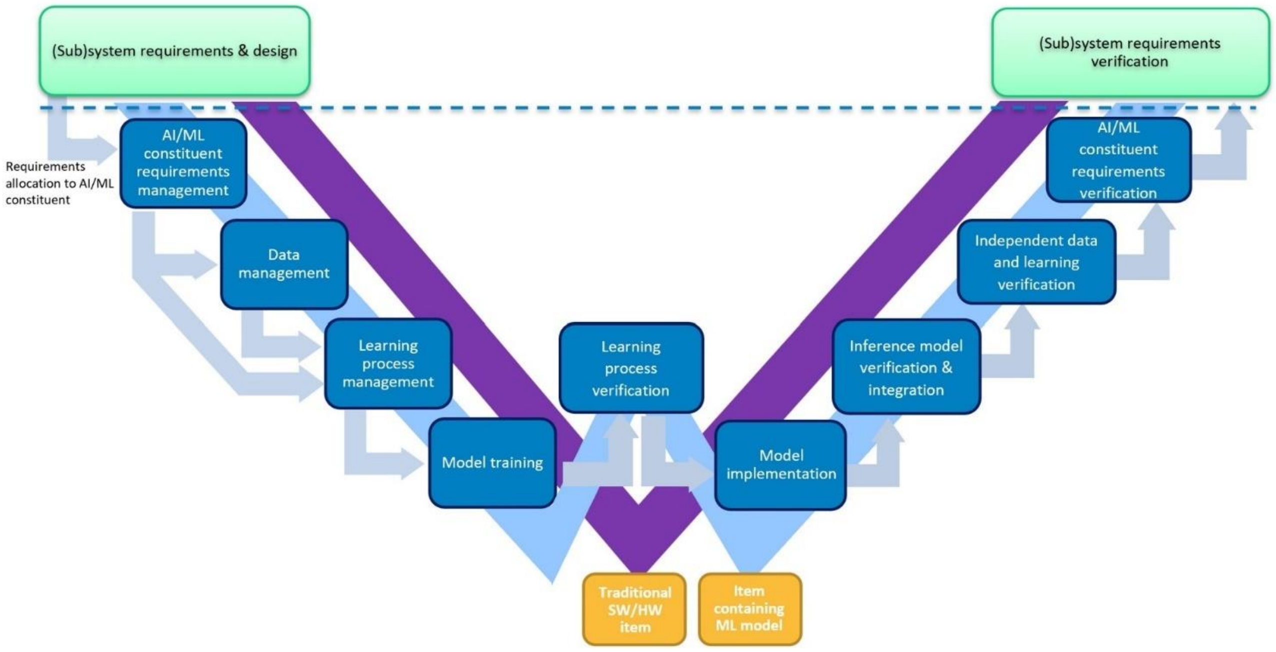
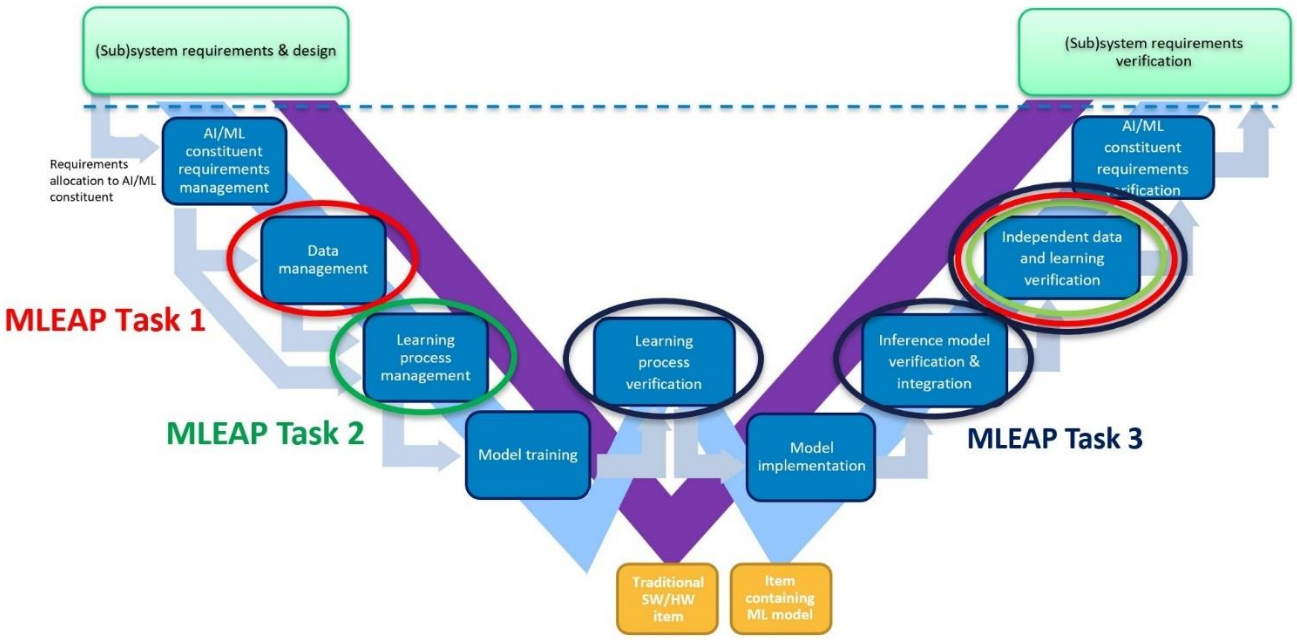

(Sub)system requirements & design
(Sub)system requirements verification
Requirements allocation to AI/ML constituent
AI/ML constituent requirements management
Data management
Learning process management
Model training
Learning process verification
Model implementation
Inference model verification & integration
Independent data and learning verification
AI/ML constituent requirements verification
Traditional SW/HW item
Item containing ML model

Figure 1 - Global view of the learning assurance W-shaped process overlapping the non-AI component V-cycle and the safety assessment process. The dashed line separates the system level (upper part) and the AI level (lower part).

The learning assurance framework is an essential building block of the AI trustworthiness concept, which adapts development assurance principles to learning-based approaches. As shown in figure 1, the learning assurance sets the specific objectives for each development step w.r.t the system level and the AI level of the whole system. Hence, this process adapts the typical software development assurance *V-cycle* to ML/DL-based applications. It allows the structure of the guidance through blocks composing it. The dotted line is here to distinguish between the use of traditional development assurance processes (above) and the need for processes adapted to the data-driven learning approaches (below), where the learning assurance processes start below the dotted line.

Focussing on the development of the AI components, the MLEAP project has been tailored to investigate the challenging objectives of the W-shaped process. Funded under the Horizon Europe framework, MLEAP aims to promote AI blocks of the W-shaped process by carrying out three main tasks, each of which will serve one or more parts of the whole process, as shown in



(Sub)system requirements & design
(Sub)system requirements verification
Requirements allocation to AI/ML constituent
AI/ML constituent requirements management
Data management
Learning process management
Model training
Learning process verification
Model implementation
Inference model verification & integration
Independent data and learning verification
AI/ML constituent requirements verification
Traditional SW/HW item
Item containing ML model
MLEAP Task 1
MLEAP Task 2
MLEAP Task 3

Figure 2 - The positioning of the various parts of the MLEAP project in the W-shaped process

The three tasks shown in Figure 2, carried out during the 2-year life of this project, correspond to:

Task 1: Dealing with data completeness and representativeness and handling the corner cases. It focuses on data quality verification and proposes a selection grid of methods that can be used to make sure that the ML pipeline is being developed with a trustable representation of the target domain and application. This representation corresponds to the construction of a complete and representative data set. Hence, this task concerns the two main steps of the W-shaped process, including the data management tasks (quality and volume assessment, preparation, and processing…) and the independent data and learning verification of the produced model during the development process. A set of empirically verified tools and methods for data qualification are expected, along with recommendations on how the completeness and representativeness of the datasets can be assessed, in addition to leveraging the learning behaviour of trained models to enhance the data quality.

Task 2: It deals with the characteristics related to the reliability of the built ML/DL model. This task revisits the model development by handling the generalisation properties. It explores how the learning process can be leveraged to promote the model's ability to scale the performances to unseen data during training. To do so, generalisation assessment tools and performance enhancement techniques are investigated, avoiding overfitting and underfitting. Hence, to reduce regressions after implementation in the target system, this task focuses on the learning process management and verification, the model training, and the performance verification as concerned steps of the W-shaped process. Evaluated generalisation bounds and other tools for generalisation assessment and assurance are expected, along with a generic AI development pipeline focussing on model performances and how to drive the development steps to achieve the target performances.

Task 3: Focuses on the model evaluation to verify the targeted features in terms of robustness and stability of the performances measured in the assessment. Since the model may be confronted with changes in the representation of input data and disturbances in the real world, it is necessary to check that the model is acceptably robust and that its performance is stable despite corrupted or naturally noisy data. Hence, this task concerns the learning process verification, the inference model verification and integration, and the learning verification as the main steps of the W-shaped process. Therefore, several questions about the robustness and stability of trained models must be answered, including edge and corner case handling and outliers' management. To do so, the expected outcome is empirically evaluated methods and approaches handling these aspects, in addition to recommendations on how the robustness and stability can be verified during the development process and what needs to be done to avoid weak performances.

This final report offers a set of anticipated concepts for evaluating and certifying AI-based systems supporting the EASA roadmap deliverables. It helps industry stakeholders plan new strategies for deploying AI in their human and technical organisations. This executive summary provides an overview of the various points covered in the project's tasks. It summarises the work to address the objectives mentioned in the EASA's concept paper (EASA, 2024), for which MLEAP is expected to bring answers. The report proposes a set of studies, including analysis of the state-of-the-art, selection and discussion of several methods, experimental results, and the main conclusions and recommendations verified empirically to meet the objectives explicitly highlighted in Chapter 1 of the full report.

Report content and main findings

The work is structured into seven chapters in the document to provide analysis and methods/verification recommendations for the various stages of the W-shaped process.

Chapter 1 corresponds to the introduction. It provides a detailed description of the research directions issued in this project while defining the boundaries of the expected work, including terminology definition, the position of the MLEAP project with regards to the W-shaped learning process, the EASA's targeted objectives, as well as the scientific protocol followed during the activities' implementation of this project. The scientific terminology the work is based upon has variable definitions from one document to another. To scope the boundaries of the project deliverables, the terminology and the scientific and technical definitions used in the project are first refined. The objective is to have a shared understanding of the different aspects involved in the activities. A comparison between the various uses and references is also provided (mainly about the intended use in the EASA's documents).

Chapter 2 is about the metrics used in the evaluation parts of this work concerning data quality and volume evaluation, as well as the model's performance evaluation. It describes the different measures, highlighting what kind of performance is targeted for each measure. Since different metrics can be used to evaluate both the generalisation performance to unseen data during training and the performance stability and robustness toward data and environment changes, a dedicated section describes the main differences between those metrics and how they can be used to evaluate the performance.

Chapter 3 is dedicated to the description of the use cases that have been selected for the experimental part of the project. It provides, for each use case, a brief background on the concerned task and a state-of-the-art analysis highlighting the position of the functions in the project, its main objectives, the challenges to be addressed in the MLEAP project, and why these are important. The goal of using different use cases is, on the one hand, to drive and lead the analysis of the state-of-the-art and the methods selection to address the main challenges of the project, as well as their applicability analysis, and on the other hand, evaluate the project findings and make recommendations for AI systems based on actual aviation use cases. These are:

Speech-to-text for air traffic control (ATC-STT) aims to correctly transcribe the spoken instructions by air traffic controllers to ensure that they are well transmitted and received by the pilots. In this use case, the data corresponds to recorded instructions provided along with the corresponding textual transcriptions. While background noise, speech rate, and spoken language accents are essential issues for an STT system, these issues must be considered in an ATC-STT application, which is a critical use case. In MLEAP, we deal with several spoken English accents, including French and Chinese. The objective is to achieve acceptable performances based on the transcriptions analysis (10%-word error rate), regardless of the noise, accents, and speech rates.

Automated visual inspection (AVI): This aims to design a system for detecting aircraft damage in-service. One of the main challenges is the diagnostic assistance for inspectors to reduce the aircraft maintenance duration for scheduled and unscheduled events. The main point is to find acceptable metrics to bring computer vision closer to classical problems, such as model development for surface damage detection. Hence, the targeted performances have been set and focus on detecting all (most of) the damages (95% accuracy). In the MLEAP project, the objective is to analyse the existing pipeline and provide recommendations to meet the expected performances in the targeted domain, which is in-service damage detection, including the lighting strike impacts and dents.

Airborne collision avoidance system Xu (ACAS Xu): is an air-to-air collision avoidance system designed for unmanned aircraft (drones). The purpose of an ACAS Xu system is to keep any intruder outside of the desired envelope of the ownship. In this use case, the objective is to produce an ML/DL model that can completely fit the discrete input lookup tables. In the MLEAP project, we consider the analysis of several models designed for this purpose, focusing on the performance analysis to meet the objective of the use case, as well as the data analysis based on different sources.

Each of the above cases uses a data set and trained ML/DL models. Airbus internal projects or open-source data sets and models either provide these. The open-source and Airbus models and data have been compared (in terms of quality and performance, as well as compliance with the use case's expectations), enabling analysis of a broader range of application contexts.

Chapter 4 deals with data management in the aspects of learning assurance. In particular, assessment methods for completeness and representativeness are presented and collated in a selection grid. In addition, approaches to managing edge cases and corner cases are explored. Indeed, the EASA guidelines consider robustness in the sense of changing inputs, including edge cases, corner cases, outliers, out-of-distribution, or even adversarial cases, which can be different in other documents (like the ISO/IEC standards). These input changes can even be refined to distinguish perturbations at different semantic levels, such as pixel, domain, object, scene, or scenario levels for images. These different semantic levels of perturbation bring insight into the problematic cases and characterise the system's robustness from various viewpoints that may be linked to the operational design domain definition.

The general problem of completeness and representativeness assessment is broken down into several factors of influence, shedding light on the task's complexity. More than fifteen factors are identified and grouped into three categories: technical requirements (i.e., elements influencing the design of the AI system's operational design domain), processes (i.e., of the AI system's development lifecycle), and other data quality requirements (i.e., apart from completeness and representativeness).

Influence factors are used to structure discussions from a normative and operational point of view by framing the contributions of more than 80 references, including academic methodologies and approaches, as well as integrated tools designed for applicative goals.

This extensive work is then summarised into a selection grid isolating methods considered applicable within the project. The goal is to test as many methods as possible to provide the future applicant with first-hand feedback and general insight into how completeness and representativeness may be assessed. This work is not prescriptive and does not pretend to be exhaustive.

Preliminary experiments and conclusions have been obtained on the most pertinent methods. Experiments followed an iterative process, first tested on a small data set that is easy to deploy and manipulate (sometimes referred to as a "toy" data set). The objective is to get to grips with existing tools or ensure the correct implementation of the methods.

Specific methods were then applied to larger-scale data sets of tasks similar to those described in the MLEAP use cases. These data sets are used as an intermediary between toy data sets (for tool validation) and actual use cases (for final analysis) because use case data sets are more complex to access, and the scalability of the methods has to be validated beforehand. Moreover, the data sets used are well-known to the experimenters, which allows for more control over the conclusions reached through the methods. Since no methodology is self-sufficient, this intermediary step is also helpful in refining the use of the methods, understanding their limitations, and determining how they can interact with one another to gain the most insight.

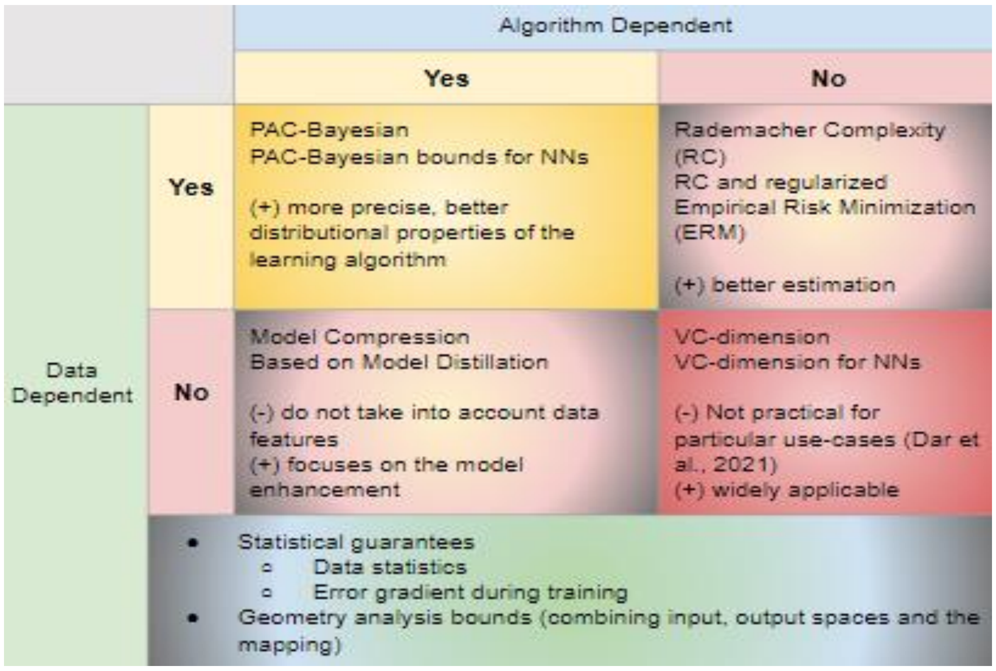
Finally, the pertinent methods were applied to the MLEAP use cases to validate the information they can provide on aviation tasks and data.

The methodologies tested provided results relative to the completeness and representativeness of tabular data and images. Academic approaches requiring prior reimplementation and off-the-shelf industrial tools have also been tested. In both cases, their limitations were experimented with first-hand and discussed in this document to provide a more concrete idea of their usability in an operational context in Chapter 4.

Identified limitations include difficulty accessing direct, quantifiable information about completeness and representativeness due to most methods or tools not being directly designed to address these specific properties. As a result, no tested solution isolates even a subset of the influence factors discussed in the previous sections (while they should ideally isolate them one by one). All methods thus remain fuzzy in the information they provide and require rigorous expert analysis, although they are undoubtedly beneficial to structuring a general assessment approach.

Besides, not all methods fit every data type. Experiments tackled low-dimensional tabular data (i.e., with ACAS-Xu as the target use case) and high-dimensional unstructured data (i.e., images and speech embeddings, although the latter did not yield helpful results). Experimentation allowed us to explore all data qualification tasks identified as relevant to meet the Data Management (DM) objectives. Thus, they need to identify their specific challenges and address them. Overall, the results are encouraging, with few methods dismissed and attractive potential highlighted. However, as mentioned earlier, experimentations have continuously reinforced that no one-size-fits-all method or even a single tool or methodology is expressive enough to be used alone on a particular problem. All the insight gathered from these experiments is synthesised in Chapter 7. A generic way to tackle the assessment of data completeness and representativeness in the general context of the W-shape process is derived. Identified pillars of data completeness and representativeness are the ODD and the trained model: the ODD allows for a priori specifications of data requirements to guide data collection and preparation processes. By contrast, the trained model provides feedback to tune the data set and alleviate biases and other model-specific behaviours.

Chapter 5 of this report is dedicated to the development and generalisation properties of a trained model. The generalisability of trained models, assessment and evaluation, is investigated, including a comprehensive overview and a state-of-the-art analysis of existing methods for evaluating ML and DL models and generalisation bounds definition and evaluation. Several generalisation issues and well-known ML/DL-related problems of underfitting and overfitting have been investigated. We analysed the existing methods and their limitations to give guarantees about the performances of trained models on unseen data. Figure 3 shows the completed grid, initially defined in the EASA's CoDANN I and updated with the latest state-of-the-art methods reviewed in MLEAP.



		Algorithm Dependent	
		Yes	**No**
Data Dependent	**Yes**	PAC-Bayesian PAC-Bayesian bounds for NNs (+) more precise, better distributional properties of the learning algorithm	Rademacher Complexity (RC) RC and regularized Empirical Risk Minimization (ERM) (+) better estimation
	No	Model Compression Based on Model Distillation (-) do not take into account data features (+) focuses on the model enhancement	VC-dimension VC-dimension for NNs (-) Not practical for particular use-cases (Dar et al., 2021) (+) widely applicable
	• Statistical guarantees ○ Data statistics ○ Error gradient during training • Geometry analysis bounds (combining input, output spaces and the mapping)		

Figure 3 - State-of-the-art classification of Generalisation Bounds methods

After the model training, evaluation measures and metrics can be used to assess the generalisability. The objective is to detect performance dropouts due to overfitting or underfitting and then boost the generalisability of trained models. While targeting a good model for the industrial application, there is a set of steps to be carried out to achieve the desired performance from the industrial and target system perspectives. Furthermore, by analysing existing AI development approaches in state of the art and the most common practices in data science, we have identified the significant pitfalls and weak practices that can harm the ML/DL system performances. These include, among others:

- The misunderstanding of the generalisation bounds, where some norm-based measures negatively correlate with generalisation;
- Several common mistakes and pitfalls in practice while developing the ML pipeline, such as the use of inappropriate training objective functions and data representation or split, in addition to inappropriate model complexity and evaluation metrics concerning the target application and results acceptability;
- There is a large gap between the expectations from the experimental evaluation compared to the real-world applications; the evaluation metrics of different machine learning applications, such as Mean Squared Error (MSE), precision, and recall, are used to measure only the

technical performance of the ML/DL component, however, in the industrial performance assessment, it is necessary to understand how far the empirical assessment reflects the actual model's efficiency and the system-level requirements that should be included;
- The acceptance of the achieved performance and how the system-level monitoring could handle the model's outputs, including errors and uncertainty, but also the meaningful 95% accuracy and the distribution of the remaining 5% of errors;
- An appropriate performance indicator for the application domain is not straightforward, and existing evaluation metrics cannot always translate it. Hence, an adaptation and combination of existing processes can be needed to bridge the gap between experimentation and industrial expectations. The classical approach that uses a set of technical metrics to assess the model's performance is limited in capturing aspects related to the industry (KPIs) and reproducibility.

Hence, generalisation evaluation is ultimately more crucial than initially thought. We have analysed the state-of-the-art ML/DL generalisation evaluation and provided our main observations about the cited methods concerning generalisation assessment, issues detection, and strategies for results improvement.

To cope with the different issues discussed above, we set the leading research and technical questions to be answered by task 2 to build a model's generalisability promoting pipeline:

- ***How do we deal with overfitting/underfitting in the industry?*** To address this problem, several techniques have been developed to help the ML/DL models generalise better. Although the original model may be too large to generalise well, regularisation techniques help limit learning to a subset of the hypothesis space, where the resulting models will have manageable complexity. Combining different methods makes the model generalise better, independently of the generalisation type (domain-based, multi-tasking, and OOD-based).
- ***How can we bridge the gap between experimentation and industrial expectations?*** To bridge the gap between empirical and industrial processes, we need to leverage the evaluation metrics to reflect the targeted performances and integrate the KPIs in the training objectives and the evaluation pipeline.
- ***How do you cope with common data processing and evaluation mistakes****?* To ensure a more rigorous evaluation pipeline, we suggest that the complete roadmap, from the data preparation and qualification step to the model validation and release, benefit from some software engineering best practices, such as building scenarios for test and iterating on the process of data set improvement, evaluation benchmarks, and the verification and validation process of the final trained model.

The questions mentioned above have been explored in Chapter 5 of the research. To bring answers to the different raised questions, a two folds experimental process is adopted:
(1) focus on the analysis of generalisation assessment and evaluation,
(2) exploration of the aviation use-cases applying the generalisation assessment and providing a complete alignment between the experimental pipeline and the target objective of each use-case.

To delve into the analysis of generalisation assessment and evaluation, a meticulous examination of how well the trained models generalise across various scenarios and datasets. By analysing the effectiveness of the produced models beyond their training environment, the aim is to gain thoughtful insights into their performances while dealing with unseen data samples. To do so, statistical methods and tools are used to perform this investigation, which lays the groundwork for subsequent analysis into the aviation use cases, where the findings are leveraged to enhance the efficacy and reliability of the experimental pipeline. Hence, to perform step (1), taking into account the impact of data type and volume, as well as the target task characteristics (the use cases selected for MLEAP), the model's architecture and characteristics, and the state-of-the-art analysis, a set of generalisation bounds have been selected, based on their applicability analysis, for generalisation bounds estimation.

The objective of the experimental work on this part is to highlight the differences between these methods and how they can be used to assess the generalisability of miniature models in small datasets. More details about the results and the behaviour of these methods can be found in Chapter 5. The first results have shown that, depending on the model's architecture, the bounds can have different behaviours, and it is not straightforward to generalise conclusions made based on one method's analysis. Nevertheless, this helped clarify several differences between the learning behaviours that models could have during training and how they can converge based on several optimisation and regularisation techniques. After application to the aviation use cases, the same behaviour was observed. The main takeaway is that using specific generalisation bounds in aeronautical scenarios confirms that when applied to moderately sized networks, they can yield precise bounds for the generalisation gap, introducing confidence that the model's behaviour observed with the test dataset will persist. However, we cannot draw definitive conclusions with deep neural networks as the computed bounds are inconclusive.

The two approaches we tested did not yield explicit positive outcomes in generalisation when mitigating the impact of unbalanced datasets on the performance of the ACAS Xu trained model. Nonetheless, enhancing model stability and robustness may have potential benefits.

In the second (2) part of the experimental analysis, the MLEAP use cases have been explored. The objective of the carried-out experiments was threefold:

I. Analysis of existing pipelines, where the models' performances, data construction and targeted objectives have been analysed and compared for each use case. This exercise highlights the main issues that resulted in a gap between experimentation and target application objectives. Several performance alternatives to improve the results have been tested to support the recommendations for a more consistent ML/DL development pipeline;
II. The already selected generalisation bounds, based on a toy use-case, have been evaluated and compared in the aviation use-cases that are of different dimensionalities and using other data types (images, audio, text…);
III. To solve the same task, compare different architectures and alternatives for data quality and model performance enhancement, using open-source models and tools with other approaches. These architectures have been compared in terms of the task's objective implementation and requirements, pointing out the advantages and weaknesses of each solution and how they can be leveraged to meet the industrial objective.

As a result, to analyse a model's performance, several pieces of information can be drawn from the model's behaviour during its training, such as the correlation between the model learning and the model complexity, the training epochs and the training dataset size, as well as the errors that the model makes after the completed training. All these analyses could help understand the elements that impact the performances and manage them better, in addition to identifying the acceptable weaknesses of the model that the system-level monitoring could handle.

Finally, chapter 5 is completed with the generalisation “assurance” definition, where the objective is to give a quantitative verification of the developed model to ensure compliance with the system-level requirements and less impact on safety. Indeed, generalisation assessment in machine learning relies on the model's ability to maintain performance with unseen data. However, providing guarantees is challenging, as generalisation bounds are based on assumptions about data quality. Various quantitative analyses are necessary to supplement this, including statistical hypothesis testing, simulation under different scenarios, and conformal prediction techniques. Loss optimisation, a wide range of performance measures, rigorous data analysis, and error analysis are crucial. These approaches provide insights into the model's generalisability, aiding decision-making, particularly in critical cases where even a tiny error margin is unacceptable.

Chapter 6 explores the issues of robustness and stability with a global view of evaluation approaches and then a specific overview of formal and analytic methods as applied to models. The literature on stability and robustness is not entirely homogeneous across standards or the state of the art. For example, the concept of stability, in the sense of an algorithmic property, differs between the ISO/IEC literature and the EASA documents (such as the concept paper and the CoDANN reports). If stability is present in the ISO/IEC literature, it is largely absent from the ISO/IEC technical literature on information technology (even outside the subcommittee studying AI). It usually refers to a property of a material or a mechanical device that is not applicable in the current context. However, the concept of robustness, and even the robustness in the context of artificial intelligence, is far more present. The different concepts of robustness that the EASA distinguishes (robustness of the training algorithm, the trained model or the inference model) are more or less aligned between the two. They both refer, to some degree, to the performance of a machine learning model, holding even in the presence of changes in its input. This notion is then considered along the life cycle of the machine learning model, for example, using the W-shaped process described in the EASA concept paper, or another life cycle model, for example, the one used in ISO/IEC 22989. In both cases, it is possible to see the correspondence between phases and the properties to be assessed during these phases.

The main conceptual difference between the notions of stability and robustness in ISO/IEC standards (from the JTC 1 / SC 42) and the EASA concept paper (CP) is that the EASA CP separates them into two different concepts. In contrast, ISO/IEC tends to unify them under the same name of robustness. For the EASA concept paper, robustness is based on input adversity, whereas stability focuses more on regular inputs. ISO/IEC views them similarly since robustness has to be defined in “any” condition, which is valid for adverse or regular inputs.

With the current state of technology, most of the robustness and stability properties that can be verified are mostly at local properties (except in some specific cases where the AI dimensionality allows some global verification to be done). This limitation does not prevent a meaningful process of validation from taking place. For this, the properties must be adequately defined. For example, properties may express some form of stability of a machine learning system (maximum stable space), and others may express some form of bounded behaviour reachable (reachability) or some form of local interpretation (relevance). These properties can be assessed using different methods, each with its advantages and drawbacks, as well as its level of industrial maturity. Chapter 6 analyses statistical (1), formal (2), and empirical (3) approaches that can be used. For each, a survey is done to distinguish which techniques can be used, what tools are identified, and their industrial availability and maturity.

To evaluate an ML system's robustness or stability, it is possible to apply a statistical methodology (1). In short, the general method consists of choosing the data set to evaluate the ML system and the metrics that will be calculated. To do this, the general process will select one or more metrics to consider together. These metrics will then be applied to the machine learning system using the testing data to assess its robustness or stability properties. Performing a testing protocol is not unique to machine learning models, and considerations include the setup of the testing environment, what and how to measure it, and data sourcing and characteristics. During testing, planned data sourcing and availability of computational resources are important considerations due to the massive amounts of data and computational resources required by machine learning models.

Corpus amplification can be used to constitute this corpus of data under the control of the operational design domain definition. The operational design domain cannot only describe perturbations that can affect the input data, but it is also possible to expand the coverage of the test set data. Beyond perturbations that can affect the input data, the test data must also reach possible edge or corner cases. For this purpose, either white box or black box testing techniques can generate edge or corner cases.

It is also possible to rely on formal methodology (2) to assess the robustness or stability of machine learning components. Several approaches are available, such as solver, abstraction interpretation, reachability, or model-checking techniques.
Solvers can rely on different representations of the property to be tested, such as a mixed-integer linear programming problem (a logical formula using satisfiability modulo theory or satisfiability modulo convex). They encode all computations of a given machine learning model as a collection of constraints and then use them to prove robustness properties. Depending on the machine learning model’s architecture, these methods can be complete or incomplete.

Abstract interpretation relies on a theory that constructs controlled approximations that can be built using different domain representations, such as boxes, pentagons, octagons, templates, polyhedrons, zonotopes, etc. It provides an incomplete, deterministic, and white-box method for verifying the robustness of large machine-learning models. Abstract interpretation proposes an inherent trade-off between precision and scalability.

Reachability techniques allow us to verify a machine-learning model's impact over time on an overall system. It can be used in deterministic or non-deterministic environments. In deterministic environments, it combines solvers on a closed-loop system to determine an over-approximation of the reachable set of the system at the next iteration of the loop. In non-deterministic environments, it is combined with probabilistic model checking to determine the probability of reaching a set of states. Probabilistic model checking determines the likelihood of achieving a particular set of states from a given initial state using dynamic programming. Adapting this framework to work with cells rather than single input states makes it possible to obtain an overapproximated probability of reaching a set of states when using a machine learning system.

Finally, model checking is a method to prove that a formal expression of a theory is valid under a particular interpretation. A theory is expressed by a vocabulary of symbols comprising constants, functions, and predicates to build sentences that state assertions about the intended semantics of an idea. Sentences of predicate logic or data patterns can express a theory. Machine learning models are algorithms designed to discover and use data pattern models. The data pattern model is checked against the input.

Empirical methods can also be an option to evaluate robustness and stability properties. Contrary to statistical or formal, they rely at some level on human expertise and expert judgment to assess. For example, in the case of a posteriori testing techniques, the field truth is ambiguous. Since it is impossible to determine all possible correct answers a priori, a-posteriori evaluations are performed. Human annotators or automated measures look at the systems' outputs to decide whether or not they are "acceptable" or "incorrect". In field trials, machine learning is integrated into a system that operates in a realistic environment for the application. In this context, data acquisition and sourcing are integral to the design and execution of experiments. Finally, benchmarking is a technique used to evaluate a machine learning-based system.

It is the first step in building confidence in an AI solution based on machine learning models. Still, it could introduce elements of subjectivity, such as in the tagging or annotation of test data sets by expert practitioners. Each method is different in suitability, ease of use, and properties to be proven.

Statistical methods are well suited for evaluating stability, bias, and variance but are not helpful for relevance or reachability properties. Formal methods are well suited for stability, relevance and reachability. Finally, empirical methods have moderate suitability for each property. Also, each method may have differences in terms of ease of setup. Statistical methods may be the most straightforward way to analyse these properties. However, they require much preparation to set up the right data sets. Also, any attempt to sample exhaustively is immediately limited by the high dimensionality of the input space. In terms of tools, many libraries provide the necessary functions to evaluate statistical metrics. However, few tackle the data issues associated with such methodologies.

While formal methods promise to overcome this limitation, they suffer partly from scalability issues that few tools can overcome. The available tools can vary in terms of maturity. Most are still academic tools, but a few industrial solutions are starting to emerge. These methods offer more substantial properties in robustness and stability; however, they are often limited in what they can prove over the input space.

Finally, empirical methods may be considered the most practical because they require the system to be up and running to be evaluated. However, these approaches can only provide a black-box understanding of the system's properties. Unlike statistical or formal approaches, they do not allow evaluation of the required properties of the system with the same level of confidence. Their use may be considered for applications of low criticality, depending on the objective that the system has to meet.

	Empirical methods	Statistical methods	Formal methods
Stability of the training algorithm	Not suitable	Suitable	Not suitable (the training algorithm is still probably too large)
Stability of the trained model	Could be used but with limited confidence in the results	Suitable	Suitable
Stability of the inference model	Could be used but with limited trust in the results	Suitable	Suitable
Bias	Not really well suited	Suitable	Not really well suited
Variance	Not really well suited	Suitable	Not really well suited
Robustness (Corner case exploration)	Could be used for very specific catastrophic scenario	Suitable	Could be used in combination with statistical approach
Relevance	Expert judgment	Not suitable since it requires some form of symbolic analysis	Suitable in combination with empirical assessment
Reachability	Not suitable since it requires strong guarantees	Not suitable since it requires strong guarantees	Suitable
Scalability	Human intervention needed	Doable but through sampling	Doable but locally
Methods	• Field trial • A posteriori testing • Benchmarking	• Combining metrics	• Solver • Abstract Interpretation • Optimization

Figure 4 - A priori assessment of the suitability of the different types of methods

Overall, combining techniques would benefit any meaningful evaluation of robustness and stability. This way, the process would cover as much of the input space as possible while maintaining operational feasibility. Chapter 6 offers a variety of results both on open-source use cases (from space, automotive and healthcare sectors) and the avionic use cases described in Chapter 3. These results verify that the requirements on stability and robustness of the EASA CP are indeed applicable and that the combination of methods allows for good practices to be defined and recommended in the context of the generic pipeline (in Chapter 7).

Chapter 7 represents a global analysis of the project's outcomes to address the already specified objectives for each task. It completes the defined ML/DL development pipeline in task 2 to enhance the model's generalisation, robustness, and performance verification in compliance with a system's target application and AI-level requirements. Chapter 7 recalls the main issues identified in the state-of-the-art concerning the analysis of the standard practice. It then brings a set of elements and a

checklist of verifications concerning data management, the ML/DL models development and delivery, and the reinforcement of the model robustness; this chapter allows us to locate the stages of the W-shaped process where the main issues have been identified and how they can be handled, through a set of suggested verifications and solutions.
Before presenting the pipeline implementing the W-shape learning assurance, the dependencies between the system-level and AI-level requirements, in terms of performances and intended functions, have been first discussed to allow a clear understanding of the correlation between these levels and how the objectives must be aligned. Besides, the results of the experimental analysis concerning data qualification and model evaluation have been served to identify the different aspects that need to be verified at each level (e.g., during the model design, during the datasets preparation, the features selection, the training and the learning behaviour analysis of the model…). This analysis highlights the importance of considering the system-level operating conditions in performance requirements for ML modules, ensuring compliance with system-level safety requirements at the AI level through cascaded requirements. In *Figure 5*, the importance of defining performance expectations in a well-known ODD is emphasised, where performance metrics and error analysis would help investigate if the target objectives were met to build trust toward the trained models.

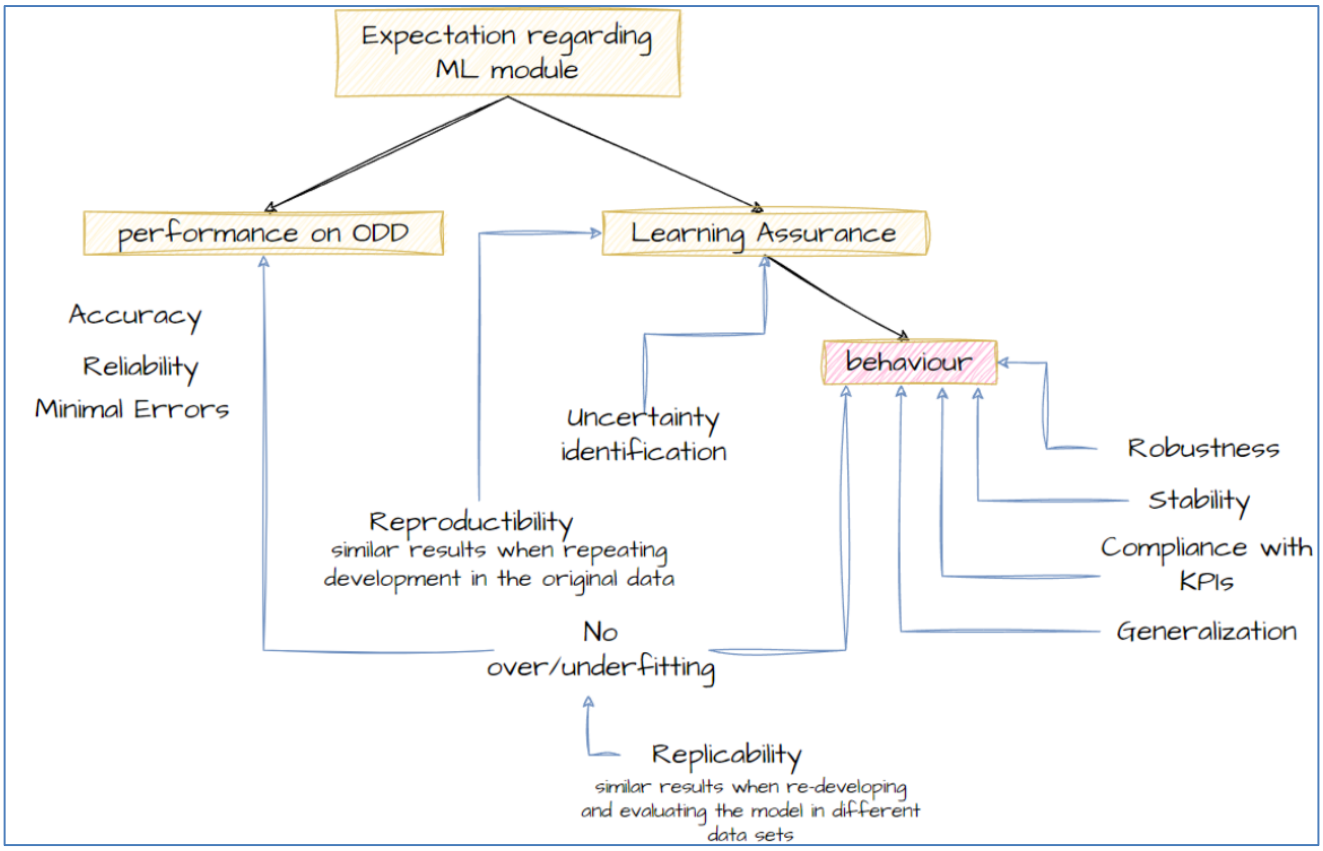

Expectation regarding ML module
performance on ODD
Learning Assurance
Accuracy
Reliability
Minimal Errors
behaviour
Uncertainty identification
Robustness
Stability
Compliance with KPIs
Generalization
Reproductibility
similar results when repeating development in the original data
No over/underfitting
Replicability
similar results when re-developing and evaluating the model in different data sets

Figure 5. Expectations regarding the ML module of a system and the related features derived from the development pipeline

Thus, by being aware of system-level requirements and taking them into account at the AI component level, the latter must enable the trustable behaviour of the trained model. This includes assurance of its robustness, generalisation, a minimum error that does not exceed the system's tolerance limit, compliance with the KPIs, and reproducibility of the results. When verified, this set of measurable qualities will provide a 'guarantee' that the model will perform well in the target system.

A generic development approach is then defined to meet the aforementioned development objectives, including specifying the target application as a mathematical model and defining a complete pipeline. The aim is to implement the W-shaped learning assurance while securing the different stages with a set of verifications to help deliver and validate industrial ML/DL models.
The generic development pipeline is built upon several steps to be performed, depending on the target application and the task being addressed, where the evaluation of ML/DL models is two folds:

A priori evaluation. The listed development and design pitfalls and mishandlings of the task should be identified while evaluating data quality, setting the model target objectives, and determining the generalisation bounds evaluation. The main inputs are the data quality, volume criteria, and generalisation bounds. This first evaluation will specify data requirements regarding completeness, representativeness, and volume necessary for model training. Finally, concerning the target system requirements, a hypothesis on the performance requirements of the ML/DL model should be made;

A posteriori evaluation. Where a set of technical metrics should be used, w.r.t target task, along with a set of the domain-specific (business) key performance indicators that verify how well the model can meet the expectations of the final application, in addition to the generalisation bounds verification. Finally, the hypothesis on the performance requirements of the ML/DL model will be either verified and hence validate the resulting model or compared to the performance of the obtained model, which will then allow us to identify ways to optimise the model better.

Hence, the pipeline[1] is composed of six steps:

1. Data evaluation and qualification (related to Task 1). Aims to determine a minimal size of data needed, perform the quality evaluation (completeness and representativeness), define the enhancement operations (data augmentation, processing, cleansing, balancing, and splitting);
2. Model development and adaptation. It takes into account the data constraints (size of inputs and type, alignments…), leverages the mappings between the inputs and outputs, includes the performances influencing elements identified during the ODD analysis in the model design and architecture enhancement, as well as the metrics selection and acceptability criteria definition to be used further in step 3 ;
3. Model training and evaluation on the optimised dataset. It includes a benchmark definition and a set of industrial KPIs to define and select adapted evaluation measures and thresholds. In the a posteriori evaluation of the trained model (also related to robustness verification in Task 3), an empirical and statistical assessment of generalisation and robustness is made;
4. If the objectives are not met based on the validated data and the optimised model, backtracking to the data management and qualification is possible to enhance the data quality and volume to be more adapted to the training requirements while reflecting the ODD definition;

[1] This process is designed to an offline model training setting. It applies to supervised models' development setting, and can be further extended to online training settings (e.g., lifelong learning frameworks) taking into account characteristics of the target application.

5. After the model implementation, a performance verification in the target environment will be made. This will consider different environment and system elements impacting performances regarding the system/target performance requirements. At this stage, an essential drop in performance can drive a step back to the design phase for a model adaptation to make sure that there will be fewer *nasty* surprises after the model integration in the target system;
6. Finally, after model implementation and during the deployment phase, there could be some updates at system-level requirements from which new AI-level requirements will cascade. Hence, it is necessary to get back to the target application definition and the system understanding to verify if the already validated ML/DL model is still compliant with the target objectives of the whole system. To do so, the verifications and evaluations that have already been made will be rerun for the new application, with updates on measures and metrics selections considered.

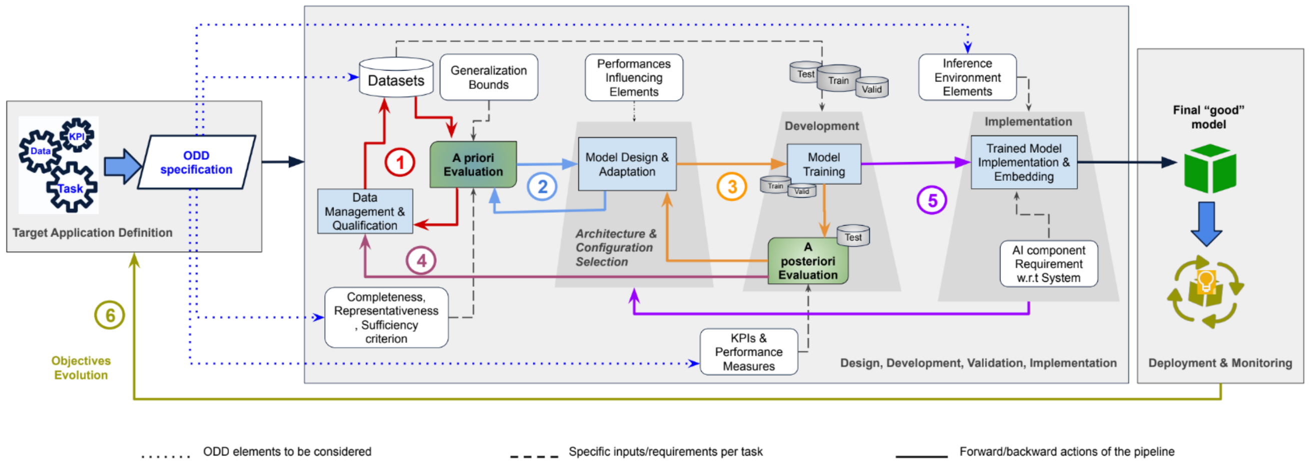

Data
KPI
Task
ODD specification
Target Application Definition
Datasets
Generalization Bounds
Performances Influencing Elements
Test
Train
Valid
Inference Environment Elements
Development
Implementation
Final "good" model
A priori Evaluation
Model Design & Adaptation
Model Training
Trained Model Implementation & Embedding
Data Management & Qualification
Architecture & Configuration Selection
A posteriori Evaluation
AI component Requirement w.r.t System
Completeness, Representativeness , Sufficiency criterion
KPIs & Performance Measures
1
2
3
4
5
6
Objectives Evolution
Design, Development, Validation, Implementation
Deployment & Monitoring
ODD elements to be considered
Specific inputs/requirements per task
Forward/backward actions of the pipeline

Figure 6. A general framework for developing and evaluating ML/DL models, implementing the W-shaped learning assurance.

Main results and perspectives

Science-wise, data quality is a complex topic because of the inherent cost of doing research in the field. Completeness and representativeness are usually not handled per se, and almost no dedicated tools exist. Thus, indicators must be built from more general metrics (such as entropy) or by leveraging different tools (like sample similarity). Intrinsically, the domain is challenging because objectively estimating completeness or representativeness requires knowing the exact extent and distributions of the phenomena to observe. In addition, there is a necessary trade-off between representativity and case coverage since rare cases must be amplified to be modelled correctly. Hence, Chapter 4 provides an analysis of the requirement for the operational design domain to set the expectation for the representativity-coverage trade-off. Hopefully, the array of tools and methods described in the selection grid should allow AI developers to document and justify if the trade-off holds. We would be remiss not to emphasise how much more scientific work is needed to reach operational solutions for more system types. At this point, classification systems are reasonably covered, but other system types, such as transcription, are much less so.

The generalisability of trained models' assessment and evaluation is investigated in Chapter 5 of this report. The generalisation of an ML/DL model depends on the data quality and the learning process. This latter has been reviewed while analysing methods to avoid under/over-fitting, considering the impact of the quality and volume of the data needed for training. We presented methodologies to right-scale the complexity and capacity of the models depending on the scope of the task under development and the volume and nature of input data while measuring the level of generalisation reached by a training session. Furthermore, the selected methods have been evaluated in a toy use case, and their analysis validated in the real aviation use cases. Finally, the selected use cases were investigated, and models and data were first analysed, highlighting the weaknesses developed in existing pipelines. These later are then deep-dived to identify the elements that resulted in a limited performance while comparing the different models in several datasets, pointing out their differences and limitations. Furthermore, to complete the conclusions on the generalisation bounds and the analysis of the gap between the results and the industrial expectations of the use cases, various alternatives were explored, including the estimation of model uncertainty, conformal predictions, and the analysis of errors and their distributions in the test samples. Other model architectures and approaches have also been explored to provide answers to questions concerning the performance limitations of use-case models and their development pipelines.

Measuring the quality of the training step takes part in the larger question of evaluating the resulting trained and inference models. Such an evaluation is driven by several guarantees that need to be gained on the models to ensure an adequate level of confidence in its intended behaviour at a given level of performance. Chapter 6 focuses on two specific guarantees of stability and robustness of machine learning models. We present multiple approaches, from pure performance measures with empirical, data-based approaches to the validation of explicit properties, particularly stability, through an array of analytic or formal methods. Those methods, while sometimes challenging to put into practice, allow for compelling analysis of the behaviour of the models, including at runtime, allowing monitoring of the whole system in a live setup. They were proven helpful in various use cases from different sectors and of varying difficulty. The experiments allowed us to verify and expand the findings of Formula IPC (EASA and Collins Aerospace, 2023) regarding formal methods. Hence, these evaluation methods on the trained models' robustness and performance stability can be leveraged in the pipeline developed in Chapter 4 to ensure better performances after implementation.

Finally, the different analyses of the tasks of this project have been leveraged to construct an operational proposal implementing the W-shaped learning process, which is proposed in Chapter 7. The objective is to leverage the project findings and the selected methods within a revisited development and implementation pipeline of ML/DL models in the industry while taking into account data-level evaluation, learning-level verifications and performance evaluation concerning industrial expectations (KPIs integration in the learning verification and management), and finally the verification of the requirements after implementation. In Chapter 7, a way toward an understanding ML-level requirements verification is first proposed. A qualitative analysis of ML-based applications is defined to help verify compliance with the system-level requirements. It highlights the main issues handled in each step of the W-shaped process and provides a set of verifications and means to overcome them in the generic pipeline. This approach is finally explored in an experimental analysis where different aspects related to data and models have been investigated, highlighting the impact on models' performances.

Description of the consortium

The consortium in charge of the project is a partnership of three entities: Airbus Protect, one of the aeronautics domains, and LNE and Numalis, two transverse entities.

Airbus Protect is an independent subsidiary of Airbus that brings together expertise in safety, cybersecurity, digital, and sustainability-related services. As a risk management company, this entity aims to offer end-to-end advisory, consulting services, training programs, and software solutions.
Pairing expertise built through large-scale projects with the latest insights from its research programs, it supports customers, partners, and their ecosystems in different industrial domains. Airbus Protect is already a trusted partner of customers in high-tech industrial manufacturing, aerospace, transportation and future mobility, energy and utilities, financial services, critical infrastructure, governments, institutions, and defence. The mission of Airbus Protect is to contribute to making its clients' businesses and products safe, secure, and sustainable. Airbus Protect brings together more than 1,600 experts in France, Germany, the UK, and Spain to create a centre of excellence to meet the clients' evolving needs. Airbus Protect combines more than 36 years of experience with industry-leading expertise to deliver services in three areas: Cybersecurity, providing consulting and managed security services to help our clients establish and maintain persistent cyber resilience; Safe Mobility, ensuring the safety of tomorrow's intelligent mobility solutions and smart cities; Sustainability developing new ways of working, new products and zero-emission energy supplies.
Airbus Protect team implements several Data/AI/engineering projects, including MLEAP:

- SmartPlanif / MaiVA (Maintenance Virtual Assistant) is an airline-centric tool that supports customers by automating and providing increased assistance to activities.
- Climate and energy challenge: aims to provide structured and semantic access to many climate and energy ecosystem data.
- eIODA (Environmental Industrial Operations DAta Foundation) aims to create a single source of truth for all departments and Airbus divisions to enable Environmental Official Reporting and Environmental Performance Management.

The French National Metrology and Testing Laboratory (in French, "Laboratoire National de métrologie et d'Essais" or **LNE**) is a public industrial and commercial establishment (EPIC) attached to the Ministry of the Economy and Finance. It is the central support body for the public authorities in testing, evaluation, and metrology. Its action aims, in particular, to examine new products and assess their impact to inform, protect and meet the needs of consumers and national industry. In this context, it carries out measurement, testing, characterisation, and certification work on systems and technologies to support breakthrough innovations (artificial intelligence, cybersecurity, nanotechnologies, additive manufacturing, radioactivity measurement, hydrogen storage, etc.) for the benefit of the scientific, normative, regulatory and industrial communities. LNE has particular expertise in evaluating artificial intelligence (AI) systems. It has carried out more than 950 evaluations of AI systems since 2008, notably in language processing (translation, transcription, speaker recognition, etc.), image processing (person recognition, object recognition, etc.), and robotics (autonomous vehicles, service robots, agricultural robots, collaborative robots, intelligent medical devices, etc.). It participates in the significant challenges of AI by developing standards to guarantee and certify these technologies. The collaborative projects that it conducts at the national, European, and international (in particular via its strategic partnership with NIST on AI and robotics) aim first and foremost to define standards and protocols (using various conformity assessment methods: literature review, testing, on-site audits), metrics and testing environments (databases, simulators, physical or mixed test benches) for AI, are varied and involve it in almost all technical and socio-economic issues, ethical questions, and sociological issues and networks of institutional actors (programmatic collaborations with the OECD, the High Authority for Health, the Cofrac and the most French Ministries) and industrial partners (agreements with Thales, Dassault, Airbus, Facebook, CEA, etc.) in the field. In December 2020, as an impartial and independent third party, it launched a working group to define the first AI certification standard in a consensual manner.

Numalis is a software editing company specialising in the reliability of AI systems. The goal of Numalis is to allow companies to accelerate the path of AI adoption by making its design, validation, integration, and deployment more reliable. Numalis is involved in several industries with safety-critical concerns, such as Defense, Aeronautics, Aerospace, Railway, (and Healthcare). For them, Numalis provides a unique set of tools and expertise to improve the maturity (and ultimately the adoption) of their use of AI technologies in their future systems. Currently, Numalis has developed Saimple, a solution based on abstract interpretation. By using only formal analysis, Saimple allows to measure the robustness of neural network or support vector machines (SVM) models against specific types of perturbation tied to the domain of use employed, visualise in a human readable fashion the robustness across the input space, and extract explainability components from the system. As robustness and explainability are critical components in most software quality models for future EU regulation (the AI Act), Numalis aims to develop standards at the international level to bring uniformity to processes across all industries. These standards are written to bring good practices in using formal methods of AI. To that effect, Numalis is currently the editor of ISO/IEC standardisation documents (the ISO/IEC 24029 series) related to assessing the robustness of neural networks. Founded in 2015 in Montpellier, Numalis employs 18 people, primarily PhDs and engineers specialising in formal methods and software development.

ABBREVIATIONS

ACRONYM	DESCRIPTION
AI	Artificial Intelligence
BC	Boundary Condition (Feature space characterization method)
CEA	Commissariat à l'Energie Atomique
CI/CD	Continuous Integration/Continuous Delivery
CoD	Curse of Dimensionality
CoDANN	Concept of Design Assurance for Neural Network
ConOps	Concept of Operations
CP	Centroid position (Feature space characterization method)
CUDA	Compute Unified Device Architecture
DL	Deep Learning
DM	Data Management
DNN	Deep Neural Network
DQR(s)	Data Quality Requirement(s)
DoC	Degree of Correspondence
E2E SLU	End-to-End SLU
EASA	European Union Aviation Safety Agency
EASA CP	EASA Concept Paper: Guidance for level 1 & 2 Machine Learning applications
EDA	Easy Data Augmentation
ELA	Equalized Loss of Accuracy
EP	Equivalence Partitioning
ERM	Empirical Risk Minimization
GA	Genetic Algorithm
GAN(s)	Generative Adversarial Network(s)
HPC	High Performance Computing
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IID	Independent and Identically Distributed random variables
ISO	International Organization for Standardization
KNN	K-nearest-neighbours
KPI(s)	Key Performance Indicator(s) (index)
LDA	Linear Discriminant Analysis

LM	Learning Management
LPV	Linear-Parameter-Varying
LoR	Logistic regression
MAR	Missing At Random
MCAR	Missing Completely At Random
MILP	Mixed-integer linear programming
ML	Machine Learning
MLEAP	Machine Learning Application Approval
MLP	Multilayer perceptron
MMD	Maximum Mean Discrepancy
MNAR	Missing Not At Random
MUP	Maximum Uncovered Pattern
NLP	Natural Language Processing
NN(s)	Neural Network(s)
OD	Operational Domain
ODD	Operational Design Domain
OECD	Organisation for Economic Co-operation and Development
ONNX	Open Neural Network Exchange
OOD	Out-of-Distribution
PAC	Probably Approximately Correct
PBC	Pair-wise Boundary Conditioning (Feature space characterization method)
PCA	Principal Component Analysis
RBF	Radial Basis Function
RNN	Recurrent Neural Networks
ReLU	Rectified Linear Units
SGD	Stochastic Gradient Descent
SLU	Spoken Language Understanding
SME	Small and Medium-sized Enterprise
SMT	Satisfiability Modulo Theory
STT	Speech-to-text
SVM	Support Vector Machines
UC	Use Case
TL	Transfer Learning
Tanh	Hyperbolic Tangent Function
WER	Word Error Rate

CONTENTS

List of Tables

List of Figures

1. Introduction

1.1 The MLEAP project

Artificial Intelligence (AI) is becoming ubiquitous, and many industrial domains, including aeronautics, aim to harness its promises to improve their performance. The most spectacular progress of contemporary AI comes from Machine Learning (ML). ML systems extract and learn behavioural patterns for a given task from data, which are samples of the operational context of the considered task. Their use in essential or even critical systems poses not-yet-solved problems, particularly in aeronautics, which is at the core of the MLEAP project.

The MLEAP project is a two-year work initiated by the EASA to collate and evaluate the state of the art on three main topics:

- Data: completeness and representativeness
- Model development: Generalisation properties
- Evaluation: robustness and stability

The aim is to build a reference document on those topics for the AI application domains concerned by the EASA certification activities and to identify possible means of compliance with related objectives of the EASA AI concept paper (EASA, 2024).

An important step in addressing these topics is to set up the adapted scientific framework. We first define the semantics and research context corresponding to the three main concepts: data, model and evaluation. To this end, a complete definition of the terminology used is provided, as well as an explanation of its use and positioning within the project. Next, we define the evaluation settings, showing the different measures and their targeted performance and use. Finally, a description of the scientific and experimental process is provided, illustrating the interaction between the various project tasks and the sequence of experimental work.

This document represents the project's final deliverable. The core of the report is a public document. The appendix contains proprietary information and shall be removed before sharing the report with third parties.

1.2 Core definitions

Those topics happen to have variable definitions from one document to another. To scope the boundaries of this document, we will first refine which definitions we will base our work on.

1.2.1 Use cases-related terminology

Depending on the target application, the data being used, and the system under development, a set of definitions can be added. In the scope of MLEAP, we provide the core definitions related to the selected use cases only (cf. Chapter 3).

- **Operational Domain (OD):** As part of the ConOps definition, the OD corresponds to the complete description of the conditions under which the whole system (including the AI part) should operate, along with the operational scenarios (EASA, 2024). The specific operational limitations and assumptions should be defined, as well as the already identified risks, associated mitigations, and impacts on the AI component.

- **Operational Design Domain (ODD):** This is a part of the OD definition and represents the operational conditions in which the ML constituent is designed to operate correctly. For every application, the ODD is an abstraction of the operational context and needs to be known to state guarantees on performance and safety (Heyn et al., 2022). According to the EASA's recommendation (EASA, 2024), the ODD should include a refinement of the OD into the AI/ML constituent. Hence, additional parameters can be identified and defined for the AI/ML constituent. Thus, their definition could vary from one use case to another (e.g., data type, initial conditions, system requirements …) while the components remain the same. To cope with this, we must define the technical requirements and initial conditions under which the system is designed to function.

- **Utterance**: used in speech recognition applications, it refers to an audio sample. As defined by the Indian Institute of Technology[2], this corresponds to *"An utterance is the vocalisation (speaking) of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences."*.

- **Intended behaviour** corresponds to the expected functionalities, enabling meeting the objectives for which the ML/DL constituent is designed. Note that this can also be referred to as "intended function" in several discussions in different working groups in AI (e.g., WG114/G34, WG63/S18) about the use of intended behaviour and intended function. In this report, we consider the EASA's recommendation, which will be part of the final concept paper Issue 2, concerning using intended behaviour to refer to the developed functionalities and expected ones for the ML/DL component.

1.2.2 Data completeness and representativeness

This section aims to analyse to what extent literature may diverge in accepting the notions or in the scope they cover. Indeed, the fact that some aspects related to quality attributes are not correctly defined in the community considerably hinders the research for appropriate methods, not only in the present deliverable but also for the users of the DQRs relative to these notions. This section thus highlights what parts of the commonly found definitions are stable among the different studies and what aspects may present difficulties and require more exploration and research results from the community.

[2] https://www.iitk.ac.in/LDP/HOWTO/Speech-Recognition-HOWTO/

In (EASA, 2024), the notions are defined as follows in section G. Annex 3 – Definitions and acronyms:

- "**Completeness** — *A data set is complete if it sufficiently (i.e., as specified in the DQRs) covers the entire space of the operational design domain for the intended application*."
- "**Representativeness** (of a data set) — *A data set is representative when the distribution of its key characteristics is similar to the actual input state space for the intended application*."

The notions are further explained in the context of the points of verification of the EASA CP, which allows the guidance reader to deepen the understanding of each attribute. The dedicated sections ("Anticipated MOC DM-13-1: Data completeness" and "Anticipated MOC DM-13-2: Data representativeness") offer operational methods and objectives. In addition, the reader can note that both attributes should be analysed "*with respect to the ML requirements and the AI/ML constituent ODD*" (section C.3.1.3), that "*the assurance process should be shifted on the correctness and completeness/representativeness of the data (training/validation/test data sets) and on the learning and its verification*" (section C.6.1), or that "C*ompleteness and representativeness of the data sets are prerequisites to ensure performance on unseen data and to derive generalisation guarantees for the trained model*" (section C.3.1.3.8).

In the following section, the term "EASA definition" will encompass both the definition from section G of the Concept Paper and the descriptions provided throughout the document.

1.2.2.1 **Completeness**

- ➢ EASA's definition of completeness is in line with literature and standards. However, all definitions include reference to notions linked to the operating conditions of the system, which are well defined in EASA's work (ODD) but seldom defined in the literature (e.g., "context of use"), hence not allowing comparison on this aspect.
- ➢ In EASA's acceptance, completeness must be considered relative to the Operational Design Domain (ODD), ML requirements, ConOps, and intended application. However, ISO/IEC 25012:2008 only refers to "a specific context of use", which is not strictly defined in the standards or literature.
- ➢ A challenge of the definition of completeness pertains to the identification and representation of the relevant scope of the system (ODD, context of use, intended application, etc.).

ISO/IEC 25012:2008 defines completeness: "*The degree to which subject data associated with an entity has values for all expected attributes and related entity instances in a specific context of use*" (ISO/IEC 25012, 2008).

The scientific literature does not always explain or define the attributes explored in the papers. In some cases, completeness is presented along with selected definitions – the article (Shrestha et al., 2022) on data analysis for urban infrastructure, for example, opts explicitly for a definition derived from (Veregin, 1999), stating that completeness is understood as "*feature completeness, which refers to the known presence and location of all* [infrastructure components]*, and attribute–value completeness, which refers to known attributes and values of each component*". In another paper dedicated to assessing completeness, availability and consistency of health record data (Nobles et al., 2015), no theoretical definition of completeness is provided. Instead, an operational definition of "presence of data" is proposed as the percentage of visits to the medical facility, which are stored in the database with all appropriate fields filled in. These studies highlight all the aspects presented in the ISO definition, namely the importance of values in the target context of use.

The study *"Data completeness measure"* (Emran, 2015) provides an overview of the definitions and methods for computing completeness. The paper mentions the direct relation to missing information and details four (4) different types of "*missing values"*:

(i) null-based missing values - where nulls represent missing values;
(ii) tuple-based missing values - the absence of "attribute-value" tuples;
(iii) schema-based missing values - missing attributes and entities from the schema;
(iv) population-based missing values - missing individuals compared to a reference population[3].

The information from the study is more of a functional description of completeness than a definition *per se*. While the concepts addressed do not seem to contradict the ISO definition, the notion of context of use is not represented in the approach, and only indirectly.

In general, one can understand that the community seems to agree that completeness assessment is linked to the study of missing information. However, the context of use provided by the ISO definition is mostly implicit. The standard (ISO/IEC CD 5259-2, 202X), in its present state, covers this notion entirely remotely by specifying that completeness should be tackled differently depending on specific usage contexts. It seems evident that the first primary step in defining operational methods of completeness assessment should encompass the formalised description of such specific usage contexts. Notwithstanding the vagueness of the "context of use" notion, the ISO 25012 definition seems coherent with the EASA definition.

1.2.2.2 **Representativeness**

- EASA's definition of representativeness aligns with literature and standards, except that EASA covers learning, validation, and test data sets, while the other sources only consider training data.
- In the literature, representativeness sometimes seems confused with the data quality attribute of relevance, which focuses on the goodness of data features as predictors for ML.

Representativeness is not addressed in (ISO/IEC 25012, 2008). In its present state, the standard (ISO/IEC CD 5259-2, 202X) refers to the degree to which a data set used for training reflects the target population under study. The target population is characterised by analysing the data expected in the AI system's target conditions.

[3] One can note that this last point seems to refer to representativeness; however, the authors of the paper confirm that this aspect is under-addressed in the literature, and its relevance cannot be confirmed.

The scientific literature does not provide strict definitions, but the works articulate similar notions, regardless of the final use of the data (for analytics or data-based AI). An OECD working paper (Bajgar et al., 2020) presents an analysis of representativeness in the context of the Orbis private companies database. The document does not define representativeness; however, the study shows that it is linked to the distribution of the population in terms of societal and demographic characteristics, a distribution deemed adapted to the object under study. This approach seems in line with the ISO approach. In another paper about AI fairness and inclusion, (Kamikubo et al., 2022) tackle the notion of representativeness in that the data sets used for training should represent the target demographic groups (in terms of age, gender and race/ethnicity). The paper highlights some challenges for the design of representative data sets, including the fact that demographic variables are sensitive and complex to annotate or the absence of certain elements ordinarily present in the population but for which research does not provide enough tools for tracking them.

One should note that representativeness is linked to relevance (see Section 4.5.3), a data quality attribute that assesses (in ML) to what extent the features used for training are good predictors. (ISO/IEC 25012, 2008), that does not provide a strict definition of representativeness, mentions the "*goodness of fit of distributions*" (in section 3.3.5 of the standard), which could correspond both to a reference to the quality of features as predictors as well as a good distribution in terms of the target population of use. It seems then compulsory that the notions be distinguished and that the intended scope of representativeness is confirmed.

1.2.3 Generalisation assessment

The set of performance indicators needs to be defined as part of the model evaluation. In this document, the following definitions, primarily aligned with the AI state-of-the-art, are used:

- **Generalisability**: Machine learning model generalisation is the capacity of an ML model to keep an acceptable level of performance on unseen input data (during the training phase) from the ODD.

- **Overfitting**: Overfitting is a concept in data science that occurs when a statistical model fits exactly training data but fails to perform accurately against unseen data from the ODD. An overfitting model fails to generalise well, as it learns the noise and patterns of the training data to the point where it negatively impacts the model's performance on new data.

- **Underfitting**: Underfitting occurs when a model cannot generalise well to new data or create a mapping between the input and the target variable.

Besides, the terminology related to model development and evaluation process is based on the following definitions, primarily used in the literature, and that we have adopted. Concerning the recommendations in the concept paper by EASA (EASA, 2024), to guarantee compliance with the objectives of the AI reliability guidelines, an overview of the concept of operations (ConOps), describing precisely how the system will be operated is expected to be established, including the task allocation and operating conditions of the AI-based system, this includes the OD specification at the system level, and the ODD definition, at the ML/DL component level. For this later, AI modules require the following elements to be defined:

- **Model architecture:** Represents a set of design features characterising the ML model, such as the type of the neural network, the ML model class or even some design-related details (e.g., size and type of the different included functions).

- **Data Dimensionality:** Based on statistical learning definitions of data dimensionality (Hastie et al., 2009), this feature can influence the choices made for AI-based systems. Hence, we rely on two levels to define the data dimensionality of the different use cases: (1) High-dimensional data are defined as data in which the number of features (variables observed) is close to or larger than the number of observations (or data points); (2) The opposite is low-dimensional data in which the number of observations far outnumbers the number of features. A related concept is broad data, which refers to data with numerous features irrespective of the number of observations (similarly, tall data is often used to denote data with many observations). Hence, analyses of high-dimensional data require consideration of potential problems that come from having more features than observations. Considering use cases from both (1) and (2) data dimensionalities, one crucial issue is the impact on the ODD definition. Note that while applications of class (2) seem to be easier to handle, compared to those of class (1), where many of the features cannot always be known (a priori, and very often a posteriori), which makes the definition of the ODD even more complex with such use cases.

- **Model Complexity:** refers to the characteristic defining how sophisticated/complicated the model is (Verhagen, 2021). Several ways can be explored to define model complexity. Still, the main feature relies on the number of trainable variables and the required data to train a performing model.

1.2.4 System robustness and stability

Robustness and stability do not necessarily constitute unambiguous notions depending on which document the reader is referring to. This section considers the literature from the EASA (and the WG114 [4]) and the one from ISO/IEC JTC 1/SC 42[5]. By comparing the definitions present, we could first draw the following table of correspondence.

[4] https://www.eurocae.net/news/posts/2019/june/new-working-group-wg-114-artificial-intelligence/

[5] For the sake of clarity, the remainder of this document will use the term ISO/IEC as a shorthand for the full name ISO/IEC JTC 1/SC 42. However, when referring to a different subcommittee of either ISO or IEC, its complete name will be utilized.

1.2.4.1 **Robustness and stability**

Definition	Stability	Robustness
ISO/IEC	Despite having some documents mentioning the concept of stability (for example in (ISO/IEC 14496, 2019)), there is no proper definition of an algorithmic stability available in the ISO/IEC.	ISO/IEC 22989:2022 (ISO/IEC, 2022a) and ISO/IEC 24029-1:2021 (ISO/IEC, 2022b) ability of a system to maintain its level of performance under any circumstances ISO/IEC TS 5723 ability of a system to maintain its level of performance under a variety of circumstances ISO 18158:2016 measure of the capacity of an analytical procedure to remain unaffected by slight variations in method parameters and indicates the method's reliability during normal usage ISO/IEC/IEEE 24765:2017 degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions
EASA CP	Stability of the learning algorithm: Refers to ensuring that the produced model does not change a lot under perturbations of the training data set. **Objective LM11**: The applicant should provide an analysis of the stability of the learning algorithms. Stability of the trained model: Refers to keeping input-output relations of the model under small perturbations	**General definition**: A system's ability to maintain its performance level under all foreseeable conditions. At the model level (training or inference), the robustness objectives are further split into 'model stability' and 'robustness' in adverse conditions. Robustness of the trained model **Objective LM-13:** The applicant should perform and document the verification of

	Objective LM12: The applicant should perform and document the verification of the stability of the trained model. Stability of the inference model: **Objective IMP-08:** The applicant should perform and document the verification of the stability of the inference model.	the trained model's robustness in adverse conditions. Robustness of the inference model **Objective IMP-09**: The applicant should perform and document the verification of the robustness of the inference model in adverse conditions.

Table 1 Correspondence between high-level concepts in EASA CP and ISO/IEC

First, we have to observe that the concept of stability, in the sense of an algorithmic property, is mainly absent in the ISO/IEC information technology technical sector literature (even outside the subcommittee studying AI). It usually refers to a property of a material or a mechanical device, which is not applicable in the current context of this document.

Second, the different concepts (of the training algorithm, the trained model or the inference model) on robustness are more or less aligned between the two. They both refer to the ML model's performance holding to some degree, even when changing input. EASA CP is analysing these changing inputs by exploring (in LM12) edge cases, corner cases, outliers, out-of-distribution or even adversarial cases. At the same time, the ISO/IEC does not overly detail those aspects.

The main conceptual difference between the notion of stability and robustness in ISO/IEC and EASA CP resides in the fact the EASA CP split them into two different concepts. In contrast, ISO/IEC tends to unify them under the same name of robustness. For EASA CP robustness resides in input adversity, whereas stability is more focused on regular inputs. ISO/IEC views them similarly since robustness has to be defined in “any” condition, which is valid for adverse or regular inputs.

In the rest of the document, the term Robustness when applied to a model, is used in the sense of the ‘robustness in adverse conditions’ definition. However, the general definition of Robustness corresponds to the notion of maintaining the performance in all foreseeable conditions which encompasses both stability and robustness of a model in adverse conditions.

1.2.4.2 **Corner and Edge Case Concepts**

In this section, we introduce some definitions of concepts related to corner cases. Then, we present a scale to classify the various kinds of corner cases. It is important to note that several terminologies are used in the literature differently. This is why, in section 1.2.4.2.4, the document will attempt to draw the relationship between the state-of-the-art and the EASA CP definitions. Corner and edge cases are essential since the variety of use cases considered in the MLEAP project (see Chapter 3) will explore several types of them.

1.2.4.2.1 Definitions from the EASA CP

The EASA concept paper (EASA, 2024) uses several related concepts that can be tied to the notion of corner cases. In particular, it revolves around the following definitions:

Corner case	This refers to a situation that rarely occurs on all of the AI/ML constituent ODD parameters, considering at least two of them (i.e., the associated values are poorly represented in the distribution for those parameters).
Edge case	This refers to a situation that rarely occurs, considering a given parameter of the AI/ML constituent ODD (i.e., the associated value is not represented in the distribution for that parameter).
ODD	The ODD defines the set of operating parameters, together with the range and distribution within which the AI/ML constituent is designed to operate, and as such, will only operate nominally when the parameters described within the ODD are satisfied. The ODD also considers correlations between operating parameters to refine the ranges between these parameters when appropriate; in other words, the range(s) for one or several operating parameters could depend on the value or range of another parameter.
Outlier	Data outside the range of at least one AI/ML constituent ODD parameter.

These definitions are then used in some of the requirements expressed in the EASA CP. For example, in DM01 (and its anticipated MOC DA-03), which states that during the different iterations which will happen during the learning phase, particular attention should be put on:

- the definition of nominal data;
- the identification of edge cases, and corner cases data in preparation for the stability of the model;
- the definition of infeasible corner cases data;
- the detection and removal of inliers;
- the detection and management of novelties;
- the definition of outliers for their detection and management.

Several other requirements expressed in the EASA CP relate directly to this notion. For example, IMP-08 (and its anticipated MOC IMP-08-01) mentioned verifying the inference model in adverse conditions, which also concerns edge and corner cases. The requirement LM-13 (and its anticipated MOC LM-13-1) is symmetrical for the trained model, which specifies the same requirement.

1.2.4.2.2 Vocabulary and Concepts for the State of the Art

Several words and concepts are commonly used in the bibliography when dealing with corner cases. It is then necessary to distinguish between corner cases and other terms used in the literature (Heidecker et al., 2021). First of all, the term edge case is used in the context of software and hardware testing. It can also refer to extreme values.

Edge case: Edge cases are situations or parameters that occur rarely but are already taken into account during development (Koopman et al., 2019). The concept of edge cases also covers extreme cases or boundary cases. When the system designers consider edge cases, they become normal cases and are no longer considered edge cases.

Next, ML literature often uses the terms outlier, novelty, and anomaly. They are strongly related to corner cases and may have an overlapping meaning. A summary is given in Figure 7. Literature also offers a wide variety of approaches for detecting anomalies and novelties (Hodge and Austin, 2004; Kumar et al., 2009; Pimentel et al., 2013).

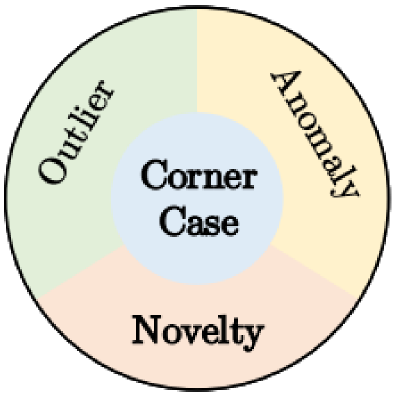


Figure 7. Relations between outliers, anomalies, novelties and corner cases (borrowed from (Heidecker et al., 2021)).

Outlier: An outlier can be defined as an observation that deviates so much from other observations that it suggests that it was generated by a different mechanism (Hawkins, 1980). In (Gruhl et al., 2021), an outlier is defined as a legitimate observation of a known process that occurs rarely and, for example, represents an extreme value.

Novelty: Novelties appear as instances or objects not seen before (Kumar et al., 2009). In (Gruhl et al., 2021), the definition of novelties is a bit broader. Novelties are described as a spatial or temporal aggregation of anomalies or a change in the distribution of an already-known process. The occurrence of new situations, new objects and new movement patterns is an essential characteristic of corner cases, which does not simplify a clear distinction to novelties.

Anomaly: Many definitions of anomalies can be found in the literature (Katsaggelos et al., 2010; Koopman et al., 2019; Kumar et al., 2009). Some definitions focus on anomalies as noisy data occurrences that prompt artefacts that hinder data analysis (Pimentel et al., 2013). Other definitions first define normal events by a high frequency of occurrence, and consequently, anomalies are defined as the opposite (Katsaggelos et al., 2010). Yet other definitions consider anomalies as patterns that are not consistent with learned ones or with general normal behaviour (Kumar et al., 2009; Popoola and Wang, 2012). Anomalies are categorised by (Kumar et al., 2009).

Corner case: While many terms related to corner cases exist in the literature, a general definition and description are missing, as seen in the former paragraphs. The lack of universally agreed-upon technical terms and definitions also makes detecting corner cases cumbersome. Corner cases can be defined in one of the following ways.

- A corner case may result from combining several normal situations or parameters that coincide, thus representing a rare or never-considered case or scene (Heidecker et al., 2021).
- A corner case may result from an entirely new situation, not just combinations of already known ones.
- A corner case may result from a deviation from normality manifested in non-conform behaviour or patterns. The terms anomaly and corner case are then almost used synonymously. Anomalies describe a deviation from normality. Hence, the term appears in the systematisation of corner cases (Fingscheidt et al., 2020).

- A corner case set can potentially comprise data samples exhibiting erroneous and unforeseen behaviours, such as adversarial data on boundaries and misclassified data (outliers) (Tinghui Ouyang, 2021), as shown in Figure 8.
- Another definition is based on prediction: according to (Fingscheidt et al., 2019), a corner case arises if there is a relevant object in a relevant context that a modern system cannot predict.

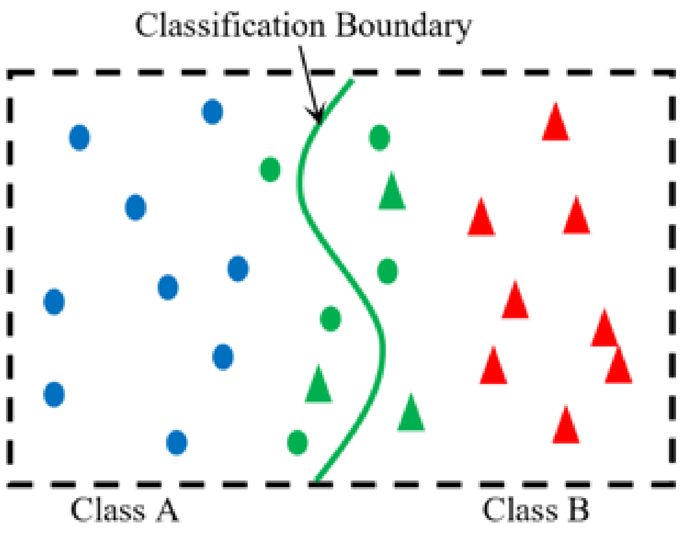


Figure 8 - Illustration taken from (Tinghui Ouyang, 2021) showing corner cases (green) that can be near the classification boundary and include both correct and incorrect classification.

Similar to software testing, corner cases can cease to exist once an appropriate number of examples of a particular corner case have been added to a perception method's training and validation data. In ML, corner cases help to validate and improve systems by retraining. The recorded data of such a situation can be used as part of the training data for the ML system, which can be used in the case of an active learning approach.

1.2.4.2.3 Systematization of Corner Cases on Different Levels

Interestingly (Fingscheidt et al., 2020; Heidecker et al., 2021) propose a systematization of corner cases for visual perception in highly automated driving, where the special cases are classified by level. While this work is dedicated to visual perception, it represents a solid basis to corner case systematization in other sectors.

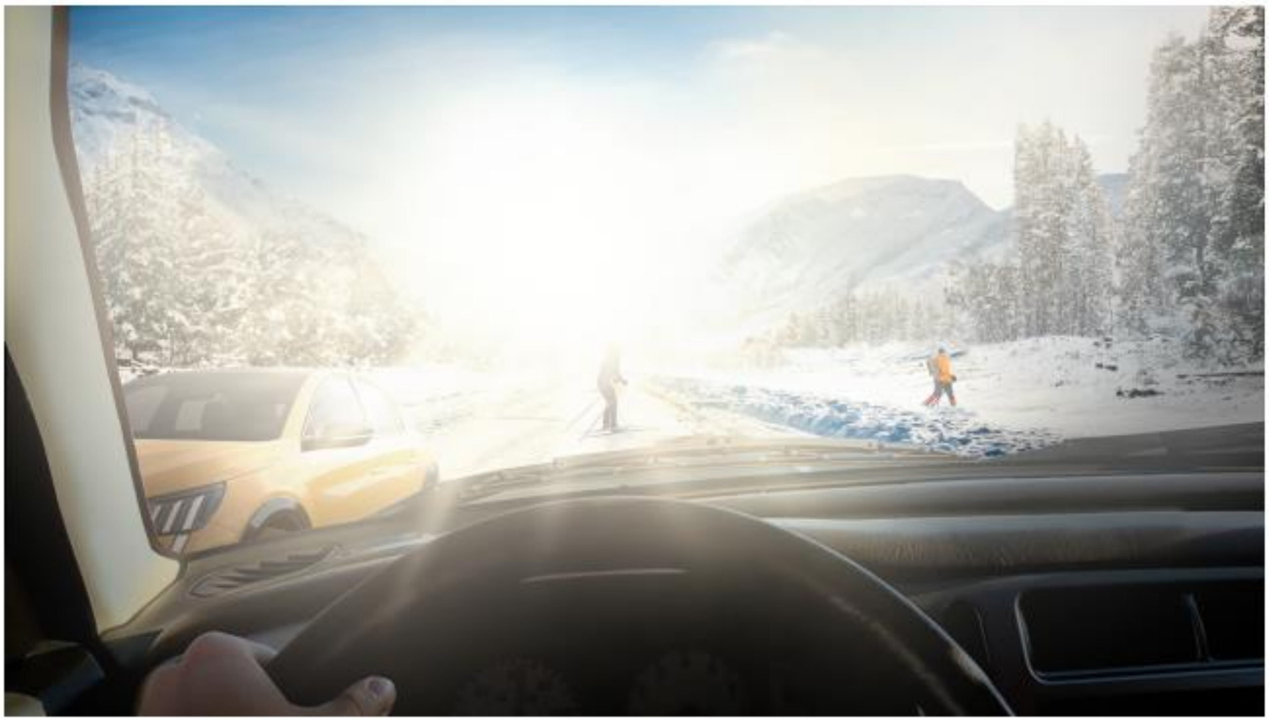


Figure 9. Multiple corner cases: A winter scene with an icy, slippery, reflective road, with low winter sun and people on cross-country skis crossing the road (figure borrowed from (Heidecker et al., 2021).)

These levels are based on the type of situation they encompass and are ordered by their theoretical detection complexity. The authors consider corner cases on the scene (see Figure 9 borrowed from (Heidecker et al., 2021)), objects (e.g., people on cross-country skis), and domain level (e.g., snowy winter), which they summarise into the content layer. In addition, the authors define the temporal layer for corner cases at the scenario level, e.g., the unusual movement of a person with cross-country skis compared to a pedestrian. They grouped the corner cases depending on whether they were concerned with single image frames and point clouds (content layer) or multiple consecutive ones (temporal layer). As the goal is to provide a more comprehensive conceptualisation of corner cases, including multi-modal sensor inputs, they distinguish on the lowest theoretical detection complexity between pixel-, and point-cloud-level corner cases (the sensor layer.)

Let us follow the order in Table 2 (borrowed from (Fingscheidt et al., 2020)) by going from low to high complexity.

- Perceived errors in the data cause the pixel-level corner cases. This type of corner case is at the bottom of Table 2 because the detection complexity is relatively low. At the pixel level, we distinguish two types of outliers. Global outliers occur when all or many pixels are outside the expected range of measurements. This is due to unnatural lighting conditions or overexposure, for example. Local outliers occur when one or a few pixel values fall outside the expected measurement range.
- In domain-level corner cases, the world model fails to explain its observations. Domain-level shifts typically cause this type of corner case, where a large and constant shift occurs in the appearance but not in the semantics. Methods for detecting domain-level corner cases are often related to domain-level adaptation (Bolte et al., 2019).
- At the object level, corner cases arise when instances not seen during training are perceived during inference. Single-point anomalies or novelties are defined by unknown objects in the

observations. Object-level corner cases include any new object appearing in a normal scene, i.e., any unknown type of obstacle or shadows of objects actually not visible in the image.

- The scene-level corner cases do not conform to the expected patterns on individual images. The need to understand the perceived scene induces the theoretical increase in detection complexity. We distinguish two types of corner cases according to the type of understanding necessary for the detection.
  - Contextual anomalies are known objects that appear in unusual locations in the scene.
  - Collective anomalies are known objects that appear in an unusual quantity, such as a large gathering of people during a demonstration or a traffic jam.
- Scenario-level corner cases denote the observation of patterns with temporal context. In addition to scene understanding, temporal understanding is necessary, requiring an image sequence for detection. We consider three different cases.
  - Risky scenarios occur when a pattern observed in a similar form during training reappears during inference and still contains a potential danger.
  - Novel scenarios occur when a pattern not observed during training appears during inference but does not increase the potential of danger.
  - Anomalous scenarios occur when a pattern not observed during training appears during inference and, additionally, increases the potential of danger drastically

This approach is highly related to the vision context, but other methods can be defined for different types of use cases. Sections 4.4.8.8.2 and 4.4.8.8.3 provide more on the topic.

Corner cases	Description	Examples	Literature
Scenario level Patterns are observed over the course of an image sequences Recognition requires scene understanding	**Anomalous Scenario** *Pattern that was not observed during the training process and has high potential for collision*	- Person suddenly walking onto the street - Car accident - Car or person breaks traffic rule	(B. Barz and Denzler, 2019; Chong and Tay, 2017; Fingscheidt et al., 2019; Hasan et al., 2016; Xu et al., 2015)
	Novel Scenario *Pattern that was not observed during the training process, but does not increase the potential for collision*	- Truck appears from a side road (but is going to stop) - Accessing the freeway	(B. Barz and Denzler, 2019; Chong and Tay, 2017; Fingscheidt et al., 2019; Hasan et al., 2016; Xu et al., 2015)
	Risky Scenario *Pattern that was observed during the training process, but still contains potential for collision*	- A car is coming towards me (potentially short time to collision) - Overtaking a cyclist	(B. Barz and Denzler, 2019; Fingscheidt et al., 2019; Xu et al., 2015)
Scene Level Non-conformity with expected patterns in a single image	**Collective Anomaly** *Multiple known objects, but in an unseen quantity*	- Demonstration, e.g., critical mass ride - Traffic jam	(Chalapathy et al., 2018)
	Contextual Anomaly *A known object, but in an unusual location*	- Tree on the street - Barrier, e.g., a fence on the street	(D. Gong et al., 2019; Kendall et al., 2015; Lakshminarayanan et al., 2017; Perera and Patel, 2019; Ruff et al., 2018; Xia et al., 2015)
Object Level Instances that have not been seen before	**Single-Point Anomaly (Novelty)** *An unknown object*	- Bear, tiger, etc. - Lost objects - Rollator	(Bendale and Boult, 2015; Creusot and Munawa, 2015; Liang et al., 2017; Lis et al., 2019; Oza and Patel, 2019; Pham et al., 2018; Yoshihashi et al., 2018; Zhai et al., 2016)
Domain Level World model fails to explain observations	**Domain level Shift** *A large, constant shift in appearance, but not in semantics*	- Weather conditions, rain, fog, snow - Traffic sign appearance - Location (Europe-USA)	(Bolte et al., 2019; Chen et al., 2018; Dai and Van Gool, 2018; Shen et al., 2017; Valada et al., 2017; Vertens et al., 2020; Zou et al., 2018)
Pixel Level (Perceived) errors in data	**Local Outlier** *One or few pixels fall outside of the expected range of measurement*	- Pixel errors (dead pixels) - Dirt on the windshield	(An et al., 2007; Buczko and Willert, 2017; Dong et al., 2019)
	Global Outlier *All or many pixels fall outside of the expected range of measurement*	- Lighting conditions - Overexposure	(Jatzkowski et al., 2018; Lee et al., 2014)

Corner detection complexity increase

Table 2 - Systematization of corner cases on different levels. The theoretical complexity of the detection typically increases from the bottom to the top (figure borrowed to (Fingscheidt et al., 2020).)

1.2.4.2.4 Comparison Literature and Concept Paper

Table 3 compares the literature and the concept paper concepts revolving around the notion of corner cases. The definitions are extracted from the literature presented in Section 1.2.4.2.2; the reader should refer to the previous sections to match them with their origin. In the following, the document uses the definitions issued from the EASA CP.

	Literature	Concept paper
Edge case	"The concept of edge cases also covers extreme cases or boundary cases. When the system designers consider edge cases, they become normal cases and are no longer considered edge cases."	This relates to a situation that, considering a given parameter of the AI/ML constituent ODD, rarely occurs (i.e., low representation of the associated value in the distribution for that parameter).
Corner case	"While many terms related to corner cases exist in the literature, a general definition and description are missing, as seen in the former paragraphs. The lack of agreed-upon technical terms and definitions makes detecting corner cases cumbersome."	This refers to a situation that occurs rarely on all of the AI/ML constituent ODD parameters, considering at least two of them (i.e., there is a low representation of the associated values in the distribution for those parameters).
Novelty	"The definition of novelties is a bit broader. Novelties are described as a spatial or temporal aggregation of anomalies or a change in the distribution of an already known process."	"Data within the ML Model ODD according to the existing ML model ODD parameters should have been considered outside the ML model ODD if it had been correctly described with the introduction of at least one new ML model ODD parameter. A novelty is generally due to a lack of characterization of the ML model ODD. It could be integrated into the ML model ODD after analysis following the upgrade policy of the ML model ODD. A novelty that is already outside the ML model ODD is, therefore, an outlier."
Outlier	"An observation deviates so much from other observations that it suggests that a different mechanism generated it."	"Data which is outside the ML model ODD."

Table 3. Comparison between the literature and the EASA concept paper on topics related to corner cases.

1.3 AI as a mathematical function

In the scope of MLEAP, we consider AI technologies based on the supervised machine learning framework. In this latter, the expected output is associated with every input element data sample. Hence, the objective is to find the function f that performs the mappings input-to-output perfectly. To define the different components of an AI system, we consider a set of data $D = \{X, Y\}$. The AI component aims to associate *"correctly"* to every data sample $x \in X$ from the input space X with a correspondent $y \in Y$ in the output space Y, using the function f. This corresponds to the ideal scenario:

$$f: X \xrightarrow{f(x)=y} Y$$

This can only be approximated in the real world using a learning algorithm. This latter aims to find the best function $\hat{f}$ (*hypothesis*) in a search space F, such that $\hat{f}$ is as close as possible to the ideal function f. This process is referred to as *training* and $\hat{f}$ is the trained model.

$$\hat{f}(x) = \hat{y}$$

The approximated function of $\hat{f}$ will gradually reduce its errors during this process These errors are the margins between the returned outputs (*predictions*) $\hat{y}$ and the expected ones (*ground truth* or *labels*) y. To do so, the set of mistakes made during the mapping process of elements of a subset $d \in D$, are aggregated in a function called "*loss function*" (or *risk function*), that can take different forms, with the same objective which is to average all the errors for the performed mappings, and can be defined as:

$$\mathcal{L} = \sum_{d} E(y, \hat{y})$$

With E is the function computing one error. This can be the difference between the prediction $\hat{y}$ and the expectation y.
To find $\hat{f}$ with a minimum $\mathcal{L}$, the dataset D can be divided into two or three parts: D_{train} for training, D_{test} for testing, and a third part D_{valid} is used for validation of the model's results during the training process (through several epochs). Where: $D_{train} \cup D_{test} \cup D_{val} = D$ and $D_{train} \cap D_{test} \cap D_{val} = \emptyset$. Hence, the loss function is in general minimised over D_{train}, compared to the loss values in D_{val}, then verified in D_{test}. The objective is to see the evolution of the learning through the error reduction, as shown in Figure 10 below:

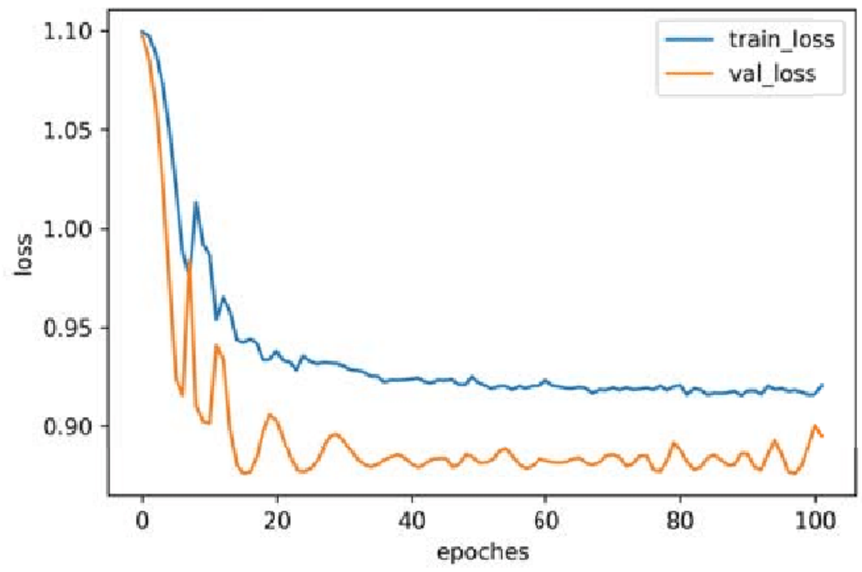


Figure 10 Loss values evolution for the train dataset and the validation dataset during the training epochs of a ML model (Mei et al., 2019)

Finally, for the purpose of data representation for the model $\hat{f}$, every element $x_i \in X$ can be represented by a weighted features (characteristics) vector $x_i = [x_i^0, \dots, x_i^n]$. Where $x_i^j \in \mathbb{R}$ if simple features are used, or $x_i^j \in \mathbb{R}^k$ if every input sample is represented by a list of k vector of characteristics. The representation space evolves depending on the application objectives in terms of input data type and the model predictions. Accordingly, the elements of the output space are also represented in a data structure that meets the target objective. For example, if the expected output corresponds to score value, it means that $Y \subset \mathbb{R}$, while for a classification objective (a specific probability is computed for every output class) the output space is $Y \subset \mathbb{R}^c$ whit c denotes the number of output classes for which the model is expected to provide a score. Hence, the predicted output will be like $\hat{y}_j = [\, \hat{y}_j^0, \dots, \hat{y}_j^c \,]$.
The rest of this document uses the same terminology and formulations. Other terms will be defined further when needed.

1.4 W-shaped process and MLEAP

The AI assurance concept[6] aims to provide a step-by-step development process for AI applications for critical and safety-related use cases. The development of this W-shaped process is part of the continuous safety assessment concept, which also aims to verify after Entry into service, in the operational phase, its ability to meet the intended objectives and how safe it is. To do so, a set of

[6] The W-shaped process, previously defined as the AI learning assurance (EASA and Daedalean, 2024), because of the extension of the scope of technologies to be tackled by the EASA AI Roadmap 2.0: https://www.easa.europa.eu/en/document-library/general-publications/easa-artificial-intelligence-roadmap-20

concepts of operations (ConOps) is provided along with the system definition during the development phase, enabling the setup of the system-level requirements. The requirements for the AI components are then defined based on the W-shaped process. From the data management to the AI constituent requirements verification, the whole process can be summarised into three primary levels, as follows:

(1) At the system level, technical requirements are allocated to different parts, including the AI constituent. The objective is to augment the existing safety assessment and development assurance guidance. This part is not explicitly included in this project's scope but can be impacted by some requirements verification that could be made on the ML component later.

(2) The AI-constituent level is fully in the scope of MLEAP. At this level, the goal is to elements to address the specificities of AI technologies, including data management, learning process management and verification, training of ML/DL models and their evaluation, and then the verification of the learned things and how the models perform in the target system.

(3) The item level is based on the existing software and hardware development assurance to ensure adequate implementation of the AI models and components.

Figure 11 highlights the different parts of the process and the position of the MLEAP along the various stages of the AI assurance, focusing on the 2nd level (AI-Constituent level).

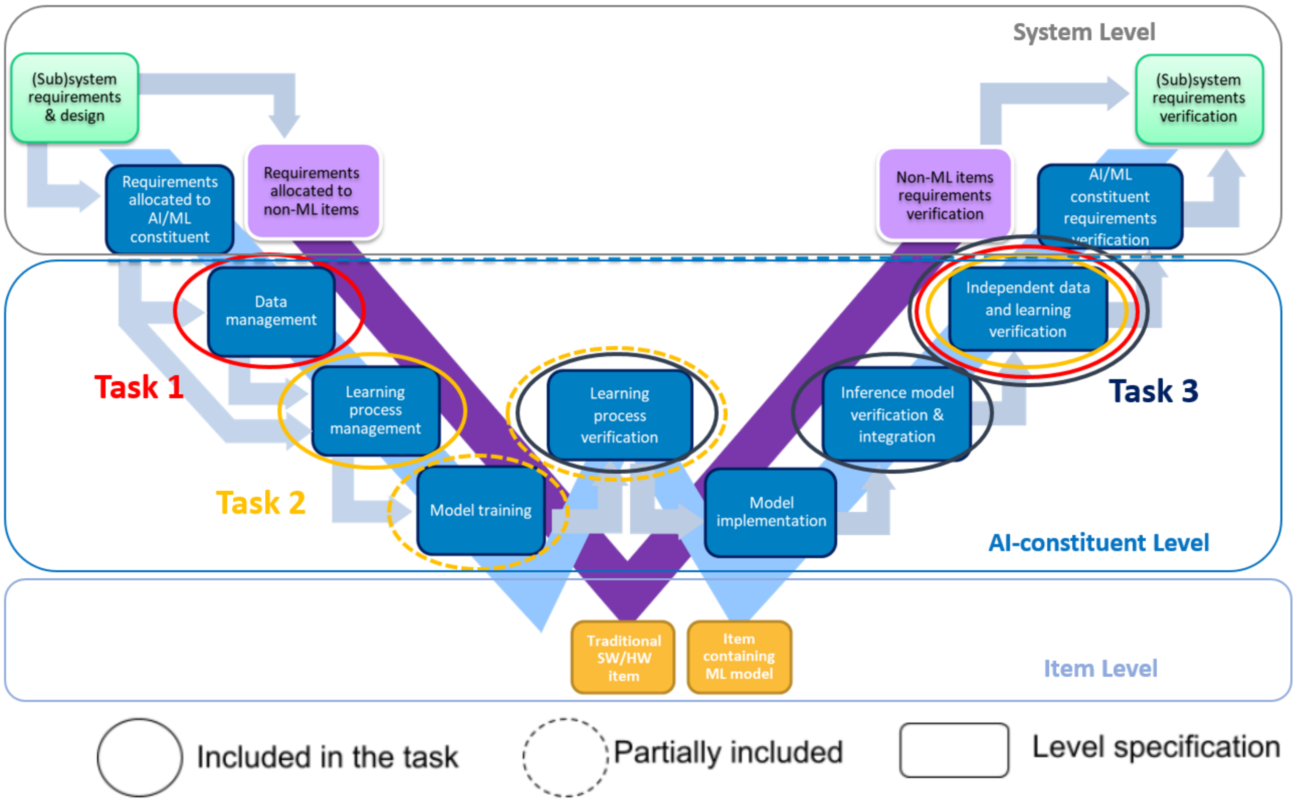

System Level
(Sub)system requirements & design
Requirements allocated to AI/ML constituent
Requirements allocated to non-ML items
Non-ML items requirements verification
AI/ML constituent requirements verification
(Sub)system requirements verification
Data management
Task 1
Learning process management
Task 2
Learning process verification
Model training
Model implementation
Inference model verification & integration
Independent data and learning verification
Task 3
AI-constituent Level
Traditional SW/HW item
Item containing ML model
Item Level
Included in the task
Partially included
Level specification

Figure 11. Global view of the learning assurance W-shaped process, overlapping the non-AI component V-cycle process and the safety assessment process. It highlights the different stages where MLEAP tasks are focused, covering completely or partially the requirements to be verified at every step.

The learning assurance concept aims to guarantee the intended function of the AI-based system at an appropriate level of performance. It also ensures that the resulting models provide sufficient guarantees of generalisation and robustness. Hence, as part of these new guidelines' definitions the MLEAP project is launched to bring recommendations and suggestions about the tools and methodologies that can be used to achieve the corresponding objectives. In Figure 11, we illustrate the position of the MLEAP project through the different steps of the W-shaped development process of learning assurance. The W-shaped cycle adapts the typical development assurance V-cycle to ML/DL. Focussing on the development of the AI components (AI-constituent level), the MLEAP project has been tailored to investigate the challenging objectives targeted by each step of this process. It aims to promote the AI blocks by carrying out the three main tasks, each of which will serve one or more parts of the whole process, as follows:

- **Task 1:** Owned by the LNE partner, this task deals with data completeness and representativeness, with the handling of the corner cases (cf. Chapter 4). This task addresses the issues related to the data management step of the W-shaped process. It is based on the ODD definition related to the targeted system, and focuses on data quality verification, setting a selection grid of methods and tools that can be used to ensure that the ML pipeline is being developed among a trustable representation of the target domain and application by a complete and representative data set.
- **Task 2:** Owned by the Airbus Protect partner, this task deals with the characteristics related to the generalisability of the developed ML/DL model (cf. Chapter 5). It addresses the issues related to several steps of the W-shaped process, where the main focus is the model learning process, management, and performance verification. All these steps are correlated and also related to the data quality. Hence, this task revisits the development model through the handling of the generalisation properties to find out how the learning process can be leveraged to promote the model's ability to maintain the performance on unseen data and how to identify common practices that are of little benefit in a critical industrial context.
- **Task 3:** Owned by the Numalis partner, this task focuses on characterising the produced model's robustness and performance stability (cf. Chapter 6). This task addresses the issues related to the evaluation, in several steps of the W-shaped process, where the main objective is to perform the model learning verification and evaluation, along with task 2, in addition to the inference model verification and integration. Since the model may be confronted with changes in the representation of input data and/or disturbances in the real world, it is necessary to check that the model is acceptably robust and that its performance is stable despite corrupted or naturally noisy data.

The distribution of the three tasks of MLEAP shows that the main parts, at the AI-constituent level, of the EASA's learning assurance are mostly covered. The remaining part is essentially the model implementation, where the trained model is converted into a form usable by the target system and requires different technologies depending on the designed system. Several validations, out of the scope of MLEAP, will be required accordingly. Finally, the post-implementation of the model includes validations and analysis from both tasks 2 and 3 since the performance that needs to be verified includes features related to the generalisation and the robustness.

1.5 EASA objectives to be addressed

This section outlines the scope of the MLEAP project. Based on the set of learning objectives and means of compliance from the latest EASA guidance (Guidance for AI applications L1 and L2), the objectives targeted by MLEAP for each technical task are highlighted, as well as how they have been tackled.

1.5.1 Targeted objectives

The EASA's call for tender to the MLEAP project (as published in the eTendering platform[7]) states the project's aim as follows: "*The subject is the approval of machine learning (ML) technology for systems intended for use in safety-related applications in all domains covered by the EASA Basic Regulation (Regulation (EU) 2018/1139). The expected short-term effect of the research results will be to streamline the certification and approval processes by identifying concrete means of compliance with the learning assurance objectives of the EASA guidance for ML applications.*"
This main objective can be broken down into two major points of interest:

- ***ML technologies to be used in safety-related applications.*** This requires a selection of relevant use cases. They should represent real-life safety-related applications where ML components are at the core of the design and have a considerable impact on the systems' behaviour.
- ***Streamline the certification objectives of the EASA guidance***. This part requires an analysis of the objectives set by the EASA AI Roadmap to approve the applicability and certification of ML systems and identify concrete means of compliance with the learning assurance objectives.

To meet the intent of (a), we have selected three use cases (cf. Chapter 3), with the objective of experimental verification and validation of the different findings of the project regarding data qualification, learning verification and model performance evaluation. A set of methods will be first selected to complete each verification and evaluation step. The following experiments on each use-case will then enable:

- Verification of the effectiveness of each approach while evaluating the models on real use cases;
- Comparative analysis of those methods with different data types, dimensions and task complexity.

Finally, conclusions will be made, enabling it to meet objective (b). This latter needs to be well defined and framed so that it is easy to achieve. On the EASA Issue 02 Concept Paper for AI applications of levels 1 (*assistance to human*) & 2 (*human-machine teaming*) (EASA, 2024), a set of anticipated means of compliance (MOCs) and guidance materials have been identified that could be used to comply with objectives of level 1 AI and level 2 AI. Several anticipated MOCs rely on the outcomes of the MLEAP project. Hence, based on those objectives[8], through the different discussions between the consortium members and EASA, a set of goals have been set for each task of the MLEAP project.

[7] https://www.easa.europa.eu/en/the-agency/procurement/calls-for-tender/easa2021hvp18
[8] For more details about the concerned objectives on the EASA document, please refer to (EASA, 2024).

Note that some of the objectives in (EASA, 2024) targeting MLEAP outcomes can be common for the different tasks. For instance, the generalisation and robustness verification on test dataset:

- DM-14: Perform a data and learning verification step to confirm that the appropriate data sets have been used for the model's training, validation, and verification and that the expected guarantees (generalisation, robustness) have been reached.

Another common objective is about verifications to be made at the implementation stage that are applicable to both robustness and generalisation:

- IMP-06: evaluate the performance of the inference model based on the test data set and document the result of the model verification.

This objective focuses more on performance verification after the model is implemented on the target system. This objective is partially covered on MLEAP since the analysis focuses on the implemented model independently from the system level, providing ways to anticipate some elements from the implementation environment that could have a significant impact on the model results after implementation on the target system (cf. Chapter 7). Provided that the tasks foreseen in the MLEAP project do not foresee activities experiment going up to the implementation of the selected use cases (cf. Chapter 3) on targeted hardware, and so the impact of MLEAP on Implementation objectives (IMP) is expected to be limited.

In the followings, an overview of the agreed goals for each task is provided, covering most of the stages of the learning assurance and verification.

1.5.1.1 **Task #1 Objectives: Data completeness and representativeness**

Task 1 aims to explore methods for the assessment of data completeness and representativeness. This involves specifying a set of characteristics related to the definition of the operational domain of the target application, as well as the restrictions and specificities related to the ML module in particular. Hence, the ML/DL design requires a specific definition of the ODD, specifying as many known inputs and data states as possible, to enable a coherent model training and evaluation in the subsequent steps. The selected "*data management*" (DM) objectives set for the data verification, based on the EASA's concept paper, are:

- DA-03: define the set of parameters pertaining to the AI/ML constituent operational design domain (ODD);
- DA-04: capture the DQRs for all data pertaining to the data management process, including but not limited to:
  - the data needed to support the intended use;
  - the ability to determine the origin of the data;
  - the requirements related to the annotation process;
  - the format, accuracy and resolution of the data;
  - the traceability of the data from their origin to their final operation through the whole pipeline of operations;

    - the mechanisms ensuring that the data will not be corrupted while stored or processed,
    - the completeness and representativeness of the data sets; and
    - the level of independence between the training, validation and test data sets.

- DM-07: ensure verification of the data, as appropriate, all along the data management process so that the data management requirements, including the data quality requirements (DQRs) are addressed.

1.5.1.2 **Task #2 Objectives: Guarantees on the model generalisation**

In this part, the selected objectives focus mainly on the learning management (LM) stages of the W-shaped process. Being highly related to the data quality and volumetry, the performances of a trained model can be evaluated based on several aspects of the target application, including the expected outputs and how the model can generalise its performance to new inputs. In order to correctly assess the model's ability to generalise, EASA recommends specifying the ODD, focusing on the capture of specific operational limitations and assumptions, which will allow to better define the training objective, as well as having a clearer view of the functional expectations for the AI module. This can be partly ensured through the verifications to be made at the data management level (cf. section 1.5.1.1), in addition to those at the learning management level.
To ensure good generalisability, several objectives are formulated in the EASA document, including:

- LM-03: document the credit sought from the training environment and qualify the environment accordingly;
- LM-04: provide quantifiable generalisation guarantees. These guarantees may then be used to support the Safety Case in Objective SA-01;
- LM-07: account for the bias-variance trade-off in the model family selection and should provide evidence of the reproducibility of the training process;
- LM-09: perform an evaluation of the performance of the trained model based on the test data set and document the result of the model verification;
- LM-10: perform a requirements-based verification of the trained model behaviour and document the coverage of the AI/ML constituent requirements by verification methods;
- LM-14: verify the anticipated generalisation bounds using the test data set.

In addition to those LM objectives, several other verifications need to be performed to enable a convincing evaluation pipeline of the models under development. In particular, models should be designed in line with the intended function. Besides, several elements should be avoided when they could prevent the produced models from being deployed, either due to an intrinsic lack of performance or a drop in performance after its implementation in the target system. To do so, in the scope of MLEAP, we consider the following points during the experimentation:

- Analysis of existing ML/DL development pipelines to highlight the main limitations;
- Awareness of the applicability conditions regarding the selected generalisation bounds to provide guarantees on the models' performances;

- Setting of risk mitigation plan concerning the possibility of guaranteeing the performances of produced models;
- Identification, through the development pipeline, of the learning verification sections where generalisation can be redefined, either by reworking evaluation objectives (metrics and KPIs) or by anticipating impact factors that can boost or undermine performance.

Finally, while training a model, the training, testing and validation data sets are assumed to verify the IID hypothesis[9], whereas in reality, this hypothesis cannot constantly, or even often, be verified. Hence, given the generalisation definition based on the test data set, *how can this be adapted to the reality of real applications?* To answer this question, the representativeness and completeness of the training and test data sets need to be correlated with a well-defined ODD. These aspects are addressed in chapter 4.

1.5.1.3 Task #3 Objectives: Robustness and Stability

Task 3 is articulated around the robustness and stability of an AI component. Following the content of the concept paper by EASA (EASA, 2024) these characteristics drive several requirements. In particular, they are identified in:

- LM-02: the applicant should capture requirements related to model robustness and stability metrics and acceptable levels;
- LM-11: The applicant should provide an analysis on the stability of the learning algorithms
- LM-12: The applicant should perform and document the verification of the stability of the trained model.
- LM-13: The applicant should perform and document the verification of the robustness of the trained model in adverse conditions.

In the objectives of the MLEAP project, Task 3 focuses on analysing how robustness and stability can be put into perspective with the existing requirements or guidelines from other normative sources. To do so, three types of methods are investigated: *statistical, formal* and *empirical*, exploring their applicability on concrete use cases, both in the design phase and during the operational phase.

Task 3 will, in particular, research what kind of evaluation techniques are available to respond to the objective of:

- Studying the stability of the training algorithm
- Studying the stability of the trained system
- Studying the ability to define and test the system against corner and edge cases

Ultimately, task 3's output will be viewed in the context of a larger evaluation process that encompasses both task 1 and task 2.

[9] IID hypothesis stands for independent and identically distributed random variables. Identically distributed variables show no overall trends, their distributions do not fluctuate and all items in the sample are taken from the same probability distribution. Variables are independent if the sample items are all independent events. In other words, they are not connected to each other in any way, and knowledge of the value of any variable gives no information about the value of the others.

1.5.2 Scientific and experimental protocol

In this section, we summarise the activities carried out, in progress and yet to come, to achieve the above-mentioned final objectives, at the level of the various tasks. The protocol and approach followed through collaborative work between the different tasks can be summarised as shown in Figure 12. This includes three main steps: (1) analysis of needs, through project objectives and selected use cases investigation, and study of state of the art then issues identification; (2) practical issues identification, based on a toy use cases exploration and setting of hypothesis about the applicability and behaviour of the method, this will help the definition of a generic development pipeline for AI applications ; (3) experimental validation, including comparative experimentations of the toy use cases and selected real use-cases, to finally provide a set of recommendations of best practices, thus completing the generic pipeline, and providing materials to better perform the recommended steps of the W-shaped process.
The three steps (1), (2), and (3) will support a continuous and iterative construction of a complete AI development methodology for industrial applications, supporting the EASA recommendations, and completing objectives of several stages of the W-shaped process for data management, models training, and learning verification.

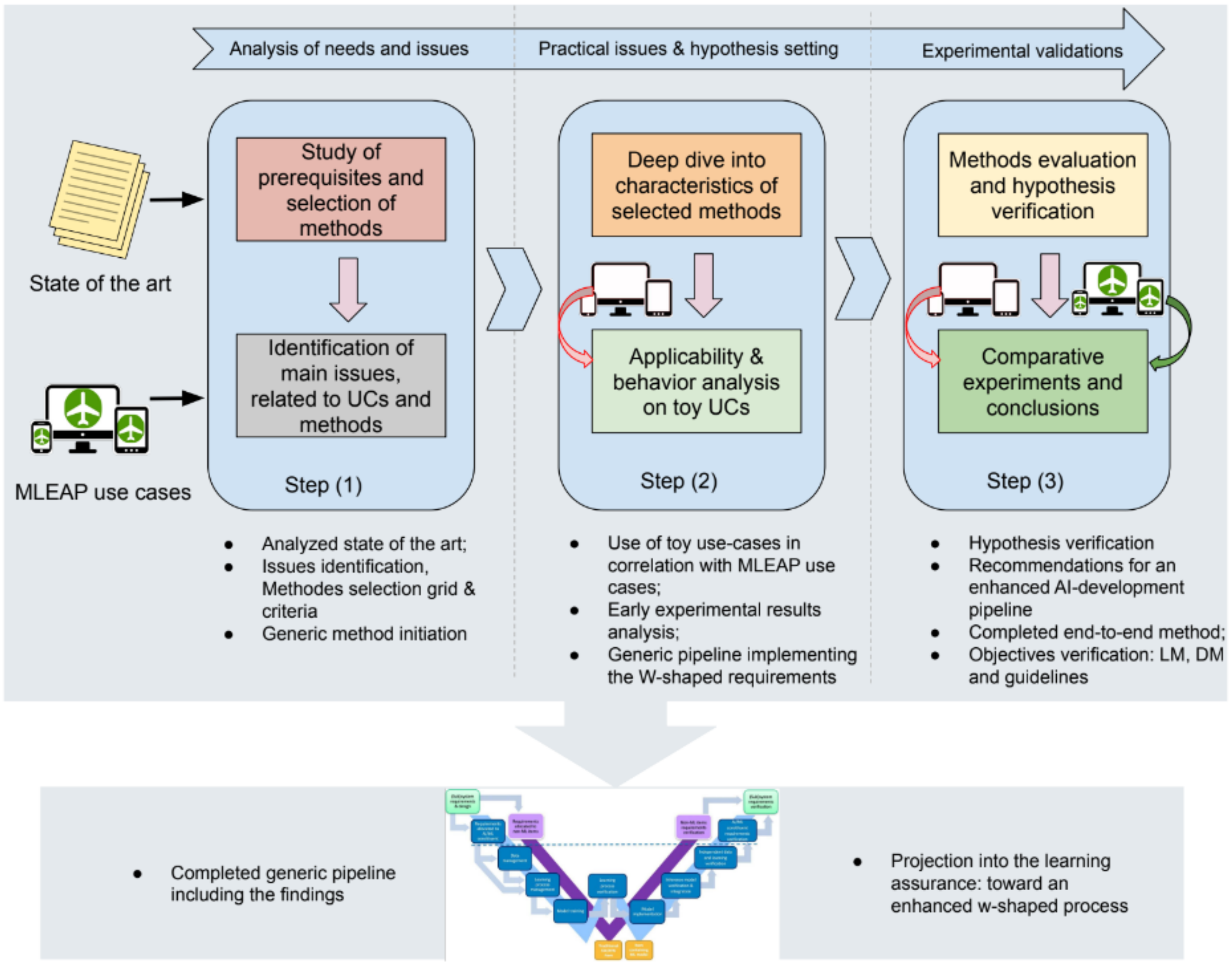

Analysis of needs and issues
Practical issues & hypothesis setting
Experimental validations
State of the art
MLEAP use cases
Study of prerequisites and selection of methods
Identification of main issues, related to UCs and methods
Step (1)
Deep dive into characteristics of selected methods
Applicability & behavior analysis on toy UCs
Step (2)
Methods evaluation and hypothesis verification
Comparative experiments and conclusions
Step (3)
Analyzed state of the art;
Issues identification, Methodes selection grid & criteria
Generic method initiation
Use of toy use-cases in correlation with MLEAP use cases;
Early experimental results analysis;
Generic pipeline implementing the W-shaped requirements
Hypothesis verification
Recommendations for an enhanced AI-development pipeline
Completed end-to-end method;
Objectives verification: LM, DM and guidelines
Completed generic pipeline including the findings
Projection into the learning assurance: toward an enhanced w-shaped process

Figure 12 Summary of the MLEAP project activities, highlighting the main objectives of each development step and the expected outcomes that will be fed into the W-shaped learning assurance

1.6 Document structure

Following this introduction, which defines the scope of the studies and the main project objectives, the next chapter presents the main evaluation set-up in terms of performance evaluation metrics and data quality and volume assessment. In Chapter 3, the three aviation use cases that are used to illustrate the analysis corresponding to the three different tasks of MLEAP (cf. section 1.5.1). These use cases are used to highlight the selected methods during the previous phases of the project and to support the findings. Then, the three following chapters (4, 5, and 6) describe the work done so far, including the state-of-the-art analysis, the selection of the method, the experimental analysis, corresponding to data qualification, model training and evaluation aspects, as well as the different conclusions made in each task. The final chapter 7 includes the generic pipeline development and implementation of the W-shaped learning assurance. This pipeline is designed to fulfil the main objective of task 2, which is the development of a generic method for generalisation evaluation. It is then extended to the findings of tasks 1 and 3 to put together the project findings and main recommendations for the development of AI applications.

Finally, the global conclusions based on the different experimental parts and results of the evaluated methods for each use case and similar applications are all synthesised in the final section of this document, summarizing the main findings and a set of perspectives and open questions to be investigated in further activities.

2. Evaluation metrics description

2.1 Introduction

As for the terminology definition, in this section, we present a set of evaluation metrics that could be used to assess models' performance in terms of generalisation and robustness during and after training. These metrics are statistical tools to analyse the model predictions, compared to ground truth. Note that the metrics that are dedicated to performance evaluation are defined following a supervised ML development workflow.
In the following sections, we first define the main elements used to compute the different metrics. These will then be presented in two main classes of evaluation measures. Then, the data evaluation, in terms of quality and volume required for training, will be presented overall.

2.2 Performances' evaluation metrics

Machine learning models are evaluated using metrics to analyse their performance on top of the loss function which is used to optimise the model during the training phase. Hence, different metrics can be used to evaluate both the generalisation performances to unseen data during training and the performance stability and robustness toward data and/or environment changes. The outputs of the metrics can reveal different aspects, depending on the data being used: the generalisability is assessed based on unseen data samples during training, and the robustness can be assessed based on corrupted data samples. Both generalisation and robustness assessment can be performed using the same metrics that can be divided into two main problem categories: *classification* and *regression*.

2.2.1 Regression

Regression methods deal with predicting a target value using independent variables. Either based on deep or shallow neural networks or classical ML algorithms, metrics grouped herein are based on point distance computation methods. These metrics contain fundamental operations and the absolute or squared value of their result can be used to provide performance values. Let $\hat{y}$ be the predicted (computed) output value (scalar or vector) by the model f, and y is the expected output (truth). The classical error value E is computed by $(y - \hat{y})$, for every instance in the data set of n training samples. As mentioned before, both robustness and generalisation performance indicators could be assessed using the same metrics. Table 4 shows an overview of the most popular evaluation metrics based on E, for different ML applications. For these metrics, the generalisation refers to the ability of the model to produce closer values to the target ones ensuring a right ranking of results (e.g., in question answering, the rank of the right answer is highly dependent on the score value computed by the trained model. Hence, the model generalising better is the one that returns a good answer in the good ranking). The robustness in these cases aims to ensure the correct sorting of results regardless of noise and/or corruption in the input data.

Metric	Formula	Description
Mean Error (ME)	$ME = \frac{\sum_{i=1}^{n} E_i}{n}$	It is the average of the simple amount of differences between a distribution and its true values. It is easy to apply and works with numeric data.
Max Error	$Max(\lvert E_i \rvert, i = 1 \dots n)$	Max Error is either an absolute or a relative metric calculating the difference between a value in the input data and the corresponding value in the prediction from the AI system. The absolute Max Error is the maximal signed difference between a value in the input data and the corresponding value in the prediction from the AI system. The relative Max Error is the percentage of the width of the variation domain on which the AI system operates.
Mean Absolute Error (MAE)	$MAE = \frac{\sum_{i=1}^{n} \lvert E_i \rvert}{n}$	It measures the difference between two continuous variables. Uses a similar scale to input data and can be used to compare a series of different scales too.
Relative Absolute Error (RAE)	$RAE = \sum_{i=1}^{n} \frac{\lvert E_i \rvert}{\lvert \hat{y}_i - \hat{y}_{mean} \rvert}$ with $\hat{y}_{mean}$ is the average value of the true labels.	Based on errors produced by a trivial model and works with numeric data. Need to handle carefully, since divisions by zero may occur (if true labels contain zeros).
Mean Relative Absolute Error (MRAE)	$MRAE = \frac{1}{n} \sum_{i=1}^{n} \frac{\lvert E_i \rvert}{\lvert \hat{y}_i - \hat{y}_{mean} \rvert}$	Based on absolute errors, it is more sensitive to outliers (especially of low values). Need to handle carefully, since divisions by zero may occur (if true labels contain zeros).
Mean Squared Error (MSE)	$MSE = \frac{\sum_{i=1}^{n} E_i^2}{n}$	Both MSE and RMSE are scale dependent. Models whose values are closer to zero present an adequate state. They are highly dependent on fraction of data that is used (low reliability).
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{\frac{\sum_{i=1}^{n} E_i^2}{n}}$	
Geometric Root Mean Squared Error (GRMSE)	$GRMSE = \sqrt[2n]{\prod_{i=1}^{n} E_i^2}$	GRMSE is also scale dependent. However, differently than MSE and RMSE, it is less sensitive to outliners.
Correlation coefficient (R)	$R = \frac{\sum_{i=1}^{n} (\hat{y}_i - \hat{y}_{mean})(y_i - y_{mean})}{\sqrt{\sum_{i=1}^{n} (\hat{y}_i - \hat{y}_{mean})^2 \sum_{i=1}^{n} (y_i - y_{mean})^2}}$	R measures the strength of association between variables. Such that values higher than 0.8 implies stronger correlations. As for R^2, it is related to R, as it is the squared one, its values that are close to 1 indicate stronger correlations too. Both R^2 and R work with numeric data.

Coefficient of Determination (R^2)	$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - \hat{y}_{mean})^2}$	
Scatter index (SI)	$SI = \frac{\sqrt{\frac{\sum_{i=1}^{n}\left(y_{\max(y)} - \hat{y}_{\max(\hat{y})}\right)^2}{n}}}{\hat{y}_{\max(\hat{y})}}$	Applied to examine whether RMSE is good or not (Mentaschi et al., 2013), if its value is less than 1, then estimations are acceptable. It shows "excellent performance" when SI < 0.1 and "poor performance" when SI > 0.3
Performance index (PI)	$PI = \frac{\sqrt{\frac{1}{n}\Sigma(\hat{y} - y)^2}}{1 + R}$	It is an indicator for the evaluation of predictivity of a model. Lower PI values result in more accurate model predictions (Rifai, 2021).

Table 4 List of commonly used performance evaluation measures for ML regression models.

Most ML/DL applications continue to incorporate traditional metrics such as R, R^2, MAE, and RMSE as primary indicators of adequacy of the regression-based ML models. This seems to stem from our familiarity with these metrics, as opposed to others, such as Golbraikh and Tropsha's (Golbraikh et al., 2003) criterion, QSAR model by Roy and Roy (Roy and Roy, 2008), Frank and Todeschini (Frank and Todeschini, 1994), and specifically designed objective functions, often used in other fields. The works of Gandomi et al. (Gandomi et al., 2010), Golafshani and Behnood (Golafshani and Behnood, 2018), as well as Cheng et al. (Cheng et al., 2014), applied a multi-criteria verification process that incorporated the use of traditional as well as modern measures. Utilising a multi-criteria process is beneficial to ensure the validity of a particular ML model, but it is also recommended to overcome some of the identified limitations of traditional metrics. For example, the Actual/Predicted correlation, corresponding to the linear correlation (in the statistical sense) between the actual values and the predicted values for every value considered in a set, must be used with another metric, RMSE, MAE or Max Error.

2.2.2 Classification

Classification applications aim to categorize data into distinct classes. This is a supervised learning approach where machines learn to classify observations into binary or multi-classes.
A set of data samples can have the following characteristics:

- *Total population*: the total number of samples in the data (input data with corresponding class);
- *Condition positive*: the number of real positive cases in the data;
- *Condition negative*: the number of real negative cases in the data;
- *Prediction positive*: the number of samples classified as positive;
- *Prediction negative*: the number of samples classified as negative;
- *Prevalence*: the proportion of a particular class in the total number of samples.

For simpler evaluation purpose, each instance in the set of samples, being classified by the system, is evaluated in one of the following ways:

- *True positive* (hit): instance belongs to the class and is predicted as belonging to the class;

- *True negative* (correct rejection): instance does not belong to the class and is predicted as not belonging to the class;
- *False positive* (false alarm, Type I error): instance does not belong to the class and is predicted as belonging to the class;
- *False negative* (miss, Type II error): instance belongs to the class and is predicted as not belonging to the class.

Based on this class evaluation, a confusion matrix (shown in Figure 13) allows a detailed analysis of the performance of a classifier and is helpful to circumvent or uncover the weaknesses of individual metrics as it achieves a more rigorous and well-rounded analysis of classifier performance (this can be computed for every candidate class, if not binary classification). By contrast, using a single metric to express classifier performance is not informative enough to conduct this analysis, as it does not indicate which classes are best recognised or the type of errors committed by the classifier.
The confusion matrix is a square matrix where entry *e*, at every row and column, are the number of instances belonging to the ¼ class or category that are labelled by the classifier as having the ¼ Class. Confusion matrices include counts of true positives, true negatives, false positives, and false negatives. Metrics such as accuracy, per-class recall, and per-class precision can be calculated from these. Further metrics, such as the entropy of the histogram represented by the matrix, can be derived from confusion matrix elements.

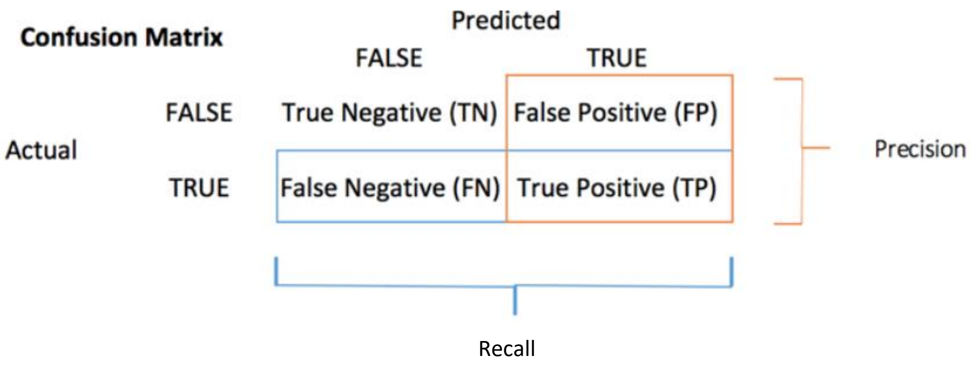


Figure 13. Construction of a confusion matrix[10] for a binary (false, true) classification.

The confusion matrix sets the foundations necessary to understand performance measurements for a specific classifier, for which the columns of the matrix correspond to the predicted class labels, while the rows correspond to the actual ones. The best classifier is the one having the maximum diagonal values. Based on this concept, several performance evaluation metrics have been defined to assess a classifier's ability to put every input instance in the corresponding target class. Hence, classification metrics are based on the number of instances that are correctly classified as positive or negative. In Table 5, most of the measures described work with categorical data. Furthermore, the generalisability and robustness of the model can be evaluated using the same metrics, where the well-generalised model is the one able to predict the right classes for instances unseen during training, while a robust

[10] https://www.guru99.com/confusion-matrix-machine-learning-example.html

model is the one that keeps the right classification scheme regardless of a corrupted or noisy data samples.

Metric	Formula	Description
True Positive Rate (TPR): ***Recall***	$TPR = \frac{TP}{TP + FN}$	Measures the proportion of actual positives that are correctly identified as positives. It does not account for indeterminate results.
True Negative Rate (TNR): ***Selectivity***	$TNR = \frac{TN}{TN + FP}$	Measures the proportion of actual negatives that are correctly identified negatives.
Positive Predictive Value (PPV): ***Precision***	$PPV = \frac{TP}{TP + FP}$	The proportions of positive observations that are true positives. The best model is the one with a precision value closer to 1 and the worst one has a value of *zero*.
Negative Predictive Value (NPV)	$NPV = \frac{TN}{TN + FN}$	The proportions of negative observations that are true positives. This has an ideal value of 1 and the worst value of *zero*.
Positive likelihood ratio (LR⁺)	$LR^{+} = \frac{\frac{TP}{TP + FP}}{\frac{FP}{FP + TN}}$	The LR⁺ evaluates the change in the odds of having a diagnosis with a positive test, it presents the likelihood ratio for increasing certainty about a positive diagnosis. While the LR^{-} focuses on the negative tests instead.
Negative likelihood ratio (LR⁻)	$LR^{-} = \frac{\frac{FN}{(TP + FN)}}{\frac{TN}{(TN + FP)}}$	
Accuracy (ACC)	$Acc = \frac{TP + TN}{P + N}$	Evaluates the ratio of number of correct predictions to the total number of samples (*P+N*).
Fβ scores	$F\beta = 1 + \beta 2 \times \frac{precision \times recall}{\beta 2 \times \text{precision} + \text{recall}}$	Computes a weighted harmonic mean of Precision and Recall, where: - F1 (β=1): Balances the weight on precision and recall. - F2 (β=2): Puts less weight on precision, and more weight on recall.
Matthews Correlation Coefficient (MCC)	$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)+(TP+FN)+(TN+FP)+(TN+PN)}}$	Measures the quality of binary & multi-class classifications analysis. It can be used in classes with different sizes. Its best values are closer to 1, worst ones are closer to -1, and when $MCC \cong 0$, the model tends to make random predictions instead.
Average precision (AP)	$AP = \sum_r (recall_r - recall_{r-1}) \times precision_r$ with r is the rank position of the result in the list.	Combines recall and precision for ranking. It describes the weighted mean of precision in each threshold with the increase in recall from the previous threshold used.

Area under the ROC curve (AUC)	$AUC = \sum_{i=1}^{n-1} \frac{1}{2}(FP_{i+1} - FP_i)(TP_{i+1} - TP_i)$	The ROC (Receiver Operating Characteristic) plots the diagnostic ability of a binary classifier as its discrimination threshold is varied. The AUC measures the two-dimensional area underneath the entire ROC curve. Works with categorical data.
Log Loss Error (LLE)	$LLE = -\sum_{i=1}^{M} \mathrm{c_i} \log\left(\hat{y}_{c_i}\right) + (1 - \mathrm{c_i})\log(1 - \hat{y})$ with M is the number of classes, $\mathrm{c_i} \in \{0,1\}$ if the current class is the correct one, and $\hat{y}_{c_i}$ is the output model probability corresponding the current class.	Measures the uncertainty of the probabilities by comparing predictions to the true labels. Values are between 0 and 1, and it penalizes for being too confident in wrong prediction. Where the perfect model has $LLE = 0$.
Hinge Loss Error (HLE)	$HLE = \max(0, 1 - \alpha y)$ Where $\alpha = \pm$ and y is the computed output probability.	It incorporates a margin or distance from the classification boundary into the cost calculation. Linearly penalises incorrect predictions, even if new observations are classified correctly, they can incur a penalty if the margin from the decision boundary is not large enough.
Lift	$lift = \frac{\frac{TP}{TP + FP}}{\frac{TP + FN}{TP + TN + FP + FN}}$	Measures the performance of a model at predicting or classifying cases. Formally, it is the ratio of correctly predicted positive examples and the actual positive examples in the test data set.
Mean Cross Entropy (MXE)	$MXE = -\frac{1}{n}\sum_{i=1}^{M} y_{\mathrm{c_i}} \ln(\hat{y}_{c_i}) + (1 - y_{\mathrm{c_i}})\ln(1 - \hat{y}_{c_i})$ Where $y_{\mathrm{c_i}}$ and $\hat{y}_{c_i}$ are respectively the expected and predicted probabilities corresponding to the current class label $\mathrm{c_i}$.	Measures the performance of a model whose output is a probability between 0 and 1. The objective, in general, is to minimize this value to improve the likelihood.

Table 5. List of commonly used performance evaluation measures for ML classifiers.

The reader can refer to ISO/IEC 24029-1 for more details on some metrics.

2.3 Data volume and quality assessment

2.3.1 Data quality

The quality of data is "*its ability to satisfy the requirements for its safe application in the end system*" (Cluzeau et al., 2020). The paper defines a set of classical measures, based on DO-200 revision B

"Standards for processing aeronautical data. The set of measures is provided with discussions highlighting the required modifications and adaptations that are needed to be applied to them in order to cope with ML/DL development and release. These requirements are:

- *Data accuracy.* The ability to provide the model with *correct* pairs (x, y) during training. Hence, statistical tests should be performed to guarantee the absence (or scarcity) of biases and statistical errors (e.g., zero mean). In addition, the following errors need to be minimised:
    - *Capture error is a noisy, biased, or distorted data collection caused* by a human factor or bad machine conditions.
    - *Single-source errors:* relying on a single source of data could introduce biases, which would lead the model to learn patterns related to the source of the data as well as the data itself.
    - *Labelling errors:* These are related to the automatic labelling of data; hence, a human verification is needed to mitigate this risk, or double (multiple) pair labelling to avoid biases and errors when the data is initially labelled by experts.
    - Data *inaccuracy* is exemplified as *Data Quality* errors at the instance level of a database in some cases. Examples of these errors are missing data, incorrect data, misspellings, ambiguous data, outdated temporal data, "*mis-fielded*" values and incorrect references (Laranjeiro et al., 2015). Hence, one of the central pre-processing steps is improving the accuracy of data by trying to predict and fill missing values in data sets (Ma et al., 2007).
- *Entity Resolution: this* is about recognising when two observations relate semantically to the same entity, despite [possibly] having been described differently. Conversely, recognizing when two observations do not relate to the same entity, despite having been described similarly.
- *Assurance level:* As the samples may be modified during data transformation and cleaning, confidence that the data will not be corrupted during storage or transport is required.
- *Traceability:* stands for the ability to determine the origin of each data item (recording) and that can be required. When inputs and outputs are both recorded from the same source, it is needed to pay attention to subsequent changes (e.g., in case of evolutionary data applications). When the data pairs are taken from different sources (x and y come from different sources), we need to take care of the matchings.
- *Timeliness:* Confidence that the data is applicable to the period of intended use.
- *Completeness:* As part of the MLEAP project, the data completeness is widely addressed in section 4. Data completeness is one of the data quality dimensions that were found to be the most significant by a previous research study (Gupta et al., 2021; Wang et al., 2006). The *incompleteness* of data is expressed in different ways, such as missing values, absent values and sparse-ness of values. Several automatic and ML-based methods for data completeness detection and improvement can be found (Juddoo and George, 2020).

  Furthermore, recent studies (Hagendorff, 2021) on the impact of data quality on resulting ML/DL applications have defined the concept of *ethical data quality*. It defines how data quality is affected by certain personality traits or modes of behaviour of individuals (e.g., in human-based data collection and labelling), and how those traits or states can be assessed from an ethical point of view. Eventually, finding quality data should not primarily serve the pursuit of improved marketability but of socially acceptable, beneficial machine learning applications. Another study by Chen et al. (H. Chen et al., 2021) defined a set of *"fit for purposes"* data quality criteria that qualify the data in a more complete manner in order to know if the data used fit the objectives of the application and the target use of the model. These qualities are *comprehensiveness* which means a data set contains all representative samples from the population (e.g., there are data samples for learning all the aspects of the target domain and associating the reality with the training data labels); *correctness* refers to the fact that a record in a data set is accurate and valid, and they are correctly labelled if they are labelled records. Inaccurate or invalid data leads to data noises, and incorrectly labelled data leads to label noises; variety represents diversity to ensure examples of variability which can be coupled to the data completeness. Hence, these qualities need to be assessed in the data preparation step since they affect a good or poor model performance.

2.3.2 Data volume

As for data quality, the amount of data that should be used to train an ML/DL model effectively needs to be defined. Here the main question to be answered is *"Do I have enough data to train the model effectively?"* (it is more about preventing overfitting a small data set by a high parameterized model), w.r.t the targeted application, and the model to be trained. Hence, as for data quality assessment, the volume of data needed to train a model can be evaluated and determined based on a set of empirical and theoretical measurements. The amount of data required for training an ML problem mainly depends on the complexities of two main elements: the problem (task) to be solved and the ML/DL *model* that will solve it. As described in the previous sections (cf. Section 5.4.1) model complexity has an important impact, as well as the complexity (cf. Section 5.4.2) in the model's generalisation estimation. Hence, we can choose one (or more) methods from the existing two classes, with respect to the target application objectives:

i) *Theoretical (heuristic-based) data bounds:*
   - *Factor of the number of classes:* It is desirable to achieve the order of tens, hundreds or thousands independent examples of each class. For example, on a binary classification, we may need a set of 20, 200 or 2000 data samples per class, depending on the model complexity and target performances.
   - *Factor of the number of input features:* The desirable features matrix should present a hundred percent (100%) more rows than columns (have more examples than a number of features describing each example). There must be x% more examples than there are input features (Jain and Chandrasekaran, 1982). For example, to train a model on a dataset where each sample is comprised of 80 features and define a heuristic of 20% more data (120% of the number of features 80 is needed for training), it is recommended to have at least 96 samples.
   - *Factor of the number of model parameters:* There must be *N-independent* examples for each parameter in the model. For example, in a classification using a linear regression and a heuristic of *N=10*, the expected input is at least 20 independent examples.

ii) *Empirical bounds for data size:* To determine the data size needed, depending on the constraints to achieve the targeted performances, several ways have been provided by (Balduzzi et al., 2021) to tie the empirical generalisation assessment and the minimal size of data needed. For instance:

- *Function of the VC-dimension* (Juba and Le, 2019): if d is the probability of failure and ε is the learning error, the amount N of data needed for learning depends on the complexity of the model
- $N = F\left(\frac{VC+\ln\left(\frac{1}{d}\right)}{\varepsilon}\right)$

  A side effect of this is the well-known voracity of neural networks for training data, given their significant complexity.

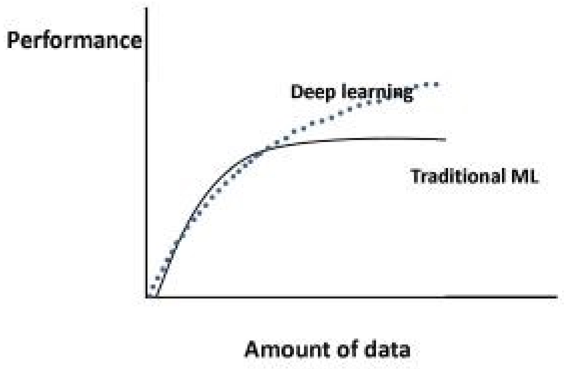


Figure 14. Difference between ML and DL w.r.t the impact of the amount of data on the performance evolution during training (Zhu et al., 2016).

- *Observing the learning curves* (Yelle, 1979): rests on a set of plots of error versus different training data sizes. An example formula to plot the learning curve is (Cho et al., 2015): $y = 100 + b_1 x^{b_2}$, where y is the classification accuracy, x is the training set, and b_1 and b_2 correspond to the learning rate and decay rate, respectively. Figure 14. Difference between ML and DL w.r.t the impact of the amount of data on the performance evolution during training (Zhu et al., 2016). shows how the performance of machine learning algorithms changes with increasing data size in the case of traditional machine learning (Zhu et al., 2016) algorithms (e.g. regression) and in the case of deep learning (Hassaballah and Hosny, 2019). Specifically, for traditional machine learning algorithms, performance grows according to a power law and then reaches and settles on a plateau value. Regarding deep learning, there is significant ongoing research as to how performance scales with increasing data size (Shahinfar et al., 2020). In this study, an empirical evaluation is provided along with an approximation formula to estimate how many images per animal species are needed for a certain accuracy level a priori. It is based on observations of the *learning curves* to show that the common behaviour is that the performance keeps increasing with data size according to a power law.

2.4 Conclusion

In this chapter, the main objective was to present the set of evaluation metrics that can be used in the following chapters for assessing the model's performances during and after training, as well as the data quality and volume. Note that the evaluation is a main part of the ML/DL development process; this helps the verification of the objectives meeting and that the model does what it is meant for. As for the data evaluation, in this chapter, we widely pointed out the main criterion that could be essential to well train and evaluate an ML/DL model. In Chapter 4, more analysis about data completeness and representativeness can be found. Besides, the generalisation and robustness of the designed models are addressed, respectively, in Chapter 5 and Chapter 6. Finally, the different metrics described in this chapter can be used in Chapter 7 to complete the different evaluation steps of the models to be developed in the complete ML/DL development pipeline.

3. Use cases description

3.1 Introduction

In this section, we provide an overall description of the different use cases chosen for the evaluation part of the MLEAP project. Note that the use cases have been selected to support our findings about the different methods to be used for data and model performance assessment. These use cases will be used to support the proof of concept associated with the recommendation of evaluation methods and approaches to be followed, in applications of similar fields and of similar complexity. They have been selected based on their relevance to the MLEAP project research directions, ensuring domain, data type, and complexity diversity. The intention is not to certify them, but to show real application examples of the different methods. These use cases may be used as sandboxes for testing some of the methodologies identified in the following chapters. Table 6 provides an overview of the main technical details, in terms of domain definition, intended tasks, models architecture, data, and system description, corresponding to three different applications: speech-to-text for air-traffic control (ATC-STT), collision avoidance (ACAS Xu), and automatic visual inspection (AVI). Note that all images, results, models, and illustrative examples that do not belong to the consortium are provided with references to proprietary public sources. Confidential details have been omitted and replaced by open-source examples (models and data sets) to allow the reproducibility of some results by an audience of this deliverable. Lastly, there is no conflict of interest with the proposed use cases. These have not been submitted to EASA for approval by any of the consortium companies prior to the MLEAP project

3.2 Overview of the UCs

With respect to the recommendations in the concept paper by EASA (EASA, 2024), the ConOps definition corresponding to the system is expected. In this section, we consider the definition of several aspects related to every AI-based application for the different use cases:

- **ML Target tasks:** Refers to the intended behaviour that the system is made to perform. In machine learning (ML), almost all automated tasks can be designed as a function.
- **System description:** This represents a set of functional and technical characteristics that could help us better understand the structure of the designed system.
- **ML component description:** This part will describe the ML component more precisely (the model and its in/outputs).
- **Data type:** this could be even text, signal, time series, or scalar data … this part gives precisions about the data type involved in the use case.
- **Data management:** finally, this will provide some details about how the data is fed to the system to perform the intended task.

	ATC-STT	ACAS Xu	AVI
High-level description of the ODD	To correctly detect the spoken instructions, the acoustic and language models should be trained on complete data sets. Full data completeness is defined by several speech-related features targeted for intended system performance: noise types, airports with specific checkpoint names, accents, and speech rates.	The data[11] represents input points of the lookup tables from the RTCA SC-147 resources[12]. This latter provides the Minimum Operational Performance Standards (MOPS) for ACAS-Xu. The ODD is divided into sub-ODDs to fit 45 ML model elements of the ML model architecture.	The ODD is defined by the set of all pictures of airframe structures in acceptable lighting & blur conditions. There may be indoors or outdoors pictures. If it is outdoors the weather conditions can influence the lighting & blur state.
ML Target tasks	Extract useful information for the pilot from ATC exchanges	Each NN shall replicate the LUT prediction in its allocated ODD (LUT property) 100% accuracy in the allocated ODD for each ML model	Automatic damage detection based on high-resolution pictures. The target accuracy for the ML item is the same as for a human inspection
Model architecture	Different architectures: LSTM, CNN, and TDNN implemented within an open-source toolkit called KALDI The one used for experiments is a TDNN-HMM	L-Lipschitz Neural Networks	CNN & Incremental learning
Data Dimensionality	High Dimension	Low dimension	High Dimension
Model Complexity	The models are made of different deep neural network architectures and shallow classical ML (KALDI models). Hence, methods based on NN architecture for identifying the model complexity (cf. section 5.4.1) apply.	The models are made of simple (shallow) neural network architectures. Hence, methods based on NN architecture for identifying the model complexity apply to this.	The models are made of deep CNN architectures dealing with high-quality images. Methods based on NN architecture for identifying the model complexity apply for this.

[11] https://hal.archives-ouvertes.fr/hal-03355299/document

[12] https://my.rtca.org/productdetails?id=a1B1R00000LoYKtUAN

	ATC-STT	ACAS Xu	AVI
System description	- The model is made of 4 main components: - **Feature extractor:** Mel filter-bank coefficients (MFCC) are used as input features along with i-vectors - **Acoustic model:** 2 models have been compared, a Hidden Markov Model (HMM), and a TDNN/LSTM for linking phonemes and audio signals. - **Language model:** gives the probability distribution over a sequence of words. - **Decoder:** beam-pruned Viterbi search (LOWERRE, 1976) that conducts a best-path approximation inside the decoding graph. - A final SoftMax layer discriminates, according to the input, what the heard phoneme is.	1) **Identification**—All input situations in which the NN and the LUT predictions are different are considered incorrect (the NN does not preserve the LUT property). 2) **Mitigation**—This part is already addressed per the subsystem architecture design: the ACAS-Xu hybrid ML-based controller switches from the ML model (NNs) to the LUTs (Safety Net) when incorrect situations are detected (this is already described in the ACAS-Xu subsystem architecture document).	The inference model is a pipeline made of: . The damage detector . The automated generation of windows . The classifier The damage detector can be seen as a data pre-processing stage before the classifier which is the AI/ML constituent.
ML component description	Audio input as a sequence of 20ms frame TDNN layers scans its input according to a splicing parameter; The final SoftMax layer discriminates, according to the input, what is the heard phoneme	the ML model is composed of several elements more precisely, 45 NNs	Transfer learning using the already trained network from the ImageNet project (e.g., VGG16); Completion of the transfer learning network with a couple of layers with a couple of hidden layers; The final layer is a cross-entropy layer; Adam optimiser is used for the training
Data type	Annotated corpus with different accents (French and Chinese)	LUTs binary data	Any type of image standard shall be addressed (.jpg, .bmp, .gif, …)
Data management	- The data corresponds ATC communications, recorded for normal situations (no abnormal or emergency communications have been recorded) - The annotated corpus of recorded real-life speech contains 114.8h, along with corresponding text manually transcripted - Audio input as a sequence of 20ms frame - Not subject to distribution or	The full raw LUTs are used	**Acquisition of pictures:** from cameras downloaded to design/deployment environment Labelling is done using VOTT tool; Every image can contain several damages of different classes A typical example using VOTT: The image is not modified, and the labelling is supported by a metadata

	ATC-STT	ACAS Xu	AVI
	sharing for non-involved third parties		file presented as a json file - Not subject to distribution or sharing for non-involved third parties
Performances and safety requirements derived from design & safety[13] processes	**System requirements—Complex background noise. The PESQ evaluation score represents operational conditions. The** PESQ returns[14] a score from 1 to 4.5, with higher scores indicating better quality. 3.8 is the acceptable score for the telephone voice. **System requirements – High speech rate** The speech rate in ATC is higher than that of speech in daily life since ATC requires high timeliness **System requirements – Accents** The system must operate with French and Chinese accent	**System requirements – real-time 1** The controller must execute with a period of 1s. **System requirements – anti-collision performance 1** Any implementation must behave similarly as the reference architecture **System requirements – ODD 1** The controller must operate on the ranges of the LUTs, i.e. $$ODD_{\text{ACAS Xu}} = \begin{cases} \tau \in [0, 101] & \rho \in [499, 185318] \\ \theta \in [-3.14, +3.14] & \psi \in [3.14, +3.14] \\ v_{int} \in [0, 1200] & v_{own} \in [50, 1200] \end{cases}$$	**ML-based requirements: The first targeted performance is to focus on true positives ~ with 80% accuracy. Detection accuracy needs to be tested across different conditions, including** variations in surface structure, camera position and viewing angle, and object obstruction. **- System requirements:** Solutions need to accommodate both indoor and outdoor environments, including night time and changing weather conditions, while extracting unnecessary information from the background of the aircraft. In addition to the ability to detect both identified types of damage (lightning strikes and dent impacts).

Table 6. Overview of the main technical information corresponding to the different use cases chosen for the MLEAP project evaluations. For more details, refer to the corresponding section for every use case description.

3.3 Speech-To-Text for Air Traffic Control

Automatic Speech Recognition (ASR), also known as Speech-to-Text (STT), is the task of transcribing a given audio signal to a corresponding text. ASR means a machine will automatically recognise what a

[13] Corresponds to the requirements that are applicable at AI/ML model level and that have been postulated based on engineering judgement, assuming that the model is part of a system and that at system level, the necessary architecture mitigation is in place to ensure that applicable safety objectives are met.
[14] Based on the state-of-the-art of ATC-STT (Lin et al., 2019)

person says, and the STT refers to the fact that it can convert speech into text (Mittal and Sharma, 2023). It has many applications, such as voice user interfaces (Fernandes et al., 2019). Several recently trained models and benchmarks are publicly available for testing use[15]. In this chapter, we consider models and approaches developed in the scope of air traffic control (ATC). First, we provide a review of some background information about the ATC-STT application, then discuss the related challenges to this use case.

3.3.1 Background and ATC-STT Description

In ATC, the ASR-STT application is more than important in handling several daily challenges and problems (Lin, 2021). Provided that the spoken instruction is transmitted in an analogue manner and can be easily impacted by environmental factors (such as communication conditions, equipment error, radio interference, etc.), it contains a wealth of contextual, situational dynamics that indicate the evolutions of the flight and traffic in the near future. These information pieces are highly significant to air traffic operations and to the communication errors that may cause potential safety risks. One of the well-known applications is spoken instruction understanding (SIU) (Lin, 2021), during radio transmissions, where the spoken instructions represent a basic information source. In order to support the ATC communications and provide reliable warnings, before the pilot performs the incorrect instruction, a SIU system is used. To do so, a wide range of research work is done to come up with better ASR models. A wide empirical study of different ASR solutions is made by (Nassif et al., 2019). The main components that constitute a generic ASR application are highlighted in Figure 15. This latter shows the general framework for speech recognition and textual transcription, where the main modules that constitute an ASR application can be defined[16] as follows:

- **Speech signal acquisition:** as a first step, the speech data is first recorded using a module that receives the speech signal (e.g., a microphone) and stores it in utterance[17] files;
- **Feature extraction:** refers to the mapping of the input acoustic signal to a vector representation, using a specific encoder (model). Speech is a 1D signal which is converted into 2D signal and chopped into 20-25ms sized frames with 10ms shift. In ASR, feature extraction is a process of converting those speech frames into feature vectors (Ashok Kumar et al., 2021). The purpose of this step is mainly to identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other ones carrying useless things: background noise, emotion...;
- **Acoustic model (AM):** responsible for the accuracy of the ASR system, it maps the acoustic feature to corresponding basic linguistic units called *Phonemes* corresponding to the hypothesis sentence. A recent review of the most used models for feature extraction can be found in (Malik and Khanam, 2022);
- **Lexical model (LxM):** gives the general structure of the language elements and attribute-value structures in a formal lexicon. It specifies the types of lexical objects and the structure of lexical

[15] Check ASR models by https://huggingface.co/tasks/automatic-speech-recognition
[16] A set of basic ASR definitions are provided in http://www.iitk.ac.in/LDP/HOWTO/Speech-Recognition-HOWTO/introduction.html
[17] An utterance is the vocalization (speaking) of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.

entries, the information associated with them, and the relations between lexical objects.

- **Language model (LM):** gives the probability distribution over a sequence of generated words (Shinde and Shah, 2018).

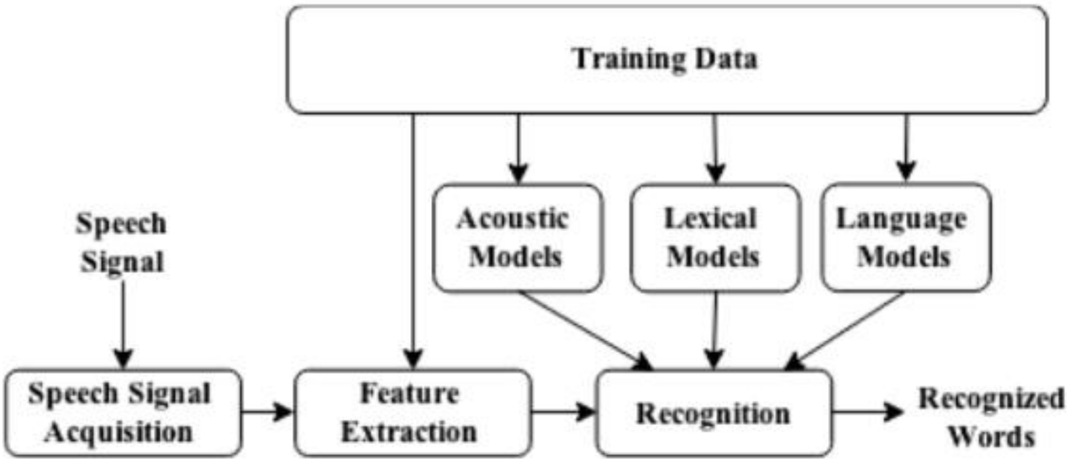


Figure 15. A domain-independent pipeline for ASR applications (Nguyen and Holone, 2015)

In order to develop an ASR system, one needs to deal with several challenges related to the input data samples. For instance, when speech has two or more languages within an utterance, it becomes more difficult to understand and correctly interpret. Known as code-switching (CS) (Mustafa et al., 2022), this challenge in spoken languages needs to be addressed in ASR systems to recognise speech in bilingual and multilingual settings, where the accuracy of the system declines with CS due to pronunciation variation. The diagram in Figure 16 shows the different research directions and taxonomy of models dedicated to different ASR applications in ATC. The main tasks that can be addressed in this use-case are three folds:

- *Language Understanding (LU)*: (Raju et al., 2021) where the systems extract both text transcripts and semantics associated with intents and slots from input speech utterances.
- *Spoken Instruction Understanding (SIU)*: (Lin, 2021) where the objective is to correctly interpret the instructions communicated between the control tower and the pilots.
- *VoicePrint Recognition (VPR)*: (Saquib et al., 2011), also known as a Speaker Recognition System (SRS), objective is the validation of a user's claimed identity using characteristics extracted from their voices.

In the scope of the MLEAP project, the ATC-STT use case is promoted by the work done in an Airbus project called BoostHLT[18]. This latter includes several topics related to ATC speech recognition, including the study of the *"callsign extraction"* (Gupta et al., 2019) challenge, which represents only one issue in ATC-STT objectives and challenges. Other problems, such as *Speech to intent extraction* (Ohneiser et al., 2014) and *Command Extraction* (Helmke et al., 2020) can be addressed when the aim is to correctly recognise all the communicated information pieces, such as those describing

[18] Similar application designed by Airbus, for a knowledge extraction from aeronautical messages (Notice to Air Missions - NOTAMs) with self-supervised language models for aircraft pilots, can be found in (Arnold et al., 2022).

movements and manoeuvre instructions received from the AT controllers (ATCO). A more extensive taxonomy of different research objectives and designs on the topic of ATC-STT is provided in (Lin, 2021), showing different directions to develop and/or enhance STT systems designed for ATC objectives, and where the vocabulary and language elements are domain-specific.

3.3.2 Speech Data

In the scope of the MLEAP project, we are provided with several datasets of discussions (manually annotated) for the evaluation of the different methods. The speech data are made of ATC interactions, in English, and contain:

- 100h discussion, using a French accent;
- 50h discussion, using a Chinese accent;

For more flexibility in the development and evaluation process corresponding to different tasks of the MLEAP project, open data samples with different accents are being collected. One of the recent works by (Lin et al., 2021a) has also provided an analysis of different speech data using several accents. The objective is to enable the public sharing of experimental results within the MLEAP's consortium and the EASA's publishable documents. Table 7 provides a set of open-source datasets for the ATC-STT use case implementation:

Data set	**Link**	**Number of samples (utterances)**	**Whole Duration**	**Spoken Accent**
ATCO2 - ASR	https://www.atco2.org/data	560	1h 6 min	Yes: Czech, Slovak, German, French, Australian
UWB	https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0001-CCA1-0	2657	20h 35 min	Yes: Czech
NIST LDC - Air Traffic Control Complete	https://catalog.ldc.upenn.edu/LDC94S14A	1	2h 02 min	No: US
ATCOSIM	https://www.spsc.tugraz.at/databases-and-tools/atcosim-air-traffic-control-simulation-speech-corpus.html	10078	10h 42 min	Yes: German, French

Table 7. Open-source ATC-STT data sets with provided text transcriptions for training and testing.

3.3.3 STT Model

As mentioned before, we rely on provided models that are already developed and trained in the scope of the BoostHLT project. The development and evaluation of the models have been performed using an open-source library called KALDI[19] (Povey et al., 2011). Several automatic speech recognition

[19] https://kaldi-asr.org/

engines have been built and tested, where the acoustic model is made of a Hidden Markov Model (HMM)[20]. The global architecture, presented in Figure 16, follows three main steps, as follows:

- Mel filter-bank and Cepstral coefficients (MFCC)[21] as input features along with i-vectors representations;
- KALDI with HMM and TDNN[22]/LSTM[23]-HMM chain system used to build the acoustic model;
- The SRILM[24] toolkit is used to build the 5-gram language model;

More precisely, the language model is built using a 5-gram model using Kneser-Ney smoothing to assign a probability for each word sequence (Chen and Goodman, 1999). The phonetic dictionary comes from the Carnegie Mellon University Pronouncing Dictionary[25] (CMU). In the context of ATC communication, some specific pre-processing is adopted to deal with named entities (typically city names) when they do not appear in the CMU dictionary and automatically output the pronunciation of out-of-vocabulary words.
Before decoding, knowledge learned in AM and LM is combined in a Weighted Finite-State Transducer (WFST) (Mohri et al., 2002) based graph. Decoding involves applying a beam-pruned Viterbi search (Lowerre, 1976) that conducts a best-path approximation inside the decoding graph.

The acoustic model is an HMM monophonic model, where each hidden state represents a monophonic. Alignments between the audio and the reference transcript are computed and used to train the HMM further. The TDNN/LSTM-HMM system adopts the same general architecture used in traditional speech recognition engines (cf. Figure 16) using hybrid DNN-HMM acoustic modelling. This deep network consists of a particular combination of TDNN[26] and LSTM layers, called TDNN/LSTM, and is made of the following bricks as described above:

1. ***Speech signal acquisition:*** aims to collect the data for training while the system is running;
2. ***Pre-processing and Feature extraction:*** after splitting the speech into a 2D signal and chopping into 20-25ms sized frames with 10ms shift, it is encoded into vectors using MFCC input features extraction;
3. ***Acoustic model:*** to map the acoustic feature to corresponding *Phonemes* an HMM and TDNN/LSTM-HMM chain system is used;
4. ***Lexical model***: to structure the language elements lexical objects a phonetic CMU dictionary is used;
5. ***Language model***: a 5-gram model using Kneser-Ney smoothing is used.

[20] Hidden Markov Model is a statistical model based on the Markov chain, for analyzing probabilities of sequences of random variables (Elliott et al., 1995).
[21] Represents a set of features that correspond to the useful data signals http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
[22] Time-Delay Neural Network (Waibel et al., 1989) is a NN that introduces a time-delay arrangement to take into account temporal relationship between its inputs in the same unit.
[23] https://intellipaat.com/blog/what-is-lstm/#no1
[24] http://www.speech.sri.com/projects/srilm/
[25] http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[26] https://kaleidoescape.github.io/tdnn/

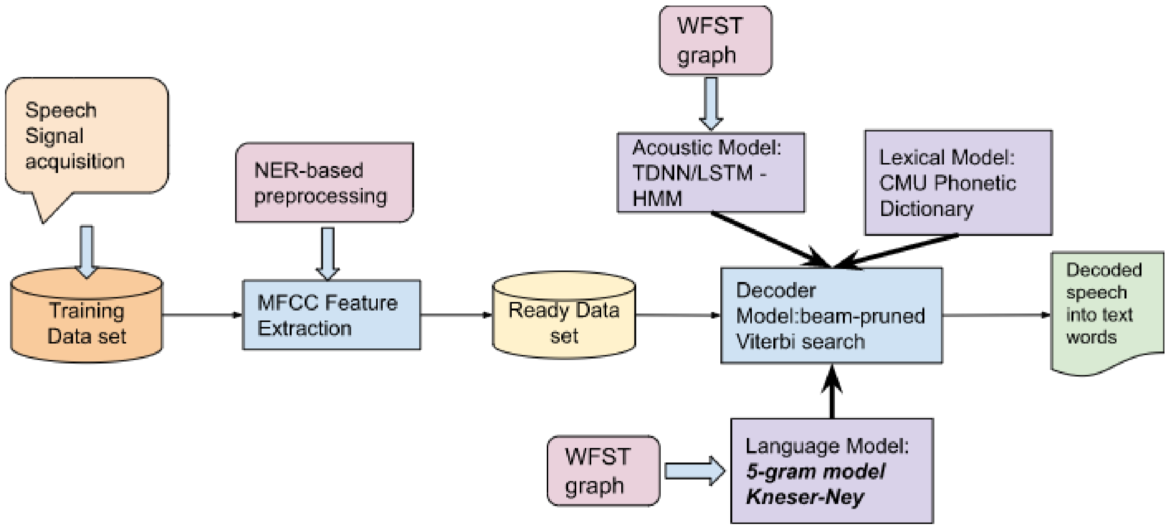

WFST graph
Speech Signal acquisition
NER-based preprocessing
Acoustic Model: TDNN/LSTM - HMM
Lexical Model: CMU Phonetic Dictionary
Training Data set
MFCC Feature Extraction
Ready Data set
Decoder Model:beam-pruned Viterbi search
Decoded speech into text words
WFST graph
Language Model: 5-gram model Kneser-Ney

Figure 16. Abstraction[27] of the ATC-STT system developed in the BoostHLT project.

Given the different technical requirements related to the ASR applications development, in general, and the ATC-STT application specifically, this use-case is considered to be of a high dimension in terms of data volume that will be processed and the complexity of the models to be evaluated. Hence, this use case is very useful for the evaluation of the methods and tools to be built within the scope of the MLEAP project, and that will be meant for approval of machine learning applicability. The ATC-STT use case aims to correctly transcribe the spoken instructions during the interactions between the control tower and the crew.

3.3.4 Challenges and Issues

In the scope of the MLEAP project, the research work concerned by the ATC-STT use-case is the evaluation of different methods and tools designed for speech recognition involving two main directions:

- *Production mechanisms* (Sohai, 2022): where the background noise and speech rate represent important issues for the acoustic models.
- *Language and accent handling* (Lin et al., 2021b): where several accents, French and Chinese among others, of spoken English need to be handled.

Hence, one of the targeted objectives of the evaluation is based on the PESQ Algorithm. This latter is designed to predict subjective opinion scores of a degraded audio sample, then returns a score from (-0.5) or 1 to 4.5, with higher scores indicating better quality of the audio (Lin et al., 2019). PESQ is also designed to analyse specific parameters of audio, including time warping, variable delays,

[27] To protect Airbus Intellectual properties, we designed this figure to give a schematic description of the system. Note that this is not the same figure provided by Airbus teams, it's not the same diagram, and nevertheless, it follows the same process.

transcoding, and noise. Therefore, this measure represents a relevant tool to evaluate the ATC-STT performances.

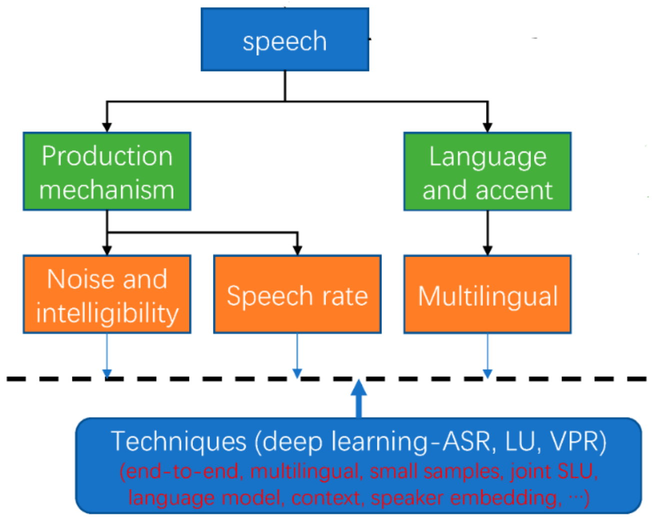

speech
Production mechanism
Language and accent
Noise and intelligibility
Speech rate
Multilingual
Techniques (deep learning-ASR, LU, VPR)
(end-to-end, multilingual, small samples, joint SLU, language model, context, speaker embedding, …)

Figure 17. The ASR research design concerned by the MLEAP project, as part of a larger taxonomy provided in (Lin, 2021)

As for any ASR application, the main challenges to be addressed in the ATC-STT use case are related to the generalisation to unseen data (e.g., the same speech can be said by different voices), how to achieve high recognition accuracy for new utterances, and the robustness of the trained models to several impacting elements (e.g., variable accent, speech rate, different pronunciations …). In order to develop a performant and trustable STT system, the model needs to deal with the following, but not only, issues:

- Speech is highly variable (different accents and speech rates) analysing records of different speaking people,
- Data sparseness problem (unbalanced in terms of voice gender, and accent),
- Adaptable methods to train and adapt acoustic models using limited/unbalanced data,
- The intended information to be recognised (all the speech, specific intents, …)
- The choice of adequate LM w.r.t the acoustic model. The phoneme's recognition must be followed by a performant language model that would construct coherent phrases (this is a more critical issue in the context of ATC applications since the phrases are a kind of semi-controlled language)
- General ML/DL issues in evaluation and implementation, including:
    - Misunderstanding and choice of the generalisation bounds, the training objectives, and data outliers,
    - Inappropriate regularisation and data representation models,
    - High/low model complexity compared to the recognition task,
    - Missing alignment between the industrial KPIs and evaluation measures in general ASR development models.

In (Lin, 2021), these issues are divided into ten (10) main challenges, in the scope of ATC application objectives:

1. *Data Collection and Annotation.* Costly and laborious work in terms of correctly annotating. It is easier to be collected than efficiently annotated.

2. *Volatile Background Noise and Inferior Intelligibility.*
3. *Unstable Speech Rate.* The rate of ATC speech is generally higher than that of daily life. This is due to traffic situations, the spoken language and emotion.
4. *Multilingual and Accented Speech.* Sometimes, non-English companies communicate with ATCOs in their local language, including English, which involves handling multilingual content and dealing with several English accents.
5. *Code Switching.* This refers to the published standard pronunciation of the vocabulary words by the International Civil Aviation Organisation (ICAO), which aims to switch the homophones and near-syllable words compared to the pronunciations recommended by the ATC departments of the different countries.
6. *Vocabulary Imbalance.* Refers to the OOV words w.r.t the published standards by ICAO and which precise the recommended vocabulary to be used in ATC communications. Since in practice, ATCOs tend to use different words, these latter need to be handled carefully.
7. *Generalisation of Unseen Samples.* Including several variable parameters in ASR such as the distribution of speech features, the vocabulary of the different ATC centres and/or locations and the different previous variation factors are key elements to be taken into account.
8. *Ambiguous Word Meaning.* Same words have different meanings and uses. This includes the distributions or the contextual correlations between digits and other words that are extremely similar. The tagging which is used in language units (LU) recognition is ambiguous (e.g., pronounced digits will be tagged as “digit” or something else such as flight number and else?).
9. *Role Recognition.* The speaker's role is important for better analysing the recognised text. Indeed, the resource conflict check is designed for ATCO speech, while the repetition check is for pilot’s speech. The ICAO requested that the ATCO instruction start with a valid aircraft identification (ACID) to specify the communication object, while the pilot instruction must end with their ACID during the repetition. However, in practice, some pilots ignore the ATC rules, whose instruction even starts with an ACID.
10. *Contextual Information.* In the spoken instruction understanding (SIU). The acoustic model recognizes different phonemes, and the LM is applied to correct the results based on semantic meanings. Hence, the standardized phraseology plays a significant role in improving the LM effectiveness. These computational parts are important to handle the contextual situational information provided from other information sources (such as radar, flight plan, etc.). This latter provides a more accurate and targeted reference for the ASR research.

In the Boost-HLT project, the speech rate is not evaluated, and the vocabulary has not been balanced. Some Chinese terms are simply omitted. The vocabulary is not standardised either. Since phraseology is different from one airport to another, these processing steps are crucial. Besides, the accent is a part of the speech that can be taken into account to help an ASR system to transcribe a given speech. Indeed, French people won’t speak most of the words the same as if they were English. There are several words that are different in the pronunciations of words included in the ATC vocabulary, according to the speaker's native language. Hence, it is preferable to indicate to the ASR engine the correct transcription of the pronounced word in a given accent. The ASR engine will then consider that the audio linked to the transcription is spoken with the corresponding accent and that the pronunciation is not systematically common to all accents (French, English, or Chinese). Based on these hypotheses, the ML/DL models are meant to use the accent annotation of the speech-ATC corpus, especially by using it with a frame-level accent embedding and in multi-task model training.

Finally, the callsign extraction from the transcribed text is one of the most important issues in STT-ATC. It is a combination of identifying codes, letters and numbers assigned to a flight, hence its correct extraction is highly important. Several solutions to perform information extraction within the text have been compared in the BoostHLT project, and could be evaluated in the scope of the MLEAP project.

3.4 Automated Visual Inspection (AVI)

The materials used in the AVI evaluation are provided by an internal Airbus project, that aims to assist inspectors in the inspection of an aeroplane surface and reduce the duration. In the scope of the MLEAP project, we use the provided models and data for evaluation purposes of selected methods, for all tasks of the project. First, we provide an overview of the AVI applications, and then focus on the challenges related to this use case in the context of MLEAP.

3.4.1 Description of the AVI application

Visual inspection is nowadays widely used in different industries including aircraft maintenance (Duvar et al., 2021). It is one of the most commonly used approaches in the production process, but not only. It entails visually inspecting the components of an assembly line to detect and repair problems. A wide analysis of AVI-related applications for the 21st century is provided and discussed by (See et al., 2017), showing the main role of AVI and how it enhanced several social, environmental, organisational and industrial tasks. However, AI-based visual inspection is frequently described as some form of optical inspection technique based on deep learning and computer vision. It is the process of monitoring and inspecting a manufacturing or service operation to ensure that products meet predetermined specifications, but not only.

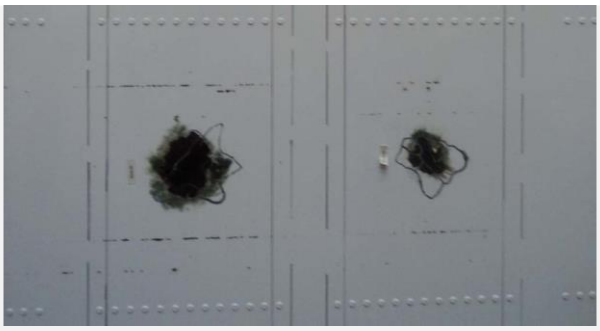
Aircraft inspection is crucial for safe flight operations and is predominantly performed by human operators, who are sometimes unreliable since they can be subjective and prone to error. Thus, a recent analysis by (Aust and Pons, 2022) compares the performance of human operators with image processing, artificial intelligence software and 3D scanning for different types of inspection. Additionally, other factors relevant to operations were assessed using the weighted factor analysis. The results show that operators' performance in screen-based inspection tasks was superior to inspection software due to their strong cognitive abilities, decision-making capabilities, versatility and adaptability to changing conditions.

Another work by (Kähler et al., 2022) has analysed the ability of such applications to detect surface defects on aircraft landing gear components. This task represents a deviation from a normal state. Visual inspection is a safety-critical, but recurring task with automation aspiration through machine vision. Various rare faults make the acquisition of appropriate training data difficult, which represents a major challenge for artificial intelligence-based optical inspection. The authors applied an anomaly detection approach based on a convolutional autoencoder for defect detection during inspection to address the challenge of lacking and biased training data. Results indicated the potential of this approach to assist the inspector, but improvements are required for deployment.

The main goal is to assist inspectors in reducing the inspection duration, which is in line with the common use of these applications in the state-of-the-art (DING et al., 2022). Specifically, the aim of the system under development by Airbus is in-service damage detection with the following technical objectives:

- Diagnostic assistance for inspectors to reduce the aircraft maintenance duration, for scheduled and unscheduled events;
- Automatic external damage detection and classification into two types: *lighting strike* impacts and *dents*;
- Detection of dents and lightning strikes using a combination of pictures and videos;
- The major point is to find acceptable metrics to bring computer vision closer to classical problems such as model development for surface damage detection.

In the MLEAP project, the targeted domain is in-service damage detection with two main kinds of damages addressed: lighting strike impacts and dents.


Lightning Strike impacts[28]

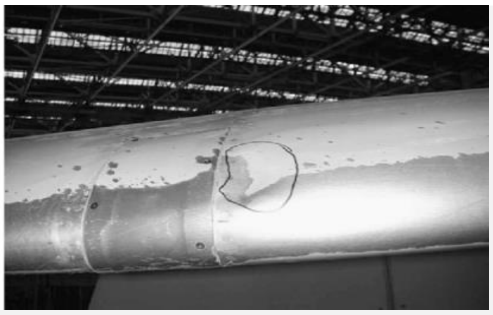

Dents Damages[29]

Figure 18. Images showing example damages on an airplane skin.

3.4.2 Provided Data Sets

The Operational Design Domain is defined by the set of all pictures of airframe structures in acceptable lighting and blur conditions. The data includes indoor and outdoor pictures. For outdoors the weather conditions will influence the lighting & blur state of the captured images. For the experimentation of the project, the provided data was not tagged with indoor/outdoor information.

Initial input Data details:

- The data set is made of two main parts, lightning strikes and dent impacts, with data augmentation (Changyu et al., 2014);
- Acquisition of pictures is done from cameras and downloaded to the design/deployment environment;
- Labelling is done using the VOTT tool[30], where every image can contain several damages of different classes;
- Weighting samples to cope with imbalanced data sets where some classes are overrepresented compared to others. Hence, it is necessary to weigh the images with the reciprocal frequency in the training data set to balance the training and avoid overfitting for

[28]https://www.researchgate.net/figure/Structural-damage-in-the-outer-skin-in-the-Airbus-A400-M-airplane-after-the-lightning_fig8_305817924
[29] https://www.researchgate.net/figure/Wing-skin-metal-dent-examples_fig3_331961295
[30] https://sourceforge.net/projects/vott.mirror/

some classes.

- Dataset statistics
  - *Lightning strike*: 28 images of 440 lightning strikes are provided for training. The impacts vary in size and shape and were photographed at different distances. It includes one class label detection *smoky* which represents the area of the lightning strike itself.
  - *Dents*: 3,659 images containing 1,225 dents are provided for training. These images are taken at different distances and include two classes: dent_al, which refers to impacts that are visible through ambient light, and dent_lb, which refers to those that can be visible only using an LED lamp.
- Example images from the datasets can be found in the annexe.

3.4.3 Model and Approach Description

The model selected was built using a transfer learning approach. The architecture is based either on VGG16[31] (Simonyan et al., 2014) pre-trained networks on ImageNet[32], or on YOLOv5[33]. Both architectures have been trained and compared. The methods using yolo were selected for the experiment and this is what we are using in the evaluation of the methods for the MLEAP project.
Specifically, a Siamese[34] network (Y. Li et al., 2022) is constructed for a multitasking framework, of which the aim is to detect both the damage type (dent impact or lightning strike) and its characterisation (corresponding severity level), to further determine its reparability. To do so, the following main steps are implemented:

- Using openCV[35] library for image stain detection, the YOLOv5 model[36] has been trained, to perform:
  - Stain detection using a binary classification: *smoky* area or not ;
  - Detection of the type of damage (multi-class damages classification): *dent_al* and *dent_lb*
- Completion of the transfer learning network with some layers: the model is based on a pre-trained YOLOv5. This latter is typically pre-trained on large-scale datasets like COCO[37] (Common Objects in Context) or ImageNet[38]. During pre-training, the network learns to detect various objects or features in images by optimising its parameters using the annotated data.

[31] More technical details can be found in this blog: https://www.mygreatlearning.com/blog/introduction-to-vgg16/
[32] https://www.image-net.org/
[33] https://github.com/ultralytics/yolov5
[34] A Siamese network is a type of neural network architecture consisting of two identical subnetworks connected by shared weights. It is commonly used for tasks involving similarity comparison, such as image recognition, text similarity, and signature verification. Siamese networks learn to extract meaningful features from input pairs and measure their similarity or dissimilarity.
[35] https://opencv.org/
[36] YOLOv5 was introduced by Glenn Jocher (Jocher et al., 2020), it significantly reduced the model size (compared to its previous version YOLO-v4 that had 244MB size whereas YOLO-v5 smallest model is 27MB), YOLO-v5 showing a higher accuracy and more frames per second than all previous versions (Fang et al., 2021).
[37] https://cocodataset.org/#home
[38] https://www.image-net.org/

This pre-training process initialises the model with knowledge about common visual patterns, enabling it to generalise better when fine-tuned on specific tasks or datasets, such as the AVI use case.

- The final layer of the model is a cross-entropy layer, and the Adam optimiser is used for the training.
- First targeted performance: >95% accuracy and recall (for the purpose of this study)

3.4.4 Challenges

This use case is considered as high dimension. It will be the opportunity to test different evaluation techniques concerning data completeness, representativeness and relevance. E.g. quality of inspection input images, data augmentation and training data set size (Goodfellow et al., 2016; Keskar et al., 2016; LeCun et al., 2012; Masters and Luschi, 2018). Besides, the diversity of input data and considered tasks is important for the assessment of the applicability of the models in terms of generalisation ability and robustness characteristics. Hence, in the scope of the AVI use case, several issues are addressed and can be considered as part of the MLEAP project experimentations, including:

- Evaluation of load and stress elements, which is part of the engineering of an aircraft development process, based on the severity level detection for the different considered damages. This challenge is not explicitly addressed; however, it contributes to the AVI target application objectives achievements ;
- The AVI use case, as part of an Airbus internal project, does not come with a final model. The experiments are still in progress, and several models are being compared in order to find the best one for damage detection and classification ;
- Automatic visual inspection-related challenges need to be handled, too, independently of the object being inspected (e.g., aeroplanes in this project). The paragraph below highlights some of the most encountered issues with AVI applications in aeronautics. While empirical research on the automated inspection of aircraft seems to be limited (Rice et al., 2018), there are clearly other technical challenges in the deployment of real-world vision systems, including:
    - ➢ AVI systems can provide accurate and timely results that are at least as good as traditional techniques. Without this, it is difficult to justify the implementation and maintenance costs of using the technology.
    - ➢ An AVI system needs to be robust enough to detect a wide range of surface defects. Technically, this is challenging given the varying physical characteristics of surface defects on an aircraft. In addition to the variability of the paints and structure finishing that are used (e.g., different material, paint colours, brightness …)
    - ➢ A usable visual inspection system needs to be scalable and deployed across different aircraft types and sizes, which will vary depending on the engines used, winglets, body shape, etc.
    - ➢ Detection accuracy needs to be tested across different conditions, such as variations in surface structure, camera position and viewing angle, and object obstruction to determine their effectiveness.
    - ➢ Solutions need to accommodate both indoor and outdoor environments, including night-time and changing weather conditions, while extracting unnecessary information from the background of the aircraft.

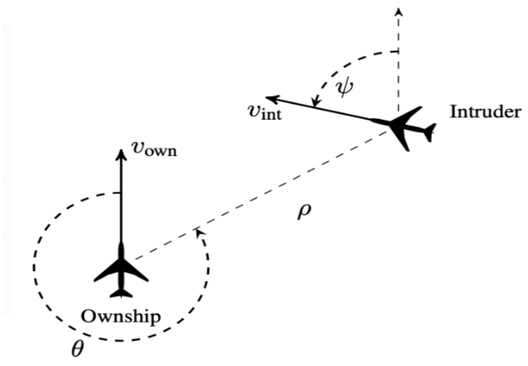
- Appropriate visual feedback needs to be provided to the ground crew, with features that can allow engineers to filter results and re-classify and archive relevant information. As such, user interfaces should be intuitive while serving a practical purpose.

3.5 ACAS Xu

This use case deals with publicly available data and models developed for monitoring flying objects and defining directions/decisions in order to avoid potential collisions. For the development and evaluation purposes of the selected methods in the MLEAP project, we rely on the data and models provided in the framework of the project DEEL[39] and activities related to the working group in EUROCAE 2020[40]. In the following sections, we first describe the main objectives of an ACAS Xu system, and where it comes from, then explore the main challenges and targeted objectives by including this use case in the MLEAP project.

3.5.1 Description

ACAS[41] stands for Airborne Collision Avoidance System, it is a universal system-to-system collision avoidance. It issues horizontal turn advisories to avoid an intruder aircraft. ACAS X stands for Next-Generation Airborne Collision Avoidance System. There are many variants, such as ACAS Xa for large aircraft, ACAS Xo for special operations, ACAS Xu for unmanned aircraft or ACAS sXu for small unmanned aircraft. The ACAS Xu is an air-to-air collision avoidance system designed for unmanned aircraft (drones); several evaluations have been made in the literature to evaluate avionic systems (Gabreau et al., 2022). In addition to the Minimum Operational Performance Standards (MOPS) by EUROCAE, promoting an implementation relying on lookup tables (LUT) without any ML constituent, recent studies by (Bak and Tran, 2022) have shown the performance of NNs in solving the collision avoidance issue, where the objective is to reduce the size of the embedded code and improve the anti-collision performance.
The purpose of an ACAS Xu system is to keep any intruder outside of the desired envelope of the ownship as illustrated in Figure 19.



[39] https://www.deel.ai/

[40] https://www.eurocae.net/about-us/working-groups/

[41] https://www.eurocontrol.int/publication/airborne-collision-avoidance-system-acas-guide

Figure 19. Horizontal ACAS Xu geometry

In

Figure 19, the collaboration between the ownship and intruder is based on a set of parameters computed to update the vehicles' trajectories. The ownship computes six parameters that enable access to the costs' tables, which give an estimation of the probability of a collision based on a set of known parameters about the position (cf. Table 8). The chosen action shall then minimise the likelihood of collision.
The ACAS Xu system does not need any communication between vehicles. Collision detection and advisories could be generated only using the ownship sensors. It enables the detection of cooperative traffic (other vehicles also equipped with the system) and noncooperative traffic, such as vehicles without ACAS Xu (small drones), birds, or ground obstacles.

3.5.2 Data

The data consists of different entries of the LUTs from the RTCA SC-147 Minimum Operational Performance Standards (MOPS)[42] for ACAS-Xu. The ODD is divided into sub-ODDs to fit the 45 ML model elements of the ML model architecture. Based on the different features defined in the previous section, different directions and movements in the space are defined. Hence, the methods using ML/DM models are provided with a complete ODD definition. The table below provides the set of known parameters upon which the LU tables are built:

Variables	Description	Unit of measurement	Definition domain
$\boldsymbol{\rho}$	Distance from ownship to intruder	*ft*	[0, 60 760]
$\boldsymbol{\theta}$	Angle to intruder relative to ownship heading	*rad*	$[-\pi, \pi]$
$\boldsymbol{\psi}$	Heading angle of intruder relative to ownship heading direction	*rad*	$[-\pi, \pi]$
$\boldsymbol{v_{own}}$	Speed of ownship	*ft/s*	[100, 1200]
$\boldsymbol{v_{int}}$	Speed of intruder	*ft/s*	[0, 1200]
$\boldsymbol{\tau}$	Time until loss of vertical separation	*s*	{0, 1, 5, 10, 20, 50, 60, 80, 100}
$\boldsymbol{a_{prev}}$	Last provided instruction	*deg/s*	{-3.0, -1.5, 0, 1.5, 3.0}

Table 8 Description of the known variables of the ACAS Xu LU tables

In total, the LU tables are made up of 600 million points. Figure 20 shows an example of a computed instruction.

[42] https://my.rtca.org/productdetails?id=a1B1R00000LoYKtUAN

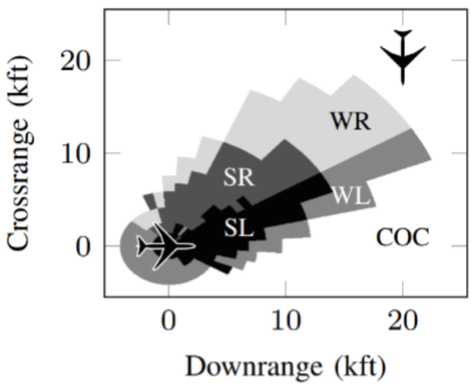


Figure 20 Example (Bak and Tran, 2022) of a provided instruction based on the known values in table 7.

3.5.3 Model

The ML model is broken down into model elements, and the architecture is validated (compatibility, consistency, verifiability, and conformity to ML design standards). Interfaces between ML model elements, in the form of data flow and control flow, are defined to be consistent between the elements. Specific ML techniques are defined or selected from analysing ML Model requirements and the ML datasets. Inadequate or incorrect inputs detected during the ML Model architecture activity are provided to the system life-cycle processes and the ML requirements process as feedback for clarification or correction. This architecture is identified as an element of the context of the main goal so that the argumentation can be based on each model element identified by the architecture. A specific goal has been added to verify the integrated ML model. Each ML model element is trained, validated and verified using the related datasets.

The architecture of the ML model is broken down into sub-model elements, as shown in Figure 21. It is verified that the union of the ODDs of all ML model elements makes the ML Controller ODD. The ML model description is created and validated with the NN architecture. The ML model elements are trained, validated, and verified. Each ML model element description is added to the ML model description.

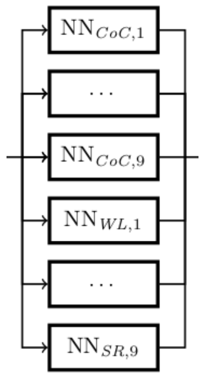


Figure 21. ML model elements of the ACAS Xu system.

3.5.4 Objectives and Challenges

In the ACAS Xu system, the drone receives instructions on how to avoid any collision with an intruder. The instructions are based on an LU table providing all possible configurations of the following cases:

- No conflict, do nothing: *Clear Of Conflict* (COC) ;
- Slightly left (at 1.5 deg/s): *Weak Left* (WL);
- Turn slightly to the right (at 1.5 deg/s): *Weak Right* (WR) ;
- Turn hard left (at 3 deg/s): *Strong Left* (SL) ;
- Turn hard right (at 3 deg/s): *Strong Right* (SR).

However, these tables require 2GB of storage space, which is not insignificant for drones. One way to circumvent this problem is to transform these tables into a neural network of smaller size. Hence, in this use case, the objective is to produce an ML/DL model that is able to fit the discrete input LU tables completely.

This is in contrast to one of the objectives of the experimental part of this project, which is to verify the generalisation of the trained models. This aspect implies good performance using data not encountered during training. Hence, one of the main challenges of the ACAS Xu use case is the fact that using the LUT as ground truth relies on their completeness and correctness (cf. section 4). Another issue is that this use case presents a risk of simplification of the generalisation problem. Differently from the other use cases, in terms of input data type and the objective of training, it represents another way to validate the proposed approaches in the MLEAP project, where the generalisation and robustness can be evaluated in a different way, and both aim to verify that the input space is fully known.

A deep neural network representation was trained to approximate the LU tables, maintaining optimal advisories while also approximating table values. The ODD is fully known, and it will be a good opportunity to test the different methods to assess the representativeness and completeness, generalisation, and validation on top of the segmentation strategy versus different subdomains of the ODD.

3.6 MLEAP dedicated server

The use cases presented in this section have been processed on a dedicated server hosted by Airbus Protect. This server has been selected to allow all tools selected by each task to have the opportunity to run on the use cases. The configuration of such a server is as follows:

- CPU: Intel Xeon Gold 5220R 2.2GHz
- RAM: 384 GB - 6x64GB
- GPU: NVIDIA RTXTM A4000, 16Go, 4DP (Precision 7920T, 7820, 5820)
- SSD: PCIe NVMe M.2 with 2TB extended to 4TB

3.7 Conclusion

In this chapter, we provided a detailed analysis of three different use cases selected for the evaluation of the different methods and tools used in the scope of the MLEAP project. These use cases address different applications in terms of data type, dimensionality, and task complexity. However, they provide an extensive benchmark for the approval of the applicability related to ML/DL models. Nevertheless, their use is not intended to certify these applications, but rather to support our conclusions and recommendations for verifications during the development of AI solutions meeting the same criteria (types of data, dimensionality, criticality, etc.). This will feed into the EASA certification process for AI applications, based on the results of this project.

Hence, STT-ATC deals with speech utterances (for a speech-to-text transcription) and text data analysis (for a correct callsign and intents analysis in the spoken instructions); AVI deals with high-quality images with two main objectives, detection of damages and their classification; ACAS Xu deals with lookup tables where entries are well defined, and hence the aim is to compress their content in a shallow NN. As we can notice, these use cases address different levels of complexity in terms of data dimensionality and ML model complexity.

Finally, for all use cases, we provided an overall state-of-the-art review, defining the application's objectives and the challenges involved. We also provide a high-level ODD definition, along with the data sets' statistics and models to be evaluated, an overall review of the safety assessment (when applicable), and references to open-source data that could be used for publication purposes.

In the next chapters, for each use case, the different data and models provided by Airbus internal projects and open-source ones will be used in a dedicated experimental section for each task.

4. Data: representativeness and completeness

4.1 Introduction

This chapter aims to provide in-depth information on how the data requirements related to completeness and representativeness can be reinforced and structured, in a way that broadly covers the different relationships that these attributes have with the wider Artificial Intelligence (AI) system environment (specifications, processes and activities, etc.).

The AI ecosystem presents an extremely shifting framework whose directions are dictated by the economic imperatives of the industrial AI producers or users, the progress of the state of the art, the different application areas involved requiring different types of approaches to AI verification and validation, and the European scene organising a regulatory framework harmonising both horizontal (AI-specific) and vertical (Sector-specific) needs. In such a context, the plethora of information and innovations bring as much confusion as progress in the concepts dealt with, and the present document serves the purpose of setting out definitions and identifying the topics of interest whose contours are mature enough to design robust requirements.

The chapter is structured as follows: section 4.2 presents and defines the main concepts to be used, particularly data quality requirements. The aspects of the AI system environment that are relevant for the assessment of completeness and representativeness are presented in three categories:

- Technical requirements, related to the technical specifications of the AI solution to implement (section 4.3).
- Processes of data management, in which activities may affect both attributes (section 4.4).
- Other data requirements that may conflict with requirements on completeness and representativeness (section 4.5).

Each of the three categories listed above includes definitions of the aspects addressed, an explanation of the relationship between this aspect and completeness/representativeness, and an identification of available tools and methods for the assessment (or for controlling their impact).

Section 4.6 summarises the information provided in the document through a selection grid allowing the identification of methods and tools for the assessment of the impact of each aspect in the three categories. Section 4.7 then details the preliminary experimentations that have been conducted to test the methods and tools presented in the selection grid in order to evaluate their practical usability. Section 4.8 extends these experimentations on the aviation use cases selected for the MLEAP project.

Finally, section 4.9 draws conclusions in light of all the discussions and experimentations covered in the chapter.

4.2 Concepts

4.2.1 Introduction

This section offers an overview of the concepts of data quality, completeness and representativeness. This section is meant to introduce the overall observations that are performed throughout the document, where most of the limitations encountered are relative to an absence of consistency or consensus in the community, and a lack of maturity in several aspects of data quality. Here, the idea is not to suggest the most suitable definitions but to present the potential variations in the acceptations and raise awareness about the usability of methods and tools in areas where even the root notions are sometimes under-defined or considered slightly differently among the stakeholders. This section can be understood as a general introduction to the topics addressed in the following sections of the document.

4.2.2 Generalities on Data Quality Requirements

While noting that it is not exhaustive, (EASA, 2024) identifies a list of Data Quality Requirements (DQRs):

- "*the data needed to support the intended use*";
- "*the ability to determine the origin of the data*";
- "*the requirements related to the annotation process*";
- "*the format, accuracy and resolution of the data*";
- "*the traceability of the data from their origin to their final operation through the whole pipeline of operations*";
- "*the mechanisms ensuring that the data will not be corrupted while stored or processed*";
- "*the completeness and representativeness of the data sets*";
- "*the level of independence between the training, validation and test data sets*".

Data quality is composed of a number of attributes[43]. The scientific community generally agrees on the existence and relevance of a number of them, such as "Accuracy" or "Completeness". However, depending on the data quality model, some other attributes may be included, such as "Conciseness" or "Clarity". Although there is some consistency among data quality models, the lack of international consensus on terminology, definitions and groupings by category creates confusion in establishing unified methods for qualification.

Table 9 presents an overview of data quality attributes. In (Mahanti, 2019), the list consists of the most commonly cited data quality attributes. (Batini et al., 2015), among numerous papers and books the authors produced on the topic offer a list of data quality attributes presented in categories. The series of international standards SQuaRE provides a data quality model (ISO/IEC 25012, 2008) based

[43] The present chapter does not develop on the selection of the best term; literature uses "property", "characteristic", "principle", "dimension", "attribute", etc. The EASA refers to Data Quality Requirements, which is understandable in the context of (EASA, 2024) but does not apply to the present analysis of completeness/representativeness. The term "property" has been selected and remains consistent throughout the chapter.

on fifteen attributes, articulated according to their nature (inherent data quality, system-dependent data quality, or both inherent and system-dependent). The status of international standards makes it liable to serve as a reference since standards rely on international consensus and are mainly designed for industrial use rather than for research. However, the model presented in this standard concerns data in the software sciences in general. In this regard, the standard misses some specific aspects pertaining to machine learning, first in the way the quality attributes are addressed, and secondly in the list of attributes itself (for example, data relevance and representativeness are not present).

Several works have been launched at the ISO level to fill this gap through the 5259 series on “Data quality for analytics and machine learning”. In particular, the standard (ISO/IEC CD 5259-2, 202X) provides a set of data quality attributes, which are leveraged in this chapter. In the series of international standards SquaRE (Systems and Software Quality Requirements and Evaluation), the standard (ISO/IEC DIS 25059, 202X) aims at expanding the scope of the framework to AI specificity, by analysing AI system quality in the light of data characteristics. Since the standards are still a work in progress and have not been published yet, the information provided here should be considered as relevant guidelines to follow, but they are not yet confirmed through publication of the standards. In this context, a full list of attributes is not provided, nor the exact definitions, but rather an explanation of the overall directions taken by the standard.

(Mahanti, 2019)	**Accuracy**	**Clarity**	**Comparability**	**Completeness**	**Conciseness**
	Consistency	Content	Currency	Efficiency	Format
	Freedom from bias	Importance	Informativeness	Interpretability	Level of detail
	Precision	Quantitativeness	Relevance	Reliability	Scope
	Sufficiency	Timeliness	Understandability	Useableness	Usefulness
(Batini et al., 2015)	Accessibility	Accuracy	Availability	Believability	Clarity
	Coherence	Cohesion	Compactness	Completeness	Comprehensibility
	Conciseness	Consistency	Correctness	Minimality	Pertinence
	Precision	Readability	Redundancy	Relevance	Reliability
	Reputation	Simplicity	Trust	Validity	
(ISO/IEC 25012, 2008)	Accessibility	Accuracy	Availability	Completeness	Compliance
	Confidentiality	Consistency	Credibility	Currentness	Efficiency
	Portability	Precision	Recoverability	Traceability	Understandability

Table 9. Several examples of data quality attributes.

4.2.3 Definitions of completeness and representativeness

In this chapter, in line with the conclusions of Sections 1.2.2.1 and 1.2.2.2, completeness and representativeness (respectively) will be used in the same sense as EASA’s Concept Paper. Unless explicitly stated otherwise, core concepts of other tasks, such as stability, will be used in the acceptation defined in the corresponding chapters.

4.2.4 Considerations of Data Dimensionality

At this stage, little to no work that explicitly tackles the notions of completeness and representativeness in their relationship with high-dimensional data could be found. The challenges of high dimensionality data are not addressed in the ISO/IEC 5259-2 definitions of completeness and representativeness; the examples currently provided in the standard focus on single isolated features of the data (e.g., the tone of individuals’ skin in images, postal codes). Dimensionality is only addressed in the context of similarity, a quality attribute of the data set that relates to the similarity between samples in exciting features. Similarity can be computed through the assessment of sample independency, through statistical measures such as the ratio of PCA and dataset dimension.

Scientific literature addressing the topic of high dimensionality in data focuses on the challenges relative to dimension reduction, namely the usability and validity of the techniques and their impact on performance.

In a survey of PCA techniques and their applicability to forensic studies, where data are of high dimensionality, (Lee and Jemain, 2021) note the importance of isolating the most relevant features for the analysis. The study notes that PCA is a powerful exploration tool, but the interpretation is derived mostly from visual inspection of the PCA plots, hence prone to cognitive bias. However, the study does not dwell on a use for representativeness and completeness analysis.

(Liu and Gillies, 2016) present a study on dimensionality reduction, performed to improve classification performance while still preserving some properties of the data. The paper argues that inter-class discrimination should minimised rather than maximised, since maximisation in the context of high-dimensional datasets may lead to severe overfitting. They demonstrate the efficiency of a method called Soft Discriminant Maps that provides an acceptable level of inter-class discrimination, namely more discriminative than PCA and less than Linear Discriminant Analysis (LDA). One should note, however, that the authors work on the concepts of data being “*efficiently represented*” and “*well-represented*”. “*Efficiently represented*” is characterised as “*the dimensionality of the data points is sufficiently small so that classification, regression or other manipulation of the data can be done quickly*”; this definition can be linked to the ISO/IEC 5259-2 acceptance of “data efficiency”. The second concept, “*well-represented*”, is characterised by “*little or no useful information is lost during the dimensionality reduction process, and redundant information is pruned out*”, which implies the absence of missing values as in completeness, but also mixes concepts that may pertain to the notion of similarity (redundant information would result in a level of similarity that would impact performance). The concepts tackled in the paper are thus slightly different from the acceptance of completeness and representativeness as defined in this chapter.

(Tsai and Sung, 2020) compare the performance of a classifier when feature selection is applied through PCA, GA (Genetic Algorithm) or C4.5 decision tree methods in the context of data sets with a high number of dimensions and a low number of samples. However, the paper does not provide relevant tools for the assessment of the representativeness of data following feature selection, since one can observe that the paper considers representativeness as "*representative features*", which is linked in the paper to features with the highest discriminative power, i.e., rather linked to the data quality attribute of diversity, as explained in section 4.5.4.

Literature seems scarce in offering studies that directly deal with completeness and representativeness in their relationship with high-dimensionality data since one can observe that the concepts presented in literature do not seem to follow strictly the definition of the concepts defended in the project. The relationship with data dimensionality and its impact on completeness and representativeness is explored further in section 4.3.4.

4.3 Technical requirements

4.3.1 Introduction

The complexity of the task (or function) to be achieved by an AI system determines several technical aspects of the solutions to implement. In turn, the increasing complexity or sophistication of the target solution influences factors such as the required quality and quantity of data. Most technical requirements discussed hereafter can affect the definition of the ODD, making it a central document in tackling questions relative to completeness and representativeness. This section identifies the technical requirements that may impact data completeness or representativeness. Each requirement is presented with its definition, source (if a reference definition could be found), and its relationship with representativeness and completeness. Finally, existing methods and tools for assessing completeness and representativeness w.r.t these technical requirements are presented. The overall objective of the discussion is to highlight the maturity of the field and the feasibility of employing such tools or methods to address the technical requirement.

The literature on the assessment of completeness and representativeness of a data set is modest, as data quality, in general, is still an emerging research domain (P. Li et al., 2021). Even though the factors of influence discussed here are known, they are rarely used to structure assessment approaches. Moreover, tools and methods designed to assess the quality of a data set generally address several aspects of data quality without focusing on completeness or representativeness specifically. Considering these factors, it seemed preferable to have separate discussions on the factors of influence on one hand and the methods and tools on the other.

4.3.2 Intended behaviour

4.3.2.1 **Definition**

The function or task the AI/ML application has to perform.

4.3.2.2 **Influence on completeness and representativeness**

The intended behaviour of an AI/ML system is first modelled by the application's operating parameters, i.e., the information that will be gathered to form the data set. These operating parameters may undergo several transformations (e.g., data augmentation, feature engineering) before yielding the input and output spaces the model will use. The completeness and representativeness of the data set, i.e., operating parameters, must be ensured before any transformation and must then be reconfirmed at each step so that the input space on which the model will learn enforces these attributes.

Moreover, the dimensionality of these spaces (i.e., the number of features) and their size (i.e., the range of each feature) influence the volume of data required to efficiently train a model, affecting the requirements for ensuring completeness and representativeness.

The degree of structure of the input also affects volumetry, as a more structured input is generally easier to handle for ML models. Consequently, as the degree of structure decreases, the number of data points required to guarantee completeness and representativeness increases, making data collection more complex. Finally, even in a given domain, e.g., computer vision, different tasks may have uneven state-of-the-art performance, which may indicate the designer to be more or less vigilant about the properties of the data set they use, including completeness and representativeness.

4.3.3 Model architecture

4.3.3.1 **Definition**

The model architecture refers to the type of Machine Learning algorithm chosen to embody the AI/ML application achieving the intended behaviour. Popular architectures include Linear Regression, Naive Bayes, Maximum Entropy or Transformers.

4.3.3.2 **Influence on completeness and representativeness**

The architecture of a model determines in large part its capacity, i.e., how much information it can learn and retain[44]. In turn, capacity has a direct impact on the amount of data required; a model with more capacity will require more data to reach a level of performance similar to a model with lesser capacity. However, the model with more capacity will be able to capture more information and thus exploit a larger data set better. Therefore, choosing a model with adequate capacity w.r.t the task at hand will limit the data collection effort and the subsequent difficulty of ensuring the completeness and representativeness of the data set.

[44] Capacity is discussed in more detail in Chapter 5.

4.3.4 Data dimensionality

4.3.4.1 **Definition**

Data dimensionality refers to the number of dimensions of the input space.

4.3.4.2 **Influence on completeness and representativeness**

On most quantitative data sets, data dimensionality corresponds to the number of features defining a typical sample. On unstructured data such as audio, text or images, defining the dimensionality of an input is an important design choice in the task of representation learning. It is defined as the task of learning representations of the data that make it easier to extract useful information when building classifiers or other predictors (Bengio et al., 2013) and is related to feature engineering and feature extraction by the authors.

When samples are defined by individual features, dimensionality reduction techniques, as well as feature selection and feature engineering, may be used as pre-processing to help fit the data to the system and improve performance. However, removing or transforming features must be done with caution as it may lead to an input space inconsistent with functional conditions, detrimental to completeness, and a loss of information detrimental to representativeness.

On the contrary, it is possible to seek to enrich a data set with more features to improve its representativeness of the complexity of the intended behaviour. However, such a strategy must also be used cautiously, as an increase in dimensionality may lead to all samples being at comparable distances in the input space. This phenomenon, called the “Curse of Dimensionality” (CoD), vastly impedes the learning process, leading to poor performance of the trained model.

Estimating the right number of dimensions to consider for the intended behaviour pertains to the assessment of the data quality attribute of relevance, which, as described in section 4.5.3, can impact representativeness and completeness. The section provides methods for the assessment of relevance but highlights the absence of standard methods to ensure a balance between the three attributes, meaning that defining the most adapted number of dimensions can only be the result of an expert trade-off.

Thus, data dimensionality is an important dimensioning factor of an ML problem. Moreover, it is also the main limitation of most data quality assessment methods, as they usually specialize in one particular kind of data (low—or high-dimensional, structured or unstructured).

4.3.5 Intended level of autonomy

4.3.5.1 **Definition**

In (EASA, 2024), the EASA distinguishes three levels of AI/ML applications, based on the degree of human oversight exercised. It is further divided into six total sub-levels from “*human augmentation*” (the highest level of oversight) to "*AI-based system performs non-overridable decisions and actions* " (the lowest level of oversight).

4.3.5.2 **Influence on completeness and representativeness**

As an AI/ML application's autonomy increases, its task allocation broadens. In turn, the volume and variety of data required to ensure its performance requirements (including robustness, resilience, and stability, discussed below) increase, and with it the spectrum of data needed to ensure the completeness and representativeness of the data set.

4.3.6 Intended level of performance

4.3.6.1 **Definition**

(ISO/IEC 22989, 2022) defines performance as "*measurable results*". However, this definition is too vague in the context of this work, and other sources can be leveraged to complement the definition. (LNE, 2021) defines performance as "*The degree to which a system or component performs its designated functions within a given set of constraints, such as speed, accuracy or memory usage, etc.*". The notion of "*designated functions*" may be understood in the sense of "*Performance requirement*" as defined in the standard (ED-79A, 2010): "*Performance requirement define those attributes of the function or system that make it useful to the aircraft and its operation. In addition to defining the type of performance expected, performance requirements include function specifics such as accuracy, fidelity, range, resolution, speed and response time*".

4.3.6.2 **Influence on completeness and representativeness**

Assuming a model capacity is consistent with the complexity of the task, reaching a higher performance level, especially in the context of robustness requirements, will usually require more data. Moreover, even considering performance solely from a system prediction quality standpoint and using a single evaluation metric, it is possible to divide the development and test sets into subsets with particular, out-of-task characteristics (e.g., meteorological conditions for object segmentation) and perform individual evaluation of each subset to further refine the analysis. Such fine-grained evaluation may intersect with constraints related to robustness and resilience and will, therefore, similarly increase the need for more numerous and varied samples to ensure completeness and representativeness of the data set.

4.3.7 Intended levels of robustness and resilience

4.3.7.1 **Definition**

The definitions of robustness and resilience from (EASA, 2024) are used, respectively:

- "*Ability of a system to maintain its level of performance under all foreseeable conditions. At model level (trained or inference), the robustness objectives are further split into two groups: the ones about 'model stability' and the ones pertaining to 'robustness in adverse conditions'*";
- "*The ability of a system to continue to operate while an error or a fault has occurred*".

4.3.7.2 **Influence on completeness and representativeness**

As discussed for the intended level of human oversight, increasing the requirements regarding robustness and resilience also broadens the spectrum of the phenomena to be covered by the data set, which in turn broadens the input space, requiring more samples (in terms of sheer quantity) and more diversity (in accordance with the phenomena of interest).

The level of oversight is not the only factor that can influence specifications regarding robustness and resilience. Higher safety requirements related to the criticality of the AI/ML applications are also to be considered, with the same effects on the data collection effort.

Finally, robustness encompasses adversarial attacks (among other cases). However, specifying robustness requirements regarding adversarial attacks is difficult, as it is, in essence, hard to anticipate their form, let alone devise dedicated sets of samples that could help the system resist them. Therefore, adversarial attacks are left out of the scope of this chapter.

4.3.8 Intended levels of stability

4.3.8.1 **Definition**

EUROCAE WG 114 and SAE G-34 are working on a definition of stability as the ability "*to provide an equivalent response within the neighbourhood of an input*". This definition is in line with the definition of "model stability" in (EASA, 2024), which distinguishes "*model stability*" from "*algorithm stability*". In this work, stability will be considered exclusively from the "*model stability*" perspective.

4.3.8.2 **Influence on completeness and representativeness**

This chapter distinguishes stability from robustness and resilience for different reasons. First, it is still a fuzzy concept without a standard definition, though efforts are ongoing by groups like the WG114, G-34 or the ISO. Second, stability relates more to performance than robustness or resilience without being explicitly encompassed. Third, its influence on completeness and representativeness is different, with a stronger impact on volumetry: as stability requires a consistent behaviour for similar input, a need arises for a finer-grained coverage of the input space (all other requirements being equal). This is achievable only by collecting a larger volume of quality samples[45].

4.3.9 Methods and tools for assessment

4.3.9.1 **Methods related to technical requirements**

Several identified methods use the model's behaviour during training to infer information from the data set. Consequently, model architecture is implicitly considered. In these approaches, coverage, i.e., the model's completeness of representation, is used as a proxy for the data set's intrinsic

[45] Chapter 6 discusses formal methods for stability, which may temper the need for supplementary data but still shows the influence of stability requirements over the data set, as these methods form an additional and non-trivial process.

completeness. Despite their subsequent limitations, these approaches enable a dynamic, model-driven process of data set improvement with a minimal number of additional samples[46] (those needed to improve model coverage).

An added advantage of using the model for the characterisation of the training data set is that it becomes possible to assess the adequation between the chosen model architecture and the data set. The overall methodology is close to *instance completion* (Dhurandhar and Sankaranarayanan, 2015), and it is used to improve a model's performance by identifying test samples that may help complete the representation of a given input sample. However, using the model for the characterisation of a data set should not be done at the expense of the independence of the validation and, most importantly, of the test data set. Such a method can similarly be used to get insight into the distribution of the phenomena in the data set and improve representativeness or completeness, though this is always done through the lens of the model's learning process.

4.3.9.1.1 Feature Space Characterisation

A more favoured angle is to use the model as a tool to characterise the data set. For example, (Mani et al., 2019) propose a methodology for Deep Learning models, based on the idea that supervised models define a feature space for each class to predict. They define the feature space as “a collection of features related to some properties of the object and the number of features determines the dimensionality of the space”. In practice, they identify those features as the activation values on the neurons in some layers. The extrema of the features define the limit of a class range. Beyond these boundaries, the model either changes class or cannot make a decision. The authors hypothesise that it is possible to use the trained model as a generator to explore the feature space and use it to characterise the data set. To this end, they devise four metrics:

- Equivalence partitioning: check that all classes are evenly represented in the data set.
- Centroid positioning: the percentage of samples in a given radius of the centroid.
- Boundary positioning: the percentage of samples close to the class decision boundary, the samples at the boundary being the less confident decisions.
- Pair-wise boundary conditioning: the percentage of samples fulfilling a boundary conditioning constraint, for any pair of two classes.

However, though it is not discussed in the paper, Equivalence partitioning introduces a bias by imposing class balance, which is a strong constraint for data collection and has a strong influence on the model's learning process. Bias mitigation must be enforced, especially as it is usually a direct condition for meeting performance and robustness requirements in rare cases, but it must be so while respecting the larger distribution of phenomena of interest in operating conditions.

This method is more of a fine-grained model performance diagnostic tool in a supervised setting. However, to be relevant as a data set characterisation tool, applying it to unsupervised models is

[46] Such process might feel reminiscent of active learning, though the logic is different because in active learning, the model embeds an ability to identify the most suited sample to maximize its learning phase, while in the discussed methods, the model remains passive and the feedback has no incidence on the current learning process.

preferable. Nonetheless, the chosen model remains a proxy with many potential caveats that must be anticipated and kept in mind when performing analysis.

4.3.9.1.2 Performance monitoring

Another approach, by (Tae and Whang, 2021) is called the Slice Tuner, a tool for maximising the accuracy and fairness of prediction by identifying the most suited samples to present to a learning model at a given point during training.

A slice is defined as a subset of samples, identified by a conjunction of features. While the method seems agnostic to data dimensionality, the authors do not discuss strategies to select slices on unstructured data such as images. Slice Tuner works by monitoring the learning curve of the model for each slice. It is based on the idea that the overall learning process of a model for a given loss function follows a decreasing power law. Slice Tuner feeds the model with a first slice, and changes slice when the model's learning curve starts to flatten. Moreover, fairness is embodied by enforcing an Equalized Error Rate over all slices, which can be extended to other forms of constraints on the learning process, making Slice Tuner a seemingly flexible tool. Besides raw performance, the speed at which the learning curve decreases may be indicative of certain slice-wise specificities of the data set. Such mechanisms can also be leveraged to monitor per-class or per-sub-class prediction performance and identify, all of these observations being useful for characterising the completeness and representativeness of the data set.

4.3.9.1.3 Neuron activity monitoring

The approach described by (Pei et al., 2017), called DeepXplore, is tailored for neural networks and aims at monitoring their learning process based on neuron activation. The core idea derives from code coverage in software engineering and is called neuron coverage. It rests on the idea that the model has fully exploited an input activating all neurons (i.e., full neuron coverage). Conversely, low input coverage shows a weakness of the model. It might be used to identify low-quality samples, i.e., samples containing either little to no information (which could indicate a sample is too hard, too easy, or missing values) or conflicting information (e.g., between the input and its label, which could be indicative of a mislabelling). Identifying these may help improve the completeness and representativeness of the data set. A limitation of the approach is that it works on a white-box assumption, assuming full knowledge and access to the model, as its initial aim is to serve as a joint training constraint.

A similar approach is presented by (Lei et al., 2018). Their tool, called Deepgauge, characterises a data set through the behaviour of a model. This is done via two observations:

- *Major function regions* are characterized by ensuring that the neurons' activation values at evaluation time (on the dev set) are in the min/max range of values obtained at training time. In addition, K-multi-section neuron coverage is used: the spectrum of activation values is binned, and Deepgauge monitors that the entire spectrum of bins is used.
- *Corner case region identification checks* if the activation values are over a given threshold. The idea behind that is to explore the decision boundary at the neuron level.

A model displaying consistent major function and corner case activation is indicative of both good learning and good adequation between the training and validation data set. Similar to DeepXplore, these criteria may be leveraged to identify low quality samples.

4.3.9.1.4 Human-in-the-loop approaches

(Kiela et al., 2021) describe a tool specifically aimed at addressing the problem of adversarial attacks. The tool, Dynabench, aims at iteratively improving a model by enriching its training data set with handcrafted adverse examples. To this purpose, Dynabench allows a user to benchmark models against data sets and provides an interface for submitting handmade adverse example to fool the models. This framework could be extended to be used as a tool to incorporate new samples, improving any dimension of the completeness and representativeness of the data set (not just adversarial attacks). A case of particular interest would be if technical specifications are exact and automated collection of corresponding samples is hard, or when there are strong control requirements over the types of samples that must be added to a data set regarding a particular technical requirement.

The Dynabench framework was initially designed for a specific set of tasks, and was not flexible enough to allow extension to new tasks. Dynatask introduced this functionality (Thrush et al., 2022), allowing users to incorporate new tasks in the framework.

4.3.9.2 **General methods**

Apart from the methods that could be related to the models' architecture and were introduced in the previous section, the technical requirements identified as influence factors here are rarely used as structuring elements in the design of assessment methods for a data set's completeness and representativeness. This is mostly because quantifying each of these attributes is a task by itself, and works in this area are scarce. To support this argument, the remaining of this section will focus on the methods found for assessing some of these attributes, to bring an idea of the state of the art in this domain.

As an example, as stated in 4.3.3, model architecture influences the model's capacity, which is a dimensioning factor for completeness and representativeness. However, capacity itself is hard to assess: the linearity or non-linearity of the model and its number of parameters are well-known factors[47], but a single formal capacity metric does not exist. Attempts at measuring capacity include (Raghu et al., 2017). The authors devise a method to assess the capacity (called expressiveness in the paper) of a neural network through three indicators:

- Transitions are defined as neuron switches from one regime to another, depending on the activation function (ReLU transitions are the switch between 0 and the linear regime, for hard tanh, when it switches from its saturated (positive or negative) regime to its unsaturated regime).

[47] All things being equal, a non-linear model has more capacity than a linear model, and more parameters also means increased capacity. A model with more capacity will be able to capture more phenomena but will generally require more data to do so, setting a different scale on the task of assessing completeness and representativeness.

- Activation patterns extend transitions of a single neuron to all neurons in the network. Thus, a specific activation pattern can be mapped for each input going through the network.
- Dichotomies is a dual formulation of expressiveness. Transitions and Activation patterns focus on expressiveness relative to the input, while dichotomies concentrate on the weights to infer how heterogeneous the general function of the network is.

The study shows that expressiveness mostly increases exponentially with the depth of the network, while the width has little impact. It also shows that the first layers of a neural network tend to act as controlling parameters for the deeper layers. Both conclusions are consistent with empirical observations but do not really go beyond a more formal description of these phenomena. These methods can be used for corner case construction, see Section 4.4.8.8 for more.

Other strategies, such as ensemble learning, fine-tuning of pre-trained models, or regularization, also affect a model's capacity and the volume of data required to reach a given level of performance. This affects the conditions to be met to ensure the completeness and representativeness of the data set[48]. However, no method has been found to quantify their impact. Overall, such limitations hinder the development of methods leveraging capacity as a clear variable in the assessment of completeness and representativeness.

The same dynamic holds true for other factors, such as the intended behaviour: It would be interesting to quantify the difficulty of a task, which in turn would enable the sizing of the model's capacity, but such a framework does not exist. The closest estimator of task difficulty is intra- and inter-annotator agreement, which is too distant a proxy to be used efficiently in a quantitative approach.

On the other hand, factors like the intended levels of performance, robustness, resilience or stability can be quantified. As an example, (Almeida and Vieira, 2011) highlights that robustness and resilience can be evaluated by using the performance metrics and degrading the operational condition of the system (with noisy inputs). (Sáez et al., 2016) even propose to integrate robustness within the performance metric with their Equalized Loss of Accuracy (ELA) whose strength is to account for the initial accuracy of the classifier, in contrast to earlier works. Yet, no work has been found that tried to leverage such objectives for the assessment of a data set's quality.

More general tools such as Whylogs[49], Cleanlab[50] (Northcutt et al., 2021) or JENGA (Schelter et al., 2021) are available. Whylogs is designed to provide a dashboard of general information about the distribution of a data set and enable data quality monitoring and other workflow around data exploration and maintenance. Cleanlab aims at identifying noisy and mislabelled samples in a data set, while JENGA is a framework to generate synthetic samples emulating common data errors (such as missing values, outliers and other noisy inputs) and evaluation tools to assess their impact on model

[48] Fine-tuning and regularization are generally used to obtain good performance from a model with important capacity using a smaller data set, implying less constraints on the completeness and representativeness of the data set, and also a less intensive effort to assess it.

[49] https://whylabs.ai/

[50] https://github.com/cleanlab/cleanlab

performance. Further testing of Whylogs and Cleanlab is required to get more insight into how these tools can be related to the technical requirements discussed here, which is why they will be part of the tools selected for the next phases of the MLEAP project (JENGA will not be tested as data synthesis is considered out of the scope of the project).

4.3.9.3 **Conclusions**

In light of the literature, it seems that technical requirements are rarely used as a prism for the assessment of a data set's completeness and representativeness. Current developments mostly focus on building systems that can compensate imperfect data sets rather than devising good data curation solutions[51] (Biessmann et al., 2021).

While most of these factors all tend to increase the need for data to achieve completeness, their effect on representativeness is more ambivalent. Indeed, higher intended levels of performance will require more (representative) data. In the meantime, achieving better robustness and resilience will require focusing on rare phenomena and corner cases, which may result in the use of oversampling strategies and other techniques that significantly alter the distribution of phenomena of interest and, therefore, the representativeness of the data set. In a similar reasoning, (Rao and Frtunikj, 2018) highlight that in autonomous driving, the raw volume of the data set is less significant than the volume of data describing anomalous events. In a sense, factors like robustness tend to shift the focus from the statistical modelling of the phenomena at hand toward the representativeness of the model in terms of expected behaviour (also referred to as coverage). Therefore, there is a trade-off to consider between the representativeness of the data set w.r.t the distribution of the phenomena of interest, and the model’s coverage of these phenomena. Improvements obtained on rare cases, regardless of their criticality, must not be detrimental to the more common situations.

[51] This does not preclude the proper application of all the processes and good practices of data management. However, as the shortcomings of the data set cannot be entirely characterized, they remain a limiting factor for the model's performance.

4.4 Processes

4.4.1 Introduction

Most learning systems[52] require data to identify patterns used to infer behaviours, which means data need to be collected beforehand to enable the system's learning and evaluation.
Therefore, several processes must or may be performed before the data are usable, each of which constitutes a potential point of alteration of the data that will bias the system's learned behaviour. Some of these biases may be introduced in a voluntary and controlled way, in order to enhance the system understanding of specific cases (such as rare but critical events). Others must be avoided or at least identified, documented and their impact minimized.

The processes presented in this section do not have the same granularity as in (EASA, 2024). Where required, the related step of data management from the Concept Paper is indicated in brackets in the title.

This section details processes in the design and implementation of an AI/ML system where the completeness and representativeness of a data set may be altered or otherwise not satisfied. Each process is defined; then its potential influence on completeness and representativeness is explained. Finally, existing methods and tools for assessment are discussed.

4.4.2 Data management requirements

4.4.2.1 **Definition**

Data management, as defined in (EASA, 2024), is the process of designing the specification of the required data set.

4.4.2.2 **Influence on completeness and representativeness**

Data management requirements encompass a variety of aspects that need to be considered to ensure that data is of quality and adapted to the intended application. Among these aspects: data must be collected from trustworthy and quality sources; dubious sources may induce data poisoning or backdoor to adversarial attacks; the quality and volume of data should comply with the task complexity and capacity of the selected AI model; the content of data should accurately represent the applicative input space. The definition of the system's data management requirements is then a fundamental stage during which specifications may conflict with completeness and representativeness, which may encompass requirements linked to all factors of influence identified in this chapter.

[52] This document leaves out AI systems based on reinforcement learning.

4.4.3 Data quality improvement (in Data Preparation)

4.4.3.1 **Definition**

Data quality improvement, as currently specified in (ISO/IEC CD 5259-2, 202X), consists in manipulating the original data to increase their amount or allow them to fulfil some requirements. Among the techniques associated to data quality improvement, this chapter will discuss in particular data augmentation and data imputation. Data augmentation consists in taking samples from the data set and applying one or more transformations in order to create a set of “new” samples. Data imputation designates methods to replace missing values in the data with inferred values in order to improve the dataset’s completeness.

4.4.3.2 **Influence on completeness and representativeness**

Data augmentation is, at its core, a strategy to improve the completeness and representativeness of factors that have been identified as insufficiently represented. It can also be used to improve the robustness, resilience and stability of an AI/ML application. However, to work properly, the relevance of the augmentation treatments (i.e., the augmentation effectively contributes to improving a specific technical requirement) and their quality (i.e., the augmented sample may be considered realistic w.r.t the original input space) must be ensured. Additionally, as discussed in 4.4.2, each sample may satisfy several technical specifications at a time, it is therefore important to ensure the completeness and representativeness of the data set are maintained when introducing augmented samples.

Data imputation improves a data set’s completeness at the cost of representativeness. Indeed, ablative methods (i.e., removing features for which samples have missing values) remove learning material and may hide correlations with other variables, altering the distributions of latent phenomena of interest. On the other hand, filling the missing values with statistical estimates may misrepresent and bias said distributions, e.g., by shifting common correlations toward rare cases and vice-versa.

4.4.4 Data Synthesis (in Data Preparation)

4.4.4.1 **Definition**

Contrary to data augmentation where new samples are created from existing ones, data synthesis is the process of generating new samples from scratch. That is, data synthesis helps extend the input space covered (in contrast to data augmentation which can only improve the already covered input space). The created samples are “realistic”, i.e., their properties and content match those of an authentic sample, which makes them usable for training the AI/ML application.

4.4.4.2 **Influence on completeness and representativeness**

As for data augmentation, data synthesis is a strategy to improve the completeness and representativeness of a data set. The quality of synthetic samples is then key to the relevance of this approach, for the same reasons discussed in 4.4.2 and 4.4.3, i.e. to ensure these new samples preserve the completeness and representativeness of the data set.

4.4.5 Data sampling (in Data Preparation)

4.4.5.1 **Definition**

Data sampling covers the methods and tools used to select a subset of data points from a larger set of data points following a sampling rule. Usually, these rules are specified so that the sampled data set has the same distribution as the original data set. It is used when the input space is too large to be processed by an AI Model in reasonable time and resources.

4.4.5.2 **Influence on completeness and representativeness**

Data sampling can impact both the completeness and representativeness of a data set, either in a positive or negative way. A very large data set may display strong biases, such as class imbalance (see section 4.5.2 on balance). In such cases, data sampling enables the AI/ML application designers to mitigate this bias w.r.t the real-life distributions of phenomena of interest as well as the technical requirements of the application. Data sampling strategies may also be used to mitigate biases in more modest-sized data sets, but this should be done in accordance with the capacity of the chosen model architecture to ensure sufficient overall volumetry.

4.4.6 Labelling (in Data Preparation)

4.4.6.1 **Definition**

Labelling consists in annotating samples in order to train and evaluate a supervised AI/ML application. Labelling can also be used for the evaluation of unsupervised AI/ML applications.

4.4.6.2 **Influence on completeness and representativeness**

Data labelling can be performed through expert annotation, semi-automatic annotation or automatic annotation, the former being the most expensive and the latter being more cost-efficient. In any case, it remains a generally costly and time-consuming process, as it requires some degree of human verification. Therefore, the time dedicated to this task, the quality of the annotation guidelines (either for annotation or verification of the annotations), the number of annotators tasked (allowing for a more robust cross-verification of their work) and their competence (either degree of expertise or incentive to perform the task, especially in the case of crowdsourcing) all influence the quality of the resulting annotations. Thus, sketchy guidelines and loose annotation consistency across annotators (due to fuzzy guidelines and hurried or unqualified annotators) lead to errors that may compromise the representativeness of the data. More specifically, either phenomena of interest that are explicitly labelled will be so incorrectly, or latent phenomena will be associated with the wrong class, hindering pattern recognition by the model.

4.4.7 Pre-processing

4.4.7.1 **Definition**

Pre-processing encompasses any treatment that modifies the data to facilitate its processing by AI/ML applications and encourage desired behaviours to emerge during the learning phase. Common pre-processing in NLP (Natural Language Processing) includes normalisation (i.e., harmonising type case and punctuation) or lemmatisation. In computer vision, images are frequently grayscale and rescaled to a unique format. Finally, pre-processing also includes feature engineering.

4.4.7.2 **Influence on completeness and representativeness**

Certain pre-processing operations may reduce the amount of information available for learning. While it is acceptable and sometimes necessary to encourage generalisation or robustness, the influence of such information loss must be monitored to not degrade the distribution of phenomena of interest in the data set to the point of compromising its completeness and representativeness.

4.4.8 Methods and tools for assessment

4.4.8.1 **Generalities**

As highlighted by (Mountrakis and Xi, 2013), the data set quality may have a more significant impact on the final model's performance than any model design choice. This is because data management implies many steps before the data even arrive at the model, and all pipeline approaches are bound to error propagation, a phenomenon discussed in (Sambasivan et al., 2021) as "Data cascade" and identified as a weak point of many AI/ML application development, because it is regarded as a tedious and less rewarding activity. Yet, data curation approaches (including some of those discussed in this chapter) are usually empirical and multifactorial, i.e., they address several aspects of data quality at once.

General methods to obtain an overview of the representativeness or completeness of the data usually rest on statistical indicators, such as the R-indicator (Schouten et al., 2009). However, these indicators are generally not adapted to unstructured data. Moreover, a limitation of completeness and representativeness assessment, especially using statistical tools, is that it usually requires knowing the actual real-life distribution of phenomena of interest, which is generally not possible (Ramsey and Hewitt, 2005). This relates to the work of (Cabitza et al., 2021), where it is pointed out that data similarity is central for ML generalisation: dissimilar data sets tend to come from different underlying distributions. Consequently, being able to gather contrastive information of distributions can be insightful for performance improvements. To do so, they extend existing work by devising a representativeness metric based on Data Agreement Criterion and Data Representativeness Criterion. Both metrics are based on KL-divergence and require *a priori* estimated parameters to compute the similarity between 2 distributions. The authors' contribution is to introduce a meta-validation method making the process non-parametric and thus not requiring *a priori* knowledge of the reference distribution.

Additionally, (Catania et al., 2022) suggest that confidence interval as well as monitoring a system's learning curve, especially coupled with techniques like cross-validation, may be useful to detect local lacks of representativeness or completeness in any type of data by the notable decrease in performance they would highlight. They also discuss strategies to circumvent such lack of representativeness, including data stratification, which is slightly outside the scope of this work. Class-wise evaluation may also refine the performance analysis and get insight into possible weaknesses in the data set's properties and be exploited as more precise feedback for the development of the model.

4.4.8.2 **Data Management Requirements**

The volume of data must not only be considered at data set scale, but from the technical specifications' standpoint as well. In particular, the ODD will specify a range of operating conditions with general performance objectives, which will ensure that a sufficient volume of samples is collected w.r.t each requirement. In addition, it is unlikely that each example would cover a single requirement. It is then essential to ensure completeness and representativeness are achieved given the combination of requirements satisfied by the samples in the data set. Tools like Whylogs (discussed in 4.3.9.3) can provide coarse-grain monitoring of the data set after collection, possibly motivating additional efforts.

More in-depth methods for characterising data sets exist, with their limitations. (Asudeh et al., 2019) Describe an approach to exhaustively identify the combinations of feature patterns represented in a data set. The method works best with low-dimensional, categorical feature spaces, though the authors indicate that it can scale to higher-dimensional, continuous inputs by pre-processing, such as binning. Assuming an example data set where each sample is an individual, defined by three features: sex (M/F), race (B/H/A/W), age class (A: 0-30; B: 31-60; C: 61+), the method rests on the construction of a tree-like structure where:

- Leaves are actual data points (individuals).
- A node is a combination of features, the values of which are either fixed or left variable. The combination is called a *pattern*. For example, FBX is the pattern encompassing all black females, regardless of age.
- Hence, the root is a pattern of only variable features (here is *XXX*), while a leaf is a pattern of only fixed features.
- A given pattern P has parent pattern P' if P has one more fixed feature than P'. For example, FXX (all Females) is a parent pattern of FBX (Black Females).

The concept of Maximum Uncovered Pattern (MUP) is then introduced. A given pattern P is a MUP if its coverage, i.e., the number of samples matching its combination of features, is less than a set threshold t while all its parent patterns have a coverage greater than t. Thus, the identification of MUPs is a direct mean of assessing the completeness of a data set, while the threshold allows a fine study of its representativeness.

The authors present three exploration algorithms to identify MUPs in a data set:

- Pattern Breaker is a top-down Breadth-First Search approach. It works by exploring the covered regions of the graph first. Consequently, it is rather inefficient in a data set with few uncovered regions.
- Pattern Combiner is the converse, Breadth-First-Search bottom-up method of exploration, prioritizing the exploration of the uncovered region. Therefore, it exhibits the opposite weakness of Pattern breaker, inefficient on mostly incomplete data sets.
- Deepdiver is the last strategy, aiming at mitigating the two previous ones and providing an algorithm with more stable performance w.r.t data set coverage by using a Depth-First-Search approach.

However, the overall method seems impractical on unstructured data such as text or images. Nonetheless, for data fitting these constraints, it is one of the most exhaustive tools found at the time of writing.

Another aspect to consider is whether the data set collection will happen from scratch, if a single pre-existing data set will be retrieved or if the data set will be assembled from several distinct data sets. The latter is called data integration and is discussed by (Paganelli et al., 2022). Their method aims at maximising data completeness when integrating text data sets, by leveraging word frequencies. Another method, presented by (Trinh et al., 2018) evaluates completeness from an end-user perspective rather than a data-specialist perspective. It is composed of two metrics:

- Data Source: a score between 0 and 3 where:
  - 0: no data;
  - 1: adapted from other references using the same nanomaterial and experimental conditions;
  - 2: adapted from manufacturer specification;
  - 3: experimentally measured by the authors.
- Measurement method: a score between 0 and 2 where:
  - 0: no information on the measurement method;
  - 1: non-standardized and less commonly used method;
  - 2: commonly used and standardised method.

A completeness score is computed as the mean of both scores for each sample in the data sets.

Other experts leverage simple metrics to assess the completeness of their data set. Notably, (C. Liu et al., 2017) provide an overview of the notion of completeness in the medical domain. It is shown that completeness is generally defined as the ratio of samples bearing the desired features in the data set over the total of samples, with variations on the counting methods based on specialties and objectives of the studies. These definitions can be implemented into algorithms that can in turn automatically gather data sets while ensuring its completeness on the fly.

Similarly, in their review, (Heinrich et al., 2018) compare different metrics for data quality, including completeness but not representativeness, with the objective of identifying those that can contribute to the system's performance and be economically sound. To do so, they define 5 requirements:

- Existence of minimum and maximum metric values.

- Interval scaling of the metric values.
- Quality of the configuration parameters and the determination of the metric values.
- Soundness of the aggregation of the metric value.
- Economic efficiency of the metric.

From these requirements, they conclude that the completeness metric fulfilling all requirements is simply: C$= 1 - \frac{T_R}{N_R}$ where T_R is the number of samples with at least one missing feature, and N_R is total number of samples.

Another data quality evaluation framework is proposed by (Even and Shankaranarayanan, 2007). The authors aim at assessing the quality of each sample by taking the applicative context into account, rather than an absolute measure of quality. Their method consists in a utility function integrating several data quality dimension, including completeness. Its computation provides insight into the individual impact of each sample on the utility function.

4.4.8.3 **Data Quality Improvement**

As discussed in 4.4.3, data quality improvement encompasses data augmentation and data imputation. Data augmentation may rely either on “non-learnable methods”[53], i.e., metamorphic transformations such as cropping or rotation (to encourage the learning of position-independent features) or by introducing some black or white pixels or gaussian noise, to create samples simulating electromagnetic noises (dead pixels or parasite noise, respectively). Alternatively, “learnable methods” using ML models may be deployed to generate more complex transformations (e.g., simulating meteorological alterations such as snow).

In the case of absent or non-usable elements (for example “NaN” values), data imputation may also be performed, for example by deleting the features containing such an element, or by replacing the absent or non-usable features with statistical estimations of the appropriate values (mean, median, fixed value, etc.).

Both techniques work on the completeness and representativeness of the data set, though their assessment prior to implementing these strategies is not the usual process. Rather, data augmentation and imputation are usually used in a performance-driven iterative process where the improvement of the evaluation metric motivates additional augmentation or imputation effort.

As an example, (Setiawan et al., 2021) describe a Generative Adversarial Network (GAN) that generates augmented samples of sensor signals and quantifies their impact by measuring the improvement of the evaluation metric of their target system compared to the performance on the raw, non-augmented data set. This is an instance of a learnable method. GANs are also used for data augmentation of image data sets by (J. Lee et al., 2020) with the same quality assessment protocol,

[53] « non-learnable » and « learnable » methods are concepts developed in Sections 5.5.4.3 and 5.5.4.4

along with a visual assessment of the realism of the augmented samples. Similarly, (Abidin et al., 2018) benchmark several ML models for data imputation and measure their performance on the improvement they enable on the downstream classifier.

On the contrary, (Caiafa et al., 2020) apply decomposition and dimensional reduction (e.g. Principal Component Analysis or PCA) beforehand to get a better understanding of the weaknesses of the data set and orient the imputation and augmentation strategies.

Similarly, (Catania et al., 2022) also use PCA to observe the distribution of the samples in the subspace, considering that the more homogeneous the distribution, the more representative the data set. While this is a more structured process than previous works, it still lacks a formal protocol for the assessment of completeness or representativeness.

Another upstream approach is described by (Dourado Filho and Calumby, 2022) with a focus on computer vision. In this work, the authors distinguish 2 notions of class imbalance:

- Sampling imbalance is defined as the difference between the number of samples for each class, which is most commonly referred to as class imbalance.
- Content imbalance focuses on the difference between the number of samples of specific sub-types inside a given class.

Imbalance is characterised at the sub-type level by computing the entropy of the sub-types in each class. Then, sub-types are binned by entropy value, allowing the comparison of distributions for sub-types and classes. Data augmentation or imputation can be applied and a new round of computation is performed to assess improvements. A limitation of this method is the need to characterise sub-types, as it requires additional annotation efforts that are generally costly and sometimes hard or impossible outside image data.

Finally, (Osman et al., 2018) offers a survey on data imputation techniques, which are considered by the authors quick and easy to implement, although they note that feature deletion, by decreasing sample size, may affect latent correlations, and lead to reduced statistical power of the data, which is also highlighted by (Lu et al., 2021). Modern techniques may seem advantageous, due to the existence of many libraries and packages, ease of understanding and implementation. Moreover, some of them preserve sample size and statistical power. However, some processes are highly complex and require complex mathematical integration that may hinder understandability and control over the underlying factors. Users should then pay specific attention to the choice of data imputation technique, and find the most adapted trade-off to limit the impacts on completeness and representativeness. The authors of the paper do not explore further the way such trade-off can be attained.

4.4.8.4 Data synthesis

For complex task such as NLP or Computer Vision applications, data synthesis can be achieved using ML or rule-based simulators. Image and video data may be gathered from tools such as 3D simulators, as is the case in the aeronautics industry when gathering in-situ video feeds is prohibitively expensive. However, since it rests on the concept of generating samples from scratch, it requires additional precautions to ensure the degree of simulation (i.e., accuracy of the rules, quality of the generated images) is compatible with the ODD of the system, as synthetic data will never equate the complexity of real-life settings.

Oversampling (i.e., generating more samples in the underrepresented classes) may be done by data augmentation but is not always possible, for example when there is too little diversity in the underrepresented samples and overrepresented classes cannot be leveraged for augmentation strategies (because classes are exclusive, e.g., to classify different types of animals). In this case, another possibility to rebalance a class-imbalanced data set is under-sampling (i.e., deleting samples in overrepresented classes). However, under-sampling may also be problematic because it may reduce the statistical power of the data set (Osman et al., 2018). In such cases, data synthesis offers an alternative way of performing oversampling.

In their work, (Goodman et al., 2022) highlight the negative effects of under-sampling are especially visible if the number of samples in the majority class is modest in absolute value. From this observation, the authors favour oversampling using data synthesis, of which they identify 2 types:

- structural methods favour the synthesis of examples improving class separation;
- statistical methods aim at modelling the underlying per-class sample distribution to guide sample generation.

Their approach leverages the KL-divergence and the distance to k-nearest neighbours to generate synthetic samples. The idea is to integrate a structural and a statistical component in a single method. However, the generative process follows the distributions of the classes in the data set, so it is most useful on small-scale data set where representativeness has been ensured.

Regarding data imputation, (Santos et al., 2019) present a review of synthetic data generation techniques for missing sample values, according to the missing data mechanism involved: Missing Completely At Random (MCAR), Missing At Random (MAR), Missing Not At Random (MNAR). The survey distinguishes between univariate configurations (a single feature has missing values), and multivariate configurations (missing values are present in all features). The authors analyse the issues and restrictions for both configurations of the approaches meant to address a type of missing data mechanisms. Among the highlighted issues, they note that MCAR approaches may tend to produce different results according to the runs, leading to an important variability of the generated data sets. In addition, the authors point out the lack of investigation of the approaches in comparison with real-world data sets, which may limit the guarantee that data synthesis is a fully reliable approach to tackle missing data issues.

4.4.8.5 Data sampling

While the limitations of under-sampling (i.e., loss of statistical power) and oversampling (i.e., degradation of representativeness) have been mentioned, data sampling methods to ensure the completeness and representativeness of a data set have been proposed. As an example, in (Celis et al., 2016), the authors' aim is, from a large data set D1, to extract a smaller data set D2 that is as much diverse and fair (i.e. balanced) as possible. The paper distinguishes 2 kinds of diversity:

- Combinatorial diversity applies to low-dimensional categorical data. It is based on computing Shannon entropy on every subset of samples, where each given feature f has value v. Reusing the example developed for (Asudeh et al., 2019), it consists of computing Shannon entropy for every sample of Female vs. Male (and can be applied to smaller subsets such as White Females vs. Black Females, etc.). The intuition is that the larger the entropy, the more diverse the subset.
- Geometric diversity, on the other hand, is aimed at high-dimensional data such as images or texts. For a set of k-dimensional samples, the idea is to compute the squared volume of the k-dimensional parallelepiped formed by all samples in the data set. The larger the volume, the more diverse the data set.

A limitation of combinatorial diversity (as is the case for most methods discussed so far) is that it works mostly for the assessment of completeness and requires explicit labelling of the attributes that define it. These attributes might go beyond the task-related labels. Therefore, it requires an additional and potentially important annotation effort, with associated costs and delays. A similarly limited but complementary approach is proposed by (A. Wang et al., 2020), which specialises in processing images.

These methods combine sampling to identify the best-suited samples for a data imputation, augmentation or synthesis strategy. The approach discussed in (Blatchford et al., 2021) is based on the estimation of a confidence interval and the entropy for each sample in a continuous-valued data set. The computation of both the interval and the entropy is performed iteratively over the data set, a technique called progressive sampling, and used as separate performance metrics. Both metrics are combined in a Probably Close Enough (PCE) criterion, to analyse quality: as the criterion converges towards a set value, it is not necessary to add samples, as they would not bring more information. The authors report the creation of a data set with the same PCE value but only 1% of the data volume than their original data set. This technique is particularly useful to sample a small, representative data set from a very large data set in the perspective of prototyping and exploratory research (such small-scale data set being easier and quicker to compute on a variety of models while staying informative of the performance to expect).

Another distance-based contrastive method is introduced by (Mountrakis and Xi, 2013). In this work, the objective consists of comparing the representativeness of a “reference” data set D1 to another data set D2. The method can thus be applied to compare a subsampled data set, or two separate data sets (e.g., train and dev). The method is also model-agnostic and does not rely on labels. However, it is presented on multi-spectral images, i.e., RGB + NIR, and its portability to other data types, such as text, is not discussed. The Euclidean distance between the samples in the reference data set and each sample of the assessed data set is computed. The more neighbours an assessed sample has in a sphere of radius r, the more representative of this part of the space it is considered. A general metric is derived for the whole data set. The authors do not point an implementation of their method.

Another type of sampling strategy consists of weighting inputs. This strategy comes from the statistical survey domain and aims at rebalancing the representativeness of response data sets. For example, (Brubaker et al., 2021) use linear regression to estimate the weights of each input in a data set of respondents regarding COVID-19 surveys in Africa, at the household and individual levels. The authors use classic statistical representativeness methods applied to pre-COVID-19 surveys to weigh household-level samples from the COVID-19 surveys, which introduces a selection bias. To alleviate it, these results are combined using linear regression on the individual-level response.

Other work uses weighting as a sampling strategy, including (Kohut et al., 2012), which exploits reference demographic statistics to weight households (by size), response method (cell phone vs landline) and respondents (gender, age, etc). Additionally, (Macgregor et al., 2017) highlight the importance of confidence interval when using weighting strategies. Indeed, for a set confidence interval value, the more variability in the data set, the more samples will be required to reach the desired value, making a formal dimensioning factor for representativeness.

(Keskes et al., 2022) adopt a different strategy by devising a method to filter out samples from a data set to improve representativeness. Starting from a data set of electrocardiograms (ECG) labelled by two classes A and B, and of varying quality, they aim at identifying and eliminating the worse quality samples while preserving the initial distribution of classes. The approach is realised in 3 steps:

- training a classifier discriminating good and bad quality samples;
- benchmarking different resampling methods;
- selecting the method that yields the best result w.r.t the performance improvements obtained by the downstream classifier (working on the ECG).

The representativeness criterion is based on a quality metric defined as:

$$DC_{quality} = \frac{\mathrm{TP} + \mathrm{FP}}{\mathrm{TN} + \mathrm{FN}}$$

After a first round of classification, they keep only the positively predicted samples, redefining the representativeness metric:

$$DC_{classes} = \frac{TP_1 + FP_1}{TP_2 + FP_2}$$

where:

- TP_1: True Positive samples of C1 (samples correctly predicted as good quality).
- TP_2: True Positive samples of C2 (samples correctly predicted as good quality).
- FP_1: False Positive samples of C1 (predicted as good but actually bad quality samples).
- FP_2: False Positive samples of C2 (predicted as good but actually bad quality samples).

This metric is used as the objective function of the quality classifier. The benchmark is performed to find the best-suited target value.

Unstructured data are especially hard to process at most steps of designing an AI/ML application. (Paganelli et al., 2022) focus on text data and devise a sampling procedure in the perspective of data integration, i.e., when assembling a data set from different pre-existing sources. On the same theme, (Simão et al., 2015) assess the completeness of genetic data by comparing the length of the genes to the mean length of a larger group, and describe a framework to automate this task. Their method rests on modelling the distribution of word frequencies across the data sets and maintaining it in the integrated data set. For time series, (Anttila et al., 2012) use Moving Block Bootstrapping to select temporally representative samples. The method focuses on the temporal aspect and is independent of the distribution of the data.

4.4.8.6 **Labelling**

So far, many identified assessment methods rest on explicit meta-data, which can be included during a tagging phase, complementary to labelling. However, in this section, tagging will not be considered and the focus will be put on labelling as the process of annotating samples with ground truth. Additionally, in this chapter, "labelling" encompasses all the phases related to the annotation of a data set built from scratch (i.e., the design of the annotation guide, labelling) or retrieved from an existing source (i.e., by partially or completely reannotating the data or integrating several data sets with different annotation schemas). Despite the additional complexities and associated externalities, labelling may be a critical phase for assessing and monitoring the completeness and representativeness of a data set. During these phases, the data set can be monitored using tools such as Cleanlab[54] (Northcutt et al., 2021). Similarly (Sánchez et al., 2019) present a framework to explore the missing values in a data set, according to defined axes such as labels, features or time. Their case study highlights the importance of distinguishing the sample-level analysis from the class-level analysis.
Besides, annotation quality is estimated through the intra- and inter-annotator agreement rates. Intra-annotator agreement assesses the consistency of a given annotator when annotating the same sample multiple times, which is particularly interesting in subjective tasks such as sentiment analysis. On the other hand, inter-annotator agreement assesses the consensus of multiple annotators on a given sample. Both agreement metrics are useful to identify annotation difficulties at small scale, be it particularly bad annotators, or a particularly difficult class. Moreover, an overall low agreement may indicate a particularly difficult or ill-defined task. Annotator agreements are usually computed using Kappa's (Cohen Kappa for pairs of annotators, Fleisch Kappa for more), but can alternatively be computed on the same metric used for system evaluation, especially if one of the annotators may be considered particularly competent.

Overall, few methods have been found that assess data completeness during the labelling phase. A related problem has been identified in the knowledge base community, where the content of data is used to assess the completeness of the bases. For example, (Balaraman et al., 2018) propose a framework around the notion of relative completeness: the content of similar instances in the database is compared and scored. The score quantifies the relative completeness of the instance. Therefore, the framework rests on two components:

- A similarity function that uses the labels of each instance and the frequency of their properties to match similar instance.
- A scoring function for assessing the relative completeness of a set of instances, based on the number of common properties.

[54] https://github.com/cleanlab/cleanlab

(Issa et al., 2021) explore in more depth the matching and scoring of instances, by reviewing the literature of such approach applied to linked databases. They distinguish four aspects of completeness:

- Schema completeness;
- Property completeness;
- Population completeness;
- Interlinking completeness.

Each aspect is then quantified using simple ratios, in line with the work of (Heinrich et al., 2018). Though the four aspects discussed could be conceptualised in a more general ML context (with population completeness being seemingly the closest to the notion of completeness as discussed in this document), the approach is too specialised to be extended outside the domain of linked databases.

4.4.8.7 **Pre-processing**

As pre-processing transforms data and assuming the original data set has been curated for completeness and representativeness, the first consideration is to apply the same methods for curation after each pre-processing step. The remainder of this section describes pre-processing steps, i.e., transformations that will be used by the model that specifically aim at addressing completeness or representativeness.

For example, (Chehreghan and Ali Abbaspour, 2018) seek to transform cartographic data (from an Open Street map) to match a reference data better (from a national cartographic institute). In the first phase, pre-processing steps are applied to align both data sets: the formats and coordinates systems are harmonised, topographical errors are removed, and the maps are converted to a graph representation. Then, a training area is selected, in which topographical objects of interest are detected and an algorithm tries to match objects from the reference and candidate maps. This matching depends on two parameters that need to be optimised, so the matching operation is repeated for several combinations of values of both parameters. The performance of each combination is evaluated by computing the F-score of the matching, and the combination of values yielding the best F-score is selected for the next phase. In this second phase, these values are used to run the matching algorithm on the rest of the data, matching the rest of both maps. Finally, completeness of the matching is computed through a combination of similarity measures based on geometric properties such as object length, orientation, area, etc. Although this approach may be too distant from the context of the MLEAP project to be used as is in its next phases, the general methodology is presented to be used as higher-level indications of what pre-processing for the completeness and representativeness of the data set (and its assessment) could look like.

On text data, (Y. Hu et al., 2020) improve the completeness of a corpus of literary works by binning data in time groups of 5, 10 and 20 years and cutting off the size of the bin at the smallest one. Though they report satisfactory results for their application, this method basically rests on data deletion, which has already been discussed as a less-than-desirable solution. As in the previous work presented, the approach in itself should not be applied directly but could serve as inspiration for the development of more adapted methodologies.

Finally, no substantial work has been found that leverages feature engineering methods to improve or assess the completeness or representativeness, though (Almaimouni et al., 2018) use a combination of PCA and K-means to group and select the most salient features of the dataset for representative sample selection in the context of power system modelling. As for both previous methods, this work is presented for exhaustivity purposes and general ideas but is not exploitable for testing in the context of the MLEAP project.

4.4.8.8 Corner case and edge case detection for ML models

4.4.8.8.1 Prediction-based methods for the vision domain

Following the definition introduced in 1.2.4.2.2, prediction-based approaches can be found mainly at the scenario level. Typically, they predict a future frame and then compare it with the true frame to detect any anomalies. Thus, they can be trained in a supervised manner, assuming that all training samples are normal. Such a method has been applied by (Fingscheidt et al., 2019) specifically for the detection of corner cases in automated driving. Another approach predicts future images in videos using a generative adversarial network architecture while guaranteeing appearance and motion constraints (W. Liu et al., 2017).

Another prediction-based method relies on the notion of surprise adequacy (Kim et al., 2019; Ouyang et al., 2021), which can be used as a test adequacy tool. Surprise adequacy's initial property describes the surprise of testing data with respect to the training data, namely to describe difference/similarity between testing and training data (Kim et al., 2020, 2019; Kim and Yoo, 2020).

This section introduces a corner case detection method based on unpredictable situations and then summarizes methods based on surprise adequacy.

<ins>Corner Cases as Unpredictable Situations</ins>

In (Fingscheidt et al., 2019), the authors identify corner cases as technically unpredictable situations. However, it is important to note that not every unpredictable situation in the field of autonomous driving is necessarily a corner case. An aircraft that suddenly enters the camera image in the sky may not be predictable, but luckily in most cases it will be irrelevant for the driving task. Following (Fingscheidt et al., 2019), a corner case is detected if there is 1) a non-predictable, 2) a relevant object or class, and in 3) a relevant location.

More precisely:

2) The method described in (Fingscheidt et al., 2019) uses an image prediction component that gives us the prediction errors for each new image. Many autonomous driving systems already predicted trajectories of other traffic participants. To identify corner cases in video streams, it is essential to understand the underlying states and dynamics within the given situations. This high-level abstraction can be learned by predictive models. For the image prediction approach, we can train a model that receives n consecutive frames $x(t-1, t-n)$: $(x_{t-n}, x_{t-n+1}, \dots, x_{t-1})$ to compute a prediction x_t of the current frame. As a metric for the corner case, we may now calculate an error $e_t = x_t - x_t$, between the predicted image and the actual image x_t. The metrics of (Mathieu et al., 2015) can be used (see Figure 22).

3) This method described in (Fingscheidt et al., 2019) uses a semantic segmentation of the input frame that allows us to classify and localize the objects in the scene, with moving objects being considered as relevant. For the semantic segmentation, the training protocol from (Chen et al., n.d.) can be used.
4) Finally the method described in (Fingscheidt et al., 2019) needs a detection system that processes the information from both image prediction and semantic segmentation by information fusion, comprising a check, whether the non-predictable relevant class is in a relevant location. For the image prediction examples of models are the well-known PredNet (Lotter et al., 2016) and the network proposed by (Hasan et al., 2016).

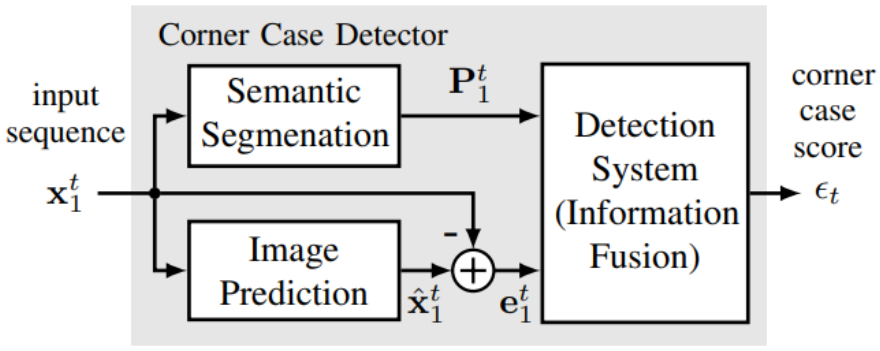


Figure 22. High-level block diagram of the corner case detector (borrowed from (Fingscheidt et al., 2019)).

<u>Corner Case Detection Based on Surprise Adequacy</u>

Following (Ouyang et al., 2021), a good way to evaluate surprise adequacy is to study neuron behaviour in terms of a given deep learning (DL) model. The authors emphasise that the more the diversity of neuron behaviours, the better the quality of testing data. For example, metrics for neuron coverage were proposed in (Pei et al., 2017), as well as for neurons' activation behaviours (Sun et al., 2018). The reader can refer to Section 4.3.9.2 for more. While compared with those metrics reflecting independent behaviours of testing and training sets, (Kim et al., 2019) an interesting idea was proposed to describe the difference between the testing set's behaviours and that of the whole training set.

Let $X = \{x_1, x_2, \dots\}$ be a set of inputs and let M be a trained DL model made of a set of neurons $N = \{n_1, n_2, \dots\}$. For a given testing data $x \in X$ and an ordered (sub)set of neurons $N \subseteq N$, the activation behaviour (namely activation trace) of x on N is expressed by the vector of activation values and it denoted as:

$$\alpha_N(x) = [a_1(x), \dots, a_N(x)]^T$$

where each element an(x) corresponds to the activation value of x with respect to an individual neuron n in N. Hence, the set of activation traces for X is denoted as $A_N(X) = \{\alpha_N(x) | x \in X\}$.

Then, $A_N(Tr)$ is calculated based on the training dataset Tr, which records neurons' activation behaviours on all samples in Tr. Similarly, the activation behaviour of testing data Te is also obtained as $A_N(Te)$. Finally, combining $A_N(Tr)$ and $A_N(Te)$, surprise adequacy (SA) is defined to describe the relative novelty of testing inputs with respect to the training data. It is actually denoted as the quantitative similarity measure between $A_N(Tr)$ and $A_N(Te)$:

SA = $SimilaryMeasure\big(A_N(Te), A_N(Tr)\big)$

Two kinds of similarity measurement are proposed in (Kim et al., 2019) to formalize SA, based on the likelihood-based SA (LSA), and on distance-based SA (DSA).

Improvements are proposed in (Ouyang et al., 2021; Tinghui Ouyang, 2021), where SA is applied for data description, especially for corner case data description. In addition, three kinds of modification on DSA definitions are developed. Finally, based on DSA, a novel corner case data detection method is proposed. Different from most corner case studies; the proposed method can be utilized as a tool for the recognition of corner case data.

The tool related above is:

- In python: dnn-tip 0.1[55].

Likelihood-based Surprise Adequacy

In (Kim et al., 2019) the authors uses the Kernel Density Estimation (KDE) to estimate the probability density of each activation value $A_N(T)$, and obtains the surprise of the new input with respect to the estimated density.

The KDE produces density function $\hat{f}$ as follows:

$$\hat{f}(x) = \frac{1}{|A_{N_L}(T)|} \sum_{x_i \in T} K_H \left(\alpha_{N_L}(x) - \alpha_{N_L}(x_i) \right)$$

With:

- N_L is part of N where N is set of neurons of the DL model
- H denotes the bandwidth matrix
- $K$is a Gaussian kernel function
- x is a new input

The LSA is defined to be the negative of the log of density:

$$LSA(x) = -log \left(\hat{f}(x) \right)$$

Distance-based Surprise Adequacy

In (Ouyang et al., 2021) the author introduce the Distance-based Surprise Adequacy (DSA) which uses the Euclidean distance for each novelty within the data test

$$DSA(x) = \frac{dist_a}{dist_b}$$

With:

- $dist_a = \|\alpha_N(x) - \alpha_N(x_a)\|$ where $x_a = \underset{D(x_i)=C_x}{argmax} \|\alpha_N(x) - \alpha_N(x_i)\|$

[55] https://pypi.org/project/dnn-tip/

- $dist_a = \|\alpha_N(x_a) - \alpha_N(x_b)\|$ where $x_b = \underset{D(x_i) \in CC_x\} \|\alpha_N(x) - \alpha_N(x_i)\|}{argmax}$
- $c_x \in C$ predicted class of the new input

This method uses different distance like Mahalanobis distance.

$$MDSA(x) = \sqrt{(\alpha_N(x) - \mu_T)^T S_T^{-1}(\alpha_N(x) - \mu_T)}$$

With:

- μ_T mean and S_T covariance matrix

The tool in the section is:

- In python: dnn-tip 0.1[56].

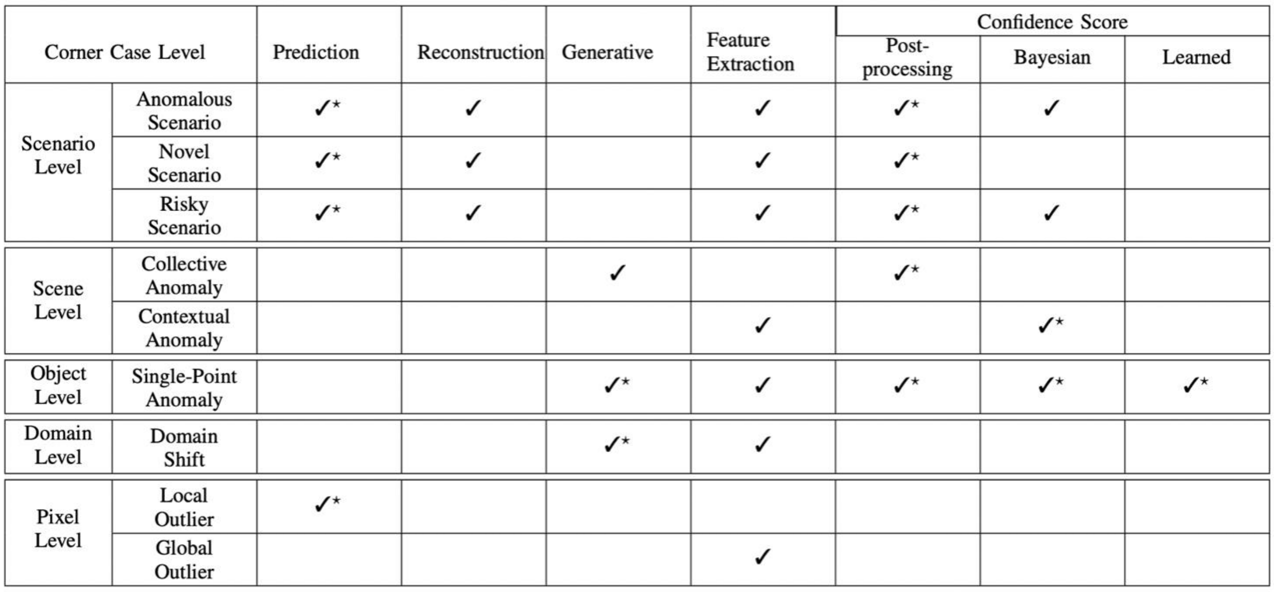
Summary of Methods Available

Corner case level	Methods available
Scenario level	• In the paper (Erdogan et al., 2019) the authors compare a rule-based: ○ Unsupervised clustering ○ Supervised deep learning approach for manoeuvre extraction on scenario level • Dangerous-driving classifiers for anomalous driving behaviour based on random forest and recurrent neural networks (Alvarez-Coello et al., 2019) • Long short-term memory and replicator neural networks (Matousek et al., 2019) • Reconstruction-based autoencoders trained on handcrafted spatio-temporal features and end-to-end implementations (Hasan et al., 2016)
Scene level	• Reconstruction-based method (Xia et al., 2015) • Learn normality with autoencoders (D. Gong et al., 2019) • Monte Carlo dropout (Gal and Ghahramani, 2016) • Bayesian SegNet uses Monte Carlo dropout for semantic segmentation (Kendall et al., 2015) • Deep ensembles for uncertainty estimation (Lakshminarayanan et al., 2017)
Object level	• Open-set recognition (Scheirer et al., 2013) • Stereo-based geometric modelling (Cordts et al., 2016) • A combination of object detection and segmentation (Pham et al., 2018)

[56] https://pypi.org/project/dnn-tip/

	• The other methods provide per-image scores (Lis et al., 2019)
Domain Level	• Measure of the domain gap between source and target distribution (Bolte et al., 2019) • Minimize cross-entropy-based metrics between the distribution (Dai and Van Gool, 2018; Zou et al., 2018) • H-divergence (Chen et al., 2018) • Wasserstein distance (Shen et al., 2017)
Pixel level	• Edge-adaptive method (An et al., 2007) • Estimation of rotation of optical flow (Buczko and Willert, 2017) • U-Net to extract features for a random forest classifier (Dong et al., 2019)

Figure 23 presents a summary of the section, where we denote, which type of method has been applied to detect which corner case level. A * symbol denotes the suggested approaches to detect corner cases on that level.



Corner Case Level		Prediction	Reconstruction	Generative	Feature Extraction	Confidence Score		
						Post-processing	Bayesian	Learned
Scenario Level	Anomalous Scenario	✓*	✓		✓	✓*	✓	
	Novel Scenario	✓*	✓		✓	✓*		
	Risky Scenario	✓*	✓		✓	✓*	✓	
Scene Level	Collective Anomaly			✓		✓*		
	Contextual Anomaly				✓		✓*	
Object Level	Single-Point Anomaly			✓*	✓	✓*	✓*	✓*
Domain Level	Domain Shift			✓*	✓			
Pixel Level	Local Outlier	✓*						
	Global Outlier				✓			

Figure 23 – The summary of approaches to detect corner case (Breitenstein et al., 2021).

4.4.8.8.2 Corner Cases in NLP

The methods for corner case detection discussed in the former sections are dedicated image and video streams. Indeed, most of the state of the art on the subject is dedicated to the kind of task associated to computed vision, especially with application to vehicles and driving. Nevertheless, corner cases do exist in other domains. For example, in NLP, consider a scenario where the CEO of a company states in an audio conference, "*Now investments will be made in Asia*". The system instead could transcribe, "*No investments will be made in Asia*". There is a meaningful difference in the implication of the two statements that could greatly influence the analysis and future direction of the company (Nourbakhsh and Bang, 2019). A few approaches have been proposed for corner case detection out of the vision domain. This section explores first the case of natural language processing (NLP) and then focuses on time series.

In the domain of NLP, some research focused on anomaly detection, for example in (Nourbakhsh and Bang, 2019). Unsupervised clustering methods have been applied to documents in order to identify outliers and emerging topics (Cheng, 2013). Deviation analysis has been applied to text in order to identify errors in spelling (Samanta and Chaudhuri, 2013) and tagging of documents (Eskin, 2000). Recent popularity of distributional semantics (Turney and Pantel, 2010) has led to further advances in semantic deviation analysis (Vecchi et al., 2011).

In (Nourbakhsh and Bang, 2019), the authors enumerate major applications of anomaly detection from text in the financial domain, and contextualize them within current research topics in Natural Language Processing. They lay out five perspectives on how textual anomaly detection can be applied in the context of finance:

1. **Anomaly as error:** Some studies have used anomaly detection to identify and correct errors in text (Eskin, 2000; Samanta and Chaudhuri, 2013). These are often unintentional errors that occur as a result of some form of data transfer, e.g., from audio to text, from image to text, or from one language to another.
2. **Anomaly as irregularity**: Anomaly in the semantic space might reflect irregularities that are intentional or emergent, signalling risky behaviour or phenomena. A sudden change in the tone and vocabulary of a company's leadership in their earnings calls or financial reports can signal risk (Nourbakhsh et al., 2017; Wendlandt et al., 2018; Zhao, 2017).
3. **Anomaly as novelty**: Anomaly can indicate a novel event or phenomenon that may or may not be risky. Breaking news stories often emerge as anomalous trends on social media. Novelty detection can also be used to detect emerging trends on social media, e.g. controversies that engulf various brands often start as small local events that are shared on social media and attract attention over a short period of time (Li et al., 2017; Liu et al., 2015; Nourbakhsh et al., 2015).
4. **Anomaly as semantic richness**: A large portion of text documents that analysts and researchers in the financial sectors consume have a regulatory nature. Annual financial reports, credit agreements are some of these types of documents. These documents can be tens or hundreds of pages long, and often include boilerplate language that the readers might need to skip or ignore in order to get to the "meat" of the content. Often, the abnormal clauses found in these documents are buried in standard text so as not to attract attention to the unique phrases (Sameena Shah and Sisk, n.d.).
5. **Anomaly as contextual relevance**: Certain types of documents include universal as well as context specific signals. As an example, consider a given company's financial reports. The reports may include standard financial metrics such as total revenue, net sales, net income, etc. In addition to these universal metrics, businesses often report their performance in terms of the performance of their operating segments. These segments can be business divisions, products, services, or regional operations. Since these segments are often specific to each business, supervised models that are trained on a diverse set of companies cannot capture them without over-fitting to certain companies. Instead, these segments can be treated as company-specific anomalies.

Let us note that unlike numeric data, text data is not directly machine-readable, and requires some form of transformation as a pre-processing step. A comprehensive survey of different technique available is presented in (Naseem et al., 2021). For example, in "bag-of-words" methods, this transformation can take place by assigning an index number to each word, and representing any block of text as an unordered set of these words (Howard and Ruder, 2018). In sentence boundary disambiguation methods (Palmer and Hearst, 1994) the text is split into sentences before being processed. In (Wu et al., 2016a) the authors propose another algorithm to process the sentences and the word within called WordPiece. Some methods focus on the representation at the document level, for example in (Cohan et al., 2020; Z. Liu et al., 2020). Or others, can operate in a semantic indexing space like in (Rosario, 2001; Sharma and Kumar, 2023).

The article (Kopf and Huh-Yoo, 2023) discusses the user-centred design approach in developing a voice monitoring system for occupational voice users (OVUs) to prevent voice disorders. It explores the current long-term voice monitoring systems available and their limitations, as well as the potential of biofeedback in VDP. The study demonstrates a UCD approach to designing an intuitive feedback display for OVUs, with the aim of creating a real-time VDP system to support proactive behaviour change for OVUs.

This article (Yan et al., 2011) proposes a new information retrieval model that takes into account similarity, popularity, and semantic granularity for domain-specific search. A concept-based computational model is developed to estimate the semantic granularity of documents, and the proposed model outperforms the similarity-based baseline in benchmark experiments. The study suggests that the proposed model resembles the implicit ranking functions exercised by humans, and its perceived relevance is significantly higher than that produced by a popular search engine for domain-specific search tasks.

This paper (Z. Shi et al., 2020) proposes the first robustness verification algorithm for transformers, which have complex self-attention layers that pose challenges for verification. The certified robustness bounds computed by the method are significantly tighter than those by naive interval bound propagation and shed light on interpreting transformers.

This paper (Wenqi Wang et al., 2019) presents a survey of adversarial techniques for generating adversarial texts in both English and Chinese characters and the corresponding defence methods, with the goal of inspiring future studies to develop more robust DNN-based text analysers against known and unknown adversarial techniques.

This paper (T. (Sherry) Wu et al., 2019) presents Errudite[57], an interactive tool for error analysis in NLP that codifies model and task agnostic principles, such as precisely defining error groups, analysing a large set of instances, and explicitly testing hypotheses about error causes. A user study shows that Errudite enables high-quality, reproducible error analysis with less effort and reveals ambiguities in prior error analysis practices.

In this section, the academic papers are referencing only prototypes that are not always available to be tested.

4.4.8.8.3 Corner cases in time series

As it is the case for NLP, little work has been done concerning corner case detection in time series. The most related work is about methods of outlier detection in medical diagnoses (Chrominski and Tkacz, 2010). The authors investigated several outlier detection methods listed hereafter:

[57] https://github.com/uwdata/errudite

- **Grubb's test**
    - Method: Grubb's Test is a test based on normal distribution, the effects of which are that the data analysed with this method should have normal distribution (Fallon A., n.d.).
    - Tools:
        - Python: from PyPI, outlier_utils
        - R: from outliers, grubbs.test
- **Dixon's test**
    - Method: Dixon's Test begins by organizing the data in an ascending order, the next step is to count some parameter Q (more details in (Chrominski and Tkacz, 2010)). When the calculated value of parameter Q is bigger than the critical value then it is possible to accept the data from the data set as an outlier (Fallon A., n.d.; J.R, 1999; Konieczka P., 2007).
    - Tools:
        - Python: outlier-detector 0.0.3
        - R: from outliers, dixon.test
- **Hampel's Test**
    - Method: For this test, one has to calculate the median Me for the whole data set and, next, to calculate the value of deviation r_i from the median value for each element i. When $|r_i| > 4.5Me_{ri}$, the deviation is greater than 4.5 times the median for the current deviation, the value from the data set can be accepted as an outlier (Ben-Gal, 2005; C, 2001).
    - Tools:
        - Python: pyhampel 0.3.7
- **Quartile Method**: In this method, one has to find the upper quartile Q_3 (75% of data in the data set are lower than this) and the lower quartile Q_1 (25% of data in the data set are higher than this). Values lower than $Q_1 - 1.5H$ and greater than $Q_3 + 1.5H$ can be considered as outlier, where $H = Q_3 - Q_1$ (Filzmoser, 2004).

4.4.8.8.4 Conclusion and Applicability

While corner and edge cases pose some serious challenges to the design and the validation of system, the current literature is still exploring new ways to tackle all of these challenges. Most of the current state of the art is currently focus on the topic of computer vision, and thus images and video streams corner and edge cases. This is in part pushed by the need to develop rapidly self-driving vehicles. The state of the art is thinner on other type of application (NLP and time series types of data) but still present some options. The current maturity of approaches is still mostly at the research level, some of them have allow some academic prototypes to be made publicly available. However, each academic tool allows to implement one approach tied to a specific research paper. No consolidation inside commercial tools or even through a standardization document had been done so far. This leaves the industry in a state where the state of the art is largely in flux, and can change in the near future, until some consolidation actions occurred. System designers can use these techniques (provided they can scale to industrial problems, which is not always the case) but would have to justify their choice under the subjective opinion they used.

Among the techniques available, test-based methods have (at least until now) the highest degree of generalisation to other use cases outside computer vision. They might be the most advanced in terms of software available and are likely to be usable at the industrial level in the short term.

However, the techniques related to prediction based methods and especially those studying unpredictable situation, are the closest to the (EASA, 2024) framework of ODD. Indeed, these techniques works under the assumptions that every scene or scenario can be decomposed in several domain (some of which contained in others). This decomposition is suitable under the requirement of EASA CP since it allows a better characterisation of the attributes of the OD and ODD, as well as a description of their distribution.

4.4.8.9 **Conclusions**

In general, and as is the case in Section 4.3, there exists a wide variety of methods integrating the assessment of completeness and representativeness into different processes of the data life cycle. However, these methods lack genericity, and their transferability (in terms of both technical feasibility and interest) to other use cases, even for the same task, is often unclear.

Nonetheless, the literature seems to indicate that well-proven tools such as dimension reduction are still relevant for today's massive data sets, even though this kind of upstream analysis tends to be forgotten in modern work. Moreover, as pointed out in (Zhang and Zhu, 2018) at the end of their review and in multiple works throughout this section, an intrinsic weakness of assessing representativeness, in particular using data sampling methods, is that it requires knowing the distribution of the phenomena of interest. This is generally hard to ascertain. Therefore, well-known, time-proven tools such as PCA and other appropriate variations (depending on the nature of the data) may be reasonable rules of thumb to gather as much knowledge of the data set's limitations as possible to outline expectations about the downstream model's performance, as hinted by (Caiafa et al., 2020) and (Catania et al., 2022).

Data improvement methods also appear to be an interesting way to tackle completeness and representativeness. Indeed, despite the limitations inherent to data synthesis and the specific weaknesses of the methods, (Salamon et al., 2017) highlight that even though synthesized data may not reflect real-world phenomena and thus allow reliable conclusions of the behaviour to expect from the model in its operational context, data synthesis allows for detailed and controlled evaluation. Moreover, data synthesis enables the creation of vast amounts of data that, by sheer number, may provide insight on the performance of different models.

Finally, in highly constrained contexts such as using data for an AI/ML application requiring high level of data trustworthiness with only few sources available, data augmentation and data synthesis may be central to increase completeness and representativeness. While it shows the developments of such methods should be encouraged, the ability to evaluate their quality (accuracy, consistency, etc.) must remain an important concern.

4.5 Relation to other data quality requirements

4.5.1 Introduction

The literature extensively highlights the influence of certain quality attributes on each other. In its current state, the standard on data quality for AI and analytics (ISO/IEC CD 5259-2, 202X) mentions the existence of conflicts between data quality attributes. Therefore, trade-offs must be made to estimate which attributes should prevail in the context of use.

This section lists and analyses the relationships as follows:

- A definition of the attribute and, when existing, methods for computing the value of the attribute.
- A description of how the attribute (representativeness, completeness, or both) is impacted.
- The existence of methods and tools for controlling this impact.
- A summary of the importance of the impact and the extent to which this can be controlled (depending, for example, on the maturity of the methods).

Coupled with an analysis of the functional requirements, this analysis shall provide the baseline for the definition of a strategy for establishing DQRs that encompasses, in an extensive manner, the whole environment of the assessment.

4.5.2 Balance

4.5.2.1 **Definition and assessment of the value**

The standard (ISO/IEC CD 5259-2, 202X) defines balance as the distribution of the data samples for all dimensions of the data set. This attribute should be checked on training, validation and test data.

The assessment of balance can be performed, according to (ISO/IEC CD 5259-2, 202X), by computing the reciprocal of the maximal ratio of the difference of a feature F impacting ML performance of a sample s, over the averaged value of the parameter for all the samples of a data set D.

As an example, the balance on the feature of brightness of images can be computed with:

$$X_F = \mathrm{MAX}\left(\frac{|A_s - B_D|}{B_D}\right)^{-1}$$

where A_s is the brightness of an image sample and B_D the averaged brightness for all the samples of the data set D.

The features that impact ML performance should be defined according to business logic and expert knowledge of the domain of application of the ML system. (ISO/IEC CD 5259-2, 202X) uses the example of image data sets for an ML application, which may require the exploration of features such as brightness, resolution, size of the categories, bounding box height to width ratio, bounding box area, and category bounding box area (averaged bounding box area of the samples in a category over the averaged area of all the samples in the data set).

As depicted in (Yu, 2021), balance can also be assessed through a comparison between the performance of the model on the whole data set (through for example accuracy and F1 score) and the performance obtained on the categories of samples representing “minority classes” (samples for which the absence or underrepresentation of values of sensitive attributes[58] may impact the fairness of the ML system). The comparison relies on nonparametric null-hypothesis significance testing (Mann–Whitney U test) and nonparametric effect size testing (Cliff’s delta). Performance is considered significantly different if the null hypothesis is rejected in the Mann–Whitney U test and the effect size in Cliff’s delta is medium or large.

4.5.2.2 Influence on completeness and representativeness

The balance data quality attribute, as described in (ISO/IEC CD 5259-2, 202X), encourages a homogeneous distribution of the values of the data items influencing the performance of the ML model. This approach allows,, in particular,, limiting the risk of biases in the data set and may present benefits for safety since edge cases are expected to be present in the data set in significant proportions and may contribute to the fairness of the system by limiting the risk of underrepresentation of certain populations. However, a balanced distribution of phenomena directly impacts the representativeness of the data set since representativeness implies that the data set matches the statistical distribution of the phenomena in operational conditions.

4.5.2.3 Methods and tools for assessment

Balance and representativeness are two attributes that seem to coexist in ML studies, but only certain aspects of the attributes are covered, for example, by ensuring balance on specific aspects (such as gender) and ensuring representativeness of other aspects. Several studies address the effect of balance on the performance of the models without considering the articulation with representativeness - for example, (Kumar et al., 2021), (Yu, 2021), (Leavy, 2018). In sociology, (Dickinson et al., 2012) address the relationship between balance and representativeness. They take the example of a survey conducted in a predominantly female company that will naturally be unbalanced in terms of gender, but representative of the target population. Additional male samples could be collected to reach balance or female samples could be deleted, but at the expense of representativeness. The study suggests relying on power analysis to determine the minimum male sample size needed for an appropriate observation of interactions between gender and the independent variables. The authors also suggest using techniques such as bootstrapping to generate standard error estimates not relying on parametric assumptions, which may increase statistical power; the authors note that such technique should only be used on variables presenting normal distributions. The study highlights that reaching a trade-off between balance and representativeness requires subtle exploration of the data and that impacts on either of the attributes should be documented by the data analyst.

Many works on ML argue about the importance of balanced data, regardless of the AI field or industry sector. For example, (Bilgic et al., 2021) offers a bibliographical analysis in AI for surgical education

[58] In the study (Yu, 2021), “sensitive attributes” are the features that may impact the suitability or acceptability of the model (sex, age, work experience, etc.).

that highlights unbalanced data as one of the limiting factors for the development or implementation of efficient AI in the sector. (Leavy, 2018) highlights the importance of data balance for the specific factor of gender, in all types of ML applications. (Zhang and Zhou, 2019) addresses unbalanced data in the context of fairness assessment in financial industry. (de la Fuente Garcia et al., 2020), in the context of a study on the collection of language data samples for Alzheimer's diagnosis, identifies data balance as one of the factors for the selection of data for AI systems. However, the bibliographic survey did not uncover a reference presenting how to attain balanced data for all influencing factors while still maintaining representativeness.

4.5.2.4 **Impact and observability**

The decisions made to enhance balance may strongly impact representativeness, which could lead to a redistribution of the items in some classes. The analysis did not reveal any standard methods to manage the articulation between the two attributes in the context of machine learning. Control should be based on expert analysis, and only a trade-off can be obtained.

4.5.3 Relevance

4.5.3.1 **Definition and assessment of the value**

According to (ISO/IEC CD 5259-2, 202X), relevance represents the degree to which a data set is suitable for a given context – the notion is refined by explaining that all features of the data used for training are good predictors. This attribute should be checked on training, validation, test and production data.

(ISO/IEC CD 5259-2, 202X) proposes to compute relevance through feature relevance and record relevance. Feature relevance consists of analysing the ratio of the number of relevant features in a data set (A) over the total number of features (B), thus $X = A/B$. One should note that this method implies that the "relevant" features have been identified. This can be done through statistical testing to identify the correlations between a feature and the outputs of the model, combined with expert analysis of the feature – for example, the weight of an individual should not be taken into account for a credit grant. The identification and determination of relevant features thus do not rely on a systematic and formal approach, which may hinder the ability to discriminate irrelevant features exhaustively. Record relevance represents the ratio of relevant data records[59] (A) over the total number of records in the data set (B), thus $X = A/B$. This approach requires determining what "relevant" data records are, but no specific method for their identification is proposed in the standard.

The assessment of data relevance relies mostly on expert analysis of the data in order to highlight what is relevant for the application. Studies such as (Van Vleck et al., 2007) present a study in a clinical context meant to determine, through structured interviews, the sentences and topics in the medical records that are perceived to be relevant for describing the patient's medical history. This approach allows leveraging the human's broad understanding of the situation when labelling the data. Although the authors highlight the relevance of the use of the output labelled data in ML, since it spots both

[59] "**Data record –** *set of related data items treated as a unit*" (ISO/IEC CD 5259-2, 202X)

relevant data and features (tagged as topics in the study), the cost of proceeding to a systematic review of data sets would not be realistic. In the domain of big data, (Doku et al., 2019) explores the determination of relevant data in an approach that relies on the voluntary actions from users who save on their mobile devices the data they find relevant to a specific domain of interest (sports, stock exchange, etc.). Topic Modelling, an NLP approach, is then applied in order to extract the abstract topics from the data. The topics are shared between all the members of the group belonging to this domain of interest, where a federated learning model coupled to blockchain updates its parameters based on the inputs received from all the users, in order to retain only relevant data. This approach, while having the advantage of limiting the need for human expertise (only a part of the whole data set is analysed by a human), is not a method for data relevance quantification *per se*. Moreover, this mode of operation cannot be generalised to all configurations of AI use, since it requires a vast quantity of different users to generate and combine data.

4.5.3.2 **Influence on completeness and representativeness**

In its section on relevance, (ISO/IEC CD 5259-2, 202X) indicates a tenuous link between relevance, and completeness and representativeness, in the sense that the assessment of relevance is pertinent only on data that have been verified on several other attributes (among which these two attributes).

(EASA, 2024) also indirectly highlights this link in section 3.3.2 Data collection, where the document notes that “data collection should identify the different sources of data of relevance to the training” and that in case of lack of completeness or representativeness, the data may be augmented. However, the document does not warn about some pitfalls linked to the subtle balance between the three attributes, and that a trade-off must be found.

The curation of a data set to increase its relevance impacts data dimensionality, which may, in turn, affect completeness and representativeness. Indeed, since relevance enhancement may lead to the deletion of features, this may lead to an inconsistency of the data set features with the input space, thus affecting completeness, which would in turn lead to a loss of information that would be detrimental to representativeness.

Although the ISO/IEC 5259-2 standard seems to recommend that relevance be analysed on data that has been validated in terms of completeness and representativeness, the strategy for managing irrelevant features without invalidating the decisions made to ensure completeness and representativeness is not straightforward. On the other side of the spectrum, one should be aware of the “Curse of Dimensionality”: the enhancement of completeness and representativeness, by providing additional dimensions to attain the precision of the input space and functional requirements, may lead to an increase in the complexity of the data that may result in performance limitation. Indeed, as the number of dimensions increases, data points tend to become equidistant in the resulting space, as there is always a neighbour matching a given subset of dimensions. The estimation of data relevance should be based on a wise trade-off in the selection of the required dimensions: enough dimensions so as to fulfil completeness and representativeness requirements, but not too many in order to both match relevance requirements and avoid high-dimensional data “curse”.

4.5.3.3 **Methods and tools for assessment**

Literature is scarce about methods dealing with relevance. The current lack of consensual and internationally validated definitions tends to reinforce the confusion about the scope of each data quality characteristic. For example, (Yang et al., 2018) study the "relevance" of training data sets through indicators of model accuracy, which suggests that the term is used with the general intent of checking that it is "fit-for-purpose".

Given the links explained earlier between relevance and completeness/representativeness, and the implications in terms of data dimensionality, two types of methods may enable an estimation of the above-mentioned trade-off:

- First, using Explainable Artificial Intelligence (XAI) solutions (Arrieta et al., 2020). As an example, decision trees can allow spotting features that are not relevant predictors. An expert comparison between the resulting model-relevant features and the functional requirements may lead to a redefinition of the most suited dimensions for the data set.
- Second, the CoD could be prevented or limited through such approaches:
    - For deep neural networks: (Poggio et al., 2017b) summarise theorems highlighting why compositional functions may prevent CoD;
    - For classifiers: (Baggenstoss, 2004) offers a probabilistic method allowing to build classifiers without a common feature space, hence avoiding the CoD;
    - For filtering algorithms: (Surace et al., 2019) offer a feedback particle filter based on optimal feedback control that circumvents the use of importance weights;
    - For high-dimensional nonlinear non-parametric systems: a method can consist in averaging derivatives and relying on one-dimensional estimates of the density function and its derivative (Bai et al., 2019).

4.5.3.4 **Impact and observability**

Enhancing the relevance of the data may impact both representativeness and completeness, and the data scientist should find a trade-off. The analysis did not reveal any standard methods to achieve an appropriate balance between the three attributes, and the methods offered here, at the academic research level, are only meant to control some aspects of the issue. Control should be based on expert analysis and only a trade-off can be obtained.

4.5.4 Diversity

4.5.4.1 **Definition and assessment of the value**

4.5.4.1.1 Diversity as "discriminative power"

The standard (ISO/IEC CD 5259-2, 202X) introduces diversity as an attribute that reflects to what extent the elements in the sample are different from each other. In (Z. Gong et al., 2019), data diversity refers to training data that "*provide more discriminative information for the model*". In (Zeng et al., 2021), data diversity is defined as "*the difference between data samples*" meant to characterize the usefulness of data samples. In the rest of the chapter, this notion refers to "diversity (discriminative power)".

Data diversity (discriminative power) can be measured by the Euclidian distance between data samples. For any data samples s_1 and s_2, their diversity can be computed by $d^2(s_1, s_2) = ||s_2 - s_1||_2^2$ (Zeng et al., 2021). (ISO/IEC CD 5259-2, 202X) proposes to compute data diversity through the indicators of label richness, relative label abundance and component richness. Label richness is the number of different labels in a data set. Relative label abundance is the ratio of the number of individual data samples having a similar label over the total number of samples in the data set. Component richness is the number of different trends in time series.

4.5.4.1.2 Diversity as "absence of non-representative sampling bias"

Another trend of studies depicts diversity as a lever for reducing bias, including ensuring appropriate representation of demographic groups. For example, the study (Leavy, 2018) promotes data diversity to reduce gender bias in machine learning. The *Assessment List for Trustworthy Artificial Intelligence (ALTAI)* delivered in 2020 by the High-Level Expert Group on Artificial Intelligence commissioned by the European Commission (HLEG, 2020) mentions the respect for diversity, non-discrimination and fairness in all stages of the AI system's life cycle, which entails the avoidance of unfair bias, the search for accessibility and universal design, and the participation of stakeholders in the design. The ALTAI does not provide a definition of the notion of diversity but highlights a link between diversity and absence of bias, which is named "non-representative sampling bias" in the ISO/IEC standard on AI bias (ISO/IEC TR 24027, 2021). In order to distinguish this notion from diversity in the "discriminative power" sense, this acceptation will refer to "diversity (absence of non-representative sampling bias)" in the following of the document.

The absence of non-representative sampling bias can be verified by performing a comparison between the performance of the model on the whole data set and the performance obtained on the categories of samples of each identified demographic group (BSA, 2021). This study offers a framework for AI bias risk management that recommends, at the stage of data acquisition, to compare the demographic distribution of the training data to the target population in the operational context, and verify that subgroups of populations are sufficiently represented. At the verification and validation stages, the framework recommends testing for bias by estimating the error rates across the identified demographic groups. The approach presented here does not refer to explicit techniques for the identification and quantification of the elements of interest, but it provides however highly relevant pointers in view of performing a risk assessment based on expert knowledge – which is a valid approach in the sense of the IEC standard on risk assessment techniques (IEC 31010, 2019). AI bias risk management can then constitute an adequate method for ML designers in order to limit the emergence of risks of non-representative sampling bias. The standard (ISO/IEC DIS 42001, 202X) provides a methodological approach to addressing management systems in the context of AI products and tackles the question of AI risk management. The standard requires the identification of risks of different nature, including risks for health and safety, but also risks of biases and absence of respect for human values. However, although the standard considers the importance of data resources, it does not go into detail about the quality requirements of the data.

The ALTAI (HLEG, 2020) offers a list of questions for self-assessment that is expected to drive expert analysis of AI systems and data sets in view of ensuring diversity. The document does not offer methods to solve the issues raised, but a list of checkpoints meant to raise awareness on the topics that need to be covered. For example, the document requires that adapted procedures to avoid biases in the use of input data are chosen and applied by the ML designer, that the diversity of end-users and subjects is characterized, and that the system's behaviour, when facing data related to problematic use cases, is tested and monitored. Several questions overlap with other notions covered by the chapter, such as fairness in general, inclusivity or representativeness. One can consider that addressing all the questions presented in the chapter may ensure that the topic of diversity is covered.

The standard (ISO/IEC TR 24027, 2021) presents several steps to follow for the identification and assessment of non-representative sampling bias, including the determination of relevant demographic characteristics, the selection of adapted "*fairness metrics to be used in detecting bias*", and the definition of acceptable margin of difference. The data can then be divided according to the values of the identified characteristics and compared with each other on the basis of the selected metrics and fixed differences. Although the standard does not provide details of adequate metrics, one can consider performance a relevant indicator in certain contexts of application (the system must work for all populations).

4.5.4.1.3 Other acceptation of diversity

The term "diversity" is also sometimes treated in a general way and not as a quality attribute in its own right. For example, (Zhan et al., 2021), a work on benchmarking for Pool-based Active Learning, uses the term as a synonym for the expression "representative sampling", and diversity is rather used in its vernacular acceptation. This acceptation of the term diversity is discarded in the remaining parts of the chapter.

4.5.4.2 **Influence on completeness and representativeness**

The standard (ISO/IEC CD 5259-2, 202X) only notes that diversity is closely related to the notions of representativeness and balance. Understandably, the quest for diversity has an impact on representativeness – a sample that is diverse enough to be discriminating may not be a true representation of the target population.

As for diversity (absence of non-representative bias), the study (Leavy, 2018) promotes data diversity to reduce gender bias in machine learning; however, it does not tackle the issue of representativeness. The ALTAI (HLEG, 2020) highlights a relationship between diversity and representativeness ("*Did you consider diversity and representativeness of end-users and/or subjects in the data?*"), but without providing an explanation of the nuance or the nature of the link between the two notions. However, one can easily understand that ensuring the absence of non-representative biases may positively impact representativeness.

4.5.4.3 **Methods and tools for assessment**

In the context of diversity as "discriminative power", the relationship with representativeness is hardly explored by the scientific community. For example, in their overview of diversity in the context of machine learning, (Z. Gong et al., 2019) address the importance of data diversification, for which they present several methods in supervised and unsupervised learning, and in active learning. However, they only scratch the surface of the impact on representativeness or completeness of the data sets. (Hyontai, 2018) provides an analysis of the impact of data diversity on machine learning performance, without mentioning the impact on representativeness or completeness.

In the case of diversity as "absence of non-representative sampling bias", since controlling diversity (absence of non-representative bias) may positively impact representativeness, the methods for diversity assessment presented in Section 4.5.4.1 may be applied.

4.5.4.4 **Impact and observability**

Data diversity is defined in terms of the discriminative power of the data, and the quest for diversity may alter completeness and representativeness. However, in this acceptance, the literature does not offer methods to address this relationship. In other trends relative to AI fairness, data diversity is presented as a crucial topic for non-representative sampling bias limitation. Due to these differences in acceptation of the notion, the topic of data diversity seems still quite immature, and one can only understand a remote link between diversity, AI bias and representativeness of the data set. In this context, the identification and assessment of non-representative sampling bias may contribute to enhancing representativeness.

4.5.5 Currentness

4.5.5.1 **Definition and assessment of the value**

According to the SQuaRE standard (ISO/IEC 25012, 2008), "*The degree to which data has attributes that are of the right age in a specific context of use*", with "*context of use*" defined as "*users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used*";

The assessment of the currentness of data may be performed by establishing some preliminary statements about the data composing the data set, which relies heavily on an expert analysis of the data items. (Iphar et al., 2015) note that a distinction should be made according to the likelihood of the data changing over time: data evolving by nature, data likely to evolve, data that may change, data unlikely to change, and permanent data. The identification of such types of data can allow computing a probability that the data remains up-to-date in a certain time range, which can lead to an estimation of the data set's currentness.

The standard (ISO/IEC CD 5259-2, 202X) suggests two indicators for data currentness: feature currentness and record currentness. Both indicators are ratios of elements that fall within a required age range. The standard notes, through examples, that the required age range should be defined based on expert analysis of the data set and the intended application. Feature currentness is described as $X = A/B$, with A is the number of data samples for a specific feature that fall within the required age range, and B the total number of data samples for the feature. Record currentness is defined by

$X = A/B$, A is the number of data records[60] that fall within the required age range, and B the total number of data records in the data set.

Literature highlights some conceptual relation between the attribute of timeliness and currentness, in the sense that the two attributes sometimes overlap in their definition. For example, (Juddoo, 2015),in an overview of data quality attributes for Big Data, summarises different trends of research that offer contradictory definitions of timeliness: "*[e]xtent to which data is sufficiently up-to-date*" and "*[e]xpresses how current the data is for the task at hand; involves currency measurement and check whether data is available before planned usage time*". The first definition is quite general and may be equivalent to the ISO/IEC 25012 definition of currentness. However, the second definition is rather linked to the time when the data recorded is available for use by the system. Also in the context of Big Data, (Jesmeen et al., 2018) seem to refer to timeliness – yet without offering a strict definition – as relative to the availability for use. In a paper on data quality for MLOps, (Renggli et al., 2021) define timeliness as "*the extent to which data are up-to-date for a task*", which seems similar to ISO/IEC 25012 definition of currentness, and considers timeliness as a synonym for "*currency*" and "*volatility*". The current absence of homogeneity of definitions in the domain should involve, before selecting works as reference, ensuring the appropriateness of the concept under exploration.

4.5.5.2 **Influence on completeness and representativeness**

The standard (ISO/IEC CD 5259-2, 202X) highlights a potential relationship between representativeness and currentness. Indeed, due to the evolving nature of certain types of data, the distribution in a data set collected at a certain point in time may not correspond to the system's intended conditions of operation, which would directly impact the representativeness of the final data set.

Many studies mention the importance of data currentness as a contributor to data quality for machine learning (Siebert et al., 2020), (Frye and Schmitt, 2020), (Challa et al., 2020), (Nitesh Varma Rudraraju and Varun Boyanapally, 2019), and no study could be found that warned of any potential conflict between representativeness and operations performed in view of enhancing currentness.

Ensuring the currentness of a data set may thus positively contribute to representativeness.

4.5.5.3 **Methods and tools for assessment**

The assessment of the data set's currentness can be considered an additional tool to ensure its representativeness.

4.5.5.4 **Impact and observability**

Currentness seems to be a prerequisite to representativeness rather than a conflicting attribute. The currentness of the data sets should be assessed along with their representativeness.

[60] "**Data record –** *set of related data items treated as a unit*" (ISO/IEC CD 5259-2, 202X)

4.5.6 Other aspects of data quality

The analysis revealed a number of other attributes that could potentially present a link with the assessment of the attributes of completeness and representativeness. However, these attributes are either only marginally explored by the community and their lack of maturity makes it difficult to confidently link these elements to precise notions, or the relationship only pertains to highly specific domains of study. These attributes are cited in this paper in order to be comprehensive about other data quality attributes that may impact, but only aim at designing an informed view of the situation.

- **Data fidelity** is described in (ISO/IEC CD 5259-2, 202X) as a group of data quality attributes including (among other attributes) representativeness, completeness and balance. The exploration of the topic is still under discussion in the standardisation committee, but it seems to refer to all attributes that may be affected by data sampling. No scientific publication could be found to explore this aspect further, but such a trend could provide interesting tracks to follow for the determination of the best trade-off among all the attributes' constraints.

- **Data availability** is seen as an important lever for completeness. Data availability is defined in (ISO/IEC 25012, 2008) as "*The degree to which data has attributes that enable it to be retrieved by authorized users and/or applications in a specific context of use*". The standard (ISO/IEC CD 5259-2, 202X) dedicated to AI, however, does not include this concept in its list of attributes. This aspect is mentioned for example in (C. Liu et al., 2017) (healthcare, not ML), and (Nobles et al., 2015) (big data) notes that "*data quality measure of availability is dependent on completeness and consistency*" and that the "[l]*ack of completeness was the largest contributor to reduced availability of data*". Some techniques can be applied to enhance data availability, as suggested by (Willemink et al., 2020) (ML): federated learning, and interactive and synoptic reporting; however, the cost and effort associated with data preparation may present a barrier. The lack of studies about the interdependence between completeness and availability leads to the latter being mentioned as an attribute of interest, without providing specific instructions for the management of the articulation of the two attributes.

- **Data integrity** is mentioned in the standard (ISO/IEC 27000, 2018) on information security management systems as the "[p]*roperty of accuracy and completeness*". Integrity is also mentioned in the SQuaRE standard (ISO/IEC 25012, 2008), with the definition "*property of safeguarding the accuracy and completeness of assets*", but the concept is only used to describe the data quality attribute of confidentiality (without any explicit links to completeness). The standard on data quality for ML (ISO/IEC CD 5259-2, 202X) does not mention integrity in its present state. Some bibliographical references on the topic of outsourcing data to the cloud confirm a relationship between integrity and completeness (Niaz and Saake, 2015), (Zhou et al., 2018). Although literature highlights a relationship, this remains confined to the domain of information security management.

4.5.7 Conclusion

Ensuring all data quality attributes of a data set may strongly impact completeness and representativeness. Figure 24 summarizes the interdependences highlighted by the literature and how they may affect the two attributes.

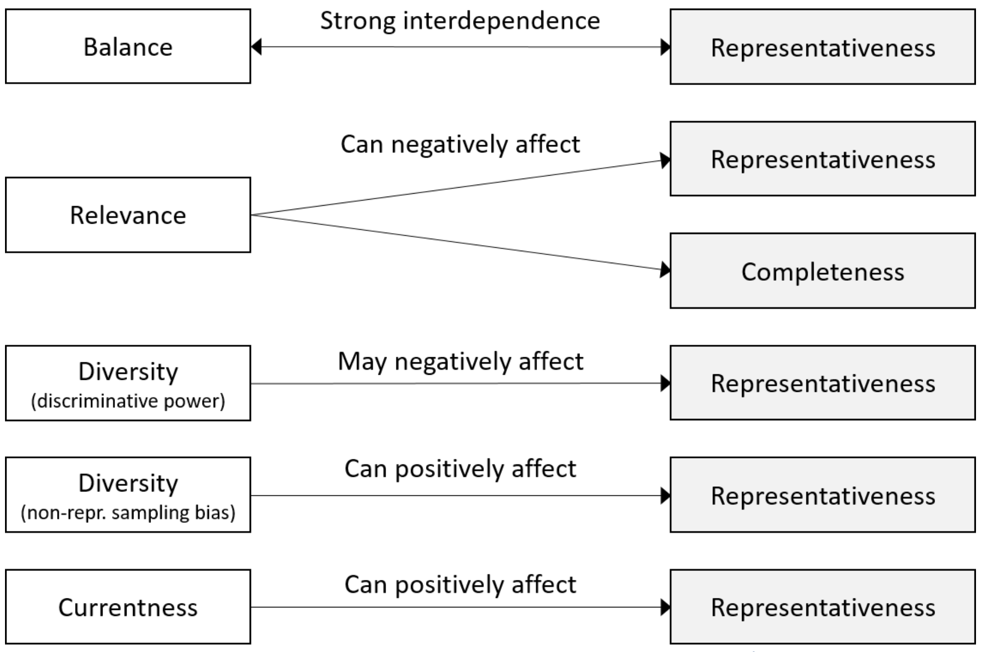


Figure 24. Interdependence between other DQRs and data completeness/representativeness.

Especially for balance, relevance and diversity, their enhancement encompasses operations meant to modify the volume and nature of the data samples or their features. This means that, in essence, the attributes operate on the same type of entity as completeness and representativeness, which may weaken the overall quality of the data used in the context of ML and supervised learning.

The main findings from the literature are that there is no standard method to reach a perfect state where all attributes may be respected. Overall, literature seems too often discard the challenges related to the conflicts between the quality attributes; the few papers dealing with the challenges tend to recommend an expert trade-off, which means that depending on the context of use, technical constraints or business logic, human analysis may lead to estimating what can be an acceptable level of quality. However, as mentioned previously, no guidance for a comprehensive analysis is offered. In addition to this lack of tools, literature shows an absence of homogeneity in the definitions and the nature of the concepts under study, which can impede the search for approaches to tackle the problem. Exploratory works should be performed, along with the determination of the best strategy for the implementation of a trade-off.

4.6 Selection grid

4.6.1 Introduction

This section offers an operational synthesis of the methods discussed in the previous sections. The first table recapitulates all the methods presented, along with a recommendation on whether they should be integrated into the selection grid (and thus tested in the future phases of the MLEAP project). The grid presents, for each selected method, the type of data it could be applied on, and to which factor of influence it relates.

Then, a series of tables summarize the document's discussions: for all factors of influence identified and detailed in the previous sections, their impact on data completeness or representativeness, as well as the existence of tools for assessing or controlling this impact, is summarized.

These last tables allow a quick overview of two elements of information:

- Understanding what type of activity[61] may impact the completeness and/or representativeness of data, to ensure that future DQRs on completeness and representativeness contain all the required checkpoints (relative to technical requirements, processes, and other data quality properties requirements);
- Estimating to what extent methods and tools may allow controlling this impact. This information would temper the selection and definition of new DQRs. It seems reasonable to envision that, in contexts where no tools or methods seem to exist, DQRs may be presented as attention points without specific DQRs being formally prescribed.

The three overview tables are presented as follows:

- Factor: the factor of influence considered.
- Impacted property: the property (**C**ompleteness and/or **R**epresentativeness) that is impacted by this influencing factor.
- Impact of the factor on C/R: the nature of the impact this factor has on **C**ompleteness and/or **R**epresentativeness.
- Impact of C/R enhancement: can the manipulations performed in order to enhance **C**ompleteness and/or **R**epresentativeness have an impact on the factor of influence? This may mean that the ML designer needs to perform trade-offs between respecting the requirements of the factor of influence and the completeness or representativeness of the data set.
- Externalities: a consequence of the factor on the data set (its quality, nature or content) or on other requirements to improve the data set quality.
- Tools/methods: a summary of the conclusions on the availability of tools and methods related to the factor. These conclusions are drawn in a general sense. The cited papers may not correspond to the selected works, and work that requires exploratory testing was not considered substantial enough to change a conclusion stating that no method stood out during the analysis.

[61] The factors of influence are all relative to an activity performed in the context of ML or data engineering (specification of the ODD, data quality enhancement, etc.).

4.6.2 Methods summary

Bibliographical references	Document references	Decision
(Mani et al., 2019)	4.3.9.1.1	Selected. The four metrics will probably be ponderously explored, as strict Equivalence partitioning may be too strong a condition.
(Tae and Whang, 2021)	4.3.9.1.2	Selected. The extension of Slice Tuner to unstructured data is unclear and could be explored.
(Pei et al., 2017)	4.3.9.1.3	Selected. Neuron coverage will be explored as a complementary observation tool but is not expected to be a strong solution on its own.
(Lei et al., 2018)		Discarded. One method based on neuron coverage is enough, and the previous one is simpler.
(Kiela et al., 2021), (Thrush et al., 2022)	4.3.9.1.4	Discarded. The adversarial examples are too narrow a problem, and manually adding samples is too tedious to explore the solution.
(Raghu et al., 2017)	4.3.9.2	Discarded. The method might be too tedious to deploy w.r.t its interest for representativeness and completeness assessment. It is preferable to allocate more time to ensure in-depth work on other methods.
(Almeida and Vieira, 2011)		Selected. The approach is simple but robust and will, at minimum, be a viable baseline.
(Sáez et al., 2016)		Discarded. The method is interesting but external assessment tools will be preferred.
https://whylabs.ai/		Selected. Whylogs is an off-the-shelf tool with interesting features, testing it should be simple and insightful for different use cases.
https://github.com/cleanlab/cleanlab		Selected. Cleanlab is an off-the-shelf tool with interesting features, testing it should be simple and insightful for different use cases.
(Schelter et al., 2021)		Discarded. JENGA focuses on data synthesis, which is not directly linked to the objectives of the project, similar to Dynabench.
(Schouten et al., 2009)	4.4.8.1	Discarded. The R-Indicator would require adaptations to be usable in assessing the completeness and representativeness of a data set, which is outside the scope and time frame of the project.
(Cabitza et al., 2021)		Selected.
(Catania et al., 2022)		Selected. The approach described mixes ML best practices and ideas from (Almeida and Vieira, 2011) as well as Slice Tuner. Its testing may be synergetic with the testing of these other methods and thus not too time-consuming or complex.
(Asudeh et al., 2019)	4.4.8.2	Selected. Despite its limitations in terms of data dimensionality, it is the only exhaustive method found.
(Paganelli et al., 2022)		Discarded. The MLEAP project does not have a text use case, which would complicate the testing of this very specific method.
(Trinh et al., 2018)		Discarded. It is an end-user methodology based on the presence or absence of descriptors of the source quality. The testing of more operational methods should be prioritised in the context of the project.

Bibliographical references	Document references	Decision
(C. Liu et al., 2017)		Selected. The method may require adjustments but seems simple yet insightful enough to be considered for exploratory work.
(Heinrich et al., 2018)		Discarded. The metric is essentially included in the exploration of (C. Liu et al., 2017).
(Even and Shankaranarayanan, 2007)		Discarded. The method is related to (C. Liu et al., 2017).
(Setiawan et al., 2021)	4.4.8.3	Discarded. The assessment method in itself is based on ML good practice.
(J. Lee et al., 2020)		Discarded. The assessment method in itself is based on ML good practice.
(Abidin et al., 2018)		Discarded. The method is based on benchmarking models, which are too complex and slightly off-topic for the later stages of the MLEAP project. Prioritize testing other methods.
(Caiafa et al., 2020)		Selected. PCA and other dimension reduction techniques are basics that should make for relevant baselines and may have even more potential.
(Catania et al., 2022)		Selected. The method is another interpretation of PCA and is thus completely synergetic with the previous one.
(Dourado Filho and Calumby, 2022)		Selected.
(Osman et al., 2018)		Discarded. The authors survey data imputation techniques without enough detail to narrow down the exploration work, which would be too time-consuming for the MLEAP project, with unclear results perspectives.
(Goodman et al., 2022)	4.4.8.4	Discarded. The method focuses on data synthesis and its results would not be the most interesting for the objectives of the MLEAP project.
(Santos et al., 2019)		Selected. Though there is no method to apply *per se*, the framework of MAR/MNAR/MCAR may be useful in later work
(Celis et al., 2016)	4.4.8.5	Discarded. The method is complex, and its results perspective is unclear. Other methods should be prioritised.
(A. Wang et al., 2020)		Discarded. The method is related to the previous one.
(Blatchford et al., 2021)		Discarded. Requires a continuous-valued data set, which is not included in the use cases of the project.
(Mountrakis and Xi, 2013)		Selected. Due to the method's contrastive nature, testing may be limited to comparing train, validation, and test sets.
(Brubaker et al., 2021)		Discarded. The method may require too much adjustment w.r.t the constraint of the project. Other methods should be prioritised.
(Kohut et al., 2012)		Discarded. The method may require too much adjustment to the project's constraints. Other methods should be prioritised.
(Keskes et al., 2022)		Discarded. The method relies on data ablation (i.e., removing samples), which is generally discouraged in the literature. Moreover, the method is too complex to be tested for exhaustivity.
(Simão et al., 2015)		Discarded. Adaptations to make the method work on text data, in general, could be explored, but there is no such use case in the MLEAP project.
(Anttila et al., 2012)		Discarded. The method focuses on temporal representativeness, which is of limited interest in the MLEAP project.

Bibliographical references	Document references	Decision
(Sánchez et al., 2019)	4.4.8.6	Selected.
(Balaraman et al., 2018)		Selected, although it may be a limited work due to the need to adapt the method.
(Issa et al., 2021)		Discarded. The approach is too narrow, focusing on linked databases.
(Chehreghan and Ali Abbaspour, 2018)	4.4.8.7	Discarded, as mentioned in the document.
(Y. Hu et al., 2020)		Discarded, as mentioned in the document.
(Almaimouni et al., 2018)		Discarded. Overlaps with (Caiafa et al., 2020)
(Kumar et al., 2021)	4.5.2.3	Discarded. The work does not focus on the link with representativeness.
(Yu, 2021)		Discarded. The work does not focus on the link with representativeness.
(Leavy, 2018)		Discarded. The work does not focus on the link with representativeness but is rather a position paper on balance.
(Dickinson et al., 2012)		Selected, although a part of the method consists of documenting data without being explicit about the criteria for selecting the elements to document.
(Van Vleck et al., 2007)	4.5.3.1, 4.5.3.3	Discarded. The approach requires an exhaustive review of data by human experts, which would not be realistic on large data sets in the MLEAP project and does not seem realistic in real ML design settings.
(Doku et al., 2019)		Discarded. The study is highly specific to a domain of application that takes its foundation in crowdsourced big data.
(Yang et al., 2018)		Discarded. Potential inconsistency with the object under study in the work.
(Arrieta et al., 2020)		Discarded. The application of, for example, a decision tree to highlight relevant predictors in view of estimating relevance may be explored. However, the relationship between relevance and representativeness is not well-founded in literature.
(Poggio et al., 2017b)		Discarded. Focus on deep neural network and the positive impact of CoD limitation on representativeness is still unclear, results may not be easily exploitable.
(Baggenstoss, 2004)		Discarded. The positive impact of the CoD limitation of representativeness is still unclear.
(Surace et al., 2019)		Discarded. The positive impact of the CoD limitation of representativeness is still unclear.
(Bai et al., 2019)		Discarded. The method is specific to high-dimensional nonlinear non-parametric systems.
(Z. Gong et al., 2019)	4.5.4.1, 4.5.4.3	Discarded. The methods do not address the link with completeness or representativeness.
(Hyontai, 2018)		Discarded. The method does not address the link with completeness or representativeness
(Leavy, 2018)		Discarded. The work does not focus on the link with representativeness and does not offer concrete methods.
(HLEG, 2020)		Discarded. The checkpoints are high-level and need to be complemented with concrete methods and a systematic approach.
(BSA, 2021)		Selected (for diversity as "absence of non-representative sampling bias"). The document offers checkpoints that remain slightly high level, but the approach is systematic enough to ensure a good coverage of the topic, and may provide relevant results.

Bibliographical references	Document references	Decision
(Iphar et al., 2015)	4.5.5.3	Selected. However, this approach to computing currentness must be completed with expert knowledge on the data sets and their application, as suggested by the other references cited in the section.

4.6.3 Selection grid

Bibliographical references	Data type	Related factor of influence
(Mani et al., 2019)	Any. The method fits any classification task (it is based on output labels observation).	Not related to a particular factor (4.3.9.1.1)
(Tae and Whang, 2021)	Any. The method monitors a system's learning curve. However, "slicing" the data set might require exploratory work, especially on unstructured data.	Not related to a particular factor (4.3.9.1.2)
(Pei et al., 2017)	Any. The method rests on the observation of neuron activation at learning time. It is, however, restricted to neural networks, and it is not clear whether complex architectures such as attention networks would respond well, which may indirectly limit the spectrum of use cases covered.	Not related to a particular factor (4.3.9.1.3)
(Almeida and Vieira, 2011)	Any. The method rests on the downstream evaluation metric and the use of samples replicating degraded operational conditions. Obtaining such samples (either through collection or data set improvement methods, e.g., data augmentation), may require additional work.	Not related to a particular factor (4.3.9.2)
https://whylabs.ai/	Unclear. Exploratory work on Whylogs will aim at identifying the limits of the suite.	
https://github.com/cleanlab/cleanlab	Any. The paper explicitly states that Confident Learning is not coupled to any data modality or model. It is however limited to labelled data.	
(Cabitza et al., 2021)	Any. The method rests on comparing two data distributions. However, defining the events in the distribution may require exploratory work. Moreover, obtaining a reference distribution (e.g., to compare the data set to real life) may be difficult or impossible, limiting the method's testing.	Not related to a particular factor (4.4.8.1)
(Catania et al., 2022)	Any. Method related to (Tae and Whang, 2021).	
(Asudeh et al., 2019)	Low-dimensional. The method is best fitted for categorical (i.e., qualitative) data but it is possible to categorise quantitative data.	Data Management Requirements (4.4.8.2)
(C. Liu et al., 2017)	Any. Using ratios is limited by the information they require. If completeness is defined w.r.t features (i.e., usually for low dimensional), any type of data can be assessed. If completeness is defined w.r.t characteristics other than features, which is common for high dimensional data and necessary for unstructured data, it may require preliminary human assessment, which may be more complex.	
(Caiafa et al., 2020)	PCA applies to quantitative variables (i.e., features), though it is possible to introduce some correlated qualitative variables. If working with qualitative variables, Multiple Correspondence Analysis (MCA) should be preferred. Factor Analysis of Mixed Data (FAMD) combines both methods to enable the study of mixed samples.	Data quality improvement (4.4.8.3)
(Catania et al., 2022)	The method uses PCA and other variations, such as (Caiafa et al., 2020), but for other purposes.	

Bibliographical references	Data type	Related factor of influence
(Dourado Filho and Calumby, 2022)	Images. Exploratory work may investigate the extension of the method to other data types, especially unstructured ones.	
(Santos et al., 2019)	The paper does not describe a method but rather a typology of data that it may be interesting to try and apply in the context of the MLEAP project.	Data synthesis (4.4.8.4)
(Mountrakis and Xi, 2013)	Images. Exploratory work may investigate extension to other types of data, especially unstructured ones.	Data sampling (4.4.8.5)
(Sánchez et al., 2019)	Most data types seem compatible with the method (axes are explicitly flexible and include labels, features and time). Exploratory work may be performed to confirm this.	Labelling (4.4.8.6)
(Balaraman et al., 2018)	The framework described is rather high level and seems able to accommodate most data types. However, high-dimensional and unstructured data may require more upstream work to fit the framework requirements.	
(Dickinson et al., 2012)	Although the method is not dedicated to AI, it provides relevant statistical methods to enhance balance, which can be explored for AI applications. Although the method focuses on the balance of a specific dimension in the data set, it takes into account interactions with other variables and may thus be explored in the context of high-dimensional data.	Balance (4.5.2.3)
(BSA, 2021)	Virtually any document does not present a limitation depending on the type of data. Exploring the framework's adaptability to high-dimensionality may be relevant.	Diversity (4.5.4.3)
(Iphar et al., 2015)	The method seems to be applicable to any type of data. The paper seems, however, to consider the currentness of samples, but not the currentness of the values of specific dimensions. An exploration would be required on the feasibility of assessing the currentness of high-dimensionality data.	Currentness (4.5.5.31)

4.6.4 Technical requirements

Factor (sections in the document)	Impacted attribute	Impact of the factor on C / R	Impact of C/R enhancement	Externalities	Existence of tools/methods
Intended behaviour (4.3.2)	C / R	The more complex the task, the more data points will be required to guarantee C/R.	Yes	Volume of data	There are no methods or tools. However, the type of data available may tailor some methods for certain AI tasks.
Model architecture (4.3.3, 4.3.9.1.1, 4.3.9.1.3)	C / R	Models with large capacity will require amounts of data that may hinder C/R.	Yes	Volume of data	Methods are usually bound by the architecture, not leveraging it. (Mani et al., 2019), (Pei et al., 2017), (Lei et al., 2018) are designed specifically for neural networks. No methods targeting specific architectures have been identified.
Data dimensionality (4.3.4, 4.4.8.2, 4.5.3.3)	C / R	Dimensionality influences the size of the input space. The larger the input space, the more data are required to achieve C/R.	No	Features used by the model and/or number of attributes in the data set	Methods are bound by the dimensionality, not leveraging it. (Asudeh et al., 2019) is relevant for low-dimensional, structured, qualitative data and may scale to high-dimensional, structured, quantitative data. No method stood out for unstructured data. Assessment of data relevance is required to determine the most adapted number of dimensions, but only a trade-off between C, R and relevance can be attained.
Intended level of autonomy (4.3.5)	C / R	The level of autonomy can impact the required amount of data needed to ensure the system's robustness, resilience, and adaptability.	Yes	Volume of data Variety of data Robustness, resilience and adaptability attributes of the system	No method taking oversight into account has been found.
Intended level of performance (4.3.6, 4.3.9.1.2, 4.4.8.1)	C / R	A high level of performance imposes constraints on robustness and resilience, which can, in turn, impact the nature and volume of data required to ensure C/R.	Yes	Volume of data Variety of data Robustness and resilience	(Tae and Whang, 2021), or evaluation methods such as cross-validation and others discussed in (Catania et al., 2022), unify the observation of the data set and the performance of the system at training time.
Intended levels of robustness and resilience (4.3.7)	C / R	Higher intended levels of robustness and resilience can require data whose nature and volume are not consistent with C/R requirements.	Yes	Volume of data Variety of data	No method based on improving robustness or resilience has been found.
Intended level of stability (4.3.8)	C / R	Stability requires a constant behaviour for similar outputs, which requires collecting large volumes of quality samples.	Yes	Volume of data Quality of data	No method based on improving robustness or resilience has been found.

4.6.5 Processes

Factor (sections in the document)	Impacted attribute	Impact of the factor on C / R	Impact of C/R enhancement	Externalities	Existence of tools/methods
Data management requirements (4.4.2, 4.4.8.2)	C / R	Specifications of the data sets can directly impact R and C. This step is central and should encompass requirements linked to all the factors of influence of this document.	*Since all requirements are encompassed, externalities and impacts include each individual requirement.*		(Caiafa et al., 2020) and (Catania et al., 2022) use dimension reduction methods to gain general insight onto the data set. Whylogs and Cleanlab are two off-the-shelf tools for the observation of data sets.
Data quality improvement (4.4.3, 4.4.8.3)	C / R	These strategies are meant to enhance R and C, the intended impact is then positive. However, R and C should be verified upon each data improvement manipulation (deletion, imputation, augmentation).	No	Volume of data Content of data	(Setiawan et al., 2021) and (J. Lee et al., 2020) describe GANs for data augmentation.
Data synthesis (4.4.4)	C / R	Data synthesis is meant to enhance R and C, the intended impact is then positive. This positive impact should be verified at the end of the process.	No	Volume of data Content of data	No data synthesis method linking C/R assessment stood out upon analysis.
Data sampling (4.4.5, 4.4.8.5)	C / R	Data sampling can be used to enhance R and C. However, its use for other objectives can impact R and C. The impact should be verified at the end of the process.	No	Volume of data Content of data	(Mountrakis and Xi, 2013) allows the comparison of C/R between train, dev and test sets. No method enables the absolute assessment of a data set without external information on the real-life distributions of the phenomena to be captured.
Labelling (4.4.6, 4.4.8.6)	C / R	Coarse granularity of labels, or a low quality, may limit the assessment of R and C.	No	Resources dedicated to labelling Complexity of the labelling task	(Balaraman et al., 2018) may be used for exploratory work on unstructured data, also leveraging the approaches by (Paganelli et al., 2022) and (Simão et al., 2015) for text.
Pre-processing (4.4.7)	C / R	Activities reducing the amount of information may affect R and C.	No	Correspondence between the function and the task	No pre-processing method enabling the assessment of C&R stood out upon analysis.

4.6.6 Other Data Quality Requirements

Factor (sections in the document)	Impacted attribute	Impact of the factor on C / R	Impact of C/R enhancement	Externalities	Existence of tools/methods
Balance (4.5.2)	R	Strong interdependence between balance and representativeness. Enhancing any of the attribute may impact the other.	Yes	Volume of data Content of data Only a trade-off can be reached	Statistical tests for balance enhancement to estimate the minimum sample size before performing data re-sampling (power analysis, bootstrapping, etc.). Limitations should be documented (description of the data sets, description of the values modified to enhance either balance or representativeness, description of limitations observed, etc.).
Relevance (4.5.3)	C / R	Enhancing relevance can negatively impact representativeness and completeness.	No	Volume of data Content of data Dimensionality of data Only a trade-off can be reached	The methods are at a research level. XAI for relevance enhancement. Methods to avoid Curse of Dimensionality: (Poggio et al., 2017b) for deep neural networks; (Baggenstoss, 2004) for classifiers; (Surace et al., 2019) for filtering algorithms; (Bai et al., 2019) for high-dimensional nonlinear non-parametric systems.
Diversity (discriminative power) (4.5.4)	R	If diversity enhancement is performed in view of maximising the discriminative power of the sample, it may negatively affect representativeness.	No	Volume of data Content of data	No tools or methods.
Diversity (non-representative sampling bias) (4.5.4)	R	If diversity enhancement is performed to control non-representative sampling bias or to ensure fairness, it may positively affect representativeness.	No	Volume of data Content of data Similar levels of AI performance for all subgroups	AI bias risk management should be performed (e.g., comparison of the demographic distribution and ensuring subgroups are sufficiently represented). Verification that the system presents similar levels of performance for all subgroups.
Currentness (4.5.5)	R	Currentness is one of the prerequisites for data representativeness.	No	/	Ensure currentness, e.g. by computing a probability that the data remains up-to-date in a certain time range (Iphar et al., 2015).

4.7 Preliminary experimentations

This section reports on the first round of experimentation on the methods identified in the selection grid.

4.7.1 Experimental setup

For the first round of experiments, the methods identified in the selection grid were first re-ranked by order of priority. Indeed, many methods need to be tested but each testing round should be anticipated to be time-consuming. It was deemed important to define strategies that would allow for testing as many methods as possible in minimum time.

Priority was defined as a mix of ease to test the method and interest w.r.t the MLEAP project objectives.

After defining the order in which the identified methods were going to be explored, the data sets to which they would be applied had to be defined. It was decided to select data sets for tasks related to the MLEAP project, i.e., computer vision, speech-to-text (STT) and multi-class classification, corresponding to the AVI, ATC-STT and ACAS-Xu use cases, respectively.

The selected computer vision data set was the ROSE data set. ROSE was an agricultural robotics challenge[62] organised by LNE between 2018 and 2022. The associated task was object detection and classification. The ROSE data set is comprised of 111 190 images, collected by four different teams participating in the challenge. The data set of only two teams will be used in the experiments, as the other resources are of lesser quality due to technical difficulties during the acquisition. The total of images available is 20 438 from one team and 28 555 for the other. Each image is a capture of the soil of a field, with crops and weeds plants annotated by polygonal bounding boxes. Each bounding box is also labelled with the species of the plant.

The ROSE dataset was selected because it covered the same task than the AVI use case, i.e., defining a bounding box around the target object and labelling it.

Regarding the ATC-STT use case, the REPERE data set from the eponym campaign organised by LNE between 2012 and 2014 was supposed to be used. However, experiments stopped at an earlier stage. The work reported in the remainder of the chapter was performed on the Fluent Speech Commands data set[63].

In both cases, these data sets were selected because LNE had extensive experience with them, which would reduce the time needed to get them up and running.

Finally, the last use case, ACAS-Xu, has many particularities that make it difficult to substitute relevantly. Considering that other use cases relied on resources that could be deployed more efficiently, it was decided to dedicate more time to learning and handling the ACAS-Xu data set rather than finding a substitute data set.

4.7.2 Selected methods: motivations and expectations

This section aims at recalling the references and associated methods that could be tested. In addition, it also describes how they were expected to be used to fit the objectives of the MLEAP project.

[62] https://www.challenge-rose.fr/

[63] https://fluent.ai/fluent-speech-commands-a-dataset-for-spoken-language-understanding-research/

4.7.2.1 **PCA-based analysis**

Principal Component Analysis (PCA) for prior data set analysis is used in (Catania et al., 2022) and briefly discussed in (Caiafa et al., 2020). The idea is to gain visual insight into the completeness of a data set by plotting its projection in the low-dimensional space (usually two or three, as it is difficult for humans to interpret visual information in more than three dimensions) computed by the PCA. The data points are expected to occupy the entire plot homogeneously. Any cluster or empty space might be indicative of some form of incompleteness (i.e., cluster density should be reduced or the data set should be enriched to reach a similar density or conversely examples should be added to fill the empty spaces).

The authors of both papers do not discuss how to backtrack from discrepancies seen in the PCA plot to actual recommendations to improve the data set. The main objective of the experiments was to gain more insight into this type of round-trip engineering step.

PCA is fit for high-dimensional quantitative data. Intuitively, the computer vision use case would fit these requirements. PCA may indeed be applied on images, but it then acts as an image compression algorithm. Such behaviour may influence the resulting analysis, and it was decided to try another data set first, possibly coming back to the image data set in later developments.

On the other hand, the ACAS-Xu data set also has quantitative features and is supposed to be complete and representative. Its low dimensionality limits the interest of dimension reduction strategies such as PCA but does not prevent it. Considering some time had to be invested in learning to manipulate the data set, it seemed an opportunity to take this time on a well-known method with off-the-shelf implementation, for which the expected results were clear. In addition, it was expected that PCA on ACAS-Xu would yield a simple and homogeneous cloud of data points that would act as a validation of the use of the implementation.

4.7.2.2 **Graph-based analysis**

The method proposed by (Asudeh et al., 2019) relies on traversing the tree-like graph of the feature combination of each sample of the data set. There is no available implementation and the paper only describes traversal strategies, leaving graph population as a problem for the developers. The method is, by design, directed towards qualitative data sets, while it is possible to extend it to quantitative features by binning them.

The method was expected to be slow and possibly intractable for data sets with too many samples or features. However, compared to other methods, it seemed easy to implement and offered clear data set exploration strategies. Thus, it could be seen as an inexpensive complementary tool to pair with other methods. Its implementation was decided on this basis.

4.7.2.3 **Entropy-based analysis**

The characterisation of samples in a data set using entropy was described in (Dourado Filho and Calumby, 2022). Entropy is a fundamental concept of information theory and while it may provide only shallow information on a data set, it appeared to be a useful and essential tool to combine with others in a more general approach.

Moreover, it is easy to deploy and can be adapted to any type of data. The main point of attention when using entropy is the type of elements in the data set from which the entropy will be computed, to ensure the metric provides useful information regarding the overall analysis process.

In the context of the MLEAP project, entropy will be used on the image data set as a first step. Its use might be extended to the speech data set in later phases.

4.7.2.4 **Sample-wise similarity analysis**

(Cabitza et al., 2021) and (Mountrakis and Xi, 2013) proposed characterization tools based on the comparison of two data sets. Such methods are intuitively useful to compare the completeness and representativeness of a data set w.r.t to another, for example, between a train and a test. Although this approach is limited for the assessment of an "absolute" or "ODD-wise" completeness or representativeness, it is essential to have tools ensuring the characteristics of the data are preserved across the different training steps (i.e., training, validation and testing).

These methods were planned to be used on the speech data set, partly because this data set had not been exploited with other methods and it was deemed important to cover all data sets within the phase, but also because raw speech is difficult to process directly.

The most common way to process speech is to extract features as proxy representations for downstream AI systems training. In recent years, the advancements of representation learning for speech led to the prevalent use of pre-trained speech embeddings (i.e., dense vectorised representation learned by dedicated large AI systems), in a way similar to text. Sample-wise similarity was deemed the most relevant testbed for studying this kind of representation.

A possible shortcoming of this approach would be its scalability to large data sets. Indeed, the time needed to encode embeddings for large volumes of long samples may be prohibitive and prevent the computation of the metrics at scale.

4.7.2.5 **Off-the-shelf tools**

There are many data quality tools and platforms available, whether proprietary or open source. Companies can also choose to develop their own in-house solution, tailored to their need and definition of data quality. This chapter examines a number of methods to illustrate how a company could develop such a customised tool. In order to cover the full scope of the discussions, it seemed important to also probe the capabilities of some of these off-the-shelf solutions.

Two candidates were identified for this task: Cleanlab and Whylogs. Both were selected because their conceptual frameworks are described in publications, which provides relative transparency regarding their inner workings (although a lot of details are left out of the reports). Even though Cleanlab has a paid solution with apparently more features and fewer constraints on commercial use, it also proposes an open-source version (free for non-commercial use) used by prominent companies such as Tesla or Google. These companies are strongly data-driven, which tends to indicate that the features available are substantial. On the other hand, Whylogs is fully open-source with no paid plan. Finally, Whylogs claims to be fully data-agnostic while Cleanlab describes specific resources to work on images, text, audio and tabular data only. It should be noted that these types of data, as well as the non-IID. Samples detection functionality tends to indicate that Cleanlab has no specific modules dedicated to the processing of time-series data, including videos.

Both tools are also complementary in their goals: Cleanlab aims to assist data qualification to improve an existing data set at the sample scale, while Whylogs is specifically designed for dataset monitoring.

As such, it has many features that are aimed at dataset development over time, in conjunction with feedback from trained models (e.g., non-regression of performance). A significant part of these features would not have been testable in the framework of the MLEAP project. Others were out of the scope of this chapter. This would have left a small number of relevant features to test, which would not have been representative of the possibilities offered by the tool. For all these reasons, it was decided not to pursue the testing of Whylogs and to rather focus on Cleanlab. This is visible in the type of data they can handle:

The goal of experimenting with Cleanlab is not to draw conclusions regarding the utility of off-the-shelf solutions as a whole. It is rather to see how a seemingly well-designed open-source tool can be leveraged for assessing data completeness and representativeness and how such a solution may be compared to a fully custom solution.

4.7.2.6 **Neuron Coverage Analysis**

Neuron coverage is discussed by (Cheng, 2013), among several other works on the subject. Neuron coverage is usually used as a metric to quantify inconsistencies in the processing of inputs by DNN. As an extension of this approach, neuron coverage is also used as an objective function to improve the quality of the learning process. This latter aspect is outside the scope of the present chapter, but the general idea of neuron coverage as quantitative feedback on a model's behaviour could be interesting to assess the data set completeness and representativeness from the model's perspective.

Indeed, a well-trained model would be expected to exhibit homogeneous neuron activation for each input, consistently across inputs. Anomalies in this expected behaviour would intuitively point out anomalous samples, i.e., possible edge or corner cases, hard cases, etc.

4.7.2.7 **Feature Space Characterization**

(Mani et al., 2019) introduces four metrics to characterise the feature space learned by a trained neural network. The authors use the gathered insight to generate new examples of the decision boundaries of the model. This offers a direct possibility of assessing the completeness and representativeness of the data set w.r.t the model's representation. The generated examples are processed through a deconvolutional model to see if they yield credible images. This is a complex step involving a supplementary layer of « AI for AI », which seemed superfluous for a preliminary approach to the method. A focus on the raw insight brought by the four metrics was deemed more appropriate. Besides, the deconvolutional approach is fit for images but would not transpose to other types of data, whereas the metrics are data agnostic.

4.7.2.8 **Completeness ratios**

Initially aimed specifically at the medical domain, the approach discussed in (Liu et al.) is general and simple enough to be abstracted for other types of data. The aim was to propose completeness ratios. More precisely, the paper is a survey and thus discusses several other works on the subject.

The method discussed by (Weiskopf et al, 2013) and relying on four dimensions of completeness was deemed the most adapted to the context of the MLEAP project.
The paper considers data completeness in the sense of ISO/IEC 25012:2008, i.e. “*The degree to which subject data associated with an entity has values for all expected attributes and related entity instances in a specific context of use*”. This aspect of data completeness seemed necessary to discuss for exhaustivity although it is not at the core of the challenges of the project.

Most of the toy data sets used in this chapter's experiments relied on high-quality, curated data sets, where completeness is not an issue. Nonetheless, the Titanic data set exhibited incompleteness, as could be the case of any data set based on data collected in sensible operational conditions.

It is important to note that the experiments reported here will not be transferred to the MLEAP aviation use cases, as they do not suffer incompleteness in the sense considered here.

4.7.3 Selected methods: Experimental protocols and results

This section describes how the methods were tested and what conclusions could be drawn from the experiments undertaken. At the end of each sub-section, a small synthesis of the results w.r.t the MLEAP objectives is proposed.

4.7.3.1 **PCA-based analysis**

4.7.3.1.1 Experimental protocol

Testing was performed using sci-kit-learn's PCA implementation. Contrary to other methods, no "toy" data set was used for prior validation. Considering the properties of the ACAS-Xu data set, 2-components PCA was run directly on the data set. The results are presented in Figure 25.

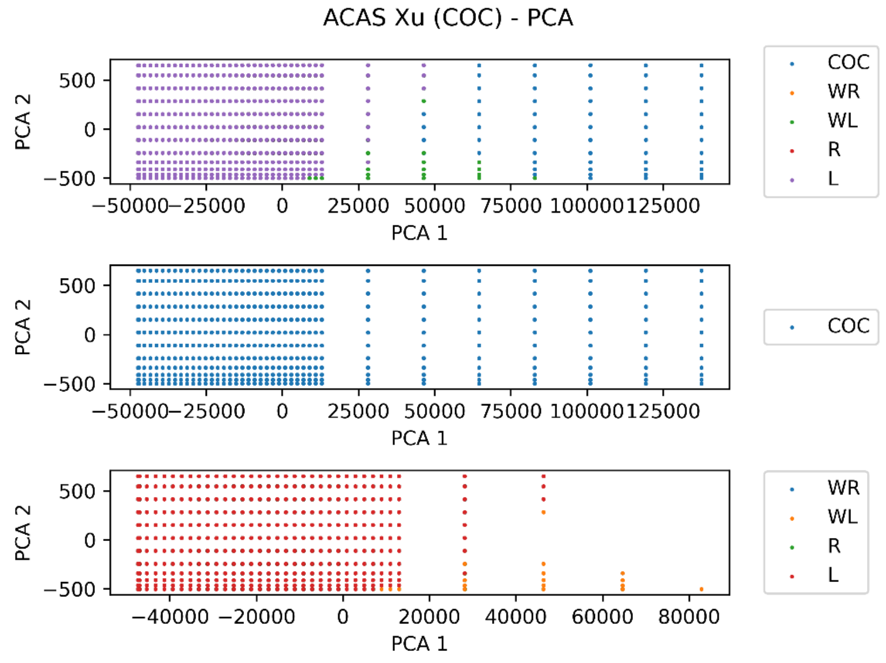


Figure 25. Results of a 2-components PCA applied on a subset of the ACAS-Xu data set. Only samples with input state COC were used.

As a reminder, the ACAS-Xu data set is comprised of eight features, including the aircraft's current state, expressed with the same class as those to predict; hence, the plot represents the *Clear Of Conflict (COC)* class (i.e., it is the input state and not the expected label)[64].

The plot shows the PCA for the input state *COC*. The output states to predict are the expected manoeuvres, either WR (Weak Right), WL (Weak Left), R (Right) or L (Left). Since PCA builds principal components (i.e., axis) by linearly combining input features with heterogeneous ranges and units to yield a new coordinate system, the actual values of the axis are meaningless. As in most cases when working with PCA, insight is mostly gathered by analysing the positions of the data points relative to each other rather than from an absolute standpoint. A strong homogeneity on the left-hand side of the graph can be observed. On the right-hand side, a structuring of the data points along vertical lines is still present but lines are horizontally separated by clear gaps. The homogeneity of the first half of the graph and of the vertical lines is interpreted as a confirmation of the exhaustive coverage of the data set.

Combined with Figure 26, it can also be noted that in a vast majority of cases, the resulting manoeuvre is *COC*. This is consistent with the general trends of the data set, where the *COC* class is highly dominant. As the data set is supposed to be representative, discussions about this imbalance will be left aside for the moment.

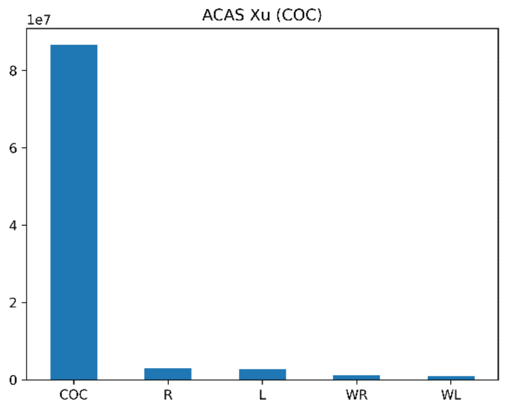


Figure 26. Distribution of the output label on the ACAS-Xu samples with input state COC.

However, ACAS-Xu being a data set compiling information about unmanned aircraft, the gaps between vertical lines cannot be explained by real-life regularities such as the use of fixed airways. This unexpected result and the lack of an intuitive explanation for it called for complementary work to assess whether they were artefacts due to errors in the use of the PCA or in the retrieval of the data set or actual phenomena from the data, in which case the work should help understand their root.

First, explained variance, i.e., the quantity of variance for each component was computed. This step can be performed prior to computing the PCA and indicates how much variance i.e., information is held in each component. It is used to choose the target number of components i.e., dimension in

[64] In the remainder of the discussion, unless indicated otherwise, the "COC" label will be used. Other labels are not shown because the observable trends are mostly identical.

which the PCA will be computed. Figure 27 shows the histogram of explained variance for the ACAS-Xu data set (still on the *COC* label).

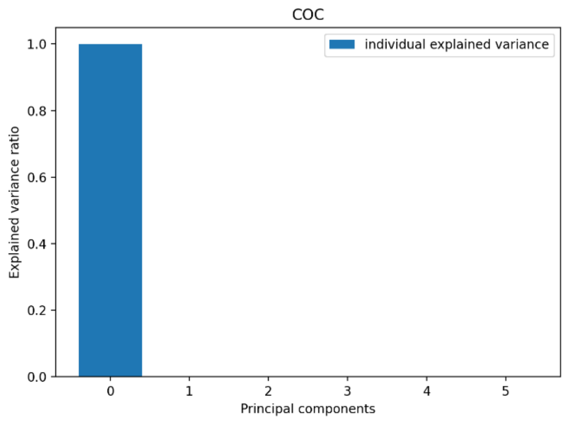


Figure 27. Explained the variance ratio of each component of the PCA. The PCA was performed on the subset of the ACAS-Xu samples with input state COC

It can be observed that the first component concentrates the entirety of the explained variance. Therefore, using the second component for a 2-dimensional projection will bring no supplementary information. Consequently, a new graph was plotted, visible in Figure 28, representing the distribution of the first component's values in function of the output decision (as cost value) for input classes *COC* and *Weak Right WR*.

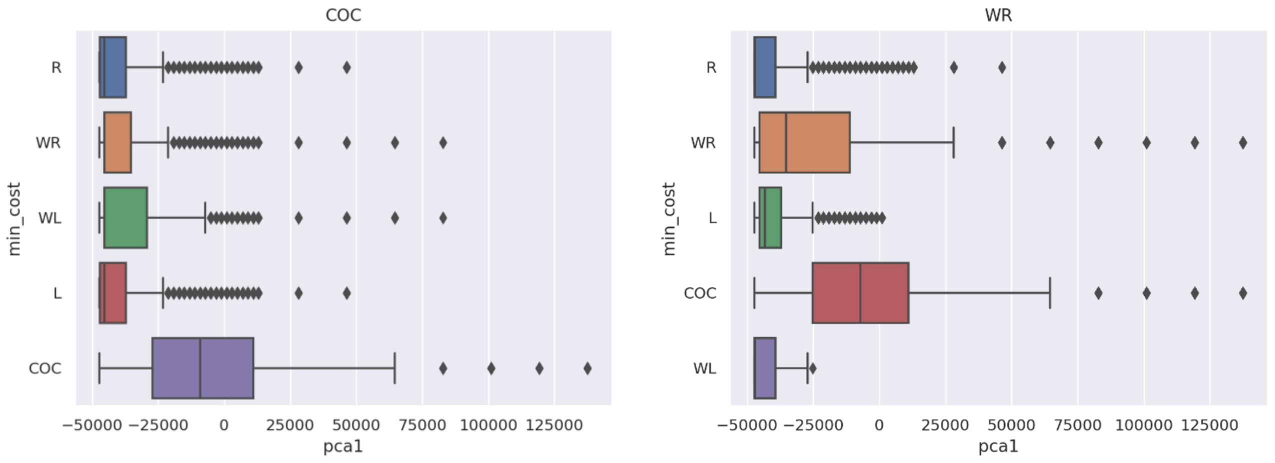


Figure 28. On the left, a boxplot of the distributions of values taken in the first component of the PCA of the ACAS-Xu subset with input state COC, in the function of the output states. On the right, the same plot for the subset of input state WR.

Boxplots are to be read as follows: the line inside the box indicates the median of the distribution (i.e., the value for which 50% of the sample is above and 50% below). The left and right extremities of the box represent the first and third quartiles of the distribution (i.e., 25% of the samples have values below the first quartile, and 25% of samples have values above the third quartile), respectively. The line at the extremities of the "whiskers" indicates the minimum value (left) and maximum value of the distribution. Finally, the diamonds represent statistical outliers, i.e., single occurrences of values outside the distribution. Note that these statistical outliers might be indicative of outliers in the sense of (EASA, 2024). The boxplots in Figure 28 confirm the class imbalance, with a strong dominance of the *COC*. The *WR* plot also shows that the dominance is shared between *COC* and the same class as the input (i.e., *WR* in this case), as illustrated by the difference in size of the boxes. There is also a large number of statistical outliers and the medians in the boxes are not visible or are very biased. All these phenomena show the heterogeneity of the distributions. No insight into the gaps was found.

To ensure there was no error with the use of the PCA, a new data set was used. The Gas Sensor Array Drift Dataset[65] was chosen. This data set comprises 13 910 measurements from 16 chemical sensors utilised in simulations for drift compensation in a discrimination task of 6 gases at various levels of concentrations. Each measurement is a 128-dimensional vector. It is a typical case where PCA may provide insight into the data set characteristics.

First, the distribution of the different gases and the histogram of explained variance were plotted, and they are shown in Figure 29 and Figure 30, respectively.

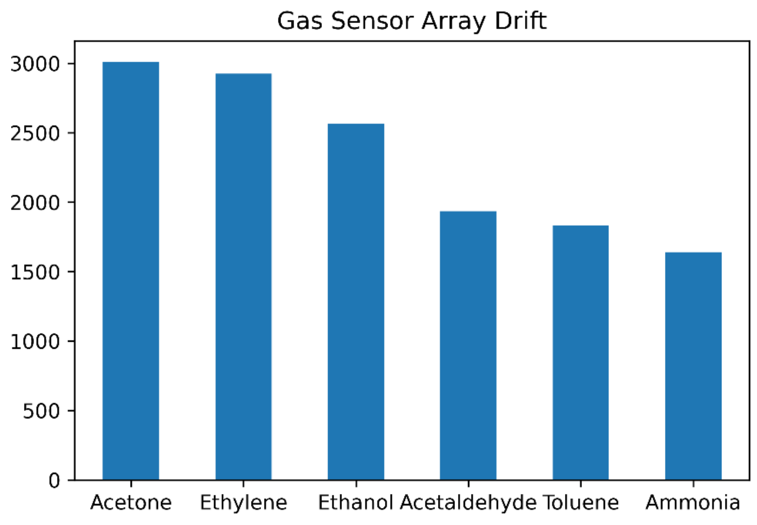


Figure 29. Distribution of the labels in the Gas Sensor Array Dataset.

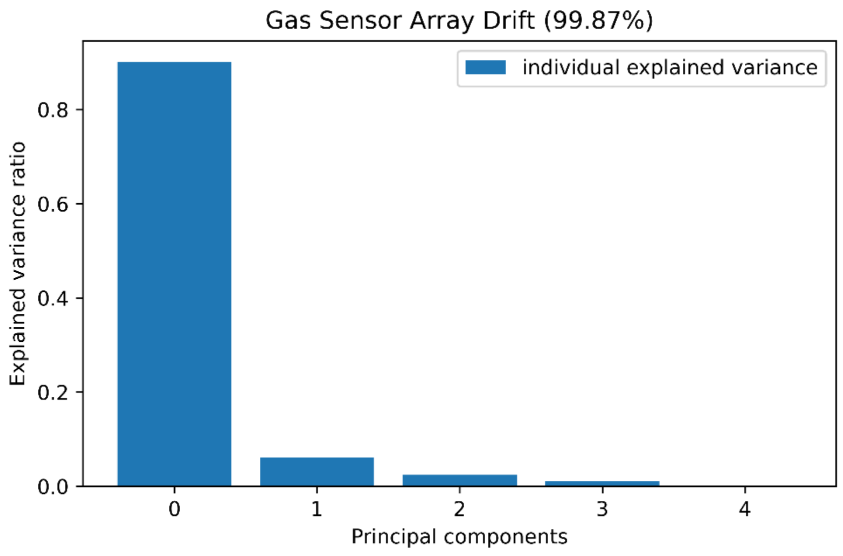


Figure 30. Explained variance ratio of each component of 5-component PCA performed on the Gas Sensor Array Dataset.

[65] https://archive.ics.uci.edu/ml/datasets/gas+sensor+array+drift+dataset#

Though not completely balanced, the data set has a globally homogeneous distribution of classes, with a 30% difference between the most represented and the least represented classes. Also, while the first component concentrates most of the variance, the second axis may still be informative. The result of the 2-dimensional PCA is visible in Figure 31.

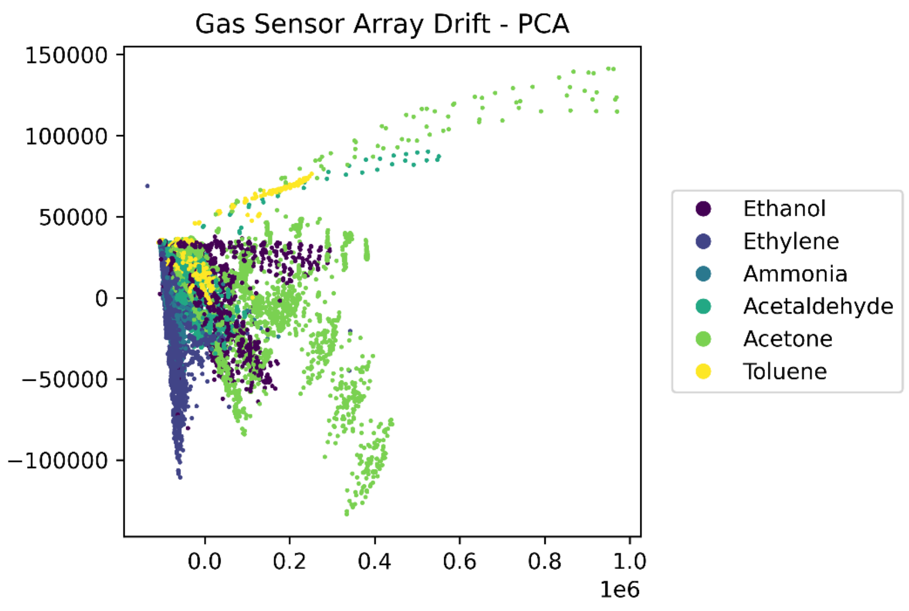


Figure 31. PCA score plot of the Gas Sensor Array Dataset.

The data points for *ethylene* form a dense and well-defined cluster. *Toluene* is less defined but each cluster is compact, as is the case for *ethanol*. However, other gases have a more diffuse distribution across the graph, especially acetone which is also the most represented, although *ethylene* is present in comparable proportions in the data set (and the best-structured in the plot). The general form of the plot validates the behaviour of the PCA, excluding errors from bad uses of the implementation.

Finally, testing of another method on ACAS-Xu required checking the value range of each feature. It was then observed that most features took only a few discrete values. This is detailed in Table 10.

Table 10. Number of unique values for each feature of the ACAS-Xu data set (entire data set).

Feature name	Number of unique values in the data set
vertical_tau	10
intrspeed	12
ownspeed	12
max_cost	5
min_cost	5
psi	41

range	39
theta	41

Features *psi*, *range* and *theta* have the largest range with around 40 different values for each, while other features are between 5 and 12, across all samples and all labels. This phenomenon probably explains the gaps observed in the PCA, although as discussed earlier, it is unexpected w.r.t the a priori knowledge on the use case.

4.7.3.1.2 Results from the MLEAP Perspective

Considering the Gas Sensor Array Drift Dataset represents real-life data, it is probably incorrect to interpret the sparsity of acetone as a lack of completeness of the data set. However, considering said sparsity, it may be relevant to anticipate a lower performance in predicting behaviours related to this gas, which could lead to a need for more data. Such a situation would be a typical illustration of the balance/coverage dilemma discussed earlier in the chapter.

The results obtained so far regarding the ACAS-Xu data set should be considered inconclusive. Hypotheses may explain the patterns observed but could not be confirmed. Moreover, these hypotheses also question the assumptions of completeness and representativeness of this data set.

More information on the data set's constitution was obtained, and complementary analysis (such as feature-component correlation or normalisation verification) is required. Both tasks had to be put aside to allow for the testing of other methods and will be continued in the later phases of the MLEAP project.

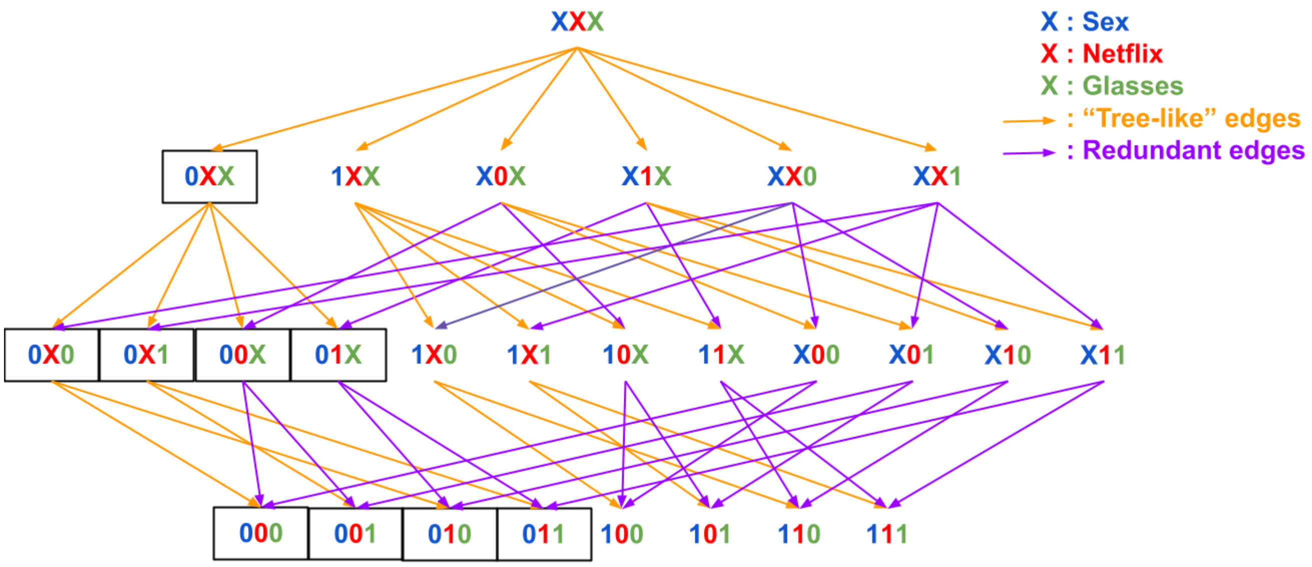
However, the study of the ACAS-Xu data set showed the importance of good practice, such as high-level verification of the data set (e.g., the expected vs. actual range of values) and the need to structure the analysis surrounding the PCA (e.g., analysing the explained variance across components). It also revealed unexpected trends in the data set related to its completeness.

4.7.3.2 **Graph-based analysis**

4.7.3.2.1 Experimental protocol

As no reference implementation was available, the necessary software had to be developed in-house. Several iterations were necessary to validate the tools' correct operation. The general idea of the method is to explore a tree-like graph of feature patterns, where a "leaf" is a sample from the data set and any upper-level node is the combination of determined and increasingly variable features.

As an example, consider a data set where each sample is a person represented by three binary features: their sex (0: male, 1: female), whether they have a Netflix account (0: no account, 1: has an account) and whether they wear glasses (0: no glasses, 1: wears glasses). Then the sample (also called pattern in the paper) [0, 1, 0] describes a man with a Netflix account who does not wear glasses. It would be a leaf of the graph and would have direct ancestors' patterns [X, 1, 0], [0, X, 0] and [0, 1, X] i.e., persons with a Netflix account not wearing glasses, men not wearing glasses and men with a Netflix account, respectively. Here, X stands as a placeholder for undetermined features. Any graph has the same root pattern with only undetermined features, here [X, X, X]. The paper discusses traversal strategies to find Maximum Uncovered Patterns i.e., patterns for which the number of attached samples is lesser than a user-defined threshold. The graph associated with this example is presented in Figure 32. Notice how the graph expands when developing fixed features, and then narrows down when reaching the fully fixed-features set. While the graph may intuitively be imagined as a tree, a large number of redundant edges actually appear. This redundancy must be taken into account in the traversal strategies for two reasons. The first is algorithmic efficiency, as it is not desirable to traverse the same regions of the graph several times. The second is to enable correct MUP identification. As an example, consider the case where the number of males would be inferior to a given threshold (blue feature = 0, upper boxed combination). In such case, all boxed combinations would be part of the MUP and it becomes useless to traverse them. However, this is not trivial as they can be reached through other combinations (purple arrows). The traversal strategies should account



for such cases.

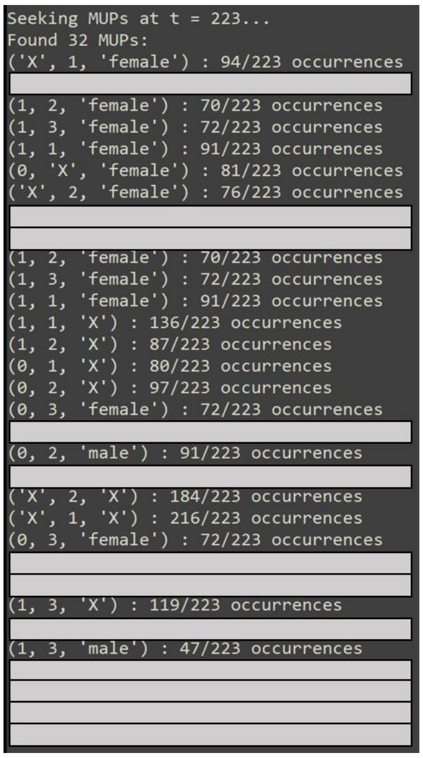
Figure 32. Graph representation of the example

The first version of the implementation included a complete building of the connection graph with separate traversal algorithms, to allow the checking of the behaviour of both the population and traversal algorithms separately and by hand if needed. This is in opposition with the original approach that relies on traversal as a pruning strategy for efficiently populating the covered portion of the graph. To enable potential manual verification, this first iteration was tested on the Titanic data set[66].

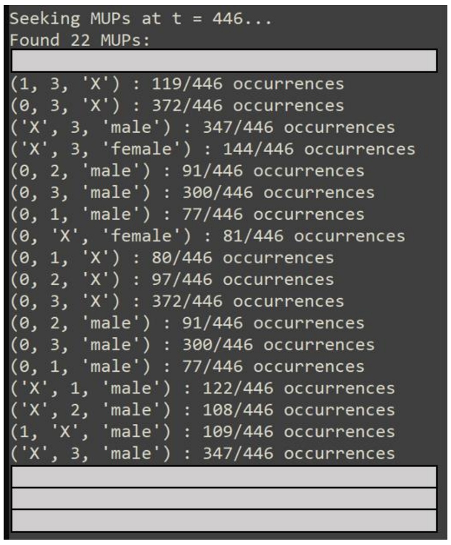
[66] https://github.com/datasciencedojo/datasets/blob/master/titanic.csv

The Titanic data set is a public data set comprised of various information about the 891 passengers of the RMS Titanic that sunk on April 15, 1912. It is a simple and popular classification data set for small ML projects. It was chosen as it is small enough to be explored by hand but large enough to hold different trends that can be used to validate the behaviour of the algorithms under implementation (thus being sometimes called a "toy" data set). While it has several features with missing values (such as age), some interesting features are complete. The three main features selected for testing were *Survived* (0: died in the wreck, 1: survived the wreck), *Pclass* (1: first class, 2: second class, 3: third class) and *Sex* (M: Male, F: Female).

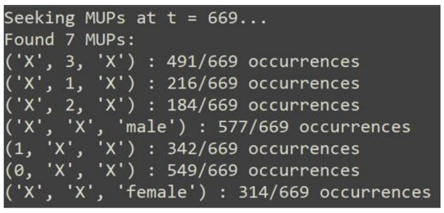
The development of this first iteration required more time than expected, in particular, because the genericity of the tool had to be preserved to allow the transparent handling of future data sets. Results of the first experiments are shown in Figure 33, setting the threshold at 223, 446 and 669 occurrences, respectively 25%, 50% and 75% of the number of passengers.



```
Seeking MUPs at t = 223...
Found 32 MUPs:
('X', 1, 'female') : 94/223 occurrences
[redacted]
(1, 2, 'female') : 70/223 occurrences
(1, 3, 'female') : 72/223 occurrences
(1, 1, 'female') : 91/223 occurrences
(0, 'X', 'female') : 81/223 occurrences
('X', 2, 'female') : 76/223 occurrences
[redacted]
(1, 2, 'female') : 70/223 occurrences
(1, 3, 'female') : 72/223 occurrences
(1, 1, 'female') : 91/223 occurrences
(1, 1, 'X') : 136/223 occurrences
(1, 2, 'X') : 87/223 occurrences
(0, 1, 'X') : 80/223 occurrences
(0, 2, 'X') : 97/223 occurrences
(0, 3, 'female') : 72/223 occurrences
[redacted]
(0, 2, 'male') : 91/223 occurrences
[redacted]
('X', 2, 'X') : 184/223 occurrences
('X', 1, 'X') : 216/223 occurrences
(0, 3, 'female') : 72/223 occurrences
[redacted]
(1, 3, 'X') : 119/223 occurrences
[redacted]
(1, 3, 'male') : 47/223 occurrences
[redacted]
```



```
Seeking MUPs at t = 446...
Found 22 MUPs:
[redacted]
(1, 3, 'X') : 119/446 occurrences
(0, 3, 'X') : 372/446 occurrences
('X', 3, 'male') : 347/446 occurrences
('X', 3, 'female') : 144/446 occurrences
(0, 2, 'male') : 91/446 occurrences
(0, 3, 'male') : 300/446 occurrences
(0, 1, 'male') : 77/446 occurrences
(0, 'X', 'female') : 81/446 occurrences
(0, 1, 'X') : 80/446 occurrences
(0, 2, 'X') : 97/446 occurrences
(0, 3, 'X') : 372/446 occurrences
(0, 2, 'male') : 91/446 occurrences
(0, 3, 'male') : 300/446 occurrences
(0, 1, 'male') : 77/446 occurrences
('X', 1, 'male') : 122/446 occurrences
('X', 2, 'male') : 108/446 occurrences
(1, 'X', 'male') : 109/446 occurrences
('X', 3, 'male') : 347/446 occurrences
[redacted]
```



```
Seeking MUPs at t = 669...
Found 7 MUPs:
('X', 3, 'X') : 491/669 occurrences
('X', 1, 'X') : 216/669 occurrences
('X', 2, 'X') : 184/669 occurrences
('X', 'X', 'male') : 577/669 occurrences
(1, 'X', 'X') : 342/669 occurrences
(0, 'X', 'X') : 549/669 occurrences
('X', 'X', 'female') : 314/669 occurrences
```

Figure 33. From left to right: MUPs for 25%, 50% and 75% of the number of passengers. Redundant MUPs across thresholds have been hidden for readability.

At $\mathrm{t} = 669$, the entire data set is captured. Identified MUPs are merely dispatched between the different features. It can be observed there was more 1st class passengers than 2nd class. For a threshold value of 446 (50% of the passengers), more trends are visible e.g., out of 342 survivors, only 109 were males (out of 577 male passengers, or 18%). Moreover, 37% of 1st class passengers, 52% of 2nd class and 61% of 3rd class passengers did not survive. Finally, examining the MUPs obtained for $t = 223$, the proportion of female survivors can be established (96% of 1st class, 92% of 2nd class and 51% of 3rd class). While unsurprising considering the nature of the data set (a vast majority of women survivors and a clearly higher fatality rate for 3rd class), these results are illustrative of what can be inferred using this method. It also shows the influence of the selected threshold: the closer it is to the cardinality of the data set, the higher the level of information, which allows for controlling the granularity of imbalance information provided. The method could be the basis of a useful visual characterization tool.

A run on the ACAS-Xu data set was then prepared in parallel with a second version of the implementation that would include pruning. Some pre-processing had to be performed. Indeed, the method is designed to work only with qualitative data, quantitative data had to be discretised. This can be achieved through binning, but different policies can be used. The first step was to check the distribution of values in the data. On this occasion, the particular distribution of the ACAS-Xu features was observed and triggered some complementary work on the PCA methods. Consequently, the method could not be tested on ACAS-Xu at the time of writing this document.

4.7.3.2.2 Results from the MLEAP Perspective

Despite seemingly light experimentations, the method's interest was confirmed. It has the potential to be a useful tool for general characterization of a data set. To improve its added value, the next round of experimentations may include the development of a threshold-setting strategy (currently a limiting factor as it requires launching and comparing runs with different thresholds, which is inefficient and impractical).

A possible strategy would be to automatically identify the thresholds that would encompass 25%, 50%, 75% and 100% of the data under a given pattern, to offer a synthetic visual tool of the imbalances in a data set, provide insight on potential completeness or representativeness shortcomings. Another would be to run the algorithm with dynamic thresholds, to ensure the data set complies with external completeness or representativeness constraints (that would have to be defined upstream).

4.7.3.3 **Entropy-based analysis**

4.7.3.3.1 Experimental protocol

In the reference paper (Dourado Filho and Calumby, 2022), entropy was used to compute the intra-class variability of a plant image data set. The data set had two "levels" of annotation: the type of the plant on the image and, for each type of plant, the part of the plant represented on the image.

As for the graph-based method, the first implementation was tested against a "toy" data set, namely CIFAR-100[67]. This particular data set was selected because, as any such data set, it is easy to handle and thus allows manual checks if necessary and is small enough to run in an algorithm without consuming significant resources while big enough to provide phenomena to observe. It also has additional advantages in the context:

- It is comprised of 20 "super-class" (i.e., coarse-grained annotations such as "insects"), each sub-divided into 5 "sub-classes" (i.e., fin grain-annotations such as "beetle" or "butterfly")
- Each sub-class has the same number of images
- Each image has the same resolution

Considering these properties, any variation in entropy would come from the information embedded in the images rather than the imbalances of the data set. In addition, the "coarse-grain"/"fine-grain" annotation scheme replicates that of the reference paper, which could be convenient for analysis (although it will actually not be exploited and should not have become a necessary configuration, as the MLEAP AVI use case is not supposed to be designed this way).

[67] https://www.cs.toronto.edu/~kriz/cifar.html

The results of computing Shannon entropy on each super-class (label-wise entropy) are shown in Figure 34. The image-wise entropy of each 20 super-classes of the data set is visible in Figure 35.

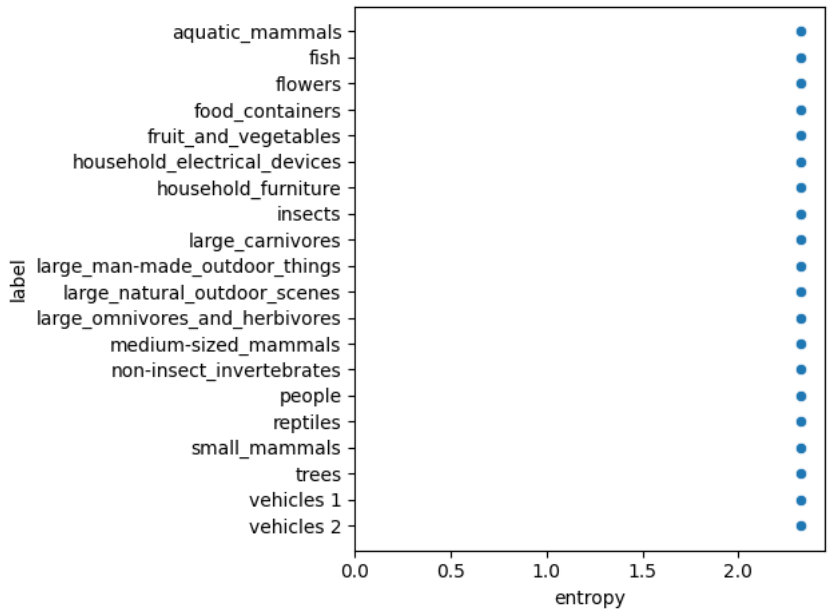


Figure 34. Entropy value of each super-class of the CIFAR-100 data set.

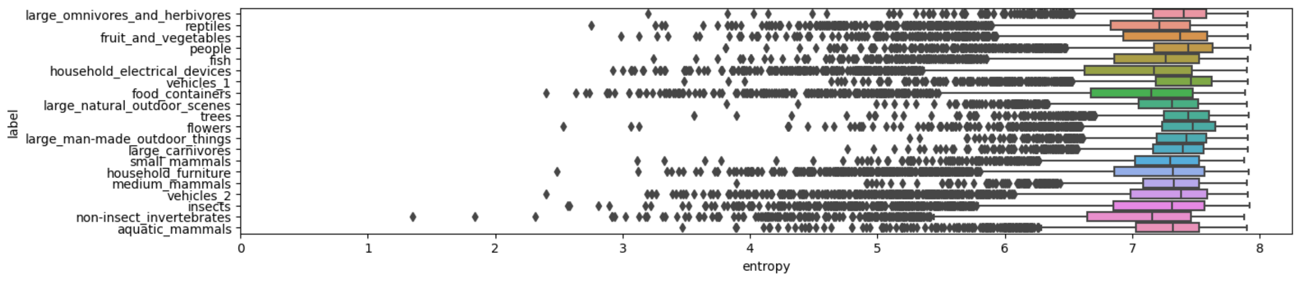


Figure 35. Boxplots of the distribution of the entropy values of every image in each 20 classes of the CIFAR-100 data set.

As a reminder, boxplots are to be read as follows: the line inside the box indicates the median of the distribution (i.e., the value for which 50% of the sample is above and 50% below). The left and right extremities of the box represent the first and third quartiles of the distribution (i.e., 25% of the samples have values below the first quartile, and 25% of samples have values above the third quartile), respectively. The line at the extremities of the "whiskers" indicates the minimum value (left) and maximum value of the distribution. Finally, the diamonds represent statistical outliers, i.e., single occurrences of values outside the distribution. Note that these statistical outliers might be indicative of outliers in the sense of (EASA, 2024). Figure 34 is merely a visual confirmation of the balance of the super-classes: entropy has the same value for all classes because they contain the same number of images. Moreover, the measured entropy is maximal. Indeed, Shannon entropy's upper bound is defined as:

$$log_2(n)$$

with n is the cardinality of the random variable considered. Here, each class has 5 sub-classes of 500 images each, so the upper bound of the entropy for each class is

$$log_2(5) = 2.32.$$

On the other hand, Figure 35 shows a little more variability among the images of each class: *large man-made outdoor things* or *medium mammals* have notably fewer outliers, while *food_containers* have significantly more. Despite these observations, the medians are roughly homogeneous and the maximum value for each variable is consistently close to the upper bound. In the case of CIFAR-100, each sample is a 32x32 image, so the upper bound is:

$$log_2(32 * 32) = log_2(1024) = 10$$

These preliminary tests validated the implementation of the method, which was then applied on the ROSE data set. As a first approach, it was decided to use single-object images, matching the setting of the CIFAR-100 data set, rather than working directly on the multi-object setting of ROSE and AVI. Thus, the data set requires some pre-processing, e.g., since every full frame image has several plants with their associated labelled bounding boxes, each bounding box had to be extracted, so that each image on which entropy is computed represents only one object. Consequently, images have heterogeneous dimensions. Moreover, each group used different sensors for image acquisition, with a group using classical CMOS sensor and the other using false colours images from multispectral cameras. Though it would be a problem in an applicative perspective, in the case of exploratory work these biases will be useful to illustrate the behaviour of the metric. Figure 36 shows the image-wise entropy for the super-classes of group A (left, team "bipbip") and B (right, team "roseau").

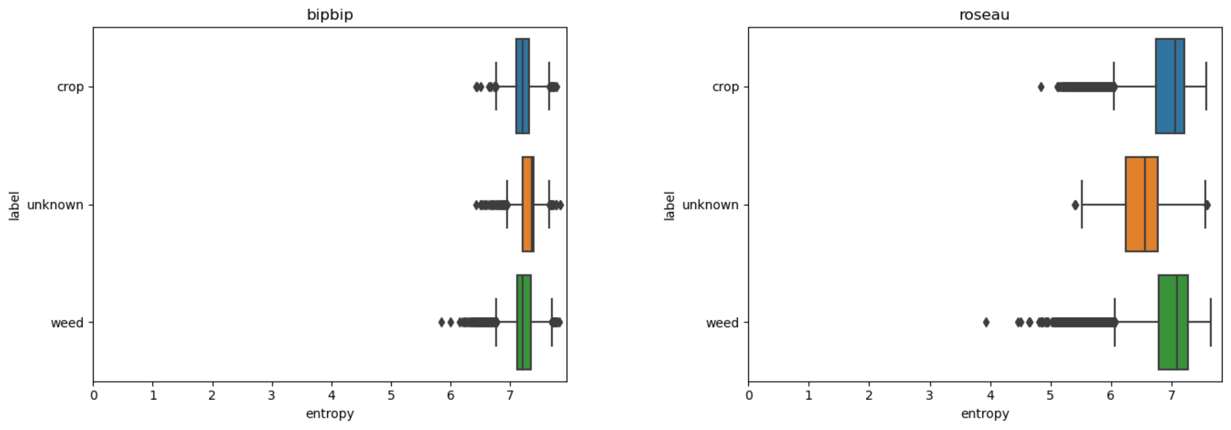


Figure 36. On the left, boxplots show the distributions of the entropy values of the images of group A (team "bipbip") in each class of the ROSE data set. On the right, boxplots show the distributions of the entropy values of the images of group B (team "roseau") in each class of the ROSE data set.

The medians of *crop* and *weed* are close, while *the unknown medians are notably different. Group B has no outliers over the maximum (except a few for unknown), contrary to group A. However, Group A's entropy distributions are notably tighter than Group B's*. Overall, both distributions are more similar than could be expected considering the difference in acquisition sensors.

Also, the average image size for group A is 502*700 pixels, yielding an average upper bound of $log_2(351400) = 18.4$. Group B has an average image size of 397*644, so an average upper bound of $log_2(255668) = 17.9$. Compared with the boxplots, it shows the images have an absolute low-complexity (probably due to the fact that most plants considered are mostly made of green leaves, with little shading). Table 11 presents the dispersion of the image sizes for both groups. The large variability explains the number of outliers display in the boxplots.

Table 11. Per-group dispersion of the image size of the ROSE data set.

Group	Min size (h*w pixels)	Mean size	Max size	25%	50%	75%
A	1 * 3	502 * 700	1449 * 2048	328 * 358	453 * 631	620 * 921
B	5 * 5	397 * 644	821 * 1228	206 * 350	381 * 647	587 * 945

Figure 37 shows the label-wise entropy of both groups.

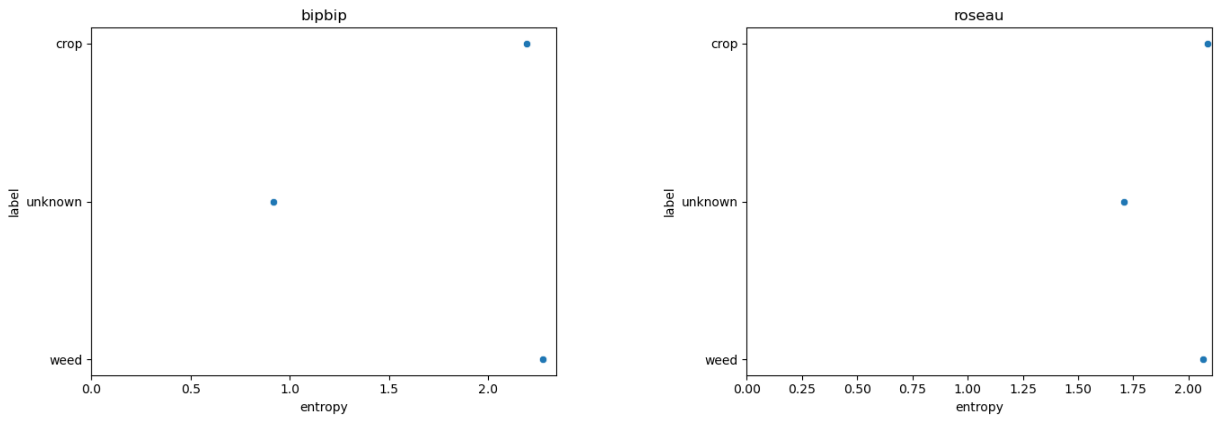


Figure 37. On the left, group A's entropy values for each label are shown. On the right, the same plot is shown for group B.

We can see the trends in entropies are similar for both groups, with *unknown* being the lowest and *crop* and *weed* being of comparable values, although the difference is smaller for group B (contrary to the image-wise entropy trends). This indicates that *crop* and *weed* images are more complex than *unknown* images. The similarity of both distributions tends to indicate that a system trained on data from group A could be used on data from group B (and vice-versa) with decent performance. However, a system using a merged data set of both data might not increase its performance beyond sheer volume benefit.

4.7.3.3.2 Results from the MLEAP Perspective

Using entropy on CIFAR-100 confirmed the balance of data set while hinting discrepancies in the amount of information available for learning each class. During evaluation, classes with the highest entropy might exhibit a higher error rate. This is because such images are somehow more complex, e.g., *large man-made outdoor things* images might share similar backgrounds and overall tones while *food_containers* might have more diverse background and object colours (especially considering the low resolution of the images), making them harder to predict for a model due to the highest number of potential features. Such phenomena might influence the volume requirement to meet representativeness criteria, though it might not be used to set a precise target value.

On the ROSE data set, the method showed that despite the fundamental differences of hardware used, group A's data set could be considered representative of group B's. Therefore, it is not farfetched to think that a system trained on the data set of one group could perform similarly on the other group's data, which is interesting.

However, the method also showed that there is a difference in the complexity of the images, which might bias the system's performance. If such a cross-data set experiment was tested, other metrics should complement the assessment, e.g., information about class-wise volume, etc. Conversely, entropy seems to be a useful metric that could be used in other assessment frameworks, either as a replacement for less suited metrics or as a complement of other metrics or tools.

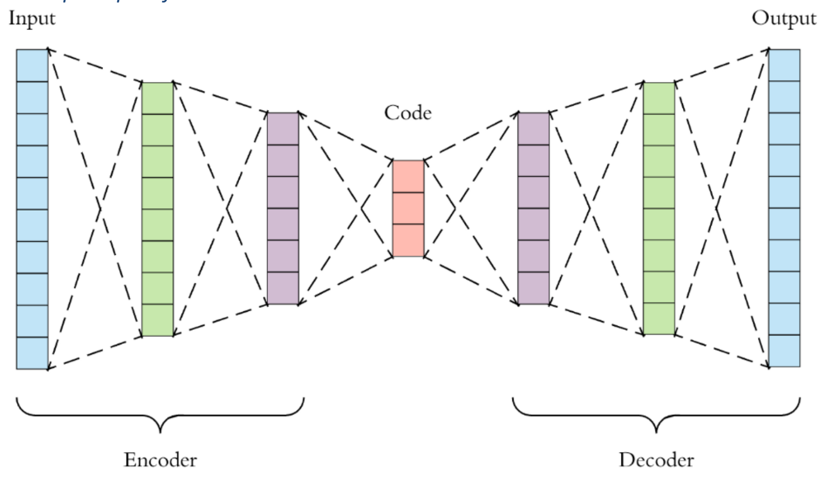
4.7.3.4 **Sample-wise similarity analysis**

4.7.3.4.1 Experimental protocol

As mentioned, the first problematic to apply the method is to be able to convert an audio signal into an embedding. Speech embeddings exist in many variants. Well-known pre-trained embeddings include i-vectors (Dehak et al., 2011) (trained using Gaussian Mixture model) and x-vectors (Snyder et al., 2018), an evolution of i-vectors trained using Neural Networks. However, both representations aim at capturing speaker-related information like prosody. Those information are not useful in the context of the MLEAP project, whose main focus is speech-to-text transcription, i.e., the semantic of the spoken exchanges, rather than the specificities of the speakers.

Nowadays, STT is treated as an end-to-end task, i.e., systems are trained with aligned speech and text as input and labels, respectively, and learn to predict the latter from the former. Most state-of-the-art STT systems are based on neural networks and usually adopt an Encoder-Decoder architecture. The Encoder takes an input (e.g., speech) and transforms it, usually through increasingly smaller hidden layers, into an internal representation (i.e., an embedding) which is fed to the Decoder. The Decoder is built as a reflection of the Encoder, thus transforming an embedding into a text sequence through increasingly large hidden layers. Training both modules jointly increase the overall performance on the task. Figure 38 illustrates the general principle of the Encoder-Decoder architecture.

Figure 38. General principle of the Encoder-Decoder architecture. The "Code" is the learned embedding space.



Some of these models are trained on an extremely large data set, with learning strategies favouring the learning of embeddings based on small units of speech (roughly comparable to phonemes). For such systems, it is possible to use the Encoder alone to extract embeddings that can then be used in downstream tasks with good performance. As these pre-trained embeddings are formed on sub-units of speech, there is no problem with Out-of-Vocabulary samples.

Four off-the-shelf pre-trained models with retrievable embeddings were found during the research step:

- End-to-End SLU[68] (Lugosch et al., 2019)
- Wav2Vec 2.0[69] (Baevski et al., 2020)
- HuBERT (W.-N. Hsu et al., 2021)
- SpeechT5 (Ao et al., 2021)

Wav2Vec 2.0, HuBERT and SpeechT5 are provided by prominent firms such as Facebook or Google and accessible via Hugging Face. They may be perceived as a more appealing resource, easier to deploy and with better support. Moreover, they are probably more accurate in absolute value, as they rely on Transformers while E2ESLU is based on their predecessors, RNN. However, since the objectives and evaluation protocols are different for all systems, comparing their strength and weaknesses is difficult (and outside the scope of the project), preventing the use of expected performance as a selection criterion. In the meantime, the embeddings of E2ESLU have fewer dimensions (256 vs 512), which makes computation less costly. Generally, the performance of E2ESLU embeddings should be sufficient for the exploratory nature of the work performed at this stage, as the focus is to have semantic embeddings in a consistent representation space rather than the absolute quality of said space. This is why E2ESLU was selected at first.

Degree of Correspondence on E2ESLU

[68] Hereafter abbreviated *E2ESLU*, implementation available at https://github.com/lorenlugosch/end-to-end-SLU
[69] Implementation available at https://github.com/huggingface/transformers

The preliminary experiment setup is basic: a subset of three audio samples from the Fluent Speech Commands Dataset was duplicated to emulate two identical datasets (called A and B respectively). Each file is then loaded and its associated embedding is generated.

(Cabitza et al., 2021)’s metric is called Degree of Correspondence (DoC) and uses a K-nearest-neighbours (KNN) algorithm as an internal mechanism. This implementation requires 1- or 2-dimensional inputs (i.e., vectors or matrices) as inputs. On the other hand, all speech embeddings (E2ESLU and others) are 3-dimensional, with a dimension of variable length, depending on the duration of the input sample. Therefore, the E2ESLU embeddings underwent a two-step pre-processing:

- First, they were padded with zeros according to the longest embedding in the data set (along the second axis), so as to all share the same shape.
- Second, they were reshaped by collapsing the first dimensions onto each other, leaving the last dimension untouched.

Finally, DoC was computed between the resulting matrices. However, the final DoC value was 0.85, whereas 1.0 should be expected considering both data sets are identical. The difference seemed too high to be imputable to variations caused by the internal KNN algorithm. Moreover, the algorithm ran two more times, yielding scores of 0.88 and 0.66. Several sanity checks were then put in place to better understand these results.

First, an element-wise comparison of each cross-data-set pair of embeddings was performed, i.e., is embedding 1 of data set A identical to embedding 1 in data set B? None of the embedding pairs were identical, confirming that the problem did not (only) come from the DoC computation. Second, the same element-wise comparison was performed for a pair of the same embeddings from the same data set, i.e., is embedding 1 from dataset A identical to embedding 1 from data set A? This time the pair was found to be identical. The same protocol was performed first on the padded and then unpadded embeddings with identical results, hinting that E2ESLU might generate its embeddings non-deterministically. The considerable variability in DoC made these embeddings unusable for further exploration. It was then decided to move to the other candidates despite their potential shortcomings.

Degree of Correspondence on Wav2Vec2, HuBERT, SpeechT5

Running the same sanity checks on Wav2Vec yielded no alert, i.e., embedding 1 of data set A is identical to embedding 1 in data set B. Running DoC between both data sets yielded a score of 0.96. To confirm that such variability came from the internal KNN, DoC was also run between data set A and data set B, yielding 0.96 again. This confirms an irreducible variability that, although small, must be taken into account in further experiments on DoC.

Then, the choice of speech embeddings for DoC and other similarity assessments was explored. To do so, two data sets of three commands were assembled.

Table 12 shows the commands chosen.

Table 12. Correspondence between the audio samples of commands used in the experiment.

Data set A	Data set B
Decrease the heating in the bathroom	Turn the heat down in the bathroom
Fetch my shoes	Go get me my shoes
Increase the sound	Volume up

It can be observed that each command in a given data set has a semantically similar counterpart in the other but with a notably different wording, except for up to one common word. Wav2Vec2 being trained for STT, it is not supposed to account for semantic similarity (focusing on the writing). However, complex representations such as these embeddings may capture a wide variety of phenomena, and investigating semantic could provide directions for further experimentations.

The sanity checks of comparing data set A against itself yielded the expected DoC score of 1.0. Considering the variability observed during the experiments on E2ESLU, several runs of the metric were performed, yielding different results. It was decided to perform ten runs of the metric on each embeddings type to get more insight on the variability. Although ten runs might seem small, it should be considered that the metric is rather expensive to compute: running it on two data sets each comprised of three embeddings of dimensions 174*512 took 10 minutes on a laptop. Using it on large datasets does not seem to be worth the high demand in computational power, considering the lack of stability of the results. Therefore, the number of ten runs was chosen with the objective to get a realistic idea of what can be expected from this metric should it be used in an operational context, rather than to get a statistically significant estimation of its variability. Results are shown in Table below.

Embedding type	Average DoC (%)	Standard deviation (% DoC)	Min DoC (%)	Max DoC (%)
HuBERT	66	2.8	63	71
Wav2Vec2	86	4.8	76	91
SpeechT5	52	5.3	44	59

Wav2Vec2 have the overall best performance it is the most inconsistent. HuBERT is more stable but its overall performance is too weak to be confidently exploited. Finally, SpeechT5 has both the most inconsistent behaviour and the worst performance in absolute value. As a side note, certain runs of Wav2Vec2 were quicker to complete (and they were usually those yielding the highest scores). This experiment illustrates the difficulty of exploiting this metric: in this experimental setting, an analysis of the representations can be made because the expected outcome is known (i.e., the datasets are supposed to be close, so the best performing representation should be chosen), whereas in a real-life use case, the metric is supposed to provide insight on the data set, which is not the case considering the disparity in behaviour of the different representations tested.

Miscellaneous similarity metrics

In (Mountrakis & Xi, 2013), the Euclidean distance is used to assess the sample-wise similarity of data set samples. Following this approach, several metrics were used on the different embeddings explored, namely cosine similarity, Euclidean distance and Manhattan distance. Euclidean was picked to match the reference method, complemented with cosine which is also commonly used to compare vectors. Manhattan was added as it tends to be considered more relevant for high-dimensional spaces.

In addition to this variety of metrics, the experimental setup computed each of them on each layer of the models used. This could not be done with the DoC due to its prohibitive computation cost. Exploring the model's layer by layer is common when working on characterizing Self-Supervised Learning models to identify what kind of information each layer identifies. In this case, the objective is slightly different, consisting in identifying if a given layer would encode more semantic information.

HuBERT is comprised of 24 layers, Wav2Vec2 and SpeechT5 of 12 layers each. None of these layers consistently exhibited better matching of the desired pairs of samples, i.e., a better Euclidean, cosine, or Manhattan compared to the other pair-wise scores.

4.7.3.4.2 Results from the MLEAP Perspective

Experiments performed with the DoC method confirmed the feasibility of using speech embeddings to apply similarity metrics. However, the metric itself exhibited an inconsistent behaviour greatly limiting its interest. Indeed, it yields a single value that is intrinsically difficult to exploit for characterizing the relation between the data sets compared, which becomes even more difficult considering how much it can vary between runs on the same pair of data sets. Moreover, it is quite costly to compute: running it on two data sets each comprised of three embeddings of dimensions 174*512 took 10 minutes, which makes it intractable for real-sized data sets.

Complementary experiments on simpler metrics tend to indicate that there is no semantic information in the embeddings used, compromising the usability of the methods on this specific use case. This highlights how the choice of embeddings is important w.r.t to the operational objectives at hand. It should be noted that speech and furthermore the use of proxy representations (i.e., embeddings) make for a difficult experimental setup. It was the only way to test the method within the scope of the MLEAP project, and was a relevant way to challenge the method. The results shown here should inform the reader on the caveats of using sample-wise similarity assessment but should not entirely dismiss the use of such methods, especially if the use case allows the direct comparison of the sample (i.e., not using proxy representations, embeddings or others).

4.7.3.5 **Off-the-shelf tools**

4.7.3.5.1 Experimental protocol

Following prior protocols, Cleanlab was tested on a toy dataset, namely MNIST. It was chosen over CIFAR-100 for internal technical constraints. Nonetheless, it remains an interesting candidate due to its good volumetry and the simplicity of its images, which is expected to enable clear results.

Cleanlab was first tested following its hands-on tutorial. The idea is that such tutorial would be the natural entry point of any user, so following it would showcase the solution's capacities while providing preliminary insight that would trigger further experimentations.
The standard data audit routine was performed on the MNIST dataset. It took roughly 6 minutes using a Google Colab configuration with 13GB RAM and GPU[70]. It diagnosed 3445 issues out of 60000 samples:

- 2602 outliers;
- 722 near duplicates;
- 120 labelling errors;
- 1 non IID sample.

The above issues are standard for any kind of data being audited with Cleanlab, but the tool also runs data-specific diagnostics. For example, on images such as MNIST, it also tries to detect:

- blurry images;
- dark images;
- light images;
- images with an odd aspect ratio;
- odd-sized images.

In the case of the MNIST data set, no occurrence of these issues was found. Outliers' detection and near-duplicates detection are performed using a k-nearest neighbours algorithm ran by the tool on the data set. If the data are tabular, the KNN algorithm directly uses the data set's features (i.e., input features for the model). On data such as images (as is the case for MNIST), the internal representation of the model is used, generally the embedding built at the second-to-last layer of the neural network. The algorithm uses the Euclidean distance and does not seem to be modifiable, although the *k* parameter can be determined by the user. The default value of 30 was used throughout experimentations. Samples very close to one another are considered near-duplicates. The case of null distance is obvious, but no documentation describes the upper threshold value of closeness. Figure 39 illustrates an example of near-duplicates for the digit class 1 of the MNIST data set. The sample's id and reference label are displayed above the image (*id* and *GL* fields, respectively). A class with a high proportion of near-duplicates will negatively affect its representativeness in the data set. Particular attention should then be provided to such class, both in terms of data collection effort (i.e., trying to gather more varied samples if possible) and training (i.e., monitoring class-wise performance at training time).

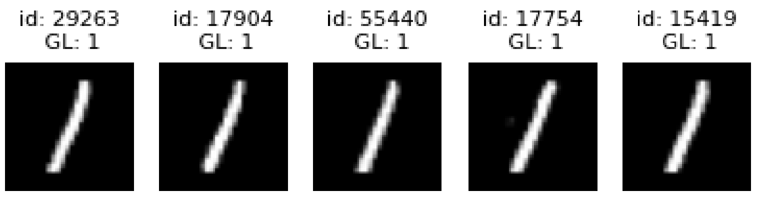


Figure 39. Example of Cleanlab's near-duplicates display.

[70] Google Colab was used because the audit could not be performed on the local setup, i.e. 8Gb RAM and no GPU.

By contrast, outliers are defined as samples that are farthest away from their neighbours. Cleanlab identifies them as the data points at the tail of the distribution of distances of the nearest neighbours, although no available documentation described the method to delimit this tail (i.e., threshold value). Figure 40 shows an example of how Cleanlab presents outliers. The first column on the left displays identified potential outliers, while the other columns contain their corresponding nearest neighbours. Sample 45525 (labelled as a digit 3) is a clear outlier and hardly exploitable sample. It would seem reasonable to expel it from the dataset and replace it with another sample. Samples 1817 and 28890 are also of very poor quality but could most probably be reliably recognized by humans. These samples are interesting to empirically question the definition of edge cases in the data set. Similarly, sample 393 is different but clearly recognizable. Such example should not be difficult for a good model but could be flagged as a sample of interest for fine-grained performance monitoring. Finally, Sample 58815 presents no visual features that significantly sets it apart from its neighbours. It is most likely an outlier caused by the parameters of the KNN algorithm and thus should be considered an artefact of the tool.

It must be noted that KNN is sensitive to biased distributions, since a class with more samples would statistically tend to be more represented amongst neighbours. This can be mitigated by using weightings methods, but this feature doesn't seem to be available in Cleanlab. It must then be handled by the developers, which implies that they must know their data and perform the appropriate Pre-processing prior to using the tool. Handling bias in distribution to enable a better KNN classification may conflict with representativeness objectives, since real-life problems are often dominated with common cases. Therefore, human assessment and comparison to the ODD and other specifications remain crucial in the exploitation of this tool.

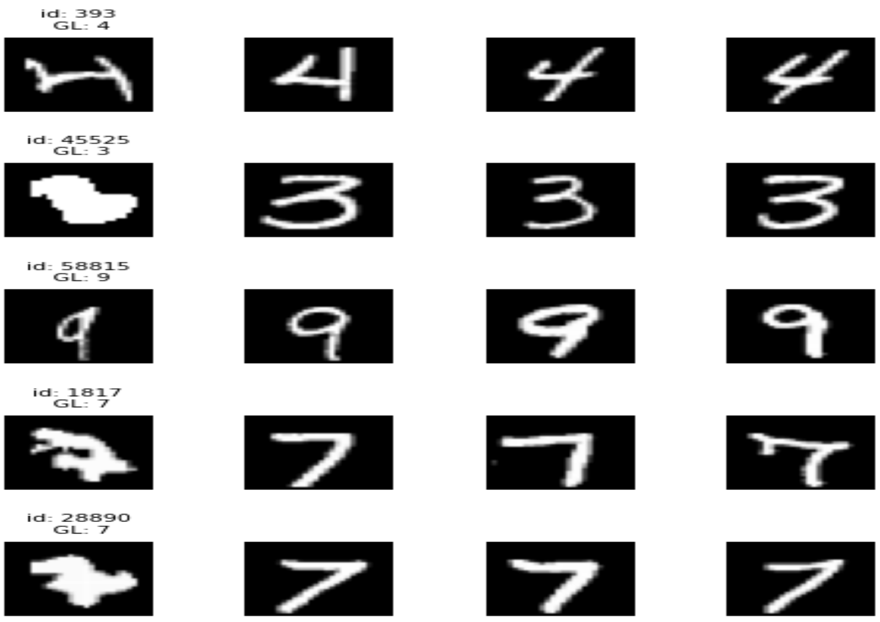


Figure 40. An example of Cleanlab's outliers presentation. The left-most column contains identified OOD samples. The three right-most columns present similar, well-labelled nearest neighbours for comparison.

Contrary to outliers and near-duplicates, labelling errors are inferred based on the feedback (i.e., results) from the model being developed on the data set. Cleanlab readily proposes two features to explore mislabelling cases. As the model predicts a distribution of probability over the classes for each

sample (of which the highest is selected as the final prediction), Cleanlab's first feature takes the highest probability (i.e., the probability of the predicted class) and normalizes it with the sample's class entropy. Entropy represents the model's uncertainty about its predictions. This *confident weighted entropy score* can be displayed for any sample. As a complement, the second feature identifies mislabels and suggests alternative ones. To do so, the predicted probability distribution of each sample is grouped by class and averaged.
These averaged values are used as a confidence index to decide whether a sample should be considered mislabelled: if the confidence index of the reference label is the highest, the sample is considered correctly labelled. On the other hand, if the confidence index of another class (or several others) is higher, then the sample will be considered mislabelled, and the class with the highest confidence index will be suggested as the correct class. An illustrative example is described below:

- Consider a validation dataset of 3 « dog » images and 4 « cat » images. A binary classifier is trained on a separate dataset of dog or cat images, is then fed these 7 validation images and yields the following predictions:

Image name (+ reference label)	Predicted class probabilities	
Dog_1	dog = 0.6	cat = 0.4
Dog_2	dog = 0.2	cat = 0.8
Dog_3	dog = 0.5	cat = 0.5
Cat_1	dog = 0.1	cat = 0.9
Cat_2	dog = 0.1	cat = 0.9
Cat_3	dog = 0.9	cat = 0.1
Cat_4	dog = 0.9	cat = 0.1

The confidence indices are:

CI_dog = $\frac{0.6+0.2+0.5}{3}$ = 0.43

CI_cat = $\frac{0.9+0.9+0.1+0.1}{4}$ = 0.5

Then, each prediction is compared to the classes' confidence indices. A confusion matrix is derived, using these indices as thresholds. For instance, Dog_1 has a « dog » probability of 0.6 and its class is dog. Since 0.6 ≥ 0.43, it is counted in the yellow cell of the matrix. Dog_2 has predicted « dog » probability = 0.2 ≤ 0.43 and predicted « cat » probability = 0.8 ≥ 0.5, so Dog_2 is counted in the cyan cell. Similarly, Cat_3 and Cat_4 will be counted in the grey cell and Cat_1 and Cat_2 in the red cell. The diagonal of the resulting confusion matrix counts the correctly labelled samples (Dog_1, Dog_3, Cat_3, Cat_4) while the off-diagonals count the inferred mislabelling (Dog_2, Cat_1, Cat_2) (cf. Table 11).

Reference/Predicted	Dog	Cat
Dog	2	1
Cat	2	2

Table 13 Confusion matrix of the example above.

To characterize an entire dataset without contaminating the results (i.e., feeding the model with examples used for its training), Cleanlab uses cross-validation. Since cross-validation implies averaging the performance of several models, it also provides an additional robustness to the analysis. It should be noted that, in the case where a sample would yield low probabilities over all classes (i.e., the model is very uncertain about how to classify the sample, which could be a typical case for an outlier), the sample would not appear in the matrix. This is a desired behaviour of the method, since it provides robustness to outliers, the detection of which is handled separately by dedicated methods.

Such homogeneous probability distributions are the main limitations of the mislabelling detection method: they may either filter out samples as outliers or yield uncertain confidence indices that may result in false alarms. Therefore, it is important to deploy it on a well-trained model, close to its final maturity, rather than using it as a preliminary step. In this regard, it is a tool more fit for jointly tuning a model and a data sets rather than to qualify a data set from scratch. Certain publications discuss ways to increase the robustness of this base method, but their impact and scope could not be clearly established.

Figure 41 shows an example of how Cleanlab presents potential mislabelling. GL stands for Given Label, i.e., the reference label as it appears in the dataset. SL stands for Suggested Label, i.e., a proposed replacement label based on Cleanlab's analysis of the model prediction.

In particular, sample 19967 is labelled as the digit 4, but Cleanlab suggests it might be mislabelled and proposes to change it to digit 7, which indeed seems like a better choice. Samples 48882 and 23687 also look like accurately identified mislabels. On the contrary, sample 37914 is referenced as an 8 and Cleanlab suggests changing it to a 4. This is clearly an error, but it can be noted that the digit is angled about 45 degrees counter-clockwise, which may indicate a sensitivity of the model to rotated images. If such phenomena were consistently observed on a larger scale throughout the data set, it could motivate a data augmentation effort to include rotated samples. By contrast, samples 308 and 183 display ambiguous samples, probably not mislabelled but rather difficult instances that can be considered edge cases. They may be used an entry points to further investigate the model's behaviour at these limits. They could also be compared to outliers identified by Cleanlab's dedicated feature, to see if they overlap or complement. Finally, samples 34139, 42317 and 42884 are clear errors to the human eye but it is also easy to see what confused the model. These samples may enable better insight on the decision boundaries set by the model.

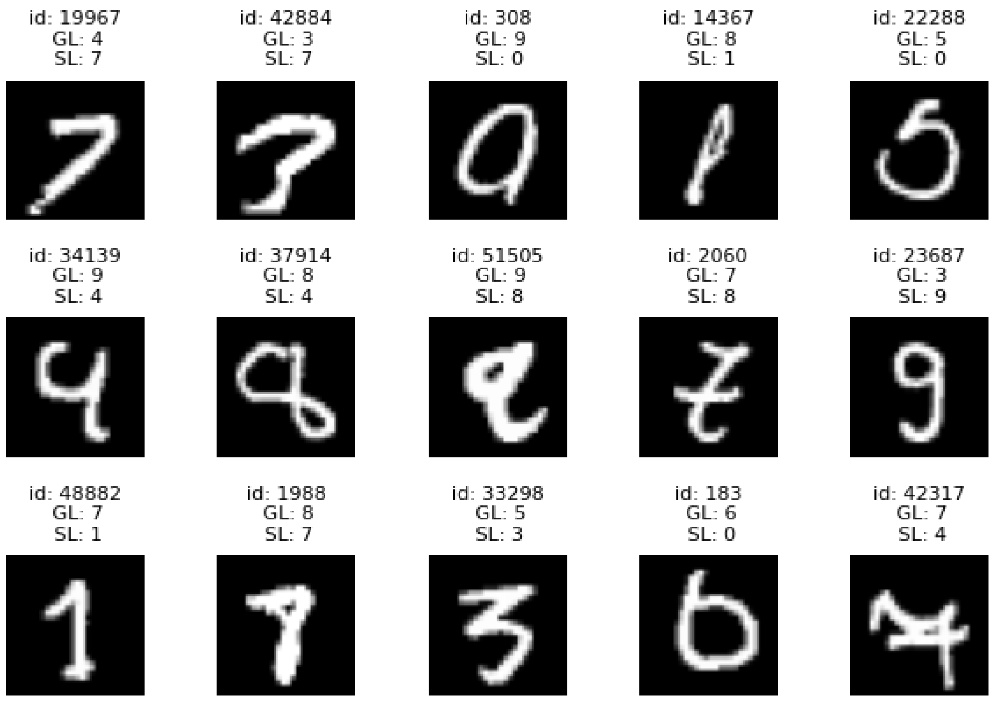
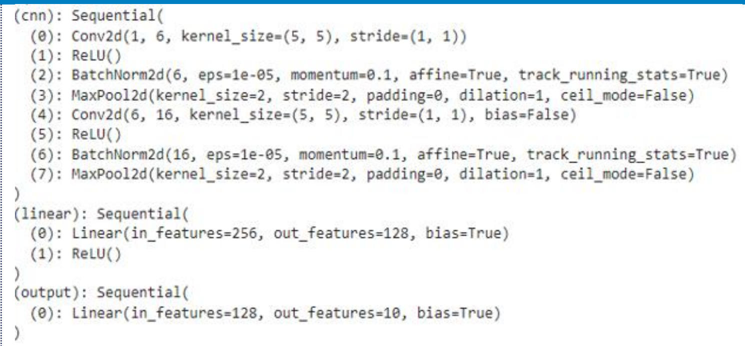


Figure 41. This is an example of Cleanlab's mislabelling cases presentation. Each image is displayed along with its ID. GL stands for Given Label, i.e., the reference label, while SL stands for Suggested Label, i.e., Cleanlab's suggestion.

To get a more empirical insight on the interaction between a trained model and Cleanlab's analyses, a contrastive study was performed. A second classifier was trained to reach a lower level of performance compared to the first one. Table 14 describes the architectures and performance levels of both classifiers, A being the first one (used for the experiments reported so far) and B the « degraded » classifier.

Table 14. Overview of the classifier's architecture and performance.

Id	Architecture	Performance (accuracy)

A	 (cnn): Sequential(
	  (0): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
	  (1): ReLU()
	  (2): BatchNorm2d(6, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
	  (4): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1), bias=False)
	  (5): ReLU()
	  (6): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
	)
	(linear): Sequential(
	  (0): Linear(in_features=256, out_features=128, bias=True)
	  (1): ReLU()
	)
	(output): Sequential(
	  (0): Linear(in_features=128, out_features=10, bias=True)
	)
		97%

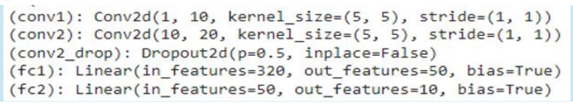
B	 (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1)) (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1)) (conv2_drop): Dropout2d(p=0.5, inplace=False) (fc1): Linear(in_features=320, out_features=50, bias=True) (fc2): Linear(in_features=50, out_features=10, bias=True)	75%

Figure 42 contrasts the outliers identified by Cleanlab based on classifiers A and B. Outliers uncovered by classifier A have already been commented and exhibit a mix of possible artefacts, hard cases, edge cases and arguably unexploitable samples. On the other hand, outliers identified by classifier B are more relatable to artefacts, as they would pose no problem to a human and are generally clear. They seem to mostly correspond to « hard » samples for the model, reflecting its poor performance.

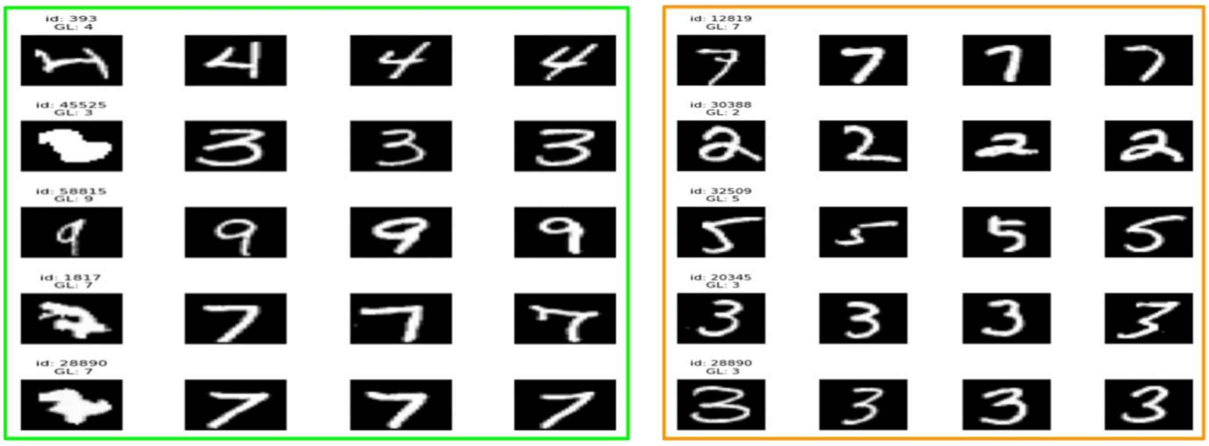


Figure 42. Comparison of the outliers found using classifier A (green box) and B (orange box).

Figure 43 shows the near-duplicates identified through classifiers A and B. The similarity is blatant in the former case: same class, slight tilt, comparable thickness. On the contrary, classifier B's results mix several classes with no apparent coherence (i.e., 1 and 0 are radically different forms). Once again, the results displayed by Cleanlab are mostly a manifestation of the model’s performance.

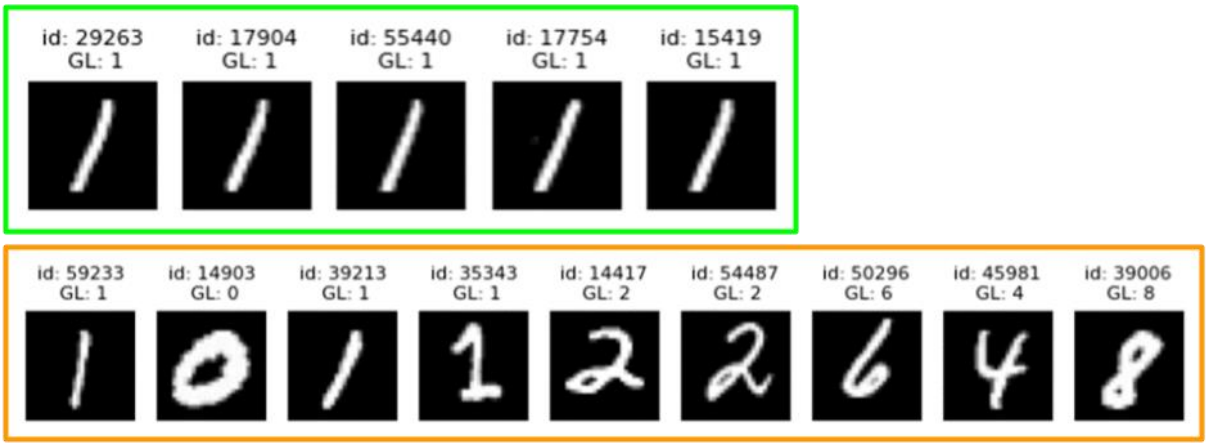


Figure 43. Comparison of the near-duplicates found using classifier A (green box) and B (orange box).

Figure 44 compares the mislabelling found in the dataset using classifier A and B. In the case of classifier, A, the results are varied and show hard or possible edge cases. As for outliers, the results on classifier B are more akin to « hard cases » for the classifier, but are unambiguous for humans. The similarity between outliers and mislabelling in the case of classifier B, together with the apparent focus on class digit 6 (and suggestions swaying between class digit 3 and 7) could indicate that classifier B would tend to predict rather homogeneous probability distributions, especially for class digit 6. In this case, as discussed earlier, the predicted individual probabilities would be close to the confidence indices and trigger false alarms, some of which are independently detected by the outlier detection module.

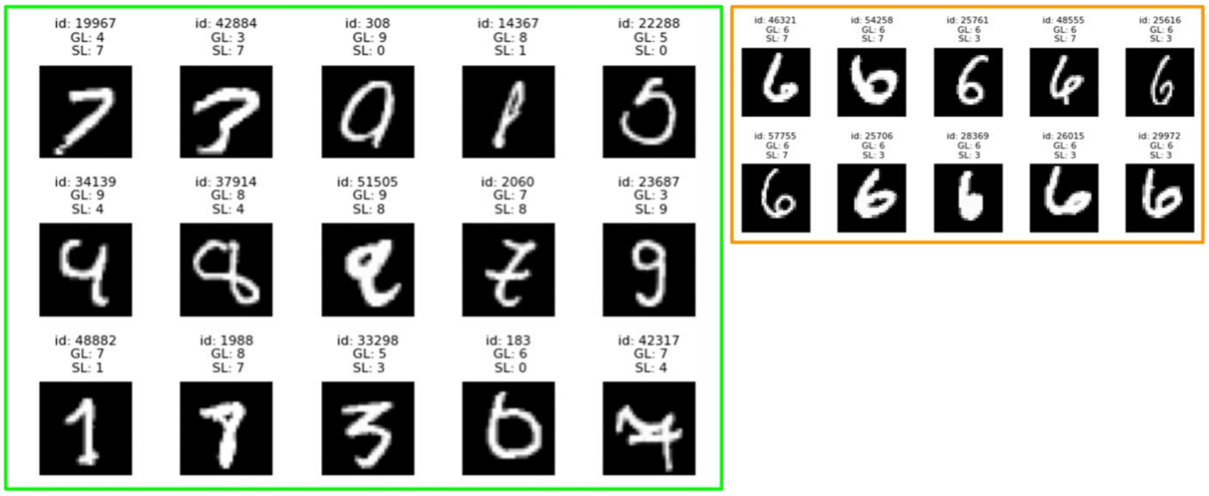


Figure 44 Comparison of the mislabelling cases found using classifiers A (green box) and B (orange box).

In general, the results proposed by Cleanlab are intuitive to analyse. On the tested use case, i.e., image classification, the visualization rests on presenting actual samples. This choice enables up-close inspection for precise assessment, but lacks a more synthetic summary of the trends that can be found in the issues identified. Figure 45 shows a proposition of such overview in the form of a heatmap of the contingent mislabelled samples. The reader should note that this heatmap was obtained using classifier A but on a different training run (with slightly different accuracy, ~99%), so it should not be related to the previous results. Contrary to previous experiments, this proposal does not aim at highlighting the influence of classifiers on the resulting Cleanlab report. Rather, it should be seen as a mock-up, to show that it is possible and potentially useful to exploit the results of off-the-shelf tools beyond their embedded visualization solutions. Such approach can be viewed as a soft intermediate between the strict use of the solution in its intended purpose and full in-house tooling.

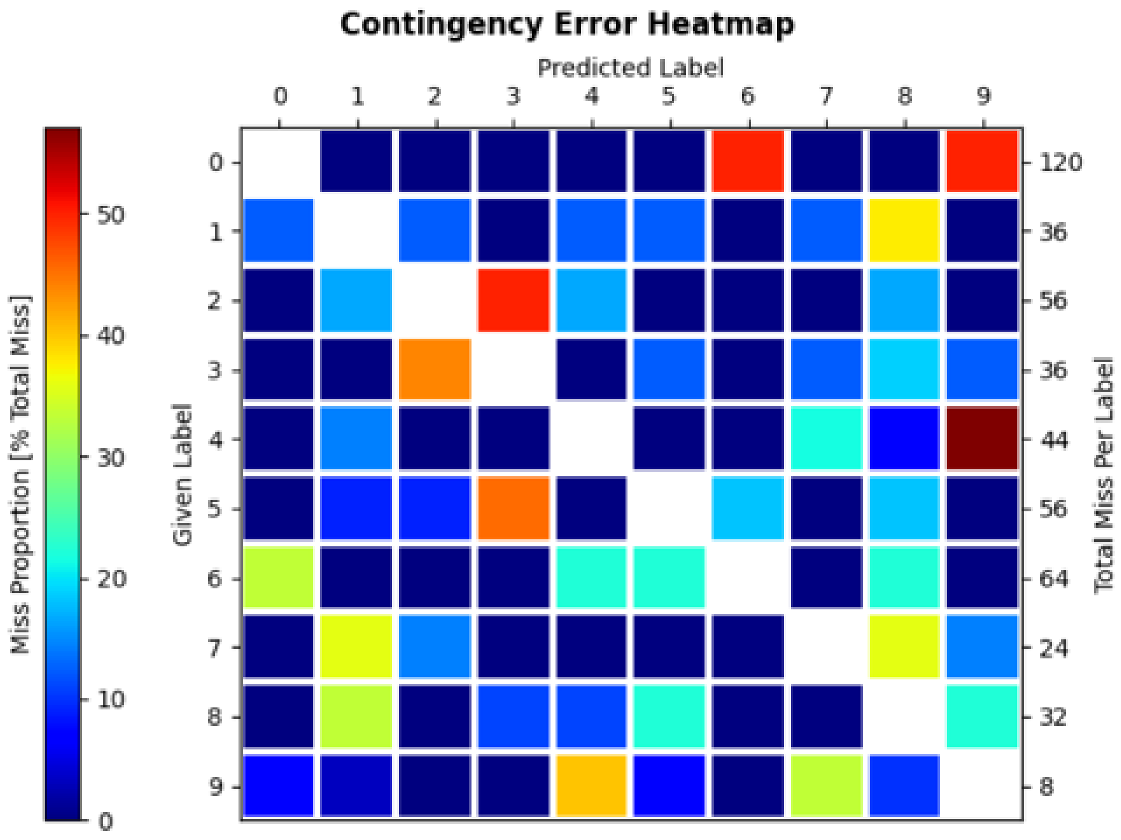


Figure 45. Contingency heatmap of the mislabelling cases identified by Cleanlab.

The map should be read row-wise, each showing the reference label, with each cell displaying the ratio of mislabels identified for the associated class (hotter cell, higher ratio). On the right, the total number of associated mislabels is indicated so the reader is reminded of the detailed volume of mislabels, along with knowing which ratios are directly comparable. As an example, it can be observed that class 0 and class 5 exhibit the same behaviour, with a tendency to be confused with one class only (6 and 5 respectively). By contrast, class 8 exhibits a similar behaviour in terms of ratio, but it significantly lowers sample count should nuance the analysis. Finally, class 6 has a similar sample count to class 0 and 5 but seems to be less robust, with its confusion spanning several classes (1, 2, 4 and 8).

4.7.3.5.2 Results from the MLEAP Perspective

Cleanlab appears to be a useful tool for data qualification. Its modules for outliers and mislabelling detection are interesting for completeness and representativeness assessment in the sense where outliers and mislabelling are complementary methods that can highlights hard cases, edge cases, mislabelling and poor-quality examples that could possibly be discarded. The limitation is that diagnosing an entire dataset requires applying a cross-validation protocol, which in turn is time-consuming and requires to know the exact architecture that will be used for the production model, without leveraging any resulting model information (i.e., not porting the weights or hyperparameters). This adds a potentially costly step before model training (depending on the model's complexity). By contrast, applying Cleanlab on a production model would require using only the test set, providing an only a partial feedback. Should the diagnostic require to backtrack to data set constitution (by either discarding samples, augmenting the data set or collecting new samples), retraining would be necessary and a completely new test set should be built. Moreover, such use would heavily depend on the model used and would thus be most efficient if relying on a mature model (i.e., following loop 4 in Figure 78), which implies it takes place late in the model's development cycle. This could challenge other assessment such as generalisation capabilities. In addition, while some highlighted samples may appear as intrinsically difficult, others may be mere artefacts of the system limitation and should actually be learned correctly. In this regard, tools like Cleanlab cannot be used alone but must be backed up by other assessment methods based on upstream technical specifications regarding data requirements and expert inspection (i.e., human and therefore costly). Moreover, the methods embedded in Cleanlab have some inherent limitations that must be known and accounted for by users to ensure correct analysis of the results. Therefore, such tool cannot be a substitute to an in-depth knowledge of the data set and a thorough analysis of its results. In particular, throughout experimentations, it consistently detected one non-IID. sample, which is impossible in the MNIST dataset. Finally, Cleanlab is not a trivial tool to use since it could not run the MNIST dataset on a desktop. Appropriate resources must be allocated to its use, introducing a trade-off between its infrastructural requirements and the insight it provides on the data.

4.7.3.6 **Neuron Coverage Analysis**

4.7.3.6.1 Experimental protocols

Neuron coverage is defined as the proportion of neurons activated by the processing of an input. A neuron is considered activated when its output value exceeds a given threshold. Other aspects of the paper are neglected to focus on the insight provided by this simple mechanism.

The neuron coverage of data set can be inferred by aggregating the coverage of its samples. It is then possible to compare data sets. As a first step, the neural network used to experiment on Cleanlab was re-used, still on the MNIST data set. Inconsistencies in coverage would be expected to reflect a lack of representativeness of one of the datasets. The comparison is made between the MNIST test set and a validation set extracted from the original train set with a 90%-10% proportion (i.e., 6000 images in the validation set).

Contrary to previous experiments, a validation set was devised here because the quality of the results provided by the neuron coverage depends heavily on the threshold value defined to consider a neuron as activated. It is, therefore, necessary to characterise its influence. Two methods of quantifying coverage have been experimented:

- Mean unique coverage: For each input, the activation of each neuron is accumulated as a binary information via an OR function. Once all inputs have been processed, coverage in computed by averaging over the input. This method shows the proportion of neurons activated at least once over the entire processing of the dataset, i.e., if unique coverage = 0.6 for threshold = 0.7, then 60% of the neurons in the network output a value > 0.7 at least on one sample, and 40% never did.
- Mean average coverage: For each input, the coverage i.e., ratio of activation is computed layer-wise then averaged over the input for the whole data set. This method indicates the average percentage of network activation for a given threshold. For instance, if coverage = 0.2 for threshold = 0.6, then 20% of the neurons in the network output a value > 0.6 over the data set.

Both methods are complementary: the mean unique coverage is more radical but gives an indication of the model's overall saturation, while the mean average coverage helps build an expectation of its activation patterns.

Moreover, two points of view have been adopted: the general, model-wise coverage and a more detailed, layer-wise representation. Results are displayed in Figure 46 and Figure 47 (unique coverage and mean average, respectively) for the validation set and Figure 48 and Figure 49 for the test set. Note how, for a given abscissa, the ordinates of any of the « model » plot is actually the average of the corresponding curves in the « layers » plot.

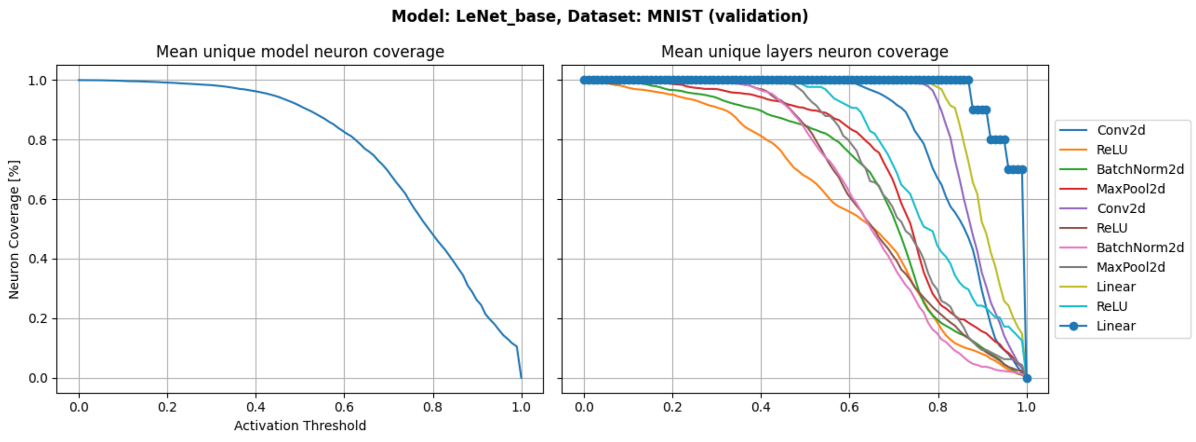


Figure 46. Unique coverage for the MNIST validation set.

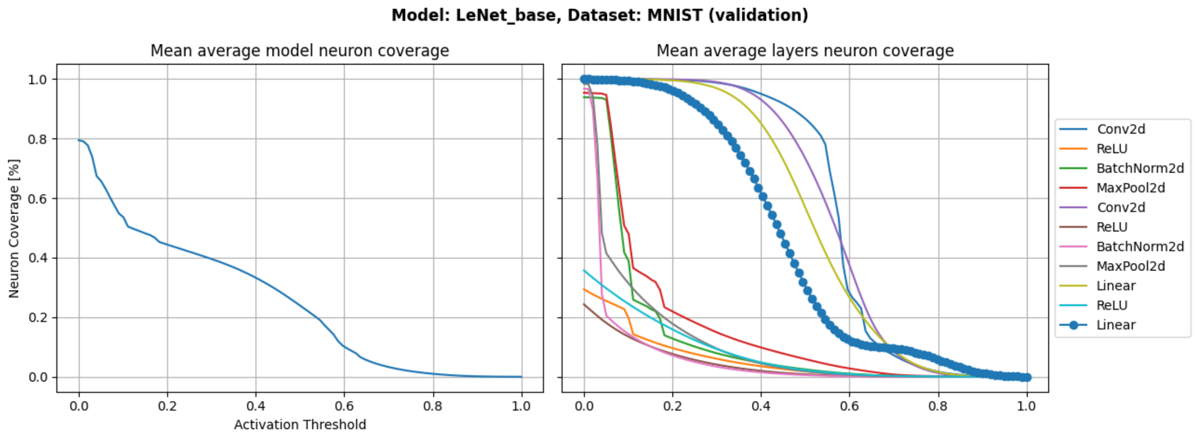


Figure 47. Average coverage for the MNIST validation set.

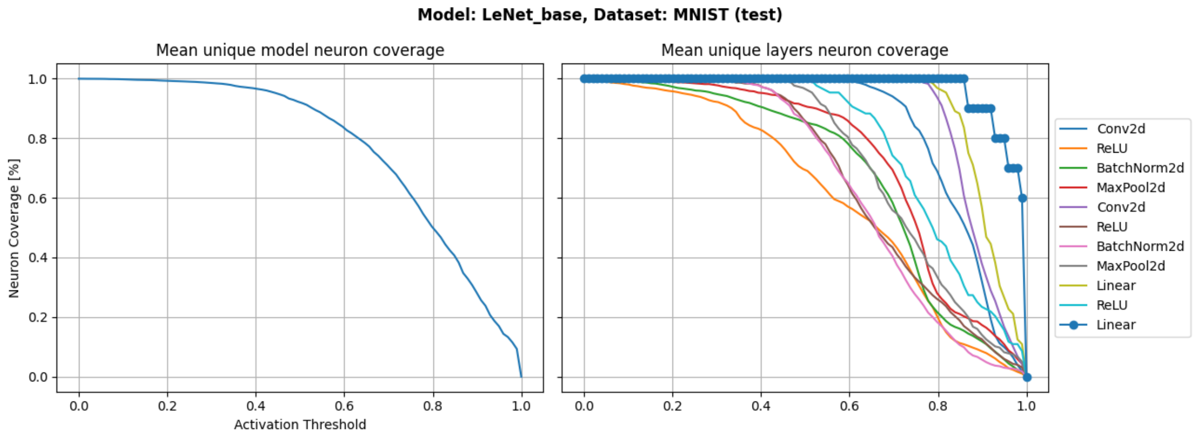


Figure 48. Unique coverage for the MNIST test set.

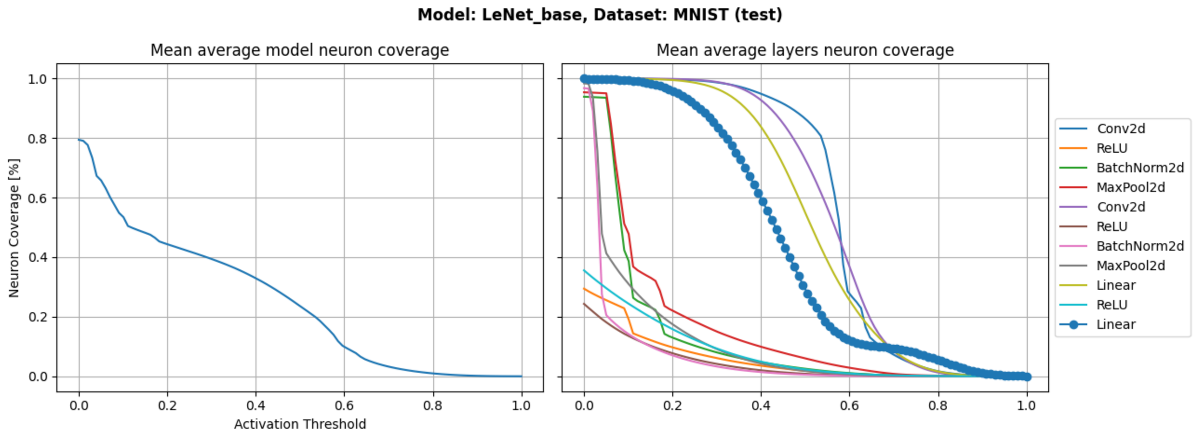


Figure 49. Average coverage for the MNIST test set.

Both mean unique and mean average coverage exhibit a tendency of the coverage to drop more abruptly in the regularization layers, i.e., ReLU, MaxPool, BatchNorm, while the feature extraction layers, i.e., conv2d, linear, hold a high coverage up to a higher threshold, although they also sharply drop at one point. In addition, the difference between the validation and the test sets is extremely small and is only in terms of amplitude, which suggests that the activation patterns learned worked equally well on the validation and test set, indicating close similarity (and thus no issue related to completeness or representativeness).

As a second step for analysis, coverage was observed for correct vs incorrect prediction (hit and miss) for each class. As a complement, the average coverage over the entire data set is also plotted, as a sort of baseline. Results are displayed in Figure 85. All classes exhibited extremely similar curves, which is why only class 9 is represented, for readability. For the same reason, the results presented are only those obtained on the test set. Besides, computation was performed using mean average coverage, as mean unique coverage is too permissive to provide insightful results.

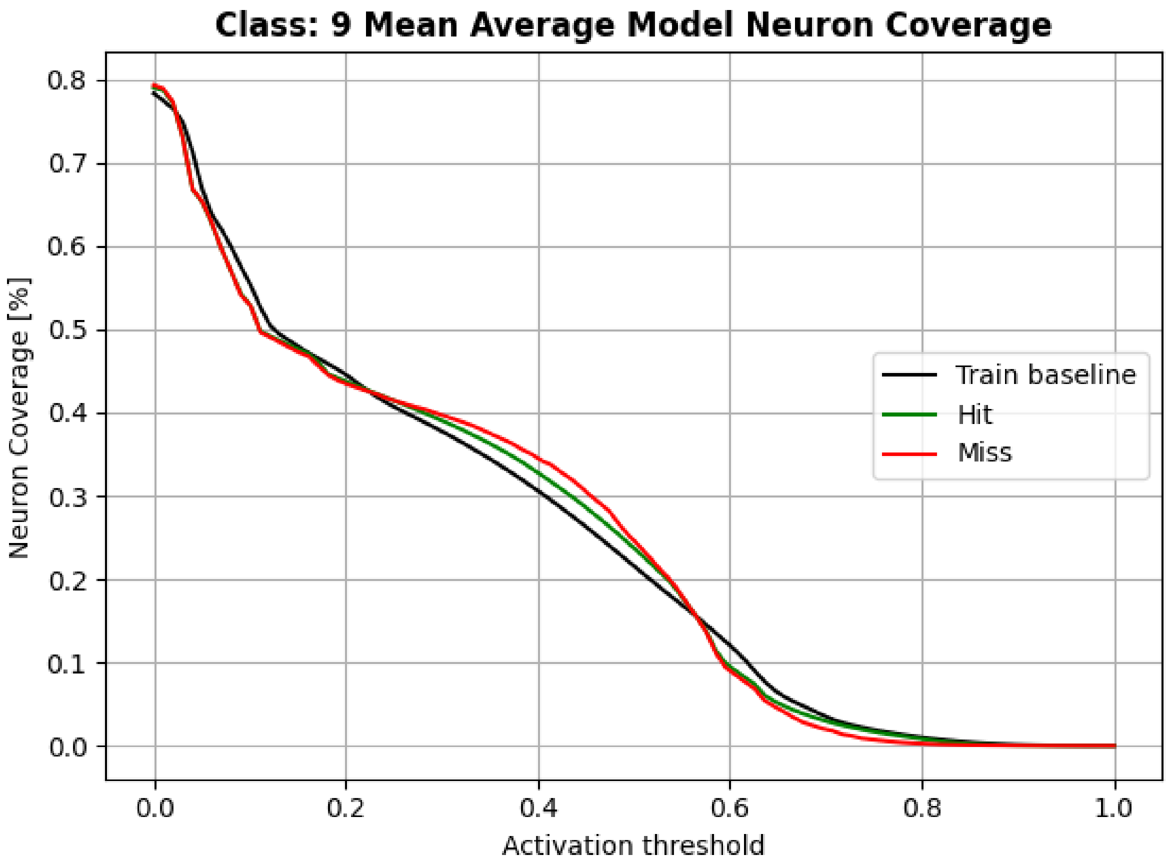


Figure 50. Hit vs Miss vs training coverage on the test set.

For the lowest and highest activation thresholds, the coverage of hit and miss tends to follow the baseline value. However, between (roughly) 0.3 and 0.5, the three trends diverge, with both hit and miss having higher coverage and miss being the most unstable (although consistently reaching the highest value). Miss having higher coverage is intuitively intriguing, as it implies that the more neurons a sample activates compared to the mean activation, the more likely it is to output an incorrect prediction. Nonetheless, this is a potentially useful monitoring tool to anticipate outliers and probe the system at the sample scale, unravelling possible problems in the data.

As final experiment, an attempt at visualizing the layer-wise activation was made. The objective was to dive at the finest possible granularity of activation to try and go beyond means and coarse trends. Figure 51 and Figure 52 show the results obtained. Again, mean average activation is the only one used because mean unique activation is too permissive. Results obtained on the validation set and on all classes are not presented to preserve readability. Instead, only classes 1 and 2 are shown, the former being the best performing one and the latter the worst performing one. Similarly, the study of thresholds has been reduced to 0.25 (where hit and miss follow the baseline) and 0.45 (where the three diverge) to focus on a minimal contrastive approach, for clarity. Following the information gathered during the first experiments, the figures only display the 4 feature extraction layers, leaving aside regularization. Layers are displayed in processing order, input entering on the left and travelling from to right. Layer n°8 is supposed to be a 128-dimensional vector that was reshaped as a 16x16 matrix for representation convenience. Any perceived geometrical pattern within this matrix is an artefact and must not be interpreted.

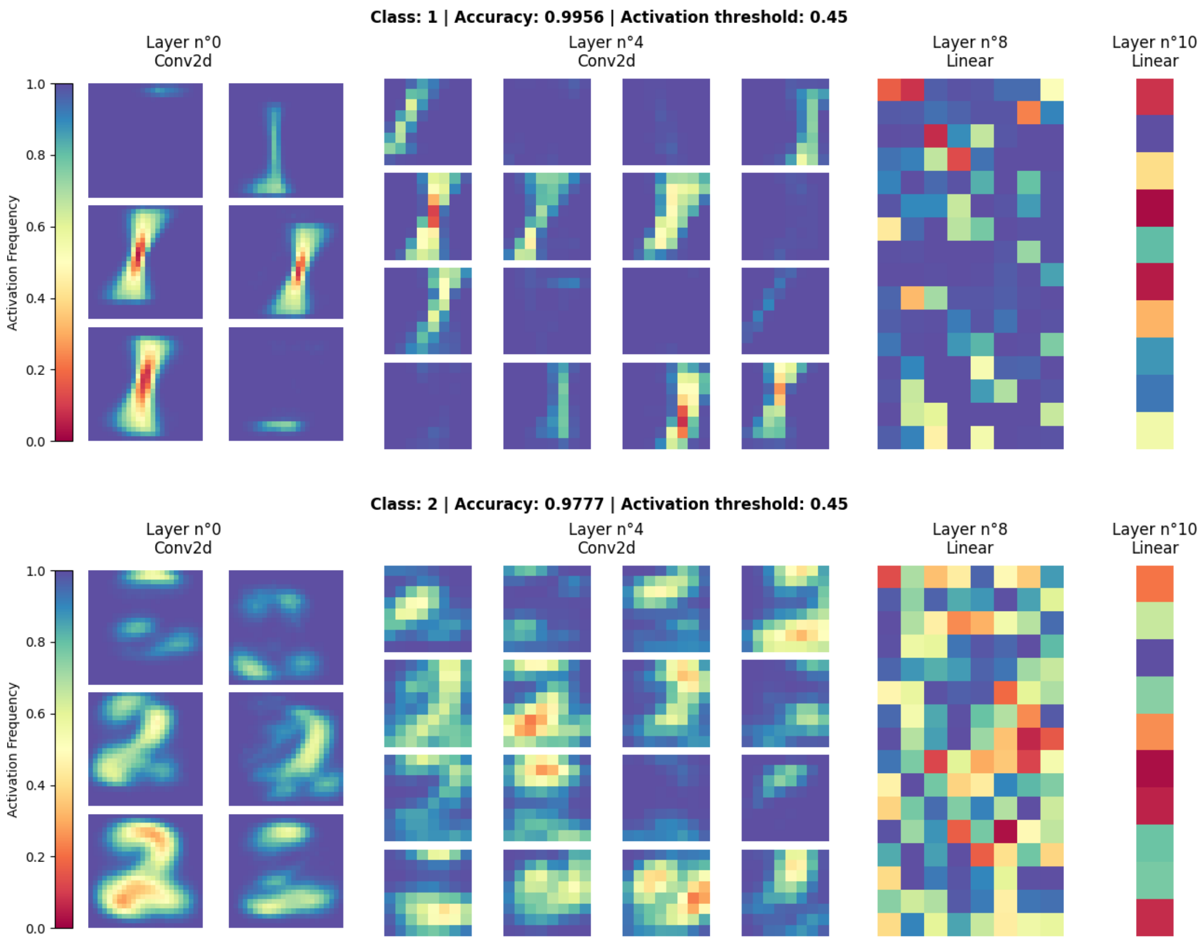


Figure 51. Layer-wise Mean average neuron coverage for the main layers of the classifier, for the best-performing class (digit 1) and the worst-performing class (digit 2), with activation threshold = 0.45 (diverging trend).

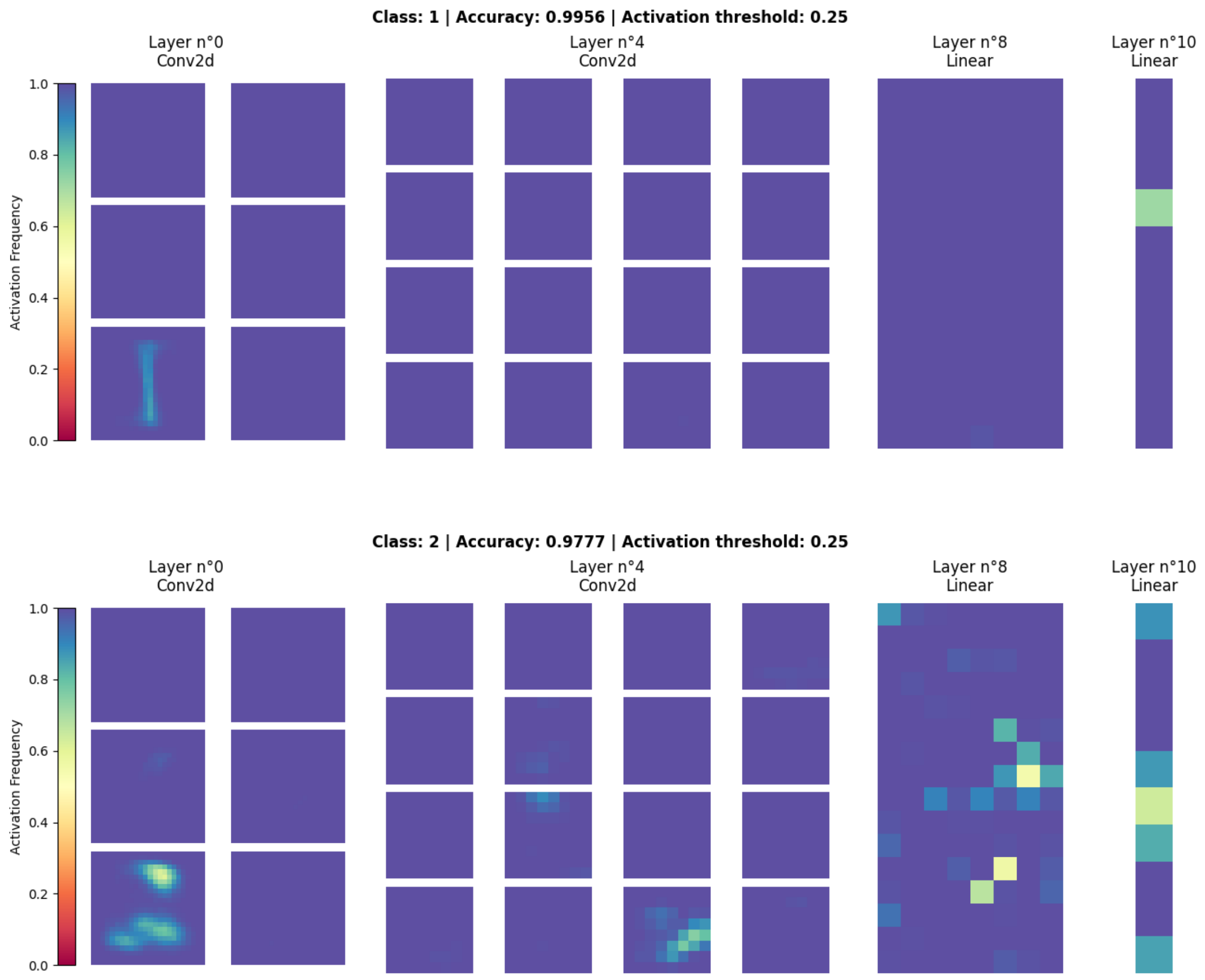


Figure 52. Layer-wise Mean average neuron coverage for the main layers of the classifier, for the best-performing class (1) and the worst-performing class (2), with activation threshold = 0.25 (converging trend).

The influence of the threshold on the coverage is clearly visible, with 0.45 (diverging curves) displaying more heterogeneous activations. Regarding classes, 1 (best performing) consistently displays higher and more homogeneous activations compared to 2, which tends to support the previous hypothesis of miss being more unstable, considering the worst performing, i.e., most error-prone class exhibits the same behaviour.

4.7.3.6.2 Results from the MLEAP Perspective

Neuron activation is a tool with potential as it can be used for different granularity of insight on the model’s behaviour. By focusing on the actual reaction that an input triggers within the model, this model-based, model-centric method acts as an interesting complement to model-based, sample-centric approaches like Cleanlab for example.

Nonetheless, it is a complex tool that requires some competence to be used effectively. In particular, the choice of the activation threshold will greatly influence the resulting analysis. This influence must be well understood beforehand and requires some preliminary experimentation as well as some expertise. Shedding light on these points was the focus of the experiments. Overall, neuron coverage seems particularly useful for representativeness assessment, in particular outlier and edge cases detection, as it tends to highlight stereotypical behaviours that can be used as baselines to compare to more specific inputs (possibly identified through other methods e.g., Cleanlab, as discussed)

4.7.3.7 **Feature Space Characterization**

4.7.3.7.1 Experimental protocols

For this experiment, the model used for testing Cleanlab was reused. Following the protocol of the source paper, the representation (i.e., embedding) of the second-to-last layer of the network is observed. The four metrics described are computed using the trained model, on the train set and the test set. The comparison between both results using the train as a reference baseline is expected to shed light on possible shifts that could be caused by completeness or representativeness issues in the data set. Equivalence partitioning is reported in

Table 15. Each class should have a score that would ideally tend to 1. In this case, values are close to this ideal and homogeneous both across classes and across sets. This confirms the known quality of the MNIST data set. As expected, this particular metric is not very informative in this case, but was reported for exhaustivity.

Table 15. Equivalence partitioning scores.

Class	Train EP score	Test EP score
0	0.987	0.980
1	1.123	1.135
2	0.993	1.032
3	1.021	1.010
4	0.973	0.982
5	0.903	0.892
6	0.986	0.958
7	1.044	1.028
8	0.975	0.974
9	0.991	1.009

The second metric to compute is centroid positioning. This metric depends on a user-defined radius parameter. Different values have thus been explored. The results are in Figure 53 (train) and Figure 54 (test).

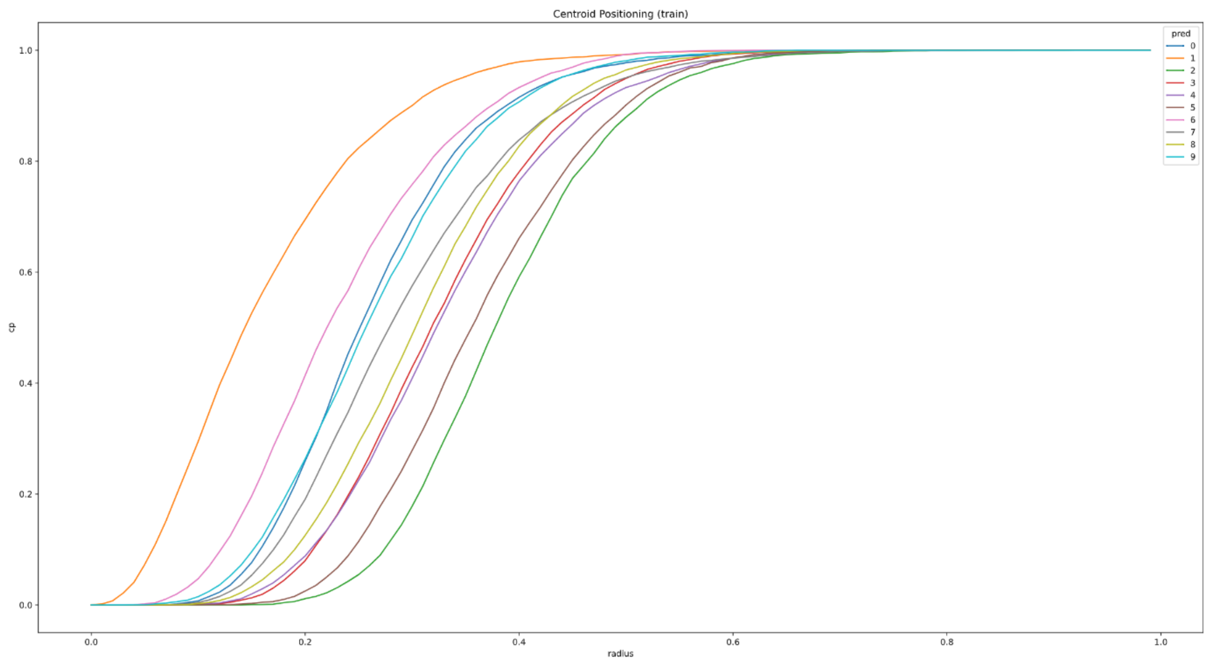


Figure 53. Centroid positioning on the train set, for different radius.

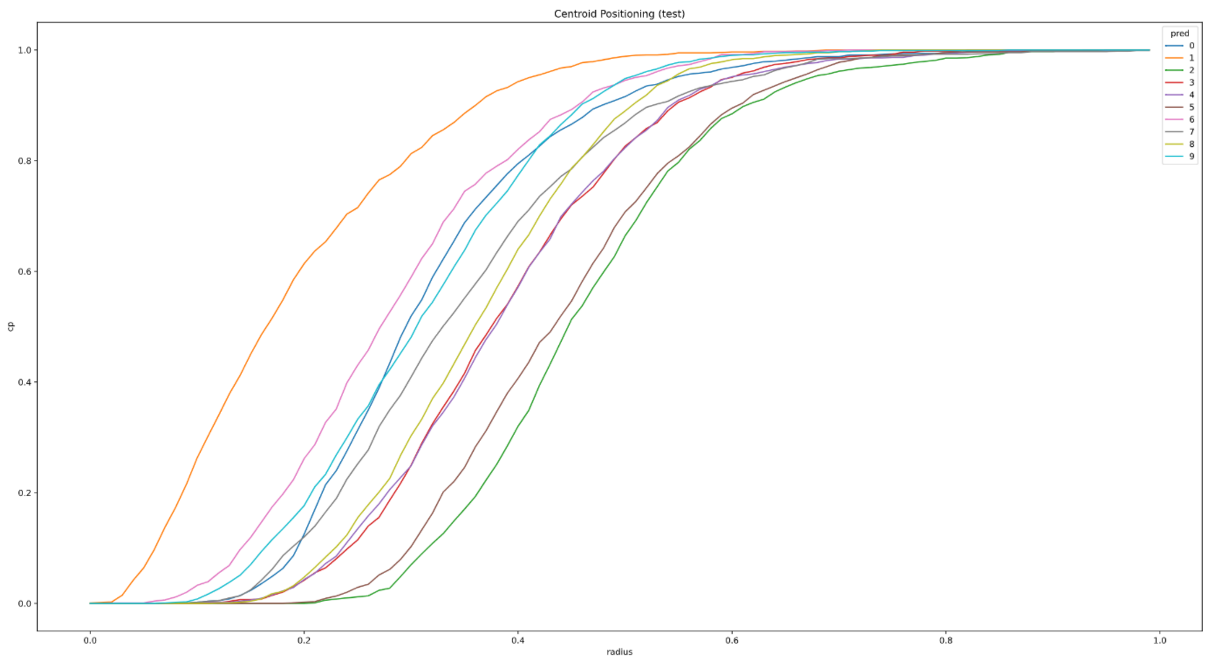


Figure 54. Centroid positioning on the test set, for different radius.

The curves of the test set are less smooth due to lower sample counts. Nonetheless, the trends are the same on both sets, with similar x-axis progress (although slightly slower on the test set). The centroid positioning score (y-axis) should ideally tend to zero for all classes[71]. Such state is reached only on a very small portion of the graph, for a radius < 0.1. The quick divergence of the positioning

[71] According to the reference paper.

scores could not be explained. Concretely, a small cp score means there are few samples within the neighbourhood of the centroid, indicating a well-spread feature space (the larger the radius for a small cp, the better the spread). In this case, the ideal case is maintained only for a very small radius and quickly becomes unreachable. This would tend to indicate that the feature space is not homogeneous. Considering the good performance of the model and the quality of the data set, this metric seems difficult to exploit directly.

The next metric is boundary conditioning. It is based on the confidence score (e.g., SoftMax results) for each sample of the data set and relies on a user-defined range. Samples whose confidence scores are within the range are considered at the decision boundary (adjacent with the second-best confidence score), and thus weakly classified. To better illustrate that idea and choose an appropriate range, the distribution of best and second-best confidence scores was explored. It is shown in Figure 55 (train) and Figure 56 (test).

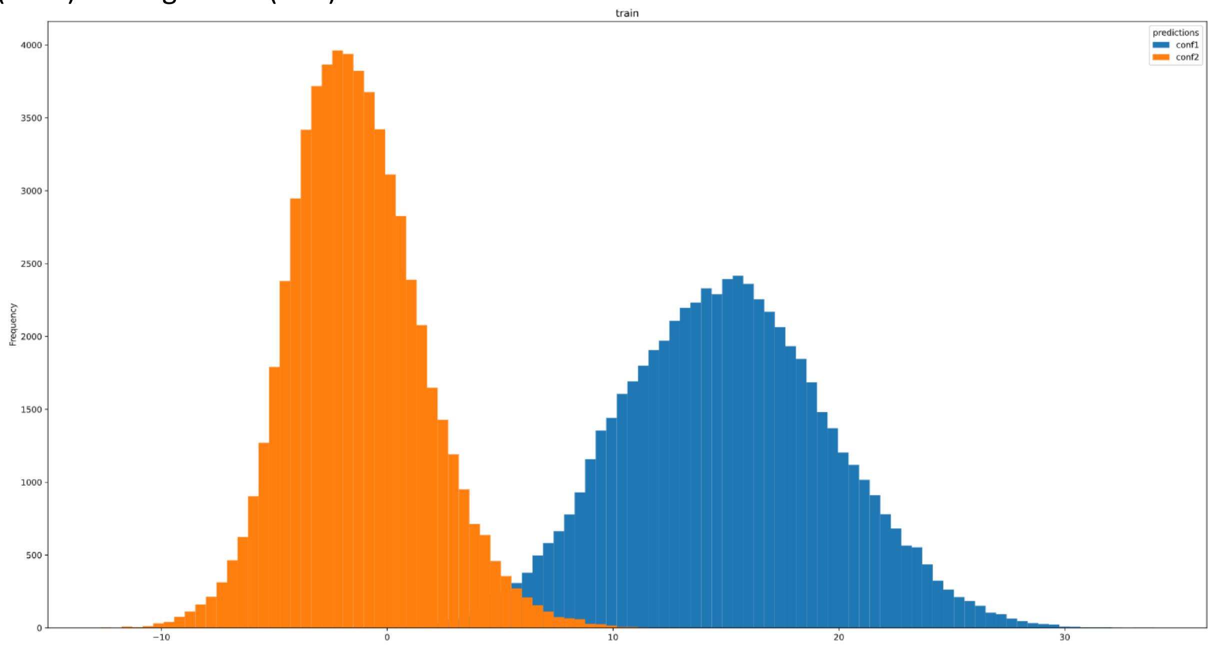


Figure 55. Confidence score distribution for the MNIST train set. In blue, the best scores (i.e., predicted classes), in orange the second-best scores.

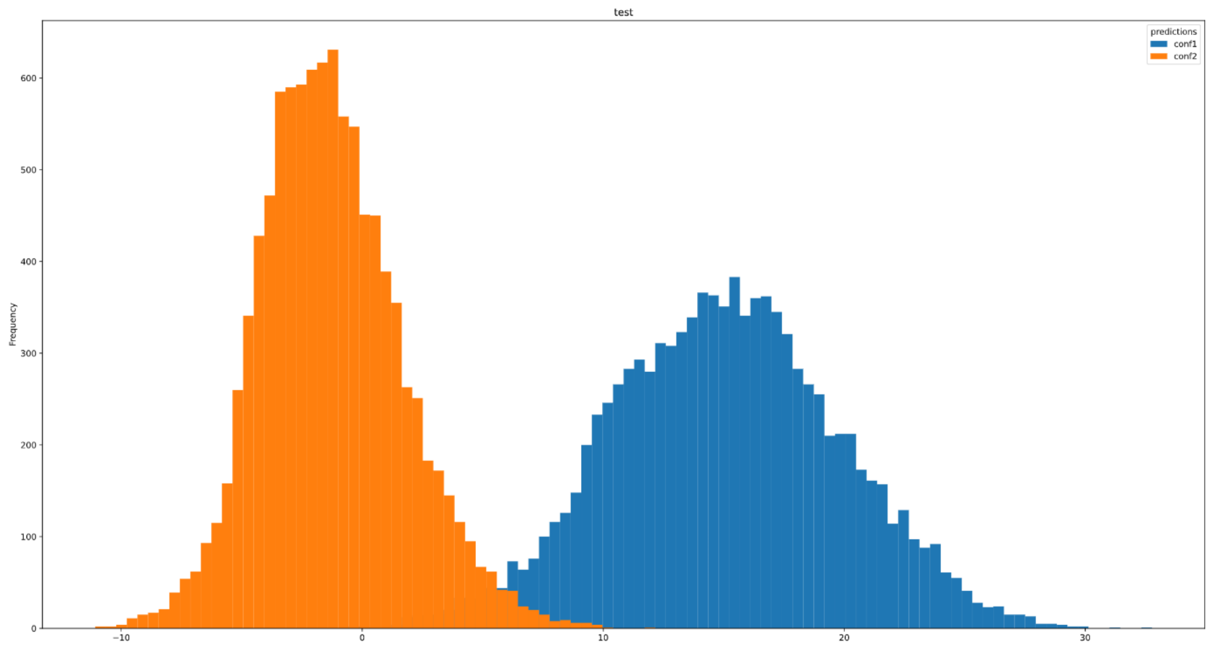


Figure 56. Confidence score distribution for the MNIST test set. The best scores (i.e., predicted classes) are in blue, and the second-best scores are in orange.

Both the top and second-best distributions are centred around the same confidence scores for the train and the test, which indicates the consistency of the model's prediction across sets. The boundary is also clearly visible, around the confidence score 5. The interest of using a range is also clear, as both distributions smoothly overlap rather than displaying a clear transition from one to the other at a given value. In light of these elements, it would seem reasonable to use the [4;6] range for this metric. As a next step, it might be insightful to visualize the samples that lie within the defined range. A corresponding proposal is displayed in Figure 57. Due to the exploratory nature of the experiment, the range used is wider than the previously proposed one, i.e. [1;10] instead of [4;6]. Each row is associated with a value within the defined range and shows three images (columns). The middle column shows a random sample whose confidence score corresponds to the target value of the range. On the left, a high confidence score sample of the top (predicted) class for the centre sample is shown. Similarly on the right, a high confidence score sample of the second-best class for the centre sample is shown.

To complement these information pieces, the sequence of values above each sample should be read as follows: reference_class, top_predicted_class (confidence_score), second_best_predicted_class (confidence_score). For instance, the first row of the figure should be read as follows: the middle sample has reference class 0, the top predicted class is 8 with confidence = 0.02 (close to target 0.00) and the second best is 9 with confidence = -1.62. Thus, the left sample shows a high confidence 8 (score = 22.61) and the right sample shows a high confidence 9 (score = 19.90). This visualization is inspired by the Cleanlab interface and their analysis may be combined for more insight. As discussed in the associated section, Cleanlab may be used to identify potential edge or corner case as well as hard cases. The approach described here follows the same logic, which could be useful to cross-identify the most problematic samples and refine the understanding of the problem, using the boundary information (i.e., between which two classes does the model hesitates). In the meantime, the approach similarly shares Cleanlab's limitation, i.e., it should be used on a mature model.

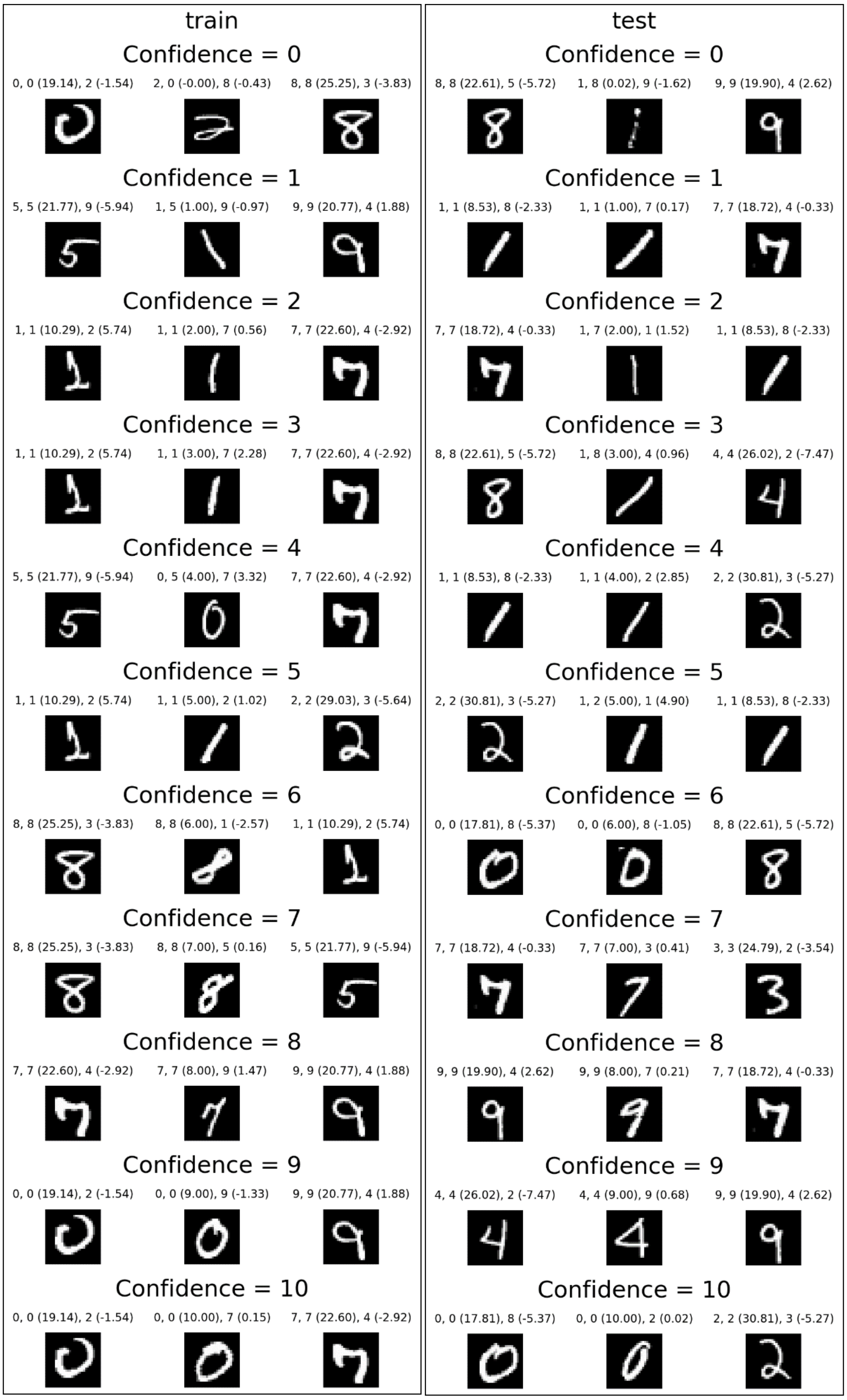


Figure 57. Boundary samples visualization on the train set (left) and test set (right).

The fourth and final metric in the method is pair-wise boundary conditioning, which refines the previous boundary conditioning metric, also relying on a range. To go beyond the painstaking process of manually analysing individual samples, a colormap visualization is proposed to provide a synthetic overview of the dynamics at play. An example is shown in Figure 58 with range [1-10] and in Figure 59 for range [4-6].

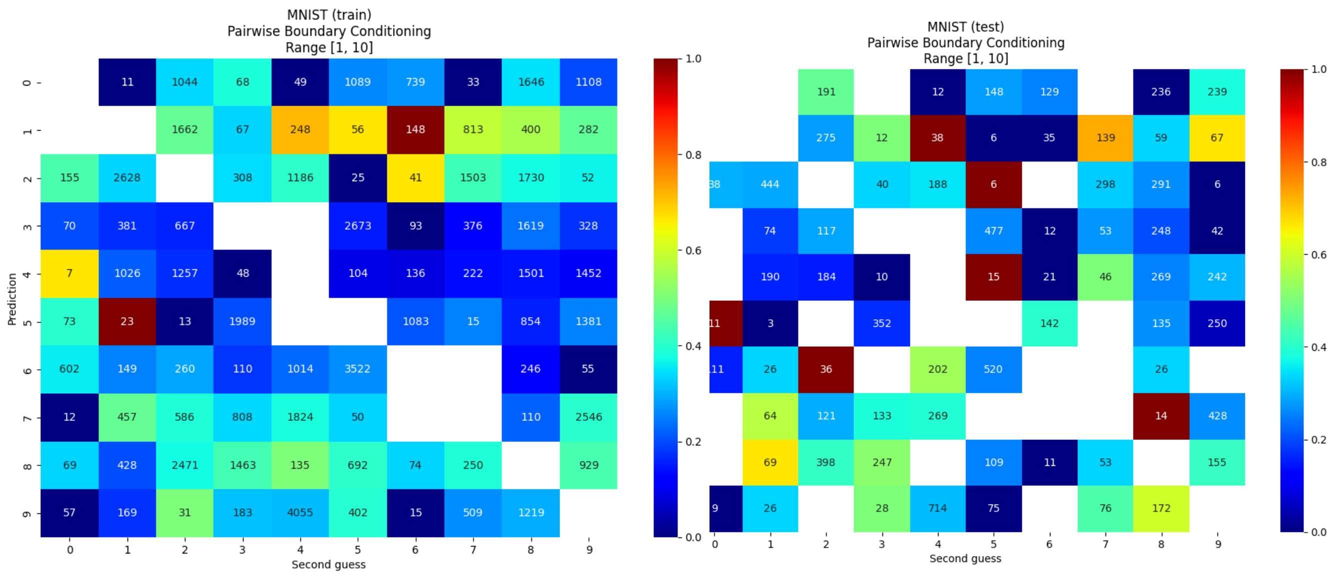


Figure 58. Colormap of pairwise boundary conditioning on the train and test sets for range [1;10]

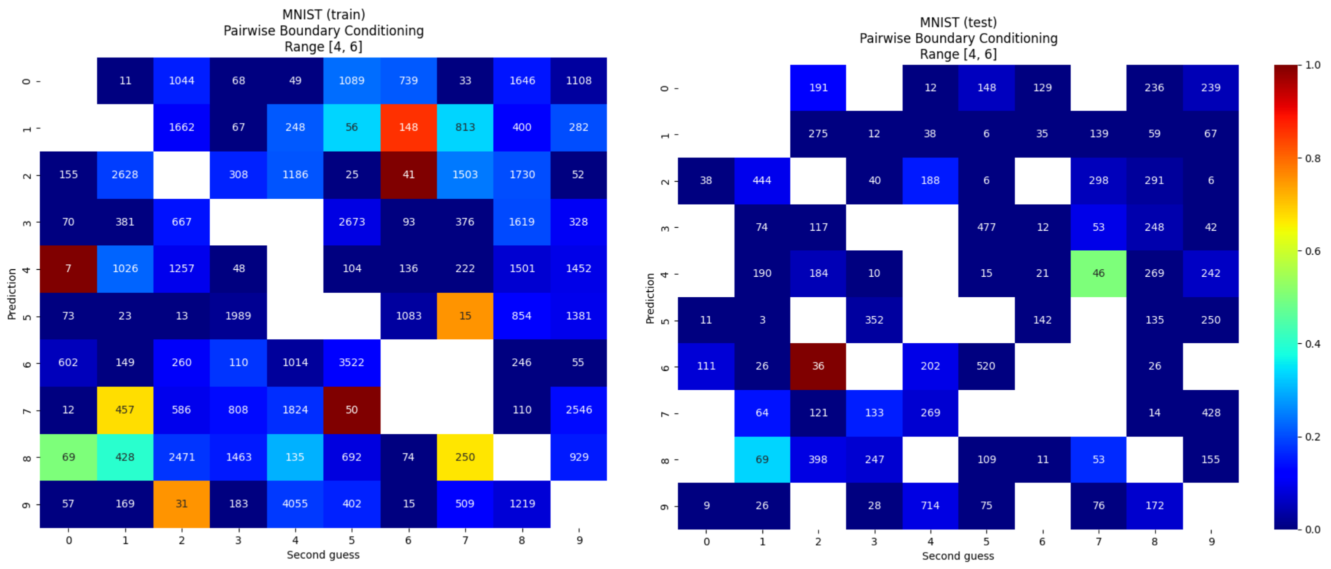


Figure 59. Colormap of pairwise boundary conditioning on the train and test sets for range [4;6]

These heatmaps allow to understand the topology of the decision boundaries. The colour indicates the degree of confusion, while the value in each cell indicates the total number of samples in the prediction class. This is an important information since the metric is a ratio based on this number, so only spots with similar values are really comparable. Hottest spots in rows or columns denote stronger boundaries between corresponding classes and therefore zones with most confusion. Note how the heatmap of the train set in the range [4, 6] has a lower maximum value (~0.35, compared to 1.0 in other heatmaps) Test sets have more zero scored pair, due to their lower sample count. This phenomenon highlights how completeness and representativeness are not only related to intrinsic properties of individual samples, but also coalesce into a combination of influence factors that is significantly harder to capture. Moreover, it can be observed that Prediction 6 and Prediction 4 exhibit more hotspots on the test set than on the train set. This could indicate that the test samples are different and could motivate further investigation to ensure the quality of the samples (for example using the visualization tool previously introduced on the specific pair).

4.7.3.7.2 Results from the MLEAP Perspective

Contrary to neuron coverage that requires some exploration to be used effectively, the four metrics used to observe the learned feature space offer some a priori structure to the analysis.
Equivalence partitioning is an interesting proposition to check the balance of the data set but does not provide ground breaking insight. Centroid positioning did not demonstrate convincing analysis properties, although it might be due to the data set used.

By contrast, boundary conditioning and pairwise boundary conditioning provided much information and appeared as useful tools for completeness and representativeness assessment. Indeed, both metrics provide complementary information on how the model represents the data. The main downside of these metrics is that they rely on threshold values of ranges, which requires some experimentation and expertise to set properly. Nonetheless, investigation into these values was more straightforward than other methods, e.g., neuron coverage.

Finally, this model-based method informs on both the model and the samples and thus should be interesting to combine with specific methods, i.e., reinforcing insight on specific samples identified by Cleanlab, or enlightening phenomena highlighted by neuron coverage. To this end, additional experimentations were conducted to explore if the thresholds for neuron coverage could be somehow transferred to this method. However, attempts were unsuccessful. This might be due to the fact that both model was explored through different spaces, and more effort should be put into ensuring that protocols are considering common points of interest. This could not be performed during the project.

4.7.3.8 **Completeness ratios**

4.7.3.8.1 Experimental protocols

As discussed earlier, the four axes of completeness described in (Weiskopf et al, 2013), namely Documentation, Breadth, Density and Predictive, served as the backbone of the proposed experiments, where:

- **Documentation** is defined as the ratio of complete samples, i.e., $\frac{\#complete\_samples}{\#total\_samples}$.
- **Breadth** is initially defined as the « availability of required multiple types of data » and implemented as the « proportion of the overall number of at least one visit with multiple types

of information to the total number of visits ». This metric is unexploitable as is and was thus modified. Instead, the distribution of feature completeness will be used.

- **Density** is defined as the « availability of sufficient numbers of data points over time » and implemented as the « proportion of the overall number of visits at least 1 day with multiple types of information to the total number of visits ». Although the metric is also unexploitable, the notion of sufficient number of data points echoed the work performed using the graph-based approach (see 4.7.2.2). For this reason, it was not tackled in this experiment.
- **Predictive** is defined as the « availability of sufficient information to predict an outcome » and the implementation relies on a logistic regression model. A similar approach will be performed using a Naive Bayes model.

Additionally, three metrics outside the work of the authors are proposed here:

- G1: the ratio of columns (i.e., features) with at least one missing value (one incomplete sample), i.e.

$$\frac{\#incomplete\_features}{\#total\_features}$$

- G2: ratio of rows (samples) with at least one missing value, i.e.

$$\frac{\#incomplete\_samples}{\#total\_samples}$$

Note that this metric is equivalent to 1-Documentation

- G3: ratio of missing values, i.e.

$$\frac{\#missing\_values}{\#total\_values}$$

As discussed earlier, experiments used the Titanic data set, as it is the only one with structural incompleteness. Results for Documentation, G1, G2 and G3 are visible in Table 16.

Table 16. Basic completeness metrics scores.

Documentation	0.21
G1	0.25
G2	0.79
G3	0.08

Documentation indicates that only 21% of the dataset is complete, which is very low (conversely, G2 shows 80% incompleteness in the data set, a somewhat alarming number). This should draw the attention of an analyst and trigger mitigation strategies such as data imputation. G1 shows that 25% of the features are incomplete. There are 12 features and the 3 incomplete ones are *Age* (of the passenger), *Cabin* (of the passenger) and *Embarked* (i.e., the port where the passenger embarked). Finally, G3 shows that 8% of all the values in the data set are missing. When compared to the Documentation metric, G3 unsurprisingly appears as the less informative score, although it puts the results in perspective, considering how removing a relatively small amount of information overall can actually impact a data set.

In addition, the graph of complete features distribution, i.e., Weiskopf breadth, is reported in Figure 60. We can see the minority of complete samples (12 features, slightly less than 200), confirming the information computed through the Documentation metric. In addition, a comparable number of samples are having 10 features only. It is interesting to observe that in this data set, there are few features missing for a lot of samples. The vast majority of samples are thus always missing up to two of the three features previously mentioned.

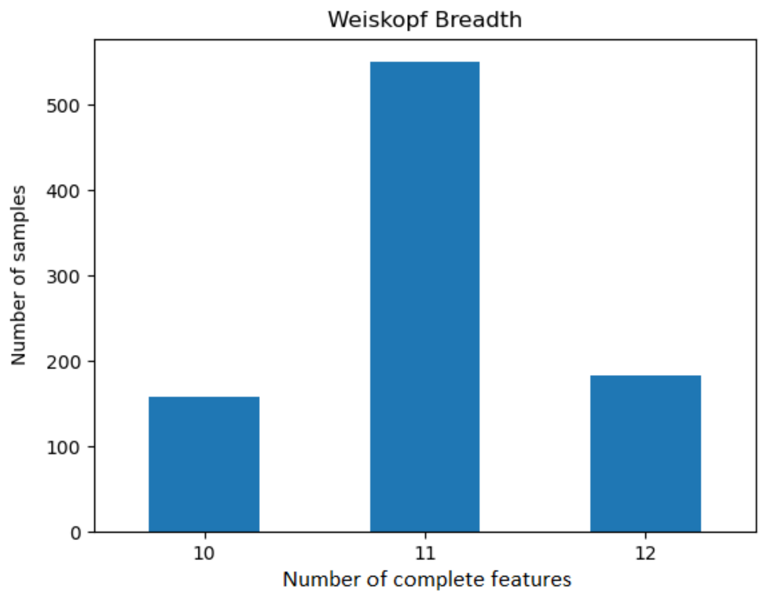


Figure 60. Graph of complete features distribution

Finally, the « predictive » completeness metric was explored using a Naive Bayes model. The data set was split into 50% train and 50% test, and classification accuracy reached 78%. The raw performance of the classifier is secondary and this level is in line with more fine-tuned models, so it was deemed sufficient.

To exploit this metric, several models were trained, each with an increasing number of features available. The results are in Table 17, with complete features in bold and the incompleteness ratio (i.e., the G2 metric for the feature considered) in parenthesis for incomplete features. The features used for the experiment were *Fare* (price paid by the passenger), *Sex* (of the passenger), *Pclass* (class of the passenger), *Age* and *Cabin*. The « Cabin » feature was not exploitable as is (because it is represented as strings), so it was split into 2 features, namely « Cabin_Deck » and « Cabin_Room »[72]. For instance, Cabin C22 would be split into Cabin_deck C and Cabin_Room 22. Other features were left out because they either were not exploitable (*Name*, *Embarked*) or to keep the feature space

[72] Following the approach described in https://www.kaggle.com/code/ccastleberry/titanic-cabin-features

simple and intuitive (*SibSp*: number of siblings of the passenger aboard the Titanic; *Parch*: number of parents of the passengers aboard the Titanic).

Table 17. Model performance vs features used. Features in bold are complete (no missing values).

Features used	Model performance (accuracy)
Fare	0.681
Sex	0.782
Pclass	0.670
Pclass + **Sex**	0.782
Pclass + **Sex** + **Fare**	0.766
Age (19% incompleteness)	0.634
Cabin_Deck (77% incompleteness)	0.692
Cabin_Room (77% incompleteness)	0.665
Age + Cabin_Deck + Cabin_Room	0.692
Pclass + **Sex** + **Fare** + Age	0.766
Pclass + **Sex** + **Fare** + Age + Cabin_Deck	0.748
Pclass + **Sex** + **Fare** + Age + Cabin_Deck + Cabin_Room	0.771

No obvious correlation between the performance of the complete and incomplete features can be observed. For instance, Pclass (complete) exhibits lower performance than Cabin_deck (incomplete). In addition, both cabin features have very high incompleteness (77% samples do not have a cabin id), yeti t exhibits more raw performance than Age (19% incompleteness) or even Pclass (complete). This also verifies when combining features: the best performance is obtained using only Pclass + Sex (both complete), with a close second using all features, including the incomplete ones. Moreover, adding Age to the set of complete features does not influence the performance, while adding Cabin_Deck over these will degrade performance. Finally, incorporating Cabin_Room boosts performance, bringing this configuration as the second-best configuration.

4.7.3.8.2 Results from the MLEAP Perspective

The Titanic data set did not allow for extensive exploration of the metrics. Incomplete data sets, in the sense considered here, are generally avoided by Machine Learning specialists as they may introduce biases. Data imputation strategies can also be implemented to mitigate such a phenomenon when incomplete data have to be used.
Despite the limitations of the experimental setup, the completeness metrics considered here can be contrasted to show how a seemingly small amount of missing data overall may actually translate to a lot of sample-wise incompleteness. It also shows how incomplete data can cripple learning.

4.7.3.9 **Risk-based approach**

4.7.3.9.1 Experimental protocol

In addition to the data set ROSE, the data set VoxCrim will be used for (BSA, 2021) because its content and operational context allows the exploration of the BSA method. The data set was created in 2020 and contains 8338 audio samples of 400 speakers extracted from French TV and radio broadcasts. It was used as a test data set for the VoxCrim research project[73], funded by the French national agency for research (ANR), aiming to assess the reliability of voice comparison systems used to identify criminals. For the experiment, the data set will be considered as a training data set.
(BSA, 2021) does not mention completeness explicitly and rather focuses on the notion of representativeness of the data. Even though it does not provide a definition of the latter, the way the term is used in the method seems aligned with EASA's definition. It considers training data representative when its demographic distribution corresponds to the population where the system will be used, which echoes the "actual input state space for the intended application" (EASA, 2021) when human subjects are involved.
In the method, representativeness is assessed at the data acquisition stage, following these two checkpoints:

- "Compare the demographic distribution of training data to the population where the system will be deployed"
- "Assess whether there is a sufficient representation of subpopulations that are likely to interact with the system"

As presented, these checkpoints do not genuinely facilitate a rigorous analysis. Indeed, the first checkpoint only indicates that the activity of comparison should be performed, with few indications about reaching an acceptable result. However, one can understand that the analysis's outcome should lead to an estimation of the degree of conformity. Furthermore, the presentation of the checkpoints is not uniform: the first one merely requires a comparative action, while the second one necessitates verifying that the representation has reached an acceptable threshold.
As the method appears somewhat akin to an audit framework, with checkpoints about the processes and the system, it is appropriate for the checkpoints to be presented in the form of criteria, the evaluation of which will determine compliance or non-compliance with specified requirements.
The checkpoints can thus be reformulated and presented as the following criteria:

- The demographic distribution of training data is reasonably similar to the distribution of the population where the system will be deployed.
- The subpopulations that are likely to interact with the system are sufficiently represented.

The definition of "subpopulations" is not specified in (BSA, 2021), therefore we will consider the term as referring to identifiable fractions of a population with distinguishing characteristics.
The conformity with the criteria will be tested on the ROSE and VoxCrim data sets. ROSE does not contain demographic data and is not intended for a human-centred application (vision detection of plants and weeds). This will allow testing the relevance of applying the method outside of its presumed context of use. VoxCrim contains human-sourced samples and is meant for applications with strong impacts on humans. This test will allow determining the actual applicability of the BSA method, in a use case presumably suitable for its application.

[73] https://voxcrim.univ-avignon.fr/#about

According to BSA's method, the assessment of the two criteria requires that several elements have been defined beforehand. Notably, the intent and purpose of the system must be identified and documented at the project conception stage, as well as the intended use cases, intended effects and operational context of the system. This is essential to determine what characterizes the population expected to interact with the system and make this information a reference to compare with the data chosen to train the model.
In the experiment, these elements will be described for each data set:

- Intent and purpose: specify the system's destination and the nature of the task to be accomplished.
- ODD: Since the intended use cases, intended effects, and operational context are included in the ODD (EASA, 2021), the latter concept will be used for clarity.
- Key features: features of the actual input state space that should drive the representativeness analysis. In the context of the BSA method, they can be considered as demographic groups.

In the context of the experiment, we will rely on the features already present in the data set and a description of the context as used in the research projects that led to the creation of these data sets. The experiment presented below is based on speculations to illustrate the reasoning; in the actual application context of the method, this information must naturally be reinforced by advanced domain expertise.
The experiment conducted on the data sets will aim to assess the representativeness of the features used to describe the data set, and the representativeness of the samples constituting the data set. This approach allows verifying 1) the type of information that needs to be represented/annotated in the data set and 2) the actual content of the samples. In both cases, the representativeness ratio prescribed by (ISO/DIS, n.d., pp. 5259–2) will be used, which consists in the ratio of features of the data sets to the relevant features in the population. A score of 1 indicates that the data set is entirely representative for a given feature, 0 means it is not representative at all. A score below 0.5 can generally be considered as low, and a score beyond 1 would indicate overrepresentation – which could be detrimental to the representativeness of another class within the data set, or lead to overrepresentation biases. The understanding of the score should however be tightly linked to the knowledge of the context of operation, which should set the thresholds of acceptability. One may note that the terms "reasonably" and "sufficiently" in the criteria are subjective and do not indicate specific thresholds. However, this approach appears entirely relevant, as the analysis relies on an expert approach, both from the perspective of the person responsible for conducting the analysis and a potential auditor tasked with conformity assessment. For reference, it is noteworthy that the European regulation for safety of machinery does indeed consider "reasonably foreseeable risks". The determination of what is reasonable or sufficient should, therefore, be delegated to sector-specific expertise and must be justifiable based on experiments or reference documents in the application domain.

- ROSE

  The intent and purpose are to improve the abilities of agricultural machines by providing a computer vision model for weed detection and to eliminate them more efficiently.

  ODD: identify weeds in European fields.

  Key features: families, species (subpopulation of "families").

Key features description in the data set: The ROSE dataset is composed of images representing 4 species of weeds:

- *Matricaria* from the family Asteraceae
- *Lolium* from the family Poaceae
- *Sinapis arvensis* from the family Brassicaceae
- *Chenopodium album* from the family Amaranthaceae

Target population: Very few studies exist on the specific subject of weeds in Europe. The most relevant one we could find is a survey on weeds increasingly spreading in Europe (Weber and Gut, 2005). According to this survey, 281 species are growing in this region, distributed among 48 families. The largest families reported are Asteraceae (61 species), Poaceae (55), Brassicaceae (15), Polygonaceae (14) and Apiaceae (11). Amaranthaceae comprises 7 species.
Criterion 1: "The demographic distribution of training data is reasonably similar to the distribution of the population where the system will be deployed."

- **Class analysis:**

$$\frac{Nb\ of\ classes\ (families)\ in\ the\ data\ set}{Nb\ of\ classes\ (families)\ in\ the\ target\ population} = \frac{4}{48} \approx 0{,}08$$

The score shows inadequate representation.

- **Sample analysis:**

For each family,

$$\frac{Nb\ of\ plants\ in\ the\ data\ set}{Nb\ of\ plants\ in\ the\ target\ population}$$

Desk research did not yield comprehensive numbering of weeds in the region of deployment. However, some assumptions can be drawn. For example, one can analyse the distribution of samples for each family of the data set, as pictured in Table 18. The survey used as reference indicates an average weediness of 2.3 for Amaranthaceae, and 1.9 for Brassicaceae, meaning that Amaranthaceae is a more invasive family than Brassicaceae. An approach could then consist in pairing weediness rate and distribution of the data set. Amaranthaceae makes up nearly half of the data set; being more invasive than the other listed families, it may confirm the importance of a high number of samples in the data set. However, this cannot justify the low percentages of Asteraceae or Poaceae, which seem almost as invasive as Amaranthaceae. This reasoning is highly speculative and does not take into account parameters such as soil nature or interplay between weeds and the surrounding plant species, and cannot replace a thorough expert analysis. The scoring should also take into account the absence of the 44 other families of the target population, which would negatively influence the ratio.
The score shows inadequate representation.

Table 18. ROSE data set: distribution of plants in each family, in comparison with weediness rates from (Weber and Gut, 2005).

Families	Number of plants	Percentage of the data set	Weediness rate
Amaranthaceae	34 218	49.1	2.3
Asteraceae	5 585	8.0	2.1
Brassicaceae	11 015	15.8	1.9
Poaceae	5 620	8.1	2.2

Unknown	10 792	15.5	-
Other	2 484	3.6	-
Total	*69 714*	*100*	

- **Conformity assessment:** the criterion is not fulfilled.

Criterion 2: The subpopulations that are likely to interact with the system are sufficiently represented.

- **Class analysis:**

For each family of the target population,

$$\frac{Nb\ of\ classes\ (species)\ in\ the\ data\ set}{Nb\ of\ classes\ (species)\ in\ the\ target\ population}$$

This results in:

- Amaranthaceae (7 species): 1/7 ≈ 0.142
- Asteraceae (61): 1/61 ≈ 0.016
- Brassicacea (15): 1/15 ≈ 0.067
- Poaceae (55): 1/55 ≈ 0.018
- Polygonaceae (14): 0/14 = 0
- Apiaceae (11): 0/11 = 0
- ... (42 additional families, none of which are represented in the data set)

Of course, the statistics used as a reference pertain to the geographical Europe in general, in accordance with the ODD established for this experiment, and therefore do not account for the specificities inherent to each geographic subzone (or any other relevant parameters), where the relevance of some plant families may vary.
The score shows inadequate representation.

- **Sample analysis:**

For each species of the target population,

$$\frac{Nb\ of\ plants\ in\ the\ data\ set}{Nb\ of\ plants\ in\ the\ target\ population}$$

The number of plants for each species in the data set is presented in Table 19. One may note that only 4 species are represented in the ROSE data set, for a target population of 281 species in total, which would lead the mean representativeness score to tend towards 0. Desk research did not yield, however, an estimation of the number of plants in the target population. The approach would perhaps need to be redefined in such a context, such as the analysis of the weediness of each species, in order to reach a trade-off between the relevance and feasibility of the verification.
The elements at our disposal do not allow a relevant computation.

Table 19. ROSE data set: distribution of plant species. " Others" are plants that have naturally grown in the experimental plots.

Plant species	Nb of plants
Matricaria	5 585
Lolium	5 632

Sinapis arvensis	11 020
Chenopodium album	34 221
Others	13 256
Total	69 714

- **Conformity assessment:** the criterion is not fulfilled.

- VoxCrim

The intent and purpose are to enable an entitled police officer or mandated expert in France to compare a suspect's voice with a recording and use the latter as proof of their guilt if the samples appear to be sufficiently similar.
ODD: Compare two different voice samples, both from a different recording source, to assess their level of similarity. The voice samples will be those of suspects arrested in France.
Key features: biological gender, age, profession.
Key features description in the data set: The VoxCrim data set is composed of audio samples and XML annotation (speaker ID, biological gender and timestamps of speech segments), from public interventions on television. For the needs of the experiment, the age and profession were collected from the Internet so as to illustrate the reasoning. The search for information yielded many missing values but provided reasonable tendencies for the analysis. The selection of these additional features is for illustrative purposes only – the relationship between relevant features and criminal context should be supported by socio-demographic expertise.
Target population: French population over 13 years old. Statistics from the French national institute of statistics and economic studies (INSEE, 2024, 2021) are used as a reference.
This selection of the target population is based on the reasoning that the demographics to use must be reflective of the population where the system will be used. Yet in the context of criminal identification, two interpretations co-exist:
- Knowing that the system will be deployed in France, we could consider any individual living in France a potential criminal. We should then refer to France's demographics.
- Knowing that the system aims to identify criminals, we could consider that the characteristics of the current prison population reflect the population where the system will be deployed. We should then refer to French prison statistics.

Both can be considered representative. However, a wise approach would be to consider fairness in the decision making. At the project conception stage, (BSA, 2021) recommends identifying fairness metrics that will be used as a baseline for assessing bias in the AI system. The method does not suggest any fairness metric and rather highlights the subjective and contextual aspects of the matter. It also mentions the need for consistency with any applicable legal requirements in the choice of these metrics. However, fairness is neither addressed nor mentioned in the process of assessing representativeness.
Fairness substantially refers to ensuring non-discriminatory practices (EASA, 2024). Considering the intent and purpose of the project relying VoxCrim data set, discriminatory outcomes would lead to unacceptable consequences for individual rights. Therefore, fairness should prevail over an interpretation of the input state space which could be characterized by discriminatory distinctions. To that end, the first interpretation implying demographic balance

seems to be the most appropriate. It will avoid from this early stage later disparities in the model's performance that could lead to the targeting of protected groups, i.e., sensitive meta-features (e.g., gender or ethnicity). More precisely, this would reduce the chances of selection or historical bias, and prevent the model from learning discriminatory patterns.

Criterion 1: "The demographic distribution of training data is reasonably similar to the distribution of the population where the system will be deployed."

- **Class analysis:**

For each key feature of the data set,

$$\frac{Nb\ of\ classes\ in\ the\ data\ set}{Nb\ of\ classes\ in\ the\ target\ population}$$

This results in:

- Biological gender: 2/2 = 1

The score shows a good representation.

- Age: (83-25)/(118-13) ≈ 0.55

Since age is represented as continuous values, its analysis can be based either on a comparison of the distributions, or a classification into categories of age. The comparison of distributions relies on statistical tests on the similarities of the distributions. However, the reference document only provides minimum, maximum, and mean values and percentages, which does not allow in-depth statistical tests. Since categories of age will be considered as subpopulations in the remaining experiment, we will rely here on the other information available in the reference, namely the age span. One can observe that the minimum age in the data set is 25, and the maximum is 83. In 2023, the eldest person alive in France was 118 years old (the youngest being naturally 0). The approach should, however, take into account the final destination of the system, which is meant to operate on persons aged 13 and over (legal age for formal investigation in France).
The score shows inadequate representation.

- Profession: 1/6 ≈ 0.17

Ten categories of occupations have been identified in the data set (journalist, politician, deputy, decision-maker, expert, healthcare professional, writer, minister, mayor and presenter) that could, according to (INSEE, 2021) categorization of professions, fall into the category of "Executives and higher intellectual professions". (INSEE, 2021) identifies 6 meta-categories in total.
The score shows inadequate representation.

- **Sample analysis:**

For each possible value of each key feature of the data set,

$$\frac{Nb\ of\ samples\ in\ the\ data\ set}{Nb\ of\ samples\ in\ the\ target\ population}$$

- Biological gender:
  - Male: 72.75/48.3 ≈ 1.51
  - Female: 27.25/51.6 ≈ 0.53

Since INSEE only provides percentages, the distribution is expressed here in terms of percentages (percentage of the data set/percentage of the population). These figures do not take into account the separation in terms of age – INSEE provides overall figures including the population under 13.

The scores show overrepresentation of males, and underrepresentation of females.

- Age: *missing information*

INSEE only states that the average population living in France is 42.4 (INSEE, 2023). The average age of the subjects in the data set is 52. The analysis of the distribution of age, or at least of age categories (young adult, etc.) would have proved more informative. In the absence of reference for establishing a relevant significance threshold, one may consider that the averages are reasonably similar, but this does not constitute relevant information for the representativeness analysis.

- Profession:
    - Executives and higher intellectual professions: 20.4/56.25 ≈ 0.36
    - Farmers and operators: 0/1.4 = 0
    - Craftsmen, shopkeepers and company directors: 0/6.8 = 0
    - Intermediate occupations: 0/26 = 0
    - Employees: 0/25.8 = 0
    - Workers: 0/19.2 = 0

All speakers belong to the socioeconomic category "Executives and higher intellectual professions", while this category only represent 20.4% of the French population (INSEE, 2021). For example, journalists alone represent 56.25% of the data set, and the other most represented jobs are politician and elected representative.

The scores show inadequate representation.

- **Conformity assessment:** the criterion is not fulfilled.

Criterion 2: The subpopulations that are likely to interact with the system are sufficiently represented.

- **Class analysis:**

For each subpopulation class in each key feature of the target population,

$$\frac{Nb\ of\ classes\ in\ the\ data\ set}{Nb\ of\ classes\ in\ the\ target\ population}$$

This results in:

- Biological gender: *missing information*

No potential subpopulations have been identified. The calculation is not relevant.

- Age: 3/4 = 0.75

The target population (over 13 years old) may be divided into teenagers (13-19), young adults (20-39), middle-aged adults (40-59), and seniors (60+), totalling 4 categories. The VoxCrim data set does not contain teenagers.

The score shows inadequate representation.

- Profession: *missing information*

Although the data set is labelled with 10 different professions (all belonging to the category of "Executives and higher intellectual professions"), the INSEE source does not provide matching details, so the score cannot be computed.

- **Sample analysis:**

For each subpopulation class in each key feature of the target population,

$$\frac{Nb\ of\ samples\ in\ the\ data\ set}{Nb\ of\ sample\ in\ the\ target\ population}$$

This results in:

- Biological gender: *missing information*

No potential subpopulations have been identified. The calculation is not relevant.

- Age:
    - Teenager: 0/6.3 = 0
    - Young adult: 16.4/23.4 ≈ 0.7
    - Middle-aged adult: 62.9/22.5 ≈ 2.8
    - Senior: 20.7/27.7 ≈ 0.7

The figures are expressed in percentages to allow comparison with (INSEE, 2024). The scores show a strong overrepresentation of middle-aged adults and an underrepresentation of seniors, young adults and teenagers.
The scores show inadequate representation.

- Profession: *missing information*

No adequate information in the reference. The score cannot be computed.

- **Conformity assessment:** the criterion is not fulfilled.

4.7.3.9.2 Results from the MLEAP Perspective

(BSA, 2021) is presented as a convenient procedural tool for assessing bias risk in AI. It offers a step-by-step approach presented in table format which makes it easy to follow.
Its approach to addressing representativeness is designed for projects involving human subject data and provides straightforward instructions on how to conduct the evaluation. This approach initiates thoughts on the data set's adequacy in light of the actual context of use from the beginning of an AI project, which can indeed prevent the appearance of biases.
However, the way the checkpoints are formulated does not clearly indicate what should be done or what results should be reached. Reformulating them was needed.
Table 20 presents an overview of the evaluation results. The most stringent requirements were adopted to support the demonstration and explore potential usability limits of the method; for example, the ODD as defined here takes into account a final ideal system. The data sets are entirely legitimate within the research projects for which they were designed, but the exercise illustrates the gap to be bridged if these corpora were to be used as they are for the development of a system.
First, the experiment demonstrates that the BSA method can reasonably be applied even in contexts where human populations ("demographics") are not considered. The experiment also highlights the importance of the availability of reference data, allowing a comparison between the parameters of the target population and the features of the data set.

In addition, the findings insist on the importance of domain expertise, which allows logical reasoning consistent with the constraints of the target application domain. In the example of ROSE, advanced knowledge in botany could drive the analysis towards patterns based on the interlink between species or with parameters in the environment. VoxCrim analysis may also be based on acoustic analysis of the samples. In-depth expert studies can establish a tight link between such factors and relevant elements for representativeness.

Table 20. Synthesis of the evaluation of conformity with the representativeness criteria, on the ROSE and VoxCrim data sets.

	ROSE	VoxCrim
Criteria 1 "The demographic distribution of training data is reasonably similar to the distribution of the population where the system will be deployed."		
Class analysis	Inadequate representation	Adapted representation for biological gender Inadequate representation for age and profession
Sample analysis	Inadequate representation	Inadequate representation for biological gender and profession *Information not available for age*
Criteria 2 "The subpopulations that are likely to interact with the system are sufficiently represented."		
Class analysis	Underrepresentation	*Not relevant for biological gender* Inadequate representation for age *Information not available for profession*
Sample analysis	*Information not available*	*Not relevant for biological gender* Inadequate representation for age *Information not available for profession*

BSA method has difficulty capturing the predominance of fairness at any stage when the AI system needs a particular awareness of human rights regarding its purpose. It does not provide details on what is understood as "fairness metrics" while (Tabassi, 2023) does provide examples of general fairness metrics (namely, demographic parity; equalized odds; equal opportunity; and statistical hypothesis tests).

Thus, the timeline for the analysis of representativeness suggested in (BSA, 2021) can come as inefficient for such AI systems (e.g., VoxCrim). The method recommends identifying relevant fairness metrics from the project conception stage, yet does not indicate when exactly they may be needed afterwards. During the data acquisition stage, the issue of having to choose between realistic and fair demographics when evaluating representativeness is not addressed. Following this evaluation, the mitigation practice suggested is to re-balance data to enhance representativeness, with additional or synthetic data.

However, a data set can be representative of the "actual input state space" (EASA, 2024) and at the same time reveal unfair biases. In this case, focusing on obtaining a representative demographic distribution rather than making fairness a priority may lead to counterproductive actions. Indeed, the method suggests revisiting earlier stages when testing reveals unacceptable levels of bias, and in particular seeking out additional data after, which impacts representativeness in the end. BSA method thus does not seem fully consistent in the way the topics of representativeness and fairness should be handled throughout the stages of development.

Finally, the only solution suggested to mitigate the risk of bias caused by an unrepresentative data set is to re-balance the data. Another way of enhancing representativeness would be to rethink the ODD of the system (e.g., by reducing its scope). With this in mind, the ROSE corpus could be used as it is

with a more limited purpose, only including the detection of species or families sufficiently represented.
In conclusion, relying on (BSA, 2021) alone would be insufficient. The ratio formula provided by (ISO/DIS, n.d.) were needed to proceed to the evaluation, as well as additional research and reflection to assess representativeness in context-relevant approach. This is especially important when fairness plays a major role for the validation of a project. In that case, (Tabassi, 2023) is a reliable additional source of information.
More generally, the evaluation should be conducted jointly with experts who are perfectly aware of the context of use and especially of the target population. This will help decide which features need to be considered and address the lack of relevant information available, such as representative statistics.

4.7.4 Discussions on other methods

During this phase, the work was divided into three rolling tasks:

- Internal discussions to confirm the selection of a method,
- Implementation of the method (or first experimentations if an off-the-shelf implementation was available), usually using a toy data set,
- Testing of the method on the reference data set (corresponding to the use case).

Implementation and testing have been extensively discussed in the previous section. However, after the first experimentations and feedback on using the data sets, the priority of certain methods was reassessed, and in some cases, reservations on their added-value w.r.t the MLEAP project were expressed.

For example, the method by (Almeida and Vieira, 2011) aims at characterizing resilience, identified in this document as an influence factor of completeness and representativeness.
However, the approach relies on incremental degradation (either of the system or of the input), which was found to be overlapping the works corner cases.

The method by (Balaraman et al., 2018) is entirely directed toward the analysis of structured data such as knowledge bases. This kind of data is outside the scope of the MLEAP project, but it was envisaged to devise adaptations to the speech or vision use cases to enable its testing nonetheless. A preliminary work was performed in parallel with the research of speech embeddings. It appeared no work in the research community was done to build a structured embedding space that would exhibit enough explicit structure to allow for the testing of this method. Such endeavour would be an entire research direction and is thus outside the scope and feasibility of the MLEAP project. However, it remains an interesting tool for structured data.

Also, the method described in (Sánchez et al., 2019) was based on a tool called TAQIH and focused on tabular data. The tool proposes an interface synthesising different information on the data set being processed. Information is dispatched into different types: on one side, a summary of general statistics on the data set e.g., the number of features, their individual type and range. On another side, correlation analysis of the features, identified outliers and missing values. In essence, the tool integrates much information from basic pre-processing of the data set. The testing of this tool was discussed after the case study of PCA and graph-based methods (PCA may even be seen has a direct extension of these basic statistics), so the tool itself could probably have been used for the ACAS-Xu data set and would have provided similar insight. Nonetheless, the tool is not generic enough to be recommended as a solution option. The study of the methods demonstrated the importance of applying basic analysis tools on the data set and studying their interaction to get insight on the next steps, but it should remain a generic methodology, adaptable to any type of data.

SliceTuner, the tool introduced by (Tae and Whang, 2021) was also put aside despite its interest because it required a learning system to be put to use, as its objective is to monitor the learning process to identify latent factors of influence and adapt the distribution of learning samples accordingly. Moreover, this objective is not directly aligned with MLEAP's scope and required adaptations, which also added to the time necessary to deploy it. However, it remains one of the priority tools to be tested in the next phases.

Finally, (BSA, 2021) being based on risk management, its testing was delayed and has been performed directly on one or more MLEAP use cases since it does not require exploration, as opposed to software tools.

4.7.5 Experimentation conclusions

The overall outcome of these preliminary experiments is positive. Many methods have been implemented and validated on small data sets. In addition, some methods have been tested on high-scale data sets, allowing for the identification of technical challenges. This important groundwork offers a solid testbed for the continuation of experimentations by enabling smoother testing of the remaining methods, along with the deepening of the analysis already implemented, either through complementary experiments or the application of the methods on new data sets.

Moreover, the results obtained, though at different levels of maturity, confirm the value of well-known but sometimes overlooked standard good practice of ML, e.g., systematic general characterisation of the data set (beyond its description) through simple metrics and tools prior to investigating deeper into problematics such as completeness and representativeness.

The work also confirms the absence of a *one-size-fits-all* methodology. Instead, a tailored, incremental examination process is preferable. It also requires combining tools and metrics to get insight from several points of view.

Also, the experiments confirm that completeness and representativeness properties are hard to assess. The expressivity of the tools tested is limited, usually allowing a fragment of information on intrinsic qualities of a data set, which may be used to anticipate trends in the learning process that may then be confirmed efficiently at evaluation time and orient modifications on the data set. In addition, a comparison of two data sets is easy and maybe more insightful, but it also has limitations and cannot be used to accurately decide a course of action regarding the use of the data set as it is. A methodology enabling a consistent and absolute assessment of the completeness or representativeness of a data set, i.e., a set of tools that could be combined to appreciate the adequacy between any data set and real-life phenomena they are supposed to capture, remains an open challenge.

4.8 Experiments on the MLEAP Use Cases

4.8.1 Experimental setup

Following the preliminary experiments, the last step of the experimental phase was to apply the relevant methods to the aviation use cases (UC). Not all methods tested in the previous step were appropriate for these. Among the relevant methods, not all are relevant to every use case.

We mostly concentrated on the AVI data set since, being a classification task on natural data, it allowed for the widest set of methods to be applied. Speech-to-text has fewer applicable methods, so we used the risk-based approach and MUP (which are complementary).

As discussed previously, similarity-based methods were not ported to the aviation UC due to their inconclusive results. The presented graph-based tool and completeness ratios were not applicable either. In some cases, methods had to be adapted or provided different types of information, which is discussed in the dedicated sections. When it comes to speech handling, embeddings have experimentally shown to not work out with the kind of statistics expected of the PCA and entropy methods, making them unusable for ATC-STT. This shows how real-life UC has its own set of constraints to build around, illustrating the constant need for adaptability in the assessment of data completeness and representativeness.

Finally, all model-dependent methods (i.e., Cleanlab, neuron coverage, feature space characterisation) were performed on the same models used in the Task 2 experimentations, with the aim of enabling a common analysis. However, contrary to the previous set of experiments, the goal is not to contrast models against each other, which is why a single model will be selected for each use case.

The AVI data set was augmented for further experimentation. In this section, the results of the non-augmented and augmented data sets will be contrasted. More details on the augmentation protocol are available in Chapter 5.

The AVI data and models used are essentially the Lightning data and sometimes also the Dents data. The model used is the YOLOV5 model provided. The ATC-STT data used is the train data of the BoostHLT project (3.3.2). We used the full training data set with more accents than just French and Chinese to make the experiments more interesting

4.8.2 Off-the-shelf tools

4.8.2.1 AVI

The AVI UC is an object detection task. The input sample will be associated with a predicted bounding box along with the expected object this bounding box contains. This is a slightly different use case than the MNIST dataset in the preliminary experiments. Cleanlab has a specific module for object detection that works differently from the module for object classification to better fit the task's particularities.

The provided model was trained on a fixed train/validation/ test set instead of a cross-validation protocol. Thus, only the test set is relevant for analysis. However, the volume of the AVI test set is minimal and due to project constraints, it was not possible to retrain a model following a k-fold cross-validation protocol (which would have allowed a complete coverage of the data set that might have let more issues emerge while avoiding data leakage). In addition, running on the test set yielded no reported problems. As a result, a re-run was performed on the training and validation sets. Consequently, the results discussed hereafter cannot be used to conclude the AVI data set and are reported only to show what features Cleanlab offers for object detection. For the same reasons, the contingency heatmap was abandoned altogether in this experiment.

In a way similar to the mislabelling's detection for object classification, Cleanlab identifies issues in images that can be seen as mislabelling, i.e., the bounding box predicted by the model is different than the ground truth yet has a high computed confidence score. Three such detections occurred on the AVI data set, once in the dent case and twice in the lightning strike case. Unfortunately, upon manual inspection, all three were false positives, i.e., the ground truth was better (the model predictions were missing objects). All detections were done on train samples, so following the previous disclaimer, they probably should be considered artefacts due to data leakage rather than factual errors attributable to the Cleanlab tools.

Finally, Cleanlab offers a generic detection of general image properties, i.e., blurriness, low information (based on entropy), dark and light images, etc. In the preliminary experiments, the MNIST dataset displayed no such discrepancies. On the other end, AVI being a real-life data set, this kind of diagnostic turned out to be more informative:

	Dents			Lightning strike		
	Train	Val	Test	Train	Val	Test
Total images	3659	1044	522	28	6	3
Blurry	284 (7.7%)	68 (6.5%)	35 (6.7%)	0	0	0
Low information	0	0	0	0	0	0
Dark	0	0	0	0	0	0
Light	0	0	0	0	0	0
Odd size	231 (6.3%)	73 (6.9%)	22 (4.2%)	0	1 (16.6%)	0
Odd aspect ratio	0	0	0	0	0	0
Grayscale	0	0	0	0	0	0
Near duplicate	143 (3.9%)	15 (1.4%)	5 (0.9%)	2 (7.1%)	0	0
Exact duplicate	0	0	0	0	0	0

Table 21 Result of the Cleanlab analysis on the AVI use case.

The dents case exhibits the same issues across all splits, which does not compromise their mutual representativeness. Moreover, the volume of such problems is not problematic. On the contrary, the lightning strike case has heterogeneous issues. Odd-sized images have a generally negligible impact on representativeness, especially considering the size of the data set, which is far more problematic. However, near-duplicates are more concerning since they significantly impact the variety and, thus, the representativeness of the data set, particularly considering the small volume of images available.

The data augmentation protocol relies on duplicating existing samples, and considering the kind of information Cleanlab makes available, running it on the augmented dataset is irrelevant.

4.8.2.2 **ATC-STT & ACAS-Xu**

Due to its inherent properties, ACAS Xu was not relevant to Cleanlab's processing. In addition, Cleanlab does not handle the speech transcription task, making it unable to handle ATC-STT.

4.8.2.3 **Conclusion from the MLEAP perspective**

Due to training constraints, Cleanlab could not provide meaningful insights that could be related to the completeness or representativeness of the AVI data set. This highlights how model-dependent assessment methods must be integrated into (or even before) the training process from the beginning, which constitutes a significant constraint. This would place the use of such solutions early in the AI component development process, i.e., at step 2 i.e., “a priori evaluation” of the generic pipeline described in 7.4, as a link between DA-03 and DA-04 (as per the discussion in section 1.5.1.1). Nonetheless, the experiments that could be performed showed the difference in features offered by Cleanlab between image classification and object detection (although they both are Computer Vision tasks). This illustrates how the richness of the solution may depend on the task at hand and makes Cleanlab a typical example of how off-the-shelf solutions can be interesting. However, they will never be a one-size-fits-all solution.

4.8.3 PCA-based analysis

4.8.3.1 **AVI**

Applying PCA on images requires converting them to grayscale, a common pre-processing in computer vision. Contrary to the image classification task, the multi-object detection task of the AVI data set cannot be tackled directly. There might be several labels within a given image, so processing images directly would combine information from different labels. Instead, the images formed by reference bounding boxes were extracted and used in the PCA, as each bounding box is unambiguously attributed to a single class. This method has a downside: bounding boxes have different sizes, which requires padding. This is not expected to be a problem for PCA as the padding will be done with zeroes and should not increase the variance. Nonetheless, considering the difference in the sizes of the bounding boxes in the dents and the lightning strike cases, Figure 61 reports the ratio of padding over valid pixels over the entire data set, for exhaustiveness. The global distribution is high with a median at ~75% but narrow, not exceeding 70% to 80% (except for extrema and outliers).

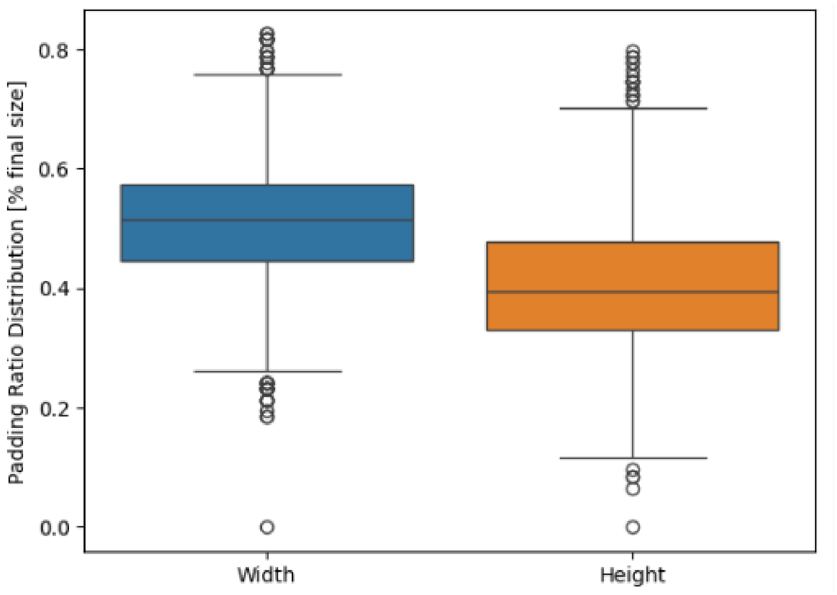


Figure 61: Padding ratio for the (non-augmented) AVI bounding boxes.

Figure 62 presents the 2-components PCA on the non-augmented data set. Although the dent_al samples tend to spread out across the first-dimension axis it generally forms a homogeneous space. The dent_lb class follows a similar space, although with more sparsity. This is due to the difference in the number of samples (1485 for dent_al vs 249 dent_lb). Nonetheless, the repartition of the samples is homogeneous, which would tend to indicate a correct coverage of the sample space. By contrast, the lightning strike samples (i.e., Smoky class) exhibit a tight cluster at the base of the dent space, and a more diffuse globular cluster that does not follow the general trend. Moreover, this globular cluster is very sparse. It extends across a large portion of the plot, indicating either a narrow sample space for which the samples from the data set are not representative, or a vast and poorly covered sample space. In this case, the poor coverage could be due to the extremely low sample count, but not only, as the extent of the space delimited by the samples would require tens of thousands of samples to be homogeneously covered. The PCA alone does not allow further analysis that could enable one to decide between both options.

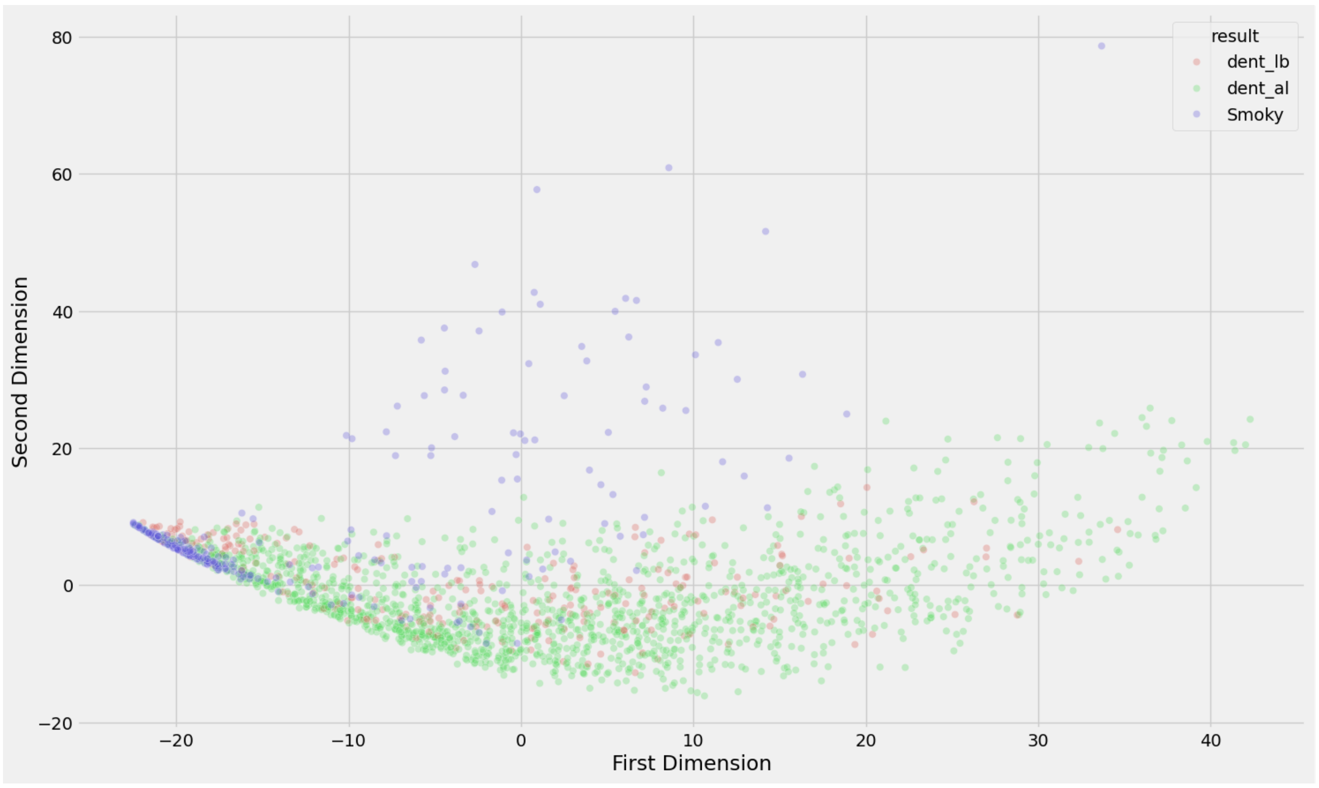


Figure 62: 2-component PCA for the non-augmented AVI dataset.

Figure 63 shows the results of the PCA on the augmented data set. First, the trend of the dent_al is drastically altered compared to the original data set, being notably shorter along the first dimension, with a stronger slope. Second, while some samples might be hidden by the abundance of dent_al and Smoky samples, dent_lb samples seem to not follow the dent_al trend as was the case on the non-augmented dataset. Moreover, the samples visible on the plot seem more scattered without any trend of their own. Finally, the aggressive augmentation protocol for the lightning data set seems to have been fruitful, as a clear trend is now visible on the plot, formed by a homogeneous sample space (although the farthest stretch of samples on the upper-right corner still remains modestly covered).

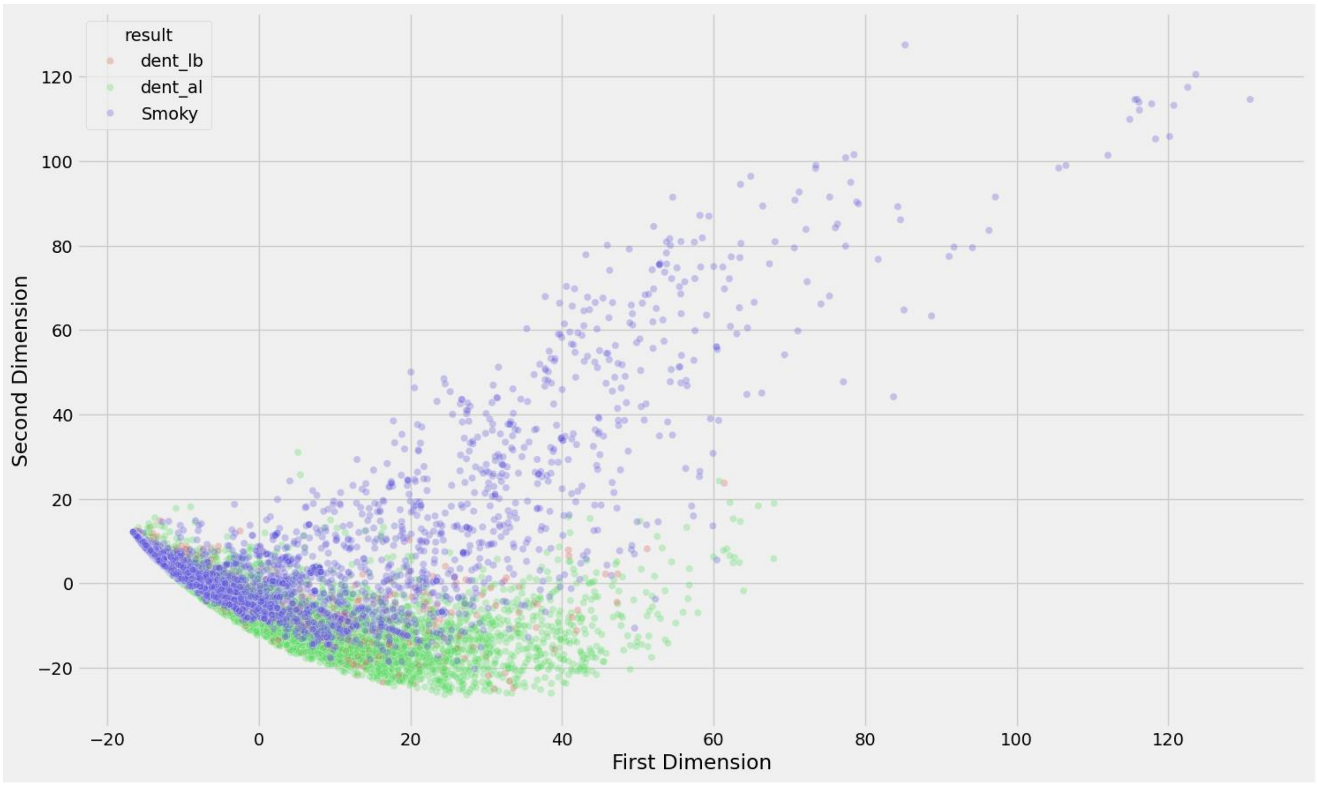


Figure 63: 2-component PCA for the augmented AVI data set.

4.8.3.2 **ACAS Xu & ATC-STT**

The reader is reminded that PCA for ACAS-Xu was tested during the preliminary experiments set. As such, it is discussed in section 4.7.3.1. Due to the lacklustre results of the preliminary experiments on audio embeddings, ATC-STT was not experimented on using PCA. This is probably due to the peculiar statistics of embeddings in general which tend to be laid out on the surface of a sphere, which does not work well with variance-based decomposition.

4.8.3.3 **Conclusion from the MLEAP perspective**

Running PCA on the non-augmented data set and then on the augmented clearly highlights the effect of the data augmentation protocol. In the lightning strike case, the coverage of the sample space was remarkably increases, suggesting a comparable improvement of the dataset completeness. On the contrary, the augmentation protocol significantly altered the trends observed in the original data set for dents classes, which could suggest a degradation of the representativeness of the samples.

PCA is a proven data analysis method that demonstrated its usefulness and limitations on the AVI data set. It should be noted that other dimension-reduction methods exist that would be best suited for images, such as HOSVD or T-SVD, which would allow images to be processed without converting to grayscale. PCA was used here for coherence with the methods discussed and for its reference status among data analysis methods.

4.8.4 Graph-based analysis

4.8.4.1 **ATC-STT**

That use case provides a number of metadata attributes which are the basis of the MUP analysis:

- Gender
- Role (P, C, M, G, A and unknown), which are undocumented (we guess that P = Pilot and C = Controller)
- Accent

Running the analysis on the data then points to the combination of attributes that are insufficiently represented. For single attributes, the roles "M," "P," and "G" are designated, with a particular emphasis on "G." In practice, the unknown labels are numerous, in fact more present than the others, and a better characterization of the speaker roles could allow for a better estimation of the completeness in that area.
As for combinations of features, the more underrepresented pairs are (female, it), (female, pt) and (female, zh). As a result, there is a risk of suboptimal performance of the model in those cases, which should be determined with a specific evaluation. A risk-based analysis taking into account the population where the system is expected to be used can help ascertain the actual level of risk associated.

4.8.4.2 **AVI & ACAS-Xu**

This analysis is dependent on the metadata associated with the input data, the AVI use-case is somewhat poor in this regard. As for ACAS-Xu, as mentioned previously, its full coverage of the possible input values makes the approach not really relevant in practice.

4.8.4.3 **Conclusion from the MLEAP perspective**

When useful metadata is available, we can see that the graph-based Maximum Uncovered Patterns analysis can give some interesting insights in the completeness of the data available. In particular, the method may uncover potential segments of the ODD where the probability of weaknesses of the model is higher, and this should be checked closely at performance evaluation time. The risk-based approach (Section 4.8.8.3) can complement that nicely by trying to estimate the impact of such weaknesses if they end up being realised.

4.8.5 Entropy-based analysis

4.8.5.1 **AVI**

Applying entropy analysis on the AVI dataset requires some adjustments compared to more straightforward datasets like CIFAR-100. Being a multiple objects detection dataset, each image may contain different objects (i.e., different classes). In such case, the image is associated with each class

it contains (i.e., if one image contains object class A and object class B, then this image will appear in the entropy computation for object class A images and object class B images.).
Moreover, due to these multiple object settings, two computation methods have been tested:

- The *unweighted* method computes each image associated with each label once, similar to the method used in the preliminary experiments.
- The *weighted* method computes the image's occurrences as many times as the number of objects of the given class it contains. For example, if image 1 contains 3 occurrences of object label 1, then it will be computed 3 times.

The first method is useful for getting an objective idea of the entropy of the dataset, allowing comparison with the theoretical bound. The second method is comparable to a weighted average version of the entropy. It was studied to see if additional insight could be gained from considering samples with more classes as somehow more critical (intuitively, because labelling them thoroughly and correctly is more challenging). This kind of hypothesis might not work for every use case and is reported here mostly for illustrative purposes. To enable meaningful reasoning about this notion, the average number of bounding boxes per image was computed for the dent case (not for the lightning strike case considering there is only one class, no empty example and dents are detected using a different model than the one detecting lightning strikes). In addition, as dent_al is significantly more represented than dent_lb, a computation method was devised to avoid the obvious bias of dividing the raw number of bounding boxes of a given class by the total number of annotated images (macro-average). Instead, the average is computed over the number of samples within the class at hand (micro-average). Results are shown in Table 22.

Class	Micro-average
dent_al	1.65
dent_lb	1.09

Table 22: Micro-average of the number of per-sample bounding boxes for the AVI/Dent data set.

Figure 64 reports unweighted and weighted image entropies for all AVI data set classes on the non-augmented data set. Figure 65 shows unweighted and weighted image entropies on the augmented data set.

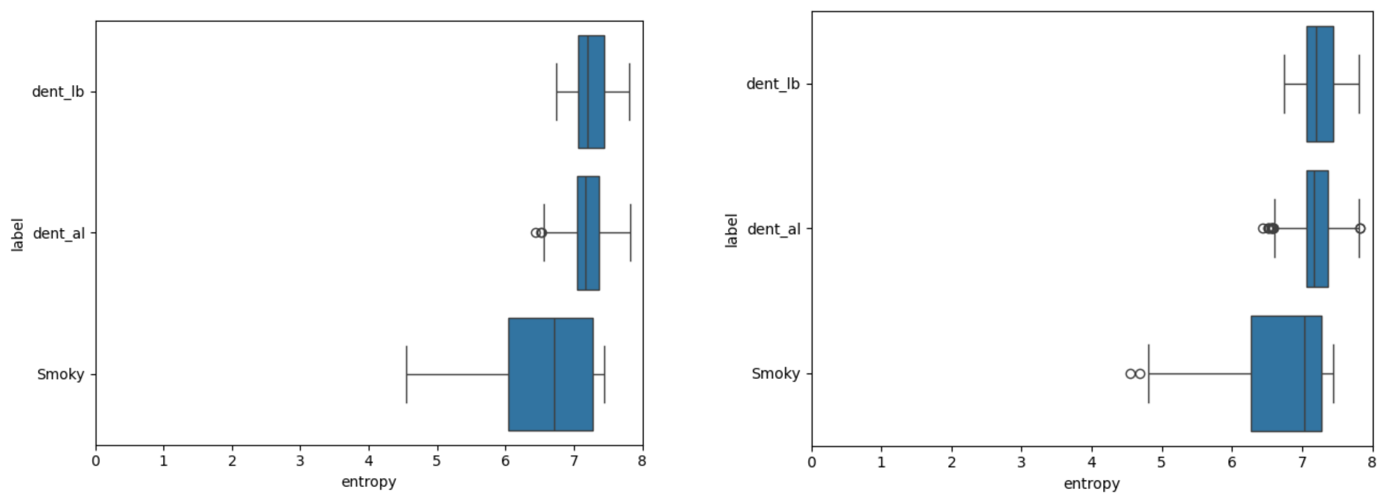


Figure 64: On the left, unweighted image-wise entropy for the AVI data set. On the right is, weighted image-wise entropy for the AVI data set. Both results were obtained on the non-augmented data set.

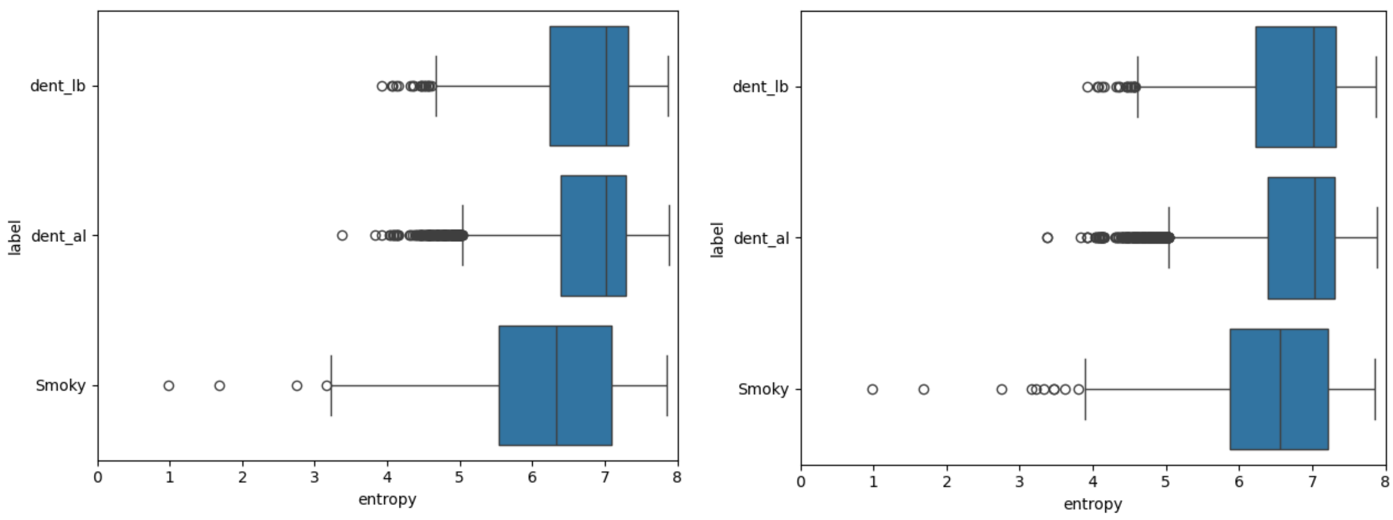


Figure 65: On the left, unweighted image-wise entropy for the AVI data set. On the right, the weighted image-wise entropy for the AVI data set. Both results were obtained on the augmented data set.

As discussed in the preliminary experiment section, the image-wise theoretical entropy bound is computed from the size of the images. The Cleanlab analysis highlighted that some images in the AVI data set have different sizes. However, this phenomenon is considered negligible for the estimation of the entropy bound. Indeed, the bound should be considered as a general horizon and not a precise limit. Therefore, the bound is estimated from the majority size, which is 1024x768 for the dents case and 4608x3456 for the lightning strike case, hence:

$$\log_2(1024 * 768) = \log_2(786432) = 19.585$$
$$\log_2(4608 * 3456) = \log_2(15925248) = 23.92.$$

First, it can be observed that despite the notable difference in theoretical bound due to the large difference in image size, both cases have similarly observed entropies, with maxima close to 8 and means close to 7. These values are slightly inferior to half the maximal entropy, denoting a consistently low complexity of the images, which is in line with the use case context, i.e., pictures of fuselages taken in the interior, with defects being rather small compared to the size of the whole image. This idea is reinforced by the fact that lightning strike has an even lower overall trend compared to dents, due to the images being larger and the defects being smaller.

Besides, the difference between the unweighted and weighted entropy is marginal, mostly materialized by outliers appearing in the weighted version, which should probably be considered an artefact related to the computation method itself.

Finally, the augmented data set exhibits wider ranges, both in terms of quartiles and extrema, while retaining similar medians compared to the non-augmented data set. This would tend to indicate that the information brought by the augmentation protocol does not contribute to adding complexity to the images and rather just spreads the existing quantity of information. The greater number of outliers (especially for dents) illustrates this trend which seems to go against the objective of data augmentation as it pulls the overall distribution towards lower entropy values, i.e., toward more randomness.

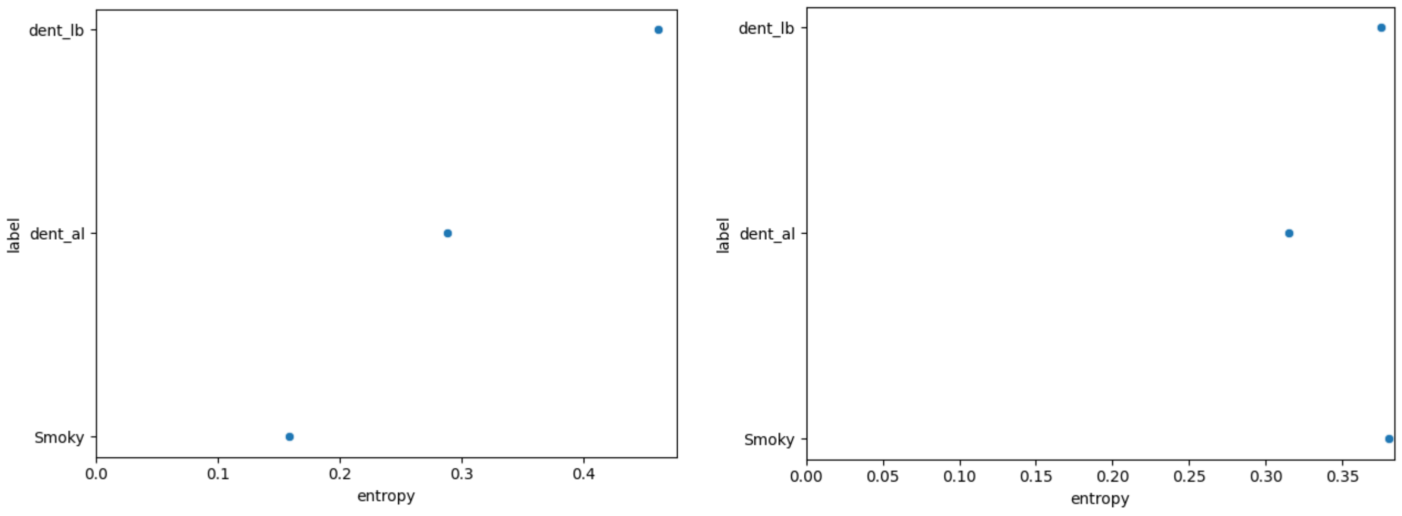
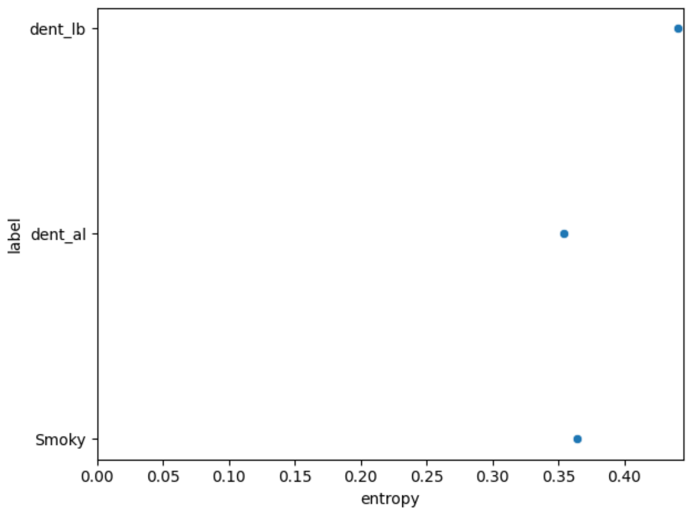


Figure 66: On the left, unweighted label-wise entropy for the AVI data set. On the right, is weighted label-wise entropy for the AVI data set. Both results were obtained on the non-augmented data set.

Label-wise non-augmented unweighted and weighted entropy are reported in Figure 66. Finally, Figure Figure 67 shows the label-wise entropies for the augmented dataset. The theoretical bound for the label-wise entropy is:

$$\log_2(3) = 1.585$$

Again, the empirical entropy is significantly smaller than this theoretical bound, indicating a low class-wise complexity of the images, in line with the image-wise results. Contrary to these results, however, data augmentation boosts the entropy values of classes Smoky (i.e., lightning strikes) and to a more modest extent, dent_al. This relative increase is not surprising considering that lightning strike is augmented with 20 supplementary augmented samples per original occurrence, while each det class is augmented with only 5 generated samples for each original occurrence. By contrast, class dent_lb only sees a marginal increase in the unweighted setting and a drop in the weighted setting. Considering that dent_lb is significantly less represented in the dataset (as discussed earlier), and also has a lower average number of occurrences per image (independently of its lower size), this last trend tends to comfort the hypothesis that, in a multiple object setting, not all sample has a similar contribution and more complex ones are more informative.



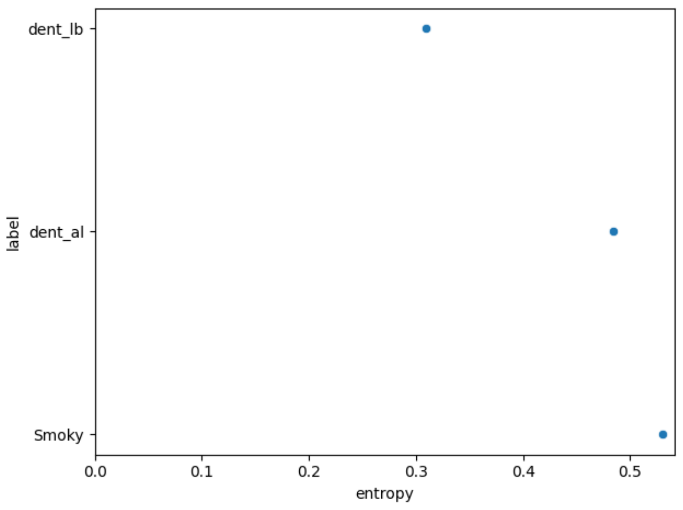


Figure 67: On the left, unweighted label-wise entropy for the AVI data set. On the right, weighted label-wise entropy for the AVI data set. Both results were obtained on the augmented data set.

4.8.5.2 **ATC-STT & ACAS-Xu**

As for the PCA experiment and due to the lacklustre results of the preliminary experiments on audio embeddings, ATC-STT was not experimented on using PCA. Similarly, ACAS-Xu is not fit for entropy analysis.

4.8.5.3 **Conclusion from the MLEAP perspective**

As in the preliminary experiments, the study of entropy ends up being a useful coarse-grain sanity check. Comparison between the theoretical bound and the empirical values allows to get a quantitative indicator of the general task setting. For instance, in the case of AVI, the low empirical information level compared to the bound can be related to operational specifications (indoor pictures, homogeneous plane colours, small elements to identify in large images, etc). In this sense, entropy can be useful for data representativeness assessment w.r.t to high-level specifications (e.g., ODD).

Experiments also show that this kind of analysis can be applied to data augmentation strategies to anticipate their effectiveness. Here, class-wise entropy shows an overall increase in the information available, but detailed analysis of the result shows strong discrepancies and even regression in certain cases. An even more refined analysis (i.e., sample by sample) might help better isolate the problem with the augmentation strategy.

4.8.6 Neuron Coverage

4.8.6.1 **AVI**

The model trained on the AVI data set is a derivative of YOLOv5. It relies on a rather complex architecture made of several layers, some of which are composed of sub-layers. Figure 68 shows a summary of the model[74].

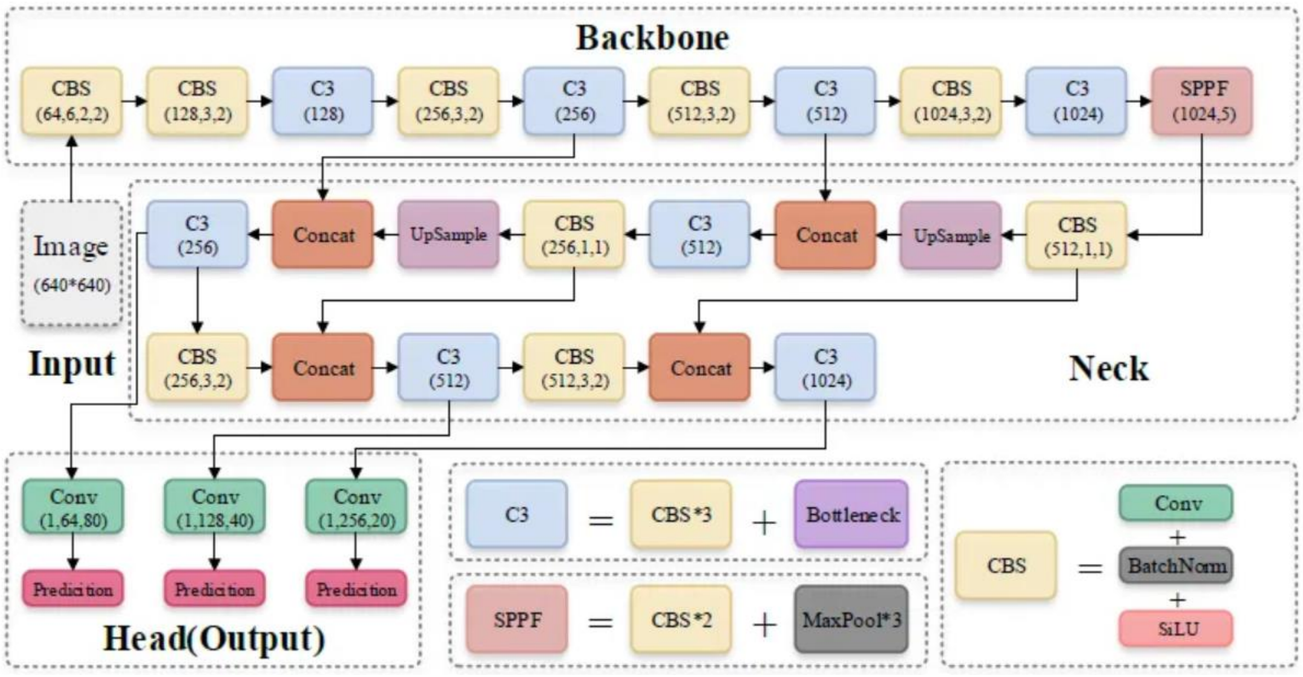


Figure 68: YOLOv5 architecture overview.

Following the same protocol described in the preliminary experiments, the first step is to get insight on the influence of the activation threshold. Figure 69 and Figure 70 show the layer-wise activation of the model using the two methods covered in section 4.7.3.6.1 (i.e., mean unique and mean average, respectively) for the non-augmented test data set of the lightning strike case. In the figure's legend, each layer is plotted in-depth order, i.e., the solid blue « Conv » line is the first layer met by the input, and the pink starred « Conv2d » layer is the last (prediction) layer.

[74] Extracted from : https://sh-tsang.medium.com/brief-review-yolov5-for-object-detection-84cc6c6a0e3a

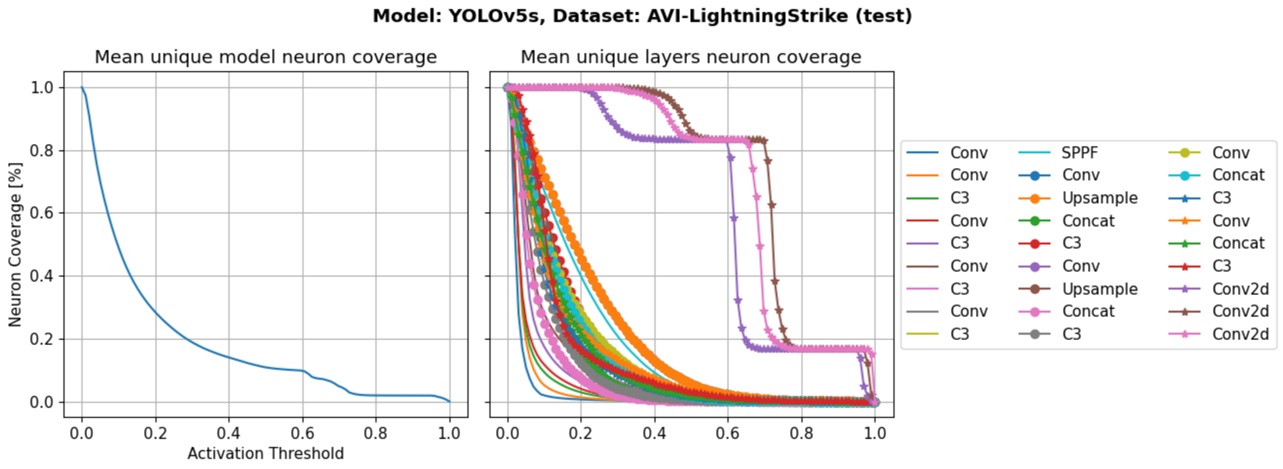


Figure 69: Mean unique coverage for the non-augmented lightning strikes test data set.

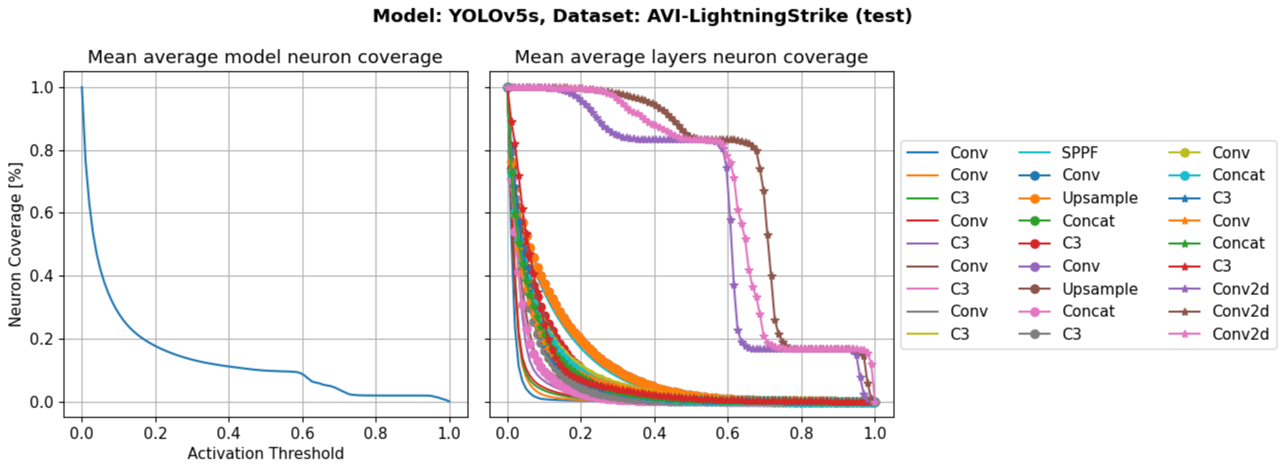


Figure 70: Mean average coverage for the non-augmented lightning strikes test data set

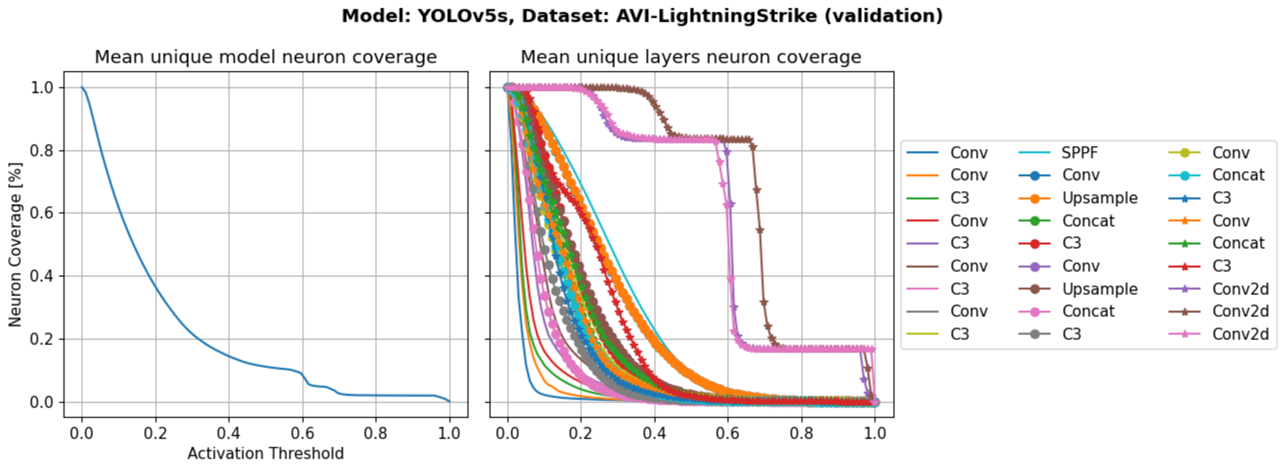


Figure 71: Mean unique coverage for the non-augmented lightning strikes validation data set.

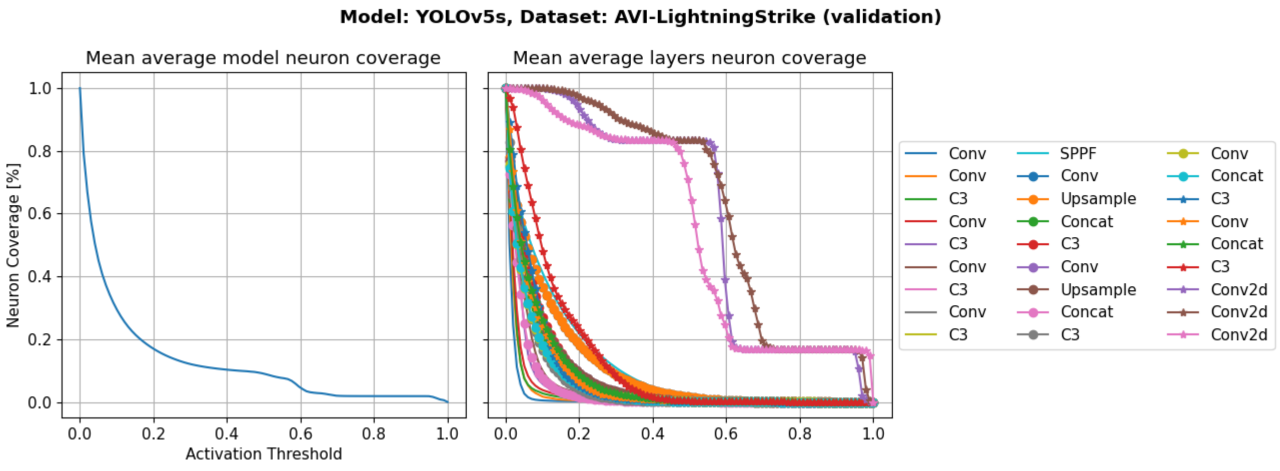


Figure 72: Mean average coverage for the non-augmented lightning strikes validation data set.

All graphs exhibit similar trends, with layers having an exponential decay in coverage that consistently smooths as the layer is positioned deeper in the network. Interestingly, here this trend is independent of the layer type, whereas in the preliminary experiments, this pattern was attributed to the regularization layers. Also, the three last layers follow a completely different behaviour of more linear decay with a sharp drop around 0.5-0.6.

Contrary to the preliminary experiments, here the threshold cannot be chosen by confronting hit-and-miss due to the fact that several classes can be represented in a single image. In MNIST, a given state of the model can systematically be bound to a specific prediction. In AVI, since there are potentially several predicted classes over an image, all these predictions would be indiscriminately bound to a single state of the model. A change of definition of the hit/miss state could enable experimentation, but results would be intrinsically lacking in information due to this entanglement of the predictions. The choice of thresholds will therefore be made following these coverage plots. The selected values are 0.1 (for being the rough value where layer activation is the most diverse while being on a generally decaying trajectory), 0.55 (because it is the rough value where the activation of the last three layers is most significant compared to the rest of the network while being similar to each other) and 0.65 (as it is the rough value where the three last layers' activation behaviours are most diverse). These three thresholds should cover a large panel of network behaviour.

The heatmaps of the network for each threshold are reported in (for threshold 0.1, 0.55 and 0.65, respectively). Considering the complexity of the model, only a few layers are displayed for readability. The index of the layer, depth-wise is indicated, i.e., Layer n°0 is the first one, n°1 the second, layer 14 is roughly in the middle of the network and 26 is the last layer. For each layer, each cell is a convolution in the layer. The contingency of the cells does not reflect an architectural reality and is just for convenience. However, pixel-contingency within a cell is interpretable in terms of geometrical patterns.

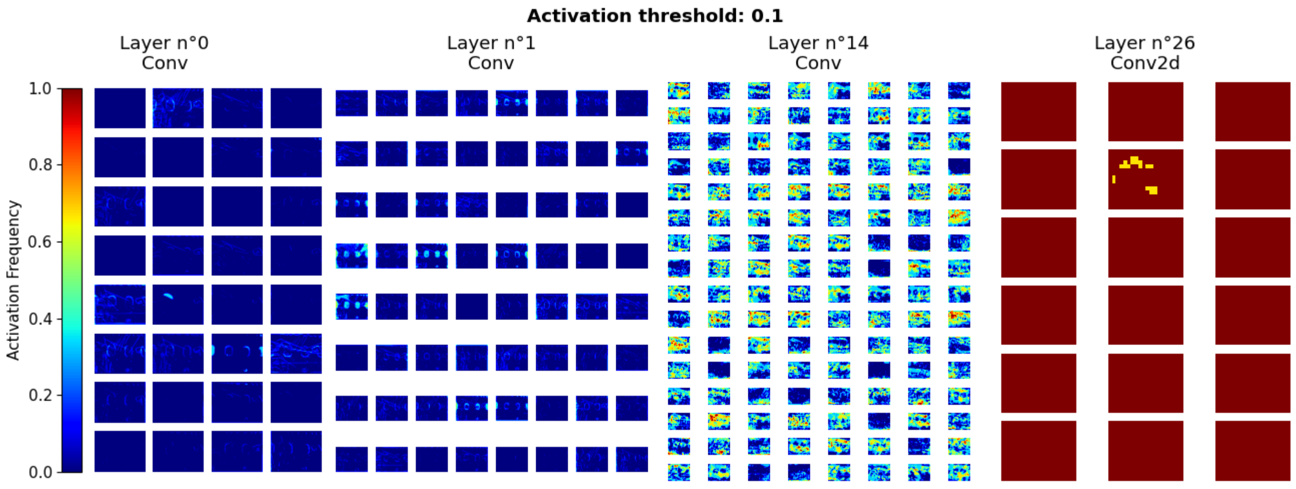


Figure 73: Layer-wise neuron activation heatmap on the non-augmented lightning strike test set for activation threshold=0.1.

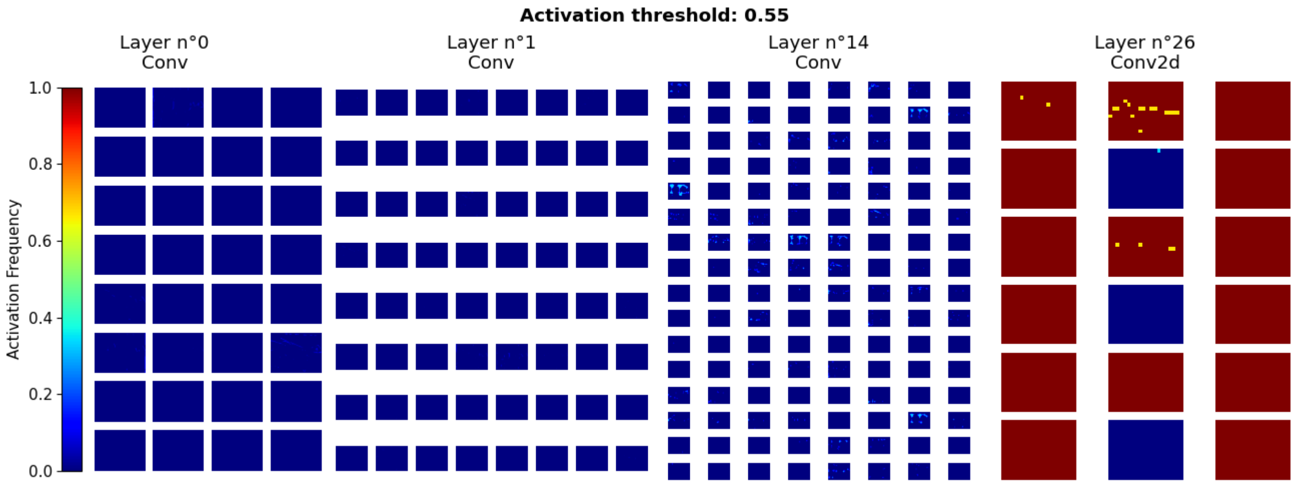


Figure 74: Layer-wise neuron activation heatmap on the non-augmented lightning strike test set for activation threshold=0.55.

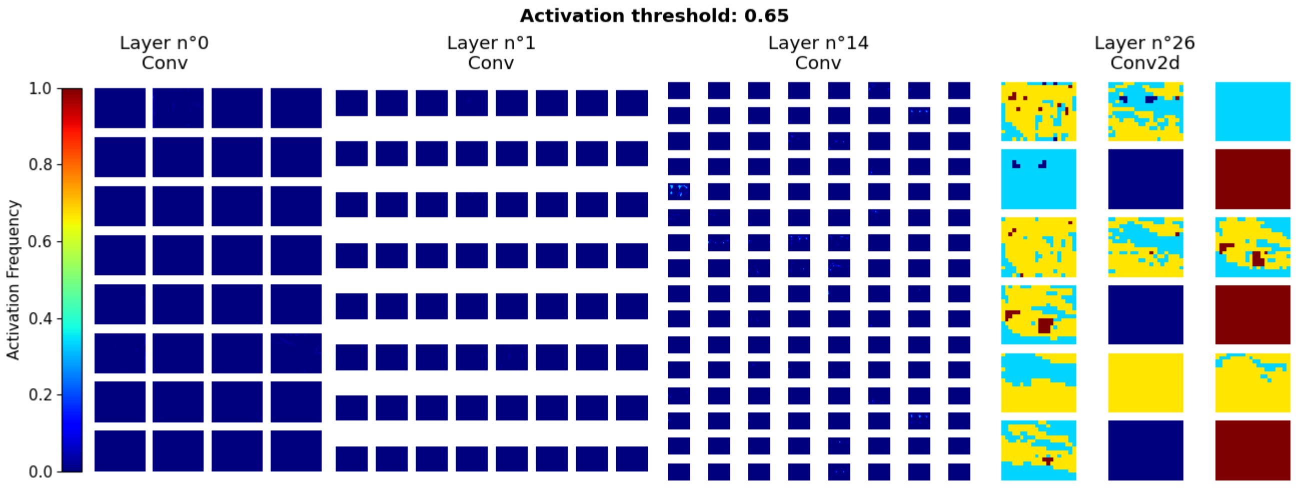


Figure 75: Layer-wise neuron activation heatmap on the non-augmented lightning strike test set for activation threshold=0.65.

At threshold = 0.1, even the first layer unsurprisingly exhibits some degree of activation compared to other threshold values. The mid-network layer has an interesting behaviour, with almost all convolutions active with various frequencies of activation within each convolution. This suggests that the network still has a lot of capacity available. Nonetheless, the last layer is uniformly active, with the exception of one convolution showing some marginally less solicited neurons.

At threshold = 0.55, the network is mostly inactive (at least in its first half). This would tend to indicate that most of the representation learning happens in the second half of the model. In the final layer, there is a sharp contrast between some convolutions being completely unused and most others being systematically activated, reinforcing the previous idea of some degree of capacity still available (which as a corollary precludes the possibility that the network overfitted the data set). This trend also highlights the phenomenon of dimension reduction and how information concentrates within the network.

Finally, threshold = 0.65 presents a similar activation behaviour to the previous value, except for the last layer which is more heterogeneous. It can be noted that the unused convolutions are the same for both threshold values. As the activation threshold is starting to become quite high, this aforementioned heterogeneity is probably explained by the fact that few neurons would regularly fire at such levels, suggesting that there are no specialized sub-structures within the network, and thus a robust training.

To illuminate this intuition, a complementary run was performed with threshold = 0.85. The resulting heatmap is visible in Figure 76. Only the three homogeneously most frequently activated convolutions are still active, confirming that they form a consistently active core that is not dependent on the class to predict.

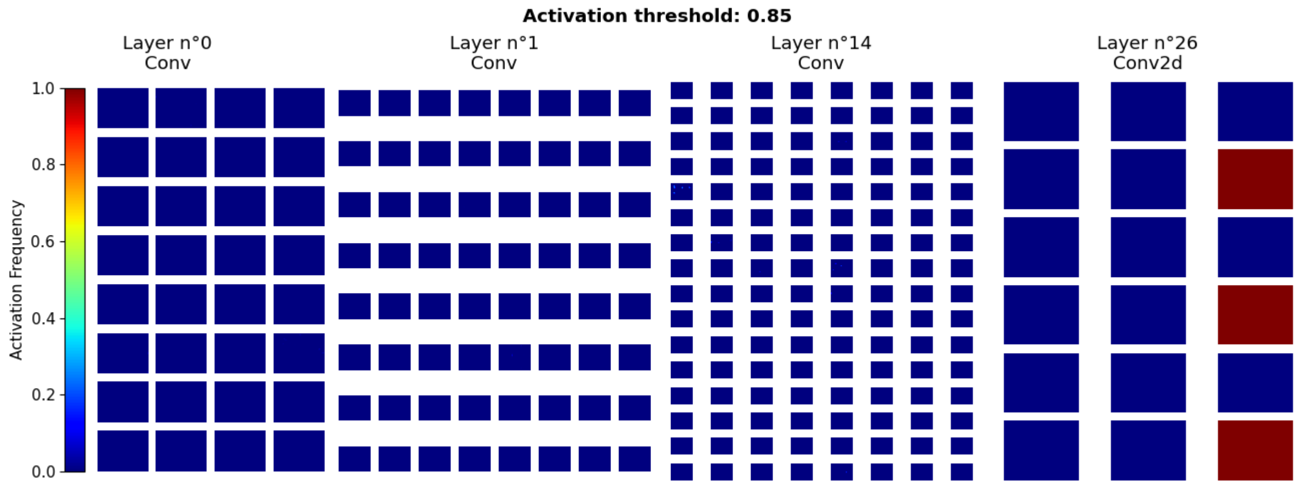


Figure 76: Layer-wise neuron activation heatmap on the non-augmented lightning strike test set for activation threshold=0.85.

The model fine-tuned on the augmented data set was not available at the time of these experiments, which prevented the completion of a contrastive study of the non-augmented vs augmented neuron coverage.

4.8.6.2 **ATC-STT**

Neuron coverage should in principle be applicable to any DNN-based system. The difficulty of the STT task is that it is not a classification task at the utterance level, so it is difficult to try to establish a correlation between the activations and the output. It should be possible still to try to ensure that normally irrelevant speech characteristics, such as gender or accent, do not induce differentiated activation patterns which could be indicative of an overfitting to fewer present data.

4.8.6.3 **Conclusion from the MLEAP perspective**

The study of the non-augmented lightning strike test set suggests that the model learned a robust representation of the problem, which hints a good degree of representativeness of the data sets, at least between the different splits used.

In any case, choosing a specific threshold value does not seem to be the best strategy for investigating data completeness or representativeness through neuron coverage. Instead, milestone values should be used and their insight cross-analysed, as proposed here. These values can be refined according to the model architecture using the layer-wise visualisation proposed at the beginning of the section. As for any model-driven assessment method (e.g., feature space characterization), neuron coverage can be used at the a posteriori evaluation step of the pipeline discussed in section 7.4.

4.8.7 Feature Space Characterization

4.8.7.1 **AVI**

The feature space characterization method had several limiting factors when applied on the AVI data set. First, Equivalence Partitioning and Boundary Conditioning could not be computed on the lightning strike case, since there is only one class and no empty example (EP would be 1.0 and there is no possible pair of classes to analyse). For technical reasons, the Dents embeddings could not be accessed as a white box, preventing the study of the Centroid Positioning in this case. Boundary Conditioning was computed, but PBC would have been only mildly insightful considering there are only two classes (three if considering « no class predicted » as a class), so it was discarded.
Figure 77 reports EP on AVI/Dents. Empty examples have been included as a standalone class (NA), which allows us to observe their importance in the data set. As a reminder, EP is supposed to tend towards 1.0 for each class.

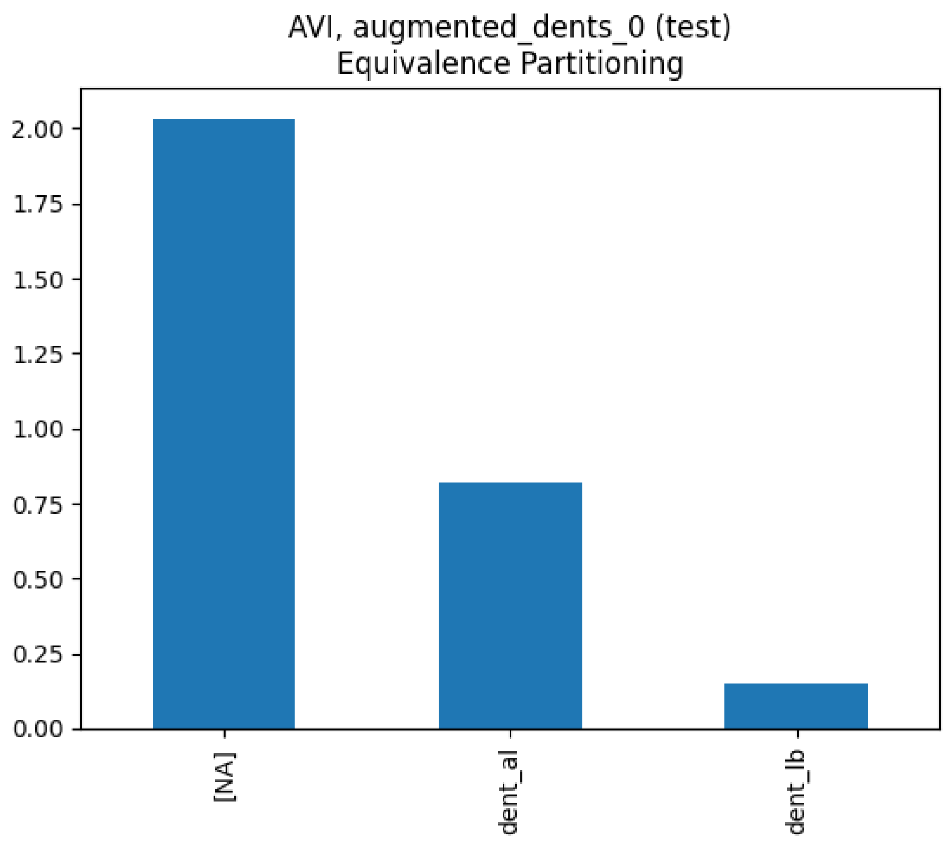


Figure 77: EP on the non-augmented AVI/Dents data set.

Following the approach set in the preliminary experiments, confidence score distributions on AVI/Dents are visible in Figure 78. For each detected object, several bounding boxes are generated and filtered using Non-Maximum Suppression to yield the final prediction. Second guesses were extracted from this filtering step. Contrary to MNIST where the intersection of two distinct overlapping gaussians could be used as a clear point of interest, here each first and second guess are closely related. This is not surprising as it simply means the best and second-best predicted bounding boxes are very close in terms of the Jaccard index (i.e., highly overlapping). However, this prevents the selection of a confidence range making it impossible to run the Boundary Conditioning and Pair-wise Boundary Conditioning metrics.

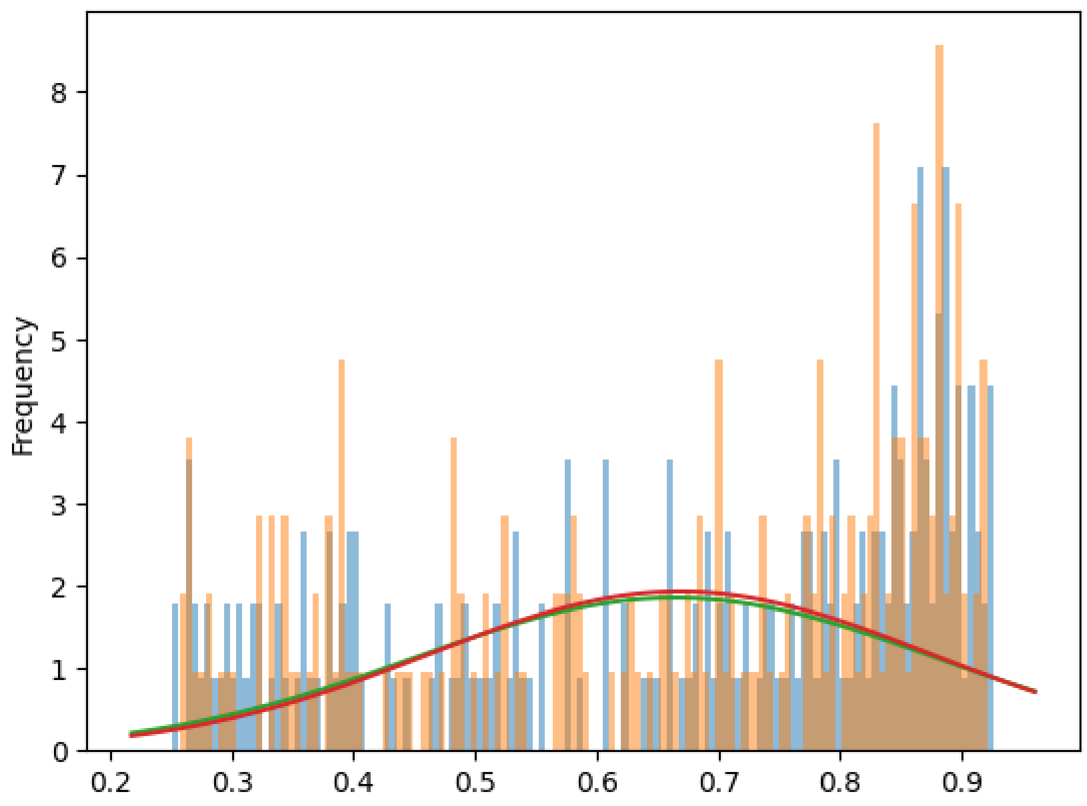


Figure 78: first (blue bars) vs second (orange bars) guess distribution for the AVI/Dents data set.

Finally, Figure 79 shows the Centroid Positioning score in function of the radius on AVI/lightning strike. This plot also follows the protocol described in the preliminary experiments, the objective being to find the largest radius possible while minimizing the CP score. The trend displayed is even sharper than for the MNIST dataset, with an instantaneous switch in value before radius = 0.1. This suggests a heterogeneous dataset, but as for the MNIST data set, this trend is so peculiar that it is difficult to exploit.

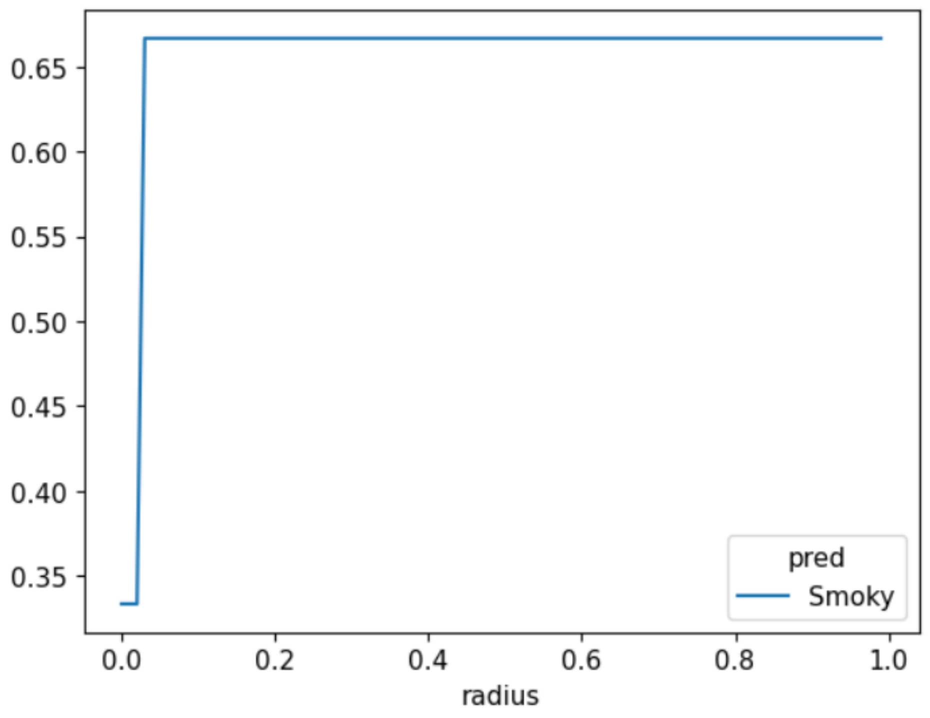


Figure 79: Centroid Positioning scores for the non-augmented AVI/Dents data set.

The limitations of the experimental setup discouraged running the experiment on the augmented data set : EP would be offset by a constant related to the number of additional samples, and fine-tuned models are unavailable for BC and CP computation.

4.8.7.2 **ATC-STT**

Similar to neuron coverage, STT's non-classification task type makes it a poor fit with methods that try to compare behaviours on different target classes. In addition, the limited results obtained on AVI did not encourage further analysis.

4.8.7.3 **Conclusion from the MLEAP perspective**

The results obtained using this method are meagre and cannot be considered otherwise than inconclusive. A main limitation of the method is its specialisation toward classification tasks, which is acknowledged by the authors themselves, an extension to other tasks being considered as future work in their paper. The experiments on the AVI data set therefore already went beyond the intended capabilities of the method. As such, the experimental attempts reported here highlight the difficulty of adapting the method to other tasks, especially due to the dependencies between BC and PBC, and the difficulty to exploit CP, even in a classification setting.

4.8.8 Risk-based approach

4.8.8.1 **Introduction**

This section presents the application of the BSA method on ACAS Xu and ATC-STT.

The AVI use case was not pursued in the experimental phase as the primary goal was to showcase the applicability of the BSA methodology beyond demographic data. This objective was successfully demonstrated through experimentation on the ACAS Xu (aerial collision detection and avoidance) and ROSE (plant detection) use cases. Given this achievement, addressing the AVI use case was deemed unnecessary as it would not have provided additional insights into the methodology's broader applicability. Thus, resources were strategically allocated to areas where they could yield the most significant advancements in proving the versatility and effectiveness of BSA.

The implementation of the BSA method takes place within the context of expert analysis, relying on an understanding of the use case and the expected context of operation of the system. For ACAS Xu, for example, this involves analysing the features used in the calculation of manoeuvre costs, namely the distance, position and speed of the aircraft. In the present analysis, the features are studied separately in order to highlight the reasoning required for the BSA method. The potentiality of a collision, however, results from a combination of all features.

The application of the BSA method requires a fine understanding of the context of operation, and as such we need to reproduce the reasoning about the expected values for the data set in relation to the knowledge of the environment of operation, hence the creation of “rules” for the definition of the desired intervals. The validity of these rules, although they must still remain within the realm of reality, is not the focus of this experiment. Indeed, we are exploring the feasibility of applying the BSA method to various types of use cases, and thus consider certain basic assumptions with the sole objective of supporting the demonstration and highlighting the prevalence of expert knowledge in the application of the method.

4.8.8.2 **ACAS Xu: Experimental protocol**

ACAS Xu use case comes with a specific ODD, as presented in Table 8, which indicates the minimum and maximum values for each feature of the data set:

$$ODD_{\text{ACAS Xu}} = \begin{cases} \tau \in [0, 101] & \rho \in [499, 185318] \\ \theta \in [-3.14, +3.14] & \psi \in [3.14, +3.14] \\ v_{int} \in [0, 1200] & v_{own} \in [50, 1200] \end{cases}$$

For the purpose of the present demonstration, we should illustrate the reasoning to be applied by the expert in charge of the assessment of representativeness. Taking these figures as the basis of the analysis would only mean that we check that the values of the data set in our hands correspond to these intervals, which presents no real added value.

> **Intent and purpose**: detect potential collision between ownship and an intruder ship (horizontal plan).

ODD: detect intruder ship and generate manoeuvre guidance for the aircraft, in order to avoid collision.

Key features: vertical tau, slant range, the relative position of the intruder to the ownship, the relative direction of the aircraft, speed of the ownship, and speed of the intruder.

Key features description in the data set: The ACAS Xu data set analysed concerns the detection on the horizontal plane. It is composed of:

- τ or vertical_tau: time until loss of vertical separation, in s;
- range: slant range between the centre of both aircraft, in ft;
- θ or theta: the angle to intruder relative to ownship heading, in rad;
- ψ or psi: heading angle of intruder relative to ownship heading direction, in rad;
- ownspeed: speed of ownship, in ft/s;
- intrspeed: speed of intruder, in ft/s.

Target population: In the context of ACAS Xu, the population of reference consists of all the realistic configurations that may occur when the detection and avoidance system is used.

A. Time until loss of vertical separation (τ)

The distribution of time values must be consistent with the normal operating conditions of the system. In the present scenario, a time before collision of 0 is not desirable but may be included in the database. However, the minimum time of interest should be considered in relation to the time required for the execution of the avoidance manoeuvre, whether automatic or by a human operator and offer an acceptable margin of safety. The maximum time can be determined based on parameters such as radar detection range (in the surveillance system) and the potential speed, position and distance of the intruding aircraft. The distribution of values within this interval should be homogeneous, assuming that no situation is more plausible than another.

B. Slant range

Slant range is defined as the Euclidean distance between the centres of the aircraft, a collision occurs when one of the aircraft enters the collision volume around the aircraft, fixed as a cylinder with a radius of 500 feet and a height of 200 feet. For the purpose of the experiment, the minimum slant range will be set to 500 ft. The maximum distance should be reasonable with respect to the capabilities of the sensors of the surveillance system. One may reasonably consider 200,000 ft for a long-range radar used for military aerial surveillance or air traffic management. The distribution of values in this interval should be homogenous.

C. Intruder angle (θ)

The intruding aircraft is likely to approach from any direction in the horizontal plane, it then appears necessary to take into account all its possible relative positions. If the intruder angle feature is considered in combination with the speed of the considered aircraft and their distance, a smaller arc may need to be considered. When considered alone, the Intruder angle feature should present a homogeneous distribution of the values.

D. Heading angle of intruder (ψ)

All possible angular positions may lead to a collision, depending on the other parameters (distance, speed, position). Here again, the combination with specific values of the other features may make an angular position more or less likely to lead to collision. When considered alone, the Heading intruder angle feature should present a homogeneous distribution of the values.

E. Speed (ownship, intruder)

The acceptable range of speed may vary considerably depending on the type of aircraft. As stated by (Owen et al., 2019), the system is expected to provide "collision avoidance capability against all aircraft". One may consider that a realistic minimal speed is stalling speed, 30 ft/s for large drones, 100 ft/s for light aircraft or 200 ft/s for commercial airliners. Excluding potentially experimental supersonic aircraft, one may consider a maximum desirable value of about 1,800 ft/s for fighter jets. For the definition of realistic possible speed values, one may consider commercial airliners and drones (about 1,000 ft/s max for airliners and 150 ft/s max for the fastest drones). All speed values in the intervals should be homogeneously distributed in the data set.

Criterion 1: "The demographic distribution of training data is reasonably similar to the distribution of the population where the system will be deployed."

- **Class analysis:**

A. Time until loss of vertical separation (τ)

Although the values may be split into categories for further analysis of the subpopulations (see Criterion 2), this feature is made up of continuous values, where each sample represents a duration, and, in itself, is not made up of classes. The class analysis can rely on a study of the range of values to ensure adequate coverage of values relative to the given reference interval. As defined previously, an acceptable span can range from 0 seconds to a duration determined according to several potential factors, such as the maximum range of detection of the surveillance system and the potential speed of the aircraft. Any positive value not exceeding 600 s may be considered acceptable.
Data show a Min of 0 and a Max of 101.
The score shows adequate representation.

B. Slant range

Similarly, to the previous case, slant range is made up of continuous values and the values may be split into categories for further analysis of the subpopulations (see Criterion 2). In the case of Criterion 1, the class analysis can rely on a study of the range of values to ensure adequate coverage of values relative to the given reference interval. As defined in this experiment, the minimum value can be 0 ft up to a distance realistic in terms of sensing capabilities, set to 200 000 ft.
Data show a Min of 499 and a Max of 185 318.
The score shows adequate representation.

C. Intruder angle (θ)

All angles can potentially be linked to a risk of collision. When the intruder angle is combined with the other parameters (speed and direction), some classes linked to a level of risk may emerge.
No potential classes have been identified. The calculation is not relevant.

D. Heading angle of intruder (ψ)

All angles can potentially be linked to a risk of collision. Some classes linked to a level of risk may emerge when the intruder's heading angle is combined with the other parameters (position and speed).
No potential classes have been identified. The calculation is not relevant.

E. Ownship speed

Even though speed is directly related to avoidance capabilities, we do not distinguish in this experiment clear speed categories that could be directly linked to a level of risk. However, speeds below the stall speed may be unrealistic and should be considered a class to be excluded – in this experiment, we set this minimal speed at 30 ft/s. The minimum speed for ownship in the data set is 50 ft/s.
Ownship speed: 1/1 = 1
The score shows adequate representation.

F. Intruder speed

The minimum speed for an intruder in the data set is 0 ft/s, which means that two categories were represented in the data set: "acceptable speed" and "below stalling speed," which may be considered unfit for the purpose of the data set.
Intruder speed: 2/1 = 2
The score shows inadequate representation.

- **Sample analysis:**

In the context of ACAS Xu, comparing the distribution of the samples with the distribution of the samples in the target population lacks meaningful significance because this would equate to comparing against all conceivable scenarios. A more realistic approach may involve ensuring that the distribution of values makes sense from an expert standpoint through an analysis of dispersion and homogeneity.

A. Time until loss of vertical separation (τ)

Given that the distribution is expected to be homogeneous (equally represented for all the categories of interest), a relevant analysis can focus on the Interquartile Range (IQR) in order to analyse the dispersion of the values. A mean value is relevant since the values do not vary significantly between the data sets.

$$IQR = \frac{\sum_{i=1}^{n}(Q_{3i} - Q_{1i})}{n} = 75$$

Where:

- *n* is the total number of data sets,
- Q_{3i} is the third quartile of data set *i*,
- Q_{1i} is the first quartile of data set *i*.

Table 23. Average distribution of samples for Time until loss of vertical separation (τ).

IQR	75
Mean	42
Std	39

Min	0
Q1	5
Q2	40
Q3	80
Max	101

These high values of the standard deviation (39) and the IQR (75), relative to the mean (42), suggest significant variability in the data. However, the data indicates an asymmetric distribution with a concentration of values on the right side of the distribution. The distribution characteristics, such as the minimum (min), quartiles (Q1, Q2, Q3), and maximum (max), indicate a tendency toward higher values.
The scores show inadequate representation.

B. Slant range

Table 24. Average distribution of samples for Slant range.

IQR	40 174
Mean	47 814
Std	44 918
Min	499
Q1	16 569
Q2	36 656
Q3	56 743
Max	185 318

The distribution appears to have a significant spread, with values dispersed over a wide range. This could suggest substantial variability in the data, with a certain distribution homogeneity.
The scores show adequate representation.

C. Intruder angle (θ)

Table 25. Average distribution of samples for Intruder angle.

IQR	3.14
Mean	5.2024E-17
Std	1.858594
Min	-3.1416
Q1	-1.5708
Q2	0
Q3	1.5708
Max	3.1416

The values show that all the possible angles are represented (from –π to π rad). The IQR is relatively narrow compared to the total range of the distribution, indicating the significant concentration of data around the median. Quartiles indicate an apparent symmetry around the median, with a notable concentration of data near zero. Overall, these indicators suggest

a relatively homogeneous distribution centred around zero with moderate dispersion, which could potentially indicate an underrepresentation of specific value ranges. Nevertheless, the distribution appears generally suitable.
The scores show adequate representation.

D. Heading angle of intruder (ψ)

Table 26. Average distribution of samples for Heading intruder angle.

IQR	3.14
Mean	5.2024E-17
Std	1.858594
Min	-3.1416
Q1	-1.5708
Q2	0
Q3	1.5708
Max	3.1416

The total range from -3.1416 to 3.1416 indicates a complete distribution of values. The relatively low standard deviation (1.858594) suggests moderate dispersion of values around the mean. The concentration of data around zero, with a mean close to zero and moderate dispersion (low standard deviation), suggests some homogeneity in the distribution. Quartiles show a significant concentration of data around the median (Q2) with apparent symmetry and a relatively balanced distribution of values across the range of angles, indicating homogeneity in the concentration of data. In summary, the data appears to have moderate dispersion, with a concentration around zero and a balanced distribution of values across the entire range of angles. The data seems to be relatively homogeneous, with a concentration around zero and moderate dispersion of values in the whole range of angles.
The scores show adequate representation.

E. Ownship speed

Table 27. Average distribution of samples for Ownship speed.

IQR	623.1
Mean	550.416667
Std	389.199882
Min	50
Q1	211
Q2	570.6
Q3	834.1
Max	1200

The mean indicates the central tendency, with an average value of 550.416667. The standard deviation is relatively high (389.199882), suggesting a considerable spread of values. The range from the minimum (50) to the maximum (1200) reflects the overall spread of the data, which encompasses the expected values defined as references for the experiment (about 1,000 ft/s

max for airliners and 150 ft/s max for the fastest drones). Quartiles show a distribution skewed towards higher values, with a substantial IQR. The data appears to have notable variability, and the distribution is skewed towards higher values. The homogeneity is somewhat affected by the dispersion of values across the range, and the quartiles and range further emphasise the variability in the dataset. The analysis may suggest that higher speeds are more prevalent in the data set, presupposing that it would be more suitable for fast ownship aircraft. Without precise information on the expected type of ownship in the ODD, we conclude that the data set is not entirely representative from the speed perspective.
The scores show inadequate representation.

F. Intruder speed

Table 28. Average distribution of samples for Intruder speed.

IQR	604
Mean	532
Std	387.13004
Min	0
Q1	203
Q2	545
Q3	807
Max	1200

The mean indicates the central tendency, with an average value of 532. The standard deviation is relatively high (387.13004), suggesting a considerable spread of values. Quartiles show a distribution skewed towards higher values, with a substantial IQR, which indicates the spread of the central half of the data. The range from the minimum (0) to the maximum (1200) reflects the overall spread of the data. The data appears to have notable variability, and the distribution is skewed towards higher values. The homogeneity is somewhat affected by the dispersion of values across the range. The analysis may suggest that higher speeds are more prevalent in the data set, presupposing that it would be more suitable for fast intruder aircraft. The presence of values below the estimated stalling speed does not seem representative.
The scores show inadequate representation.

- **Conformity assessment:** the criterion is not fulfilled.

Criterion 2: The subpopulations likely to interact with the system are sufficiently represented.

- **Class analysis:**

A. Time until loss of vertical separation (τ)

Some classes may be designed according to expert knowledge to depict the level of risks associated with different time intervals. These intervals may vary depending on the type of aircraft considered and the type of collision avoidance manoeuvre (automated by a pilot in the plane or remotely by an operator), which would impact the required duration.
However, One may consider that the longer the duration, the safer the aircraft are from collision. For the experiment, we split arbitrarily the values into “extreme risk” (0 to 10 seconds), “high risk” (11 to 20), “average risk” (21 to 40), and “safe” (above 40 seconds).

Naturally, these values would be designed by air traffic control experts in natural settings. Since the data set presents values for each of these categories, one may verify that:
Time until loss: 4/4 = 1
The score shows adequate representation.

B. Slant range

Although the values may be split into categories for further analysis of the subpopulations (see Criterion 2), this feature is made up of continuous values, where each sample represents a duration, and, in itself, is not made up of classes. The class analysis can rely on a study of the range of values to ensure adequate coverage of values relative to the given reference interval. This feature is made up of continuous values. Similarly, to the time of vertical separation loss, the values can be discretized according to safety considerations, which may vary heavily depending on the type of aircrafts considered. For the experiment, we split arbitrarily the values into "extreme risk/collision" (0 to 500 ft), "high risk" (501 to 1,000), "average risk" (1,001 to 2,000), and "safe" (above 2,000 feet).
Since the data set presents values for each of these categories, one may verify that:
Slant range: 4/4 = 1
The score shows adequate representation.

C. Intruder angle (θ)

No classes were identified in the conformity assessment for Criterion 1. Therefore, it is not possible to pinpoint any subclasses further.

D. Heading angle of intruder (ψ)

No classes were identified in the conformity assessment for Criterion 1. Therefore, it is not possible to pinpoint any subclasses further.

E. Ownship speed

No classes were identified in the conformity assessment for Criterion 1. Therefore, it is not possible to pinpoint any subclasses further.

F. Intruder speed

In the conformity assessment for Criterion 1, no classes were identified. Therefore, it is not possible to further pinpoint any subclasses.

- **Sample analysis:**

A. Time until loss of vertical separation (τ)

According to the figures in Table 8, the classes determined arbitrarily for the experimentation are not equally distributed, with an underrepresentation of "high risk" and "average risk" classes.

Table 29. Distribution of classes for Time until loss of vertical separation.

Classes	# of samples	Proportion
Extreme risk (0-10 s)	37 761 984	40 %
High risk (11-20 s)	9 440 496	10 %
Average risk (21-40 s)	9 440 496	10 %
Safe (40+ s)	37 761 984	40 %

Total	94 404 960	100 %

The scores show inadequate representation.

B. Slant range

The figures presented in Table 8 a substantial underrepresentation of the first three classes.

Table 30. Distribution of classes for Slant range.

Classes	# of samples	Proportion
Extreme risk / collision	2 420 640	2.56 %
High risk	2 420 640	2.56 %
Average risk	0	0 %
Safe	89 563 680	94.87 %
Total	94 404 960	100 %

These results indicate a glaring error in the execution of the method, as it is unlikely such a bias exists in the ACAS Xu data set. Here, the definition of classes does not appear suitable for the use case, as 94% of the corpus lies within the class that we initially deemed "safe". The classes could have been reworked in a second step, in order to bring them closer to an estimation induced by an analysis of the present data since our first definition of classes seems to be invalid. However, such an approach would be incorrect in the light of the method, as it should not be derived from the data; rather, it should be derived from expertise on the considered use case and then applied to the data. We will stick to these results for the purpose of the demonstration.
The score shows inadequate representation.

C. Intruder angle (θ)

Not applicable.

D. Heading angle of intruder (ψ)

Not applicable.

E. Ownship speed

Not applicable.

F. Intruder speed

Not applicable.

- **Conformity assessment:** the criterion is not fulfilled.

4.8.8.3 ATC-STT: Experimental protocol

The corpus is described in section 3.3.2. We used the ATCO2-ASR dataset to make the analysis more interesting as it includes more accents.

Intent and purpose: transcribe speech signal in order to extract relevant information for the pilot or the air traffic controller.

ODD: correctly detect the spoken instructions.

Key features: gender, accent.

ACT-STT presents several other potential key features of interest for representativeness analysis, such as the speech rate or noise types. The decision to focus solely on gender and accent in the study is justified by their sufficiency in demonstrating the BSA methodological approach. Indeed, both parameters encapsulate significant dimensions of communication dynamics and operational relevance within the aviation context, while being directly linked to demographic characteristics.

Key features description in the data set:

A. Gender: female, male and unknown;
B. Accent: 67 different accents, noted as "language_COUNTRY" with ISO 639 codes.

Target population: In the context of ATC-STT, the population of reference consists in all the realistic configurations that may occur in the use of the STT system within the limits of the *Intents and Purpose* section, e.g., pilots and ATC controllers.

Criterion 1: "The demographic distribution of training data is reasonably similar to the distribution of the population where the system will be deployed."

- **Class analysis:**

A. Gender

The data set presents three classes: female, male, and unknown. "Unknown" does not represent a class in itself but an absence of information and will not be included in the analysis.
Biological genre: 2/2 = 1
The score shows adequate representation.

B. Accent

The data set contains 24 languages, comprising several official languages of the European Union, and languages outside of Europe. In the operation context, the system may have to face additional languages coming from outside Europe or from other European countries not listed (Bulgarian, Maltese, etc.), which would increase the required number to more than 6,000 languages worldwide[75]. The analysis should however take into account the countries that are indeed involved in aviation, for which figures could not be found.

The data set represents 24 countries, both in Europe and outside, with 11 European countries missing. With 195 different countries worldwide, the data set is not representative. Here again, however, the analysis should take into account countries involved in aviation, and the data set should be realistic in terms of combination language/country.

[75] https://www.worlddata.info/languages/

The data set may be representative in terms of the most frequent language/country combinations, but without reference value, a meaningful comparison is not possible.

Without appropriate reference, the assessment of representativeness for this criterion is not possible.

- **Sample analysis:**

A. Gender

The analysis should consist in verifying that the distribution of gender classes in the data set is similar to the distribution of the population. According to a review performed by the International Civil Aviation Organisation[76], "4.9% of all pilots, air traffic controllers, and maintenance technicians are women".
According to the results in Table 31, the proportion of women in the data set is 25% in terms of sample number, and 28% in terms of speaking time, which is far above the proportion of reference.

Table 31. Distribution of samples and speaking time according to gender.

Gender	Number of samples	Percentage	Speaking time (s)	Percentage
Female	19 219	25.3	102 949.46	28.21
Male	56 504	74.5	261 490.90	71.64
Unknown	113	0.1	546.09	0.15
Total	75 836	100	364 986.45	100

The score shows inadequate representation.

B. Accent

Without an appropriate reference value, the comparison would not prove relevant. However, Figure 80 shows a strong representation of the French language in the data set (in terms of the number of samples), which may indicate inadequate representation.

[76] Several results of the report are presented in https://www.sesarju.eu/news/mind-gap-why-gender-equality-air-traffic-management-matters#:~:text=When%20we%20zoom%20in%20on,globally%20and%2021.4%25%20in%20Europe.

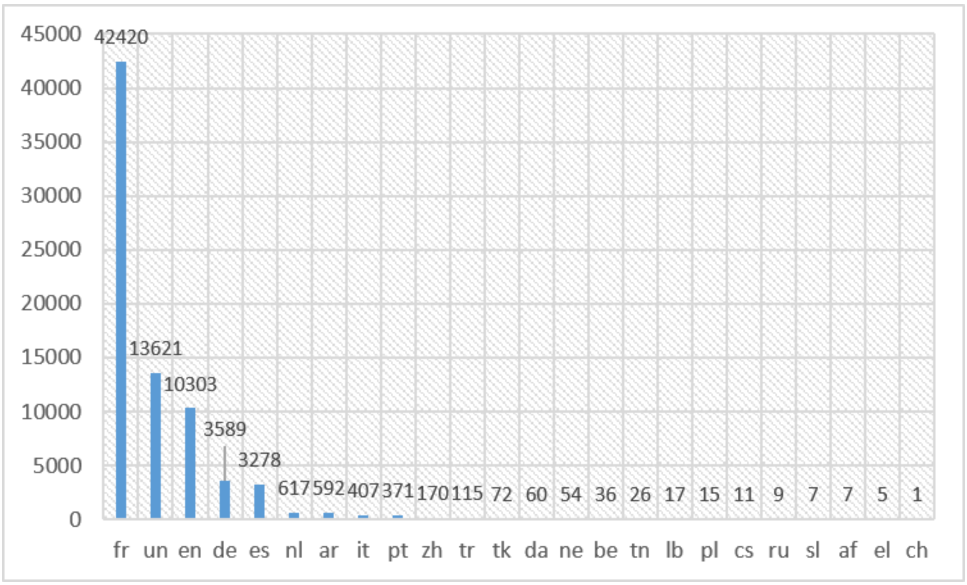


Figure 80. Distribution of sample number per language.

The scores show inadequate representation.

Criterion 2: The subpopulations likely to interact with the system are sufficiently represented.

- **Class analysis:**

A. Gender

Although the Role, as labelled in the data set (ground staff, pilot, controller, etc.) is not explicitly a subpopulation of gender, it allows a finer analysis of the distribution in the Gender class. One would then verify that for each role, the gender distribution is representative of the target population.
The roles presented in the corpus (P, C, A, M and G, see hypothesis on the roles in 4.8.4.1) likely correspond to the roles expected in the target domain of use, with no logical reasoning or source allowing us to state otherwise, which provides a score of:
Roles: 5/5 = 1
The score shows adequate representation.

B. Accent

No relevant subpopulations could be identified.

- **Sample analysis:**

A. Gender

Information can be found about the proportion of females in civil aviation and air traffic controllers, but almost no sources are publicly available for the other roles noted in the data set (ground staff, ATIS, etc.).
For the analysis, we will take the report from the Centre for Aviation mentioned above, which indicates that around 5% of pilots worldwide are female. Table 32 presents the gender distribution for the role of the pilot. The figures show that the data set consists of 5.6% females, which is in line with the reference.

Table 32. Distribution of samples and speaking time for the role Pilot (P) according to gender.

Gender	Number of samples	Percentage	Speaking time (s)	Percentage
Female	2 310	5.6	9 734.87	5.8
Male	38 783	94.2	157 386.25	94.0
Unknown	59	0.1	249.32	0.1
Total	41 152	100	167 370.44	100

The same reference indicates that 20% of women worldwide work as air traffic controllers. As shown in Table 33, the distribution in the data set is 50% (in terms of number of samples and speaking time), which is above the reference value.

Table 33. Distribution of samples and speaking time for the role Controller (C) according to gender.

Gender	Number of samples	Percentage	Speaking time (s)	Percentage
Female	16 532	49.5	81 423.12	50.1
Male	16 856	50.5	80 889.65	49.8
Unknown	15	0.0	95.33	0.1
Total	33 403	100	162 408.10	100

The score shows inadequate representation.

B. Accent

No relevant subpopulations could be identified.

4.8.8.4 **Results from the MLEAP Perspective**

The BSA method was applied to the use cases VoxCrim, ROSE, ACAS Xu, and ATC-STT. The experimentation aimed to verify whether the application of the technique seemed far-fetched in certain use cases, or if it made sense and provided relevant insights into the data quality.
BSA is a method leveraging expert human analysis. This method is not equipped in the reference text itself, thus it was necessary to define an experimentation plan to verify the compliance of the data set with the method's criteria. The criteria aim at analysing the general distribution of the population within the data sets and the distribution of subpopulations. The analyses conducted are statistical.
The BSA method's reference text refers to demographic characteristics and, therefore, seems designed for human populations only. However, it is not unreasonable to extend the notion to “population” in the statistical sense, which represents the complete set of elements or individuals subject to a study.

No particular barriers presented themselves to applying the BSA method, which seems potentially applicable to all use cases.

Table 34 summarizes the information and application domains contained in each use case.

Table 34. Overview of the applicability of BSA method on the use cases.

Use case	Data type	Annotated object	Application domain	Applicability of BSA method
VoxCrim	Voice segments	Speakers	Forensics	X
ROSE	Plant images	Plants	Agriculture	X
ACAS Xu	Flight features tables	Aircrafts	Air safety	X
ATC-STT	Voice segments	Speakers	Air safety	X

The application of the BSA method underscores the paramount importance of expert knowledge for determining threshold values and acceptable or potentially unacceptable ranges, considering both the expected system capabilities and realistic field conditions. Expert analysis should also determine the most relevant statistical indicators to describe the degree of data representativeness. As illustrated in the case of Criterion 2 for "slant range" in ACAS Xu, where an error in class definition led to aberrant results, the analysis should not be data-driven but expert-driven.

The experimentation results presented in this document are not based on a thorough expert analysis of the application domains, since this would require expertise in each specific domain (agriculture, forensics, etc.); they mainly rely on external references providing partially adaptable information to the considered use case, supplemented by the experimenters' common sense. The representativeness or non-representativeness results presented in this document should therefore not be regarded as indicators of the actual quality of the datasets but as a demonstration of the application of the method.

The analysis of use cases has highlighted the relationship between the application of the BSA method and the consideration of specific domain context characteristics, such as ethics or operational safety. For example, the analysis of VOXCRIM emphasized the importance of the fairness dimension for preliminary dataset analysis. In the case of ACAS Xu, context analysis should include risk management, allowing to determine conditions or configurations likely to represent risks, and thus conditioning the analysis of data representativeness.

It should be noted that in the experimentation context, the study only focused on present textual annotations and not on the data itself or additional metadata. For instance, in the case of ROSE, the representativeness study could have also focused on an analysis of image characteristics, or additional phenotypic indicators related to the presented plants; the only constraint being that the parameters under study are relevant, for a human expert, in terms of representativeness analysis. It is the responsibility of the person in charge of the assessment to determine the appropriate compromise between the feasibility of obtaining information (cost, existence, etc.), the possibility of obtaining human-understandable information, and the relevance of the information concerning the representativeness study.

4.9 Conclusion

In light of all the elements discussed in this document, the field of data quality management is at a low degree of maturity. While general processes are mostly defined, normative efforts about definitions and terminology are still ongoing. Moreover, the importance of data curation often remains neglected due to its cost-intensive and painstaking nature. The dependency on the nature of both the data and the task makes the possible combinations numerous and not well covered. Validating a dataset really requires trying to make the best of the existing methods yet considering the results with some trepidation. Consequently, it is paramount to never rely on a single assessment method and rather to combine several of them, both prior and after model training. This double-ended approach is central to ensure coverage of the targeted objectives stated in section 1.5.1.1, with a priori methods enabling compliance with DA-03 and DA-04, the a posteriori (or model-driven) methods helping with compliance with DM-07.

Meanwhile, most tools and methods publicly available lack operationalizability. Their many individual shortcomings encourage the multiplication of empirical, specialized works. This fragmentation is incompatible with industrial requirements; thus, a consolidation step of the most generic and efficient method is needed but cannot be expected without incentives. On the research side, such incentives may take the form of challenges and other events to stimulate the development of theoretically sound, general-purpose solutions. On the industrial side, this can be done by encouraging the adoption and development of such methods, through labels, standards, certifications and regulations.

In addition, an important point is that many methods do not explicitly address completeness or representativeness but rather data quality in general. Moreover, assessment methods do not identify and leverage influence factors as structuring elements of their methods. This results in various approaches, from devising data-quality-aware objective functions that work at training time, to assessment methods working directly on the data set, with in-between propositions such as using the model's learning process as a proxy to get insight on the data set. The heterogeneity of the solutions makes the choice of a method even more difficult.

The experimentations empirically confirmed this combination of scarce information, complex operationalizability and lack of focus on the specific issues of completeness and representativeness. A framework for such an assessment would require significant engineering and expert knowledge. While it is probable that such a framework would be portable to similar tasks, at least each different task would require a tailored solution aggregating several metrics and tools and requiring substantial analysis. A reasonable first step to require from the system designers would be to provide documentation about whether completeness and representativeness have been ensured in relation to the factor of influence listed in this document, the methodology chosen to do so and the justifications behind this choice. Experts may then be consulted to evaluate the soundness and sufficiency of the approach.

One of the major difficulties in assessing completeness and representativeness is to have reliable information about the distributions of phenomena of the intended behaviour in its operational context. Such assessment must be performed on a case-by-case basis and in most cases requires extensive expert work. No off-the-shelf methodology exists to define clear requirements prior to data collection. Finally, it is important to keep in mind that this assessment task must also take into account the necessary trade-off posed by robustness and resilience requirements, i.e., ensuring completeness by enabling sufficient performance on specific cases while preserving an overall degree of representativeness.

5. Model development: Generalisation properties

5.1 Introduction

This part is about evaluating Model Generalisation in AI. It aims to review existing methods of machine learning (ML) and deep learning (DL) generalisation guarantees and evaluation, distinguishing between methods from the two fields. Problems and limitations are analysed, and new evaluation methods are developed. This section includes a comprehensive overview and state-of-the-art analysis of existing methods for ML and DL models in terms of generalisation process and bounds. In this section, we first present the generalisation issues and known ML/DL-related underfitting and overfitting problems. Then, we provide an overview of existing methods to address these aspects in a general way and how to detect overfitting/underfitting cases that prevent models from generalising. Furthermore, we comprehensively review the available methods and tools to evaluate generalisation bounds. Finally, we identify the barriers to the tractability of the objective of quantification of generalisation guarantees for a given AI model and provide a generic development and evaluation pipeline, dealing with the identified limitations to promote generalisation after training and model implementation.

5.2 Background concepts

In this section, we introduce the main concepts in machine learning, deep learning, and artificial intelligence and give general terminology definitions, w.r.t the definitions and terms used in EASA[77] 's second paper (EASA, 2024).

5.2.1 AI modelling

In Chapter 1, we briefly described using and implementing artificial intelligence applications as a mathematical function *f*. The latter is then fitted to some training data in a given context to process additional data and make predictions. Several models have been defined for AI applications. There are two main categories: (1) ML models, which could be supervised[78] or not, and based on different architectures and theoretical approaches. A detailed taxonomy can be found in (Shyam and Singh, 2021): (2) DL models, which are made of several computational layers stacked on each other. DL uses supervised and unsupervised strategies to learn multi-level representations and features in hierarchical architectures for several tasks, such as classification or regression. A detailed description can be found in (Sarker, 2021).

[77] https://www.easa.europa.eu/

[78] Supervised learning (Caruana and Niculescu-Mizil, 2006) includes a set of algorithms that reason from a set of instance samples where input examples are provided with expected outputs, in contrast to unsupervised learning (Celebi and Aydin, 2016) where the objective is to discover latent connexions and signals between the input samples (e.g. creating clusters, learning semantic representations). Note that in this document, we are more interested in supervised models, especially in the different formal descriptions of the models, without neglecting the unsupervised approaches. Examples of works from both approaches will be cited indifferently.

Independently of the model's class, developing ML/DL-based applications requires a series of steps from the design phase to the operational phase. Given a generic data set D made of three disjoint subsets D_{train} for training, D_{test} for testing, and D_{val} for validating (called also development set), such that $D = D_{train} \cup D_{test} \cup D_{val}$ and $\emptyset = D_{train} \cap D_{test} = D_{train} \cap D_{val} = D_{test} \cap D_{val}$.
Given a hypothesis space F such that $f \in F$. We consider two main steps to put an ML/DL model into production following the pipeline defined in (Cluzeau et al., 2020), as shown in Figure 81.

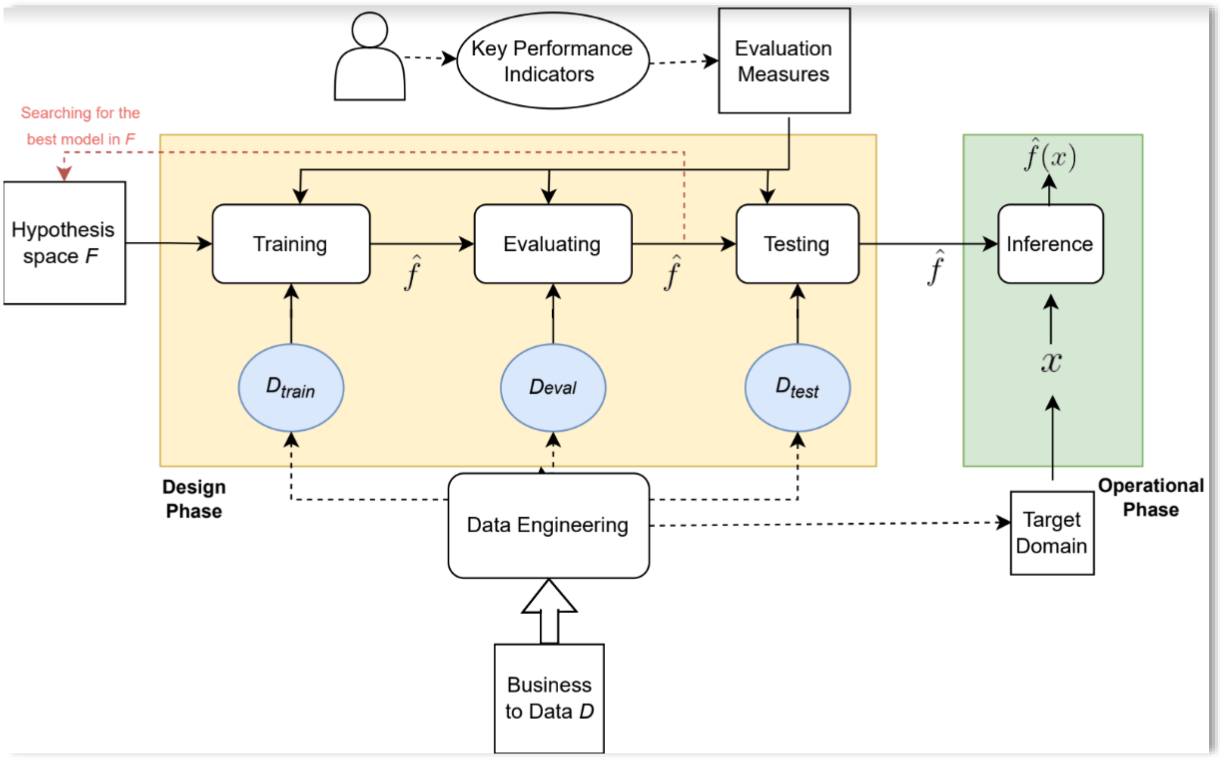


Figure 81. From the design to the operational[79] phase of machine learning modelling and production.

1. *The design phase includes choosing an ML model (algorithm), training on a dedicated data set, evaluating the best model, and testing its performance.* In this phase, it is widespread for a model to be changed or modified due to a lack of performance in the evaluation set.
2. *The operational phase:* at that point, the model should have provided satisfying results and can keep the same performance level on unseen data samples. The trained model is simply used to predict new inputs (inference).

In both phases, design and operational, we assume that the same data engineering pipeline is used to process data samples. The evaluation measures (cf. section 2.2) can be derived from the key performance indicators provided by the domain application expert.

[79] This phase can include an implementation step of the model. Where the latter can be embedded in the target system and be part of its pipeline.

5.2.2 Learning process

The learning process aims at going through a hypothesis space $F,$ using a learning algorithm and repeatedly updating the model's parameters. The goal of a (supervised) learning algorithm (Cluzeau et al., 2020) is to learn a function $f : X \rightarrow Y$ from an input space X to an output space Y, using a finite number of example pairs $(x, f(x))$, with $x \in X$. These sample pairs represent parts of the target domain for which the application is being designed. More precisely, given a finite training data set D_{train}:

$$D_{train} = \{(x_i, f(x_i)) : 1 \leq i \leq n_{train}\}, f(x_i) = y_i \in Y$$

the goal of the training algorithm is to generate a function

$$\hat{f}(D_{train}) : X \rightarrow Y$$

That approximates f "well", as measured by some error metric. $\hat{f}_\theta(D_{train})$ is the result of a learning algorithm trained on data set D_{train}, where θ is the parameters of the final function $\hat{f} \in F$ whose parameters (weights and biases, referred to as hyper-parameters) are θ and enables to better perform the ongoing task.

5.2.3 Error metrics

The pointwise quality of the approximation of f by $\hat{f}$ is measured by using a predefined choice of *error* or *loss* function $\mathcal{L}: Y \rightarrow \mathbb{R}^+$ expecting that

$$\mathcal{L}(\hat{f}) = \sum E\left(\hat{f}(D_{train})\right)$$

is low on all $x \in X$. These *"metrics"* can emphasize that *"lower value means better performance"*. For example, if Y is a subset of the real numbers, one could simply use the absolute values (resp. squares) of the error $E(y, \hat{y}) = |y - \hat{y}|$ (respectively $(y - \hat{y})^2$).
These metrics help identify whether the model can maintain high performance in the face of data changes and perturbations (robustness aspect) and how it would behave in new data instances that have not been evaluated during training (generalisability aspect).

5.2.4 Robustness

In ML/DL, the term *"robustness"*, according to the EASA's CoDANN-1 paper (Cluzeau et al., 2020), is used to refer to the ability of the system to perform the intended behaviour in the presence of abnormal or unknown inputs and to provide an equivalent response within the neighbourhood of an input.
In the literature, the robustness is defined differently. For instance, in (Doshi-Velez and Kim, 2018), the robustness evaluates how ML models are effective in unseen data samples, which can also refer to the generalisation ability. In (Xu and Mannor, 2012), the robustness of a model is defined based on the property that if a testing sample is "similar" to a training sample, then the testing error is close to the training error. This means that, for similar domains, the errors at testing time and training time should be correlated. In the MLEAP project, the robustness of ML/DL models is covered by Chapter 6,

which provides a more detailed definition and analysis of the *"robustness"* as a quality of AI applications.

5.2.5 Generalisability

Once the model has been trained, it is evaluated in a test set D_{test}. This set is used to evaluate the model's ability to generalise the learned knowledge to a new context or environment. The most important success indicator is to produce a model that can perform well in unseen data set $X \neq D_{train}$ (*out-of-sample*[80] examples), and not simply memorise the *in-sample*[81] examples of the subset D_{train}. The memorisation aspect is used to describe an overfitting behaviour of the learning process. The following sections will explain these aspects and show how to recognise the lack of generalisation.

5.2.6 Stability

In analysing a model's robustness, the *stability* concerns the learning algorithm and the model itself. According to (Cluzeau et al., 2020), performance stability can be observed based on the behaviour of the trained model $\hat{f}$ when it deals with a noisy data set. Hence, it is a measure of how much changing a data point in D_{train} can change the learned model (Gonen and Shalev-Shwartz, 2017; Hardt et al., 2016; Kuzborskij and Lampert, 2018). Associated with model robustness, stability ensures that the produced model keeps a defined level of performance under perturbations of the training data set. In other works (Subbaswamy et al., 2021), model stability, associated with robustness, is defined as the ability of the model to preserve its robustness level while dealing with varying contexts and environments. Hence, such stability analysis should demonstrate the range of environments and operational domains in which a model performs excellently and which types of changes in the environment will degrade performance. The stability characteristic of a model is essential so that environmental changes to the model will not affect its performance, especially in critical applications such as those in the medical field (Bai et al., 2021) and aeronautics (Torens et al., 2022). Further, in this document, we will focus on the performance stability of a learning algorithm and the trained model. This notion is further developed in Chapter 6.

5.3 Overfitting and underfitting

5.3.1 Overfitting vs Underfitting

Overfitting and underfitting are two aspects that show that the ML/DL model is not learning well or that it cannot perform on unseen data or training samples. According to (Cluzeau et al., 2020), on the

[80] The out-of-sample data, in some references (Chu and Qureshi, 2022) and this document, refers to data examples that have not been used during training. In the CoDANN paper (Cluzeau et al., 2020), this term is used to define error based on the expected loss (on sampled data), compared to the empirical loss (on the training data). The out-of-sample data, in some references (Chu and Qureshi, 2022) and this document, refers to data examples that have not been used during training. In the CoDANN paper (Cluzeau et al., 2020), this term is used to define error based on the expected loss (on sampled data), compared to the empirical loss (on the training data).

[81] According to the CoDANN paper (Cluzeau et al., 2020) the in-sample data includes all the training samples.

one hand, as the model becomes more complex, it can fit the data better (i.e. bias decreases); on the other hand, it will become susceptible to it (i.e. variance increases). These two facts yield a specific and different evolution of error values. Figure 82 shows the variance and bias trade-off that optimises the learning errors.

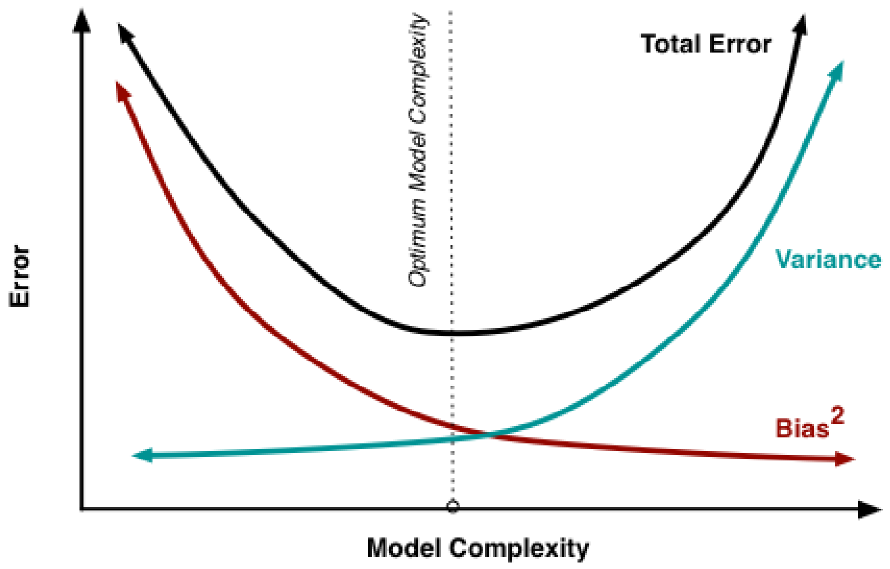


Figure 82. Bias-variance trade-off[82], for better error optimisation, based on the model's complexity.

Simple models usually have high bias and low variance (sometimes called underfitting), while more complex ones (e.g., deep neural networks) have lower bias but higher variance (sometimes called overfitting). This can easily be observed by taking the simple case where the algorithm chooses among a finite set of models (hypothesis), such that if F contains a single model, the variance will be zero, but the bias might be significant if the single model does not approximate well the target; in contrast, when F contains different models, the bias should be more negligible, since there is more capacity to find a model $\hat{f}$ that approximates f well, but the variance will be non-zero.
Both overfitting and underfitting have risks: overfitting leads to models that do not generalise well (and may not perform well in unseen contexts), while underfitted models do not achieve a satisfactory trade-off. A trade-off between these two extremes must be reached depending on the performance and safety requirements.
For more highlights, an empirical study comparing these aspects can be found in (Koehrsen, 2018), and a case study in adversarial learning is detailed in (Z. Li et al., 2020).

5.3.2 Under/Over-fitting detection

As explained above, overfitting occurs when an algorithm reduces error through the memorisation of training examples, and sometimes of noisy or irrelevant features, instead of learning the regularities in the data samples from the input space X and the output space Y (Krueger et al., 2017; C. Zhang et al., 2021). Overfitting prevents models from perfectly generalising the same performances, from

[82] https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

observed data during training to unseen data during testing (Ying, 2019). This happens primarily because of the presence of noise (randomness), the limited size of the training set, and the unsuitable complexity of the model (Krueger et al., 2017; C. Zhang et al., 2021). Overfitting prevents models from perfectly generalising the same performances, from observed data during training to unseen data during testing (Ying, 2019). This happens primarily because of the presence of noise (randomness), the limited size of the training set, and the unsuitable complexity of the model.
Underfitting occurs when there is insufficient model capacity or training to thoroughly learn the exact relationship between input and output spaces, whether through memorisation or not. To determine a model's generalisation ability and limitations (Bashir et al., 2020), we need to estimate its capacity, knowing the approximated true risk (expected loss), that has a central role in generalisation. Here, we first define the aspects related to the training algorithm, then bridge the relationship with the resulting model's ability to generalise:

- ***Capacity:*** describes the learning capabilities of a model $f \in F$, as opposed to the complexity, which indicates the expressiveness of functions in the algorithm's hypothesis space F (e.g., linear functions for a regression model). The capacity C_f of a model f is the maximum amount of information I that f can extract from a training data set D_{train}, when selecting its output hypothesis (Bashir et al., 2020), formally,

$$C_f = sup_D \, I(f \mid D_{train})$$

  Note that the maximum amount of information that a model may transfer from a data set, w.r.t. a hypothesis, is the number of bits required for it to memorise a one-to-one mapping between each *feature-label* pair in that data set. Hence, the capacity aspect greatly impacts the model's generalisation.

- ***True risk (loss):*** is the aggregated expected loss values over the different data samples x of the test data set: $\mathcal{L}_{D_{test}}(f)$ is estimated by a sampling of the test data sets.

- ***Overfitting:*** *is diagnosed by comparing a model's losses* on training and test data sets, where the error on the test set (average observed loss) is intended to approximate the true risk. Observationally, if the true risk $\mathcal{L}_{D_{test}}(f)$ exceeds the empirical risk $\hat{\mathcal{L}}_{D_{train}}(f)$ (the risk on the training data set), the algorithm seems to overfit.

  Formally, an algorithm F overfits a data set D_{train} if it selects a hypothesis $\hat{f} \in F$ such that

$$\mathcal{L}_{D_{test}}(\hat{f}) > \hat{\mathcal{L}}_{D_{train}}(\hat{f})$$

  Where:

$$\mathcal{L}_{D_{test}}(\hat{f}) = \frac{1}{m}\sum_{j=1}^{m} \mathcal{L}(\hat{f}, x_j) < \mathcal{E}$$

  for some fixed scalar $\mathcal{E} > 0$ for any data distribution D_{test}, and for every element in the training data set:

$$\hat{\mathcal{L}}_{D_{train}}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\hat{f}, x_i)$$

- ***Underfitting:*** An algorithm F underfits at iteration i of the model f, if after training for i times, its capacity is strictly less than the estimated one for a given model $f \in F$, on the training data set, ie: $C_f^i < C_f$.

- ***Double descent risk:*** Based on the error evolution during training (training risk), and the overfitting and underfitting definitions, the *double descent* is used in recent ML/DL works (Belkin et al., 2019; Nakkiran et al., 2021) to refer to the bias-variance trade-off in the context of complex models. It bridges the gap between the observed evolution in simple ML models and deep models like NNs and Gradient Boosting. Empirically (Belkin et al., 2019), given a training data set of size n and a deep model of complexity N, the shape of the risk curve displays the double decent evolution, such that while N is increasing, the risk initially decreases, attains a minimum, and then increases until $n = N$. This twofold descent is called the *double descent risk*.

  As we can see, capacity as a property of the training algorithm and the resulting model enables us to estimate the underfitting and overfitting of a trained model $f \in F$. Machine Learning algorithms will perform well when their capacity is in adequation with the complexity of the task that they need to perform and the amount of training data they are provided with. This will have a direct impact on the capacity of the trained model:
  - An algorithm with low capacity struggles to fit the training set, which means that the resulting model will have insufficient capacity and, hence, may underfit;
  - An algorithm with a high capacity could solve more complex tasks; however, when the capacity is higher than needed, it will make the models memorise all the training set, including irrelevant signals (noise), leading to overfitting;

  One way to deal with this problem is to control the capacity of a learning algorithm and the resulting model by choosing the hypothesis space or through an empirical analysis carried out beforehand to restrict the search space of the models.

5.4 Generalisation in AI

Generalisation is one of the most fundamental aspects of ML and DL. For centuries, scientists have exploited the empirical fact that unknown outcomes of a given process, whether future or unobserved, often trace regularities in past observations. This is called generalisation (C. Zhang et al., 2021): finding rules consistent with available data that apply to instances we have yet to encounter. Thus, a variety of theories have been proposed to explain generalisation. Uniform convergence[83] (G. H, 1918), margin theory (Wei et al., 2019), and algorithmic stability[84] (T. Liu et al., 2017) are some of the essential conceptual tools to reason about generalisation. In addition, the capacity of a model can

[83] A sequence of functions f_n converges uniformly to a limiting function f on a set E if, given any arbitrarily small positive number ε, a number N can be found such that each of the functions $f_N, f_{N+1}, \ldots$ differ from f by no more than ε at every point x in E.

[84] Captures stability of the hypothesis output by the learning algorithm in the normed space of functions from which hypotheses are selected.

be evaluated to help predict generalisation. When the complexity of a model is very high, regularisation introduces algorithmic tweaks intended to reward models of lower complexity.

5.4.1 Model Complexity

As explained in (Hu et al., 2021), the model complexity is highly dependent on its architecture and other important factors, including the model framework, model size, optimisation process, and data complexity. In deep learning, the model complexity concerns the network architecture and how complicated problems the model can express (Bianchini and Scarselli, 2014; X. Hu et al., 2020). In classical machine learning models, the model complexity definition differs from one model to another (Bohanec and Bratko, 1994; Bulso et al., 2019). For example, in decision trees, the complexity is measured by tree depth and the number of leaf nodes. At the same time, logistic regression is investigated using several metrics, such as the perspectives of Vapnik-Chervonenkis (VC) theory[85] (Tempo et al., 2013), Rademacher complexity (Kakade et al., 2008), Fisher Information matrix (Bulso et al., 2019), and the razor of the model (Balasubramanian, 1997).

Model complexity can be categorised into:

1) *Expressive capacity*: also known as representation capacity, expressive power, and complexity capacity (Liang et al., 2019; Poggio et al., 2017a). It describes how well a deep learning model can approximate complex problems. Informally, the expressive capacity describes the upper bound of the complexity in a parametric family of models. It is based on:
    1) Depth efficiency, which analyses how deep learning models gain performance (e.g., accuracy) from the depth of architectures;
    2) Width efficiency, which analyses how the widths of layers (e.g., Vectors dimensionality in fully connected networks) in deep learning models affect model expressive capacity;
    3) Expressible functional space that investigates the functions that can be expressed by a deep model with a specific framework and specified size using different parameters;
    4) VC Dimension and Rademacher Complexity are two classic measures of expressive capacity in machine learning.
2) *Effective* complexity: *practical complexity*, practical expressivity, and usable capacity (Hanin and Rolnick, 2019; Novak et al., 2018). It reflects the complexity of the functions represented by deep models with specific parameterisations. It is based on two different aspects:
    1) General measures of practical complexity that design quantitative measures for the practical complexity of deep learning models. e.g., the maximum number of samples on which the model must be trained to obtain a training error close to zero (Nakkiran et al., 2021) ;
    2) Investigations into the high-capacity low-reality phenomenon find that the effective complexity of deep learning models may be far lower than their expressive capacity.

[85] In Vapnik-Chervonenkis theory, the Vapnik-Chervonenkis (VC) dimension is a measure of the capacity (complexity, expressive power, richness, or flexibility) of a set of functions that can be learned by a statistical binary classification algorithm.

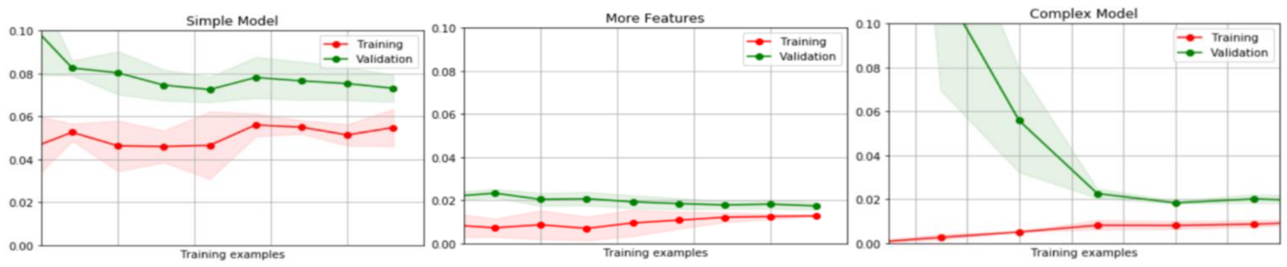


Figure 83. Learning curves[86] showing the error evolution of different models with different complexities

Figure 83 shows how model complexity impacts the training. We can see that adding more features reduces errors in both training and validation data. However, when complexity is increased again with even more features, training error improves, but validation error does not, which means the model is beginning to overfit. A trade-off needs to be made:

- Using fewer features reduces model complexity. However, inadvertently removing key features necessary for the prediction can quickly make a model too simplistic to perform well.
- Increasing the number and size of layers used in a neural network model, or the number and depth of trees used in a random forest model, increases model complexity.

High-Capacity Low-Reality Phenomenon

The *effective complexity*, reflecting the model's characteristics that make it practical enough to address target problems, and the *expressive capacity*, reflecting its ability to express those problems, are highly related and must be balanced. Several studies explore the gap between these two properties in deep learning models to define standards that could help design effective DL-based systems. In (Ba and Caruana, 2014), the empirical study shows that shallow, fully connected neural networks can learn complex functions, and deep neural networks can. Given a well-trained deep model, a shallow model can be trained based on the outputs of the deep model. As a result, the shallow mimic model can achieve an accuracy as high as the deep model's. However, the shallow model cannot be trained directly on the original labelled training data to achieve the same accuracy. This model derivation is called *knowledge distillation* (Hinton et al., 2015). This study suggests that there may be a big gap between the practical effective complexity of a deep learning model and its expressive capacity, which is called *the high-capacity low-reality phenomenon*. This is why effective model complexity is a relatively promising and useful research topic in deep learning. Detecting effective model complexity during training helps to investigate the usefulness of optimisation algorithms and explain the model efficiency and results (Kalimeris et al., 2019). Furthermore, effective model complexity can be defined based on the model's and target data's relationship. Hence, it can be considered a reflection of the information volume in the model (Du, 2016) and can be used for model selection and design to balance resource utilisation and model performance according to every use case's technical requirements and characteristics.

5.4.2 Generalisation Error Bounds

Generalisation bounds are statistical tools that take several features of the data and the model being trained into account and output a performance estimate for unseen data — that is, they estimate how

[86] https://www.pico.net/kb/overfitting-variance-bias-and-model-complexity-in-machine-learning/

well the predictor generalises to unseen data. Knowing the generalisation error bound (gap) means ensuring theoretical guarantees that the model will perform well on unseen data samples. It is a statement about the predictive performance of a learning algorithm or class of algorithms (Reid, 2010). Under the assumption that the performance of a learning algorithm can be expressed in terms of the expected loss of its hypotheses, given randomly selected training samples, a generalisation bound is a theorem which holds for any distribution and states that, with a high probability P_D, applying the learning algorithm to a randomly drawn sample D will result in a hypothesis with a loss that is no greater than some value ε:

$$P_D\left(\left|\mathcal{L}_{out}(\hat{f},m) - \mathcal{L}_{in}(\hat{f},D,m)\right| < \varepsilon\right) > 1 - \delta$$

$\mathcal{L}_{out}$ is the out-of-sample loss value, and $\mathcal{L}$ is the in-sample loss. $\delta \epsilon (0, 1)$ represents the probability tolerance, and ε is the generalisation gap tolerance. The *Probably Approximately Correct* (PAC)-learning setting is then defined by the inverted form (Cluzeau et al., 2020):

$$P_D > 1 - \delta\text{:}\ G(\hat{f},D) < \varepsilon(\delta, |D|, bounds)$$

The bounds represent several involved theoretical aspects, and they almost share the same intuitive idea: more complex models are more prone to overfitting and lesser generalisation (Cluzeau et al., 2020), especially in a domain with few annotated data. All bounds have a complexity term on the right-hand side, which controls looseness. A general form of existing bounds is as follows:
With the probability $P_D > 1 - \delta$, for any estimated model $\hat{f} \in F$:

$$G(\hat{f},D) \ \leq \ \sqrt{\frac{func(model\ class\ F\ complexity) + \log(1/\delta)}{\|D_{train}\|}}$$

$$G(\hat{f},D_{train}) \ \rightarrow 0 \quad as \ \ \|D_{train}\| \rightarrow \infty$$

In the following, we review some of the most used bounds in ML, grouped according to the taxonomy proposed in CoDANN-1 document (Cluzeau et al., 2020).
Note that the generic formulation of generalisation bounds, *func* can depend on many various terms; for example, it can rely on the hypothesis class *H*, the volume of samples, delta, which is the probability of which the bound fails, the data distributions, the architecture of the algorithm we use and may be the parameters of the trained model.

5.4.2.1 Data-independent, algorithm-independent

This is the first bounds class derived by the Vapnik-Chervonenkis theory (VC) (Tempo et al., 2013). It is based on the complexity of the hypothesis space (model class). The main component of these bounds is the VC-dimension defining how powerful the models are in the hypothesis space F. This power describes their ability to contain as much information as possible about the data. In DL, the VC-dimension of feedforward networks can be bounded in terms of the number of parameters $dim(\theta)$ and can be defined as:

$$VC_{dim} = \tilde{O}(d * \dim(\theta))$$

With θ is the set of trainable parameters (weights and bias), and d (depth) is the number of layers of the network being evaluated. Note that the notion of VC-dimension does not take into account the properties of a particular data set, which is important in practice for a better estimation of generalisation bounds (Dar et al., 2021).

5.4.2.2 Data-dependent, algorithm-independent

Using data set information, such as structure and size, could help tighten generalisation bounds better. One of the data-dependent measures for estimating these bounds is the Rademacher complexity (Kakade et al., 2008), measuring the degree to which a hypothesis space can fit random noise. Compared to the VC-dimension, the Rademacher complexity considers data distribution and provides finer-grained model complexity. A higher Rademacher complexity means that the model can fit a larger number of random labels, and thus, the model has a higher expressive capacity (Bartlett and Mendelson, 2002). Generalisation bounds that are data-dependent and algorithm-independent scale inversely with data margin[87] (a training-data dependent quantity). These bounds are directly proportional to the Rademacher complexity. They can explain good generalisation for classification, in some cases of overparameterisation (more parameters than data points) (Bartlett et al., 1998), where the "effective dimension" is sufficiently small and there is no label noise in the data. Yin et al. (Yin et al., 2019) have proven that the lower bound for the Rademacher complexity of a given algorithm exhibits an explicit dependence on the input's dimension. However, other studies (Neyshabur, 2017; C. Zhang et al., 2021) suggest that deep learning models are often over-parameterized in practice and have significantly more parameters than samples. In this case, the VC-dimension and Rademacher complexity of deep learning models is always too high, so their practical guidance is weak. Recently, (D. Li et al., 2022) investigated the domain generalisation problem, providing a novel learning-theoretic generalisation bound that bounds unseen domain generalisation performance regarding the model's Rademacher complexity. This analysis suggests that domain generalisation should be achieved by performing regularised Empirical Risk Minimization (ERM) with a leave-one-domain-out cross-validation objective.

5.4.2.3 Data-dependent, algorithm-dependent

The methods used in this class consider the learning algorithm's distributional properties. One of the most insightful methods is the PAC-Bayesian framework (McAllester, 2003), which operates with distributions over models involving a prior distribution (i.e. chosen before seeing any data and defining the complexity of possible solutions). It defines the foreseen complexity of the model category and the posterior distribution (i.e., distribution learned once seeing the data), which determines the complexity of the models trained on given data. Hence, these bound measures are both data and model-dependent.

[87] In machine learning the margin of a given data point corresponds to the distance from the data point to a decision boundary (Maji and Berg, 2009). This latter is the reference for a margin classifier to decide to which class every instance belongs.

5.4.2.4 **Overview of Recent Bounds**

In several cases (Dziugaite and Roy, 2017), the values of the generalisation upper bound G w.r.t the values δ and ε for large models (DNN) are less important for small data sets compared to larger ones. Hence, in practice, a validation data set D_{val} is used to compute validation errors, in addition to the D_{test} testing data set, as suggested in (Cluzeau et al., 2020). The indices D_{val} can then be used to optimise the model better during training epochs. Besides, bias and variance need to be estimated and minimised in the train data set. The analysis in (Cluzeau et al., 2020) suggests that one would aim for a model whose complexity is high enough to provide a low bias, but not too high as to cause a high variance. For better estimating the $\text{bias}(F, \|D_{train}\|)$ and the $\text{variance}(F, \|D_{train}\|)$, the following random resampling methods could help:

1) *Bootstrapping* (Efron, 1992): consists of sampling k subsets of the training set D_{train} to train the model in the subsets separately. Note that bootstrap sampling uses random sampling with replacement. This means that it is much possible for an already chosen observation to be selected again. The resulting subsets $D_{i=1\ldots k}$ will help estimate the variance for the different subsets separately.
2) *Jackknife* (Miller, 1974): remains at sequentially removing a single data point from the data set D_{train} and re-training on such a "*reduced*" version of the original data set. Here, the most important (representative) data point will have more impact and hence produce the most different trained instances That is why it is useful to estimate the $\text{bias}(F, \|D_{train}\|)$.
3) *Margin distributions* (Lyu et al., 2022): margins measure how much the input has to be altered to change the output classification. Recent works (Glasgow et al., 2022) have shown that max margins indicate good generalisation behaviour while large ones fail.

Another taxonomy of the different theoretical and empirical generalisation bounds is provided by (Valle-Pérez and Louis, 2020), where the models are grouped based on assumptions about the data, the algorithm or even according to the dependence of their capacity on the training set. Moreover, the classical bounds above in DL models cannot be easily applied due to the over-parameterized setting and the non-linearity related to NN models. To handle these aspects, the classical bounds have been updated, and new bounds have been defined for DL model generalisation:

1) *VC-based bounds* (Maass, 1995)*:* even though a corrected definition of VC-dimension for NNs has been proposed recently (Bartlett et al., 2019), there is still a misinterpretation of the VC-theoretical bound. The analysis (Lee and Cherkassky, 2022) of this bound in a double descent[88] (Nakkiran et al., 2021) shows that it can be fully explained by classical VC-generalisation bounds. This is by applying analytic VC-bounds for modelling double descent in classification problems, using empirical results for several learning methods.
2) *Model compression bounds* (Meir and Fontanari, 1993)*:* Based on the idea of Occam's Razor, which aims to gradually reduce an input space, including data (Talbot and Ting, 2022) or model's parameters (Sun and Nielsen, 2019). In this approach, obtaining stronger generalisation bounds on the simple model might be possible if a complex model can be replaced by a simpler one, up to some small admissible error. Hence, the objective is to perform a model reduction process to reduce the complexity of the model in a low-resource setting (Choudhary et al., 2020).

[88] The double descent means that multilayer neural networks can be trained to achieve zero training error, while generalizing well on test data.

3) *Based on Model Distillation* (D. Hsu et al., 2021)*VC-based bounds* (Maass, 1995): even though a corrected definition of VC-dimension for NNs has been proposed recently (Bartlett et al., 2019), there is still a misinterpretation of the VC-theoretical bound. The analysis (Lee and Cherkassky, 2022) of this bound in a double descent[89] (Nakkiran et al., 2021) shows that this latter can be fully explained by classical VC-generalisation bounds. This is by applying analytic VC-bounds for modelling double descent in classification problems, using empirical results for several learning methods.
4) *PAC-Bayesian bounds for NNs* (McAllester and Akinbiyi, 2013): these bounds are based on the stochasticity of training loss minima. Hence, the original PAC-Bayes bound has been optimised (Dziugaite and Roy, 2017) and shown that one can bound the generalisation gap of a two-layer-stochastic neural network. Another method (Nagarajan and Kolter, 2019), namely *Deterministic PAC-Bayesian* provides a generalisation bounds computation method for DL models via generalising noise-resilience, showing that if on training data, the interactions between the weight matrices satisfy certain conditions that imply a wide training loss minimum, these conditions themselves generalise to the interactions between the matrices on test data.
5) *Statistical guarantees* (V. N. Vapnik, 1999): in this class, the generalisation bounds are defined based on statistics either from data, especially for classes of estimators that are used in practice (Taheri et al., 2021), or the error gradient evolution during training w.r.t the training algorithm (Neu et al., 2021). In the latter, the upper bounds are defined based on the generalisation error that depends on local statistics of the stochastic gradients evaluated along the path of iterations computed by the Stochastic Gradient Descent (SGD) algorithm. The main difference between these bounds resides in the variance of the gradients (concerning the data distribution), the local smoothness of the objective function along the SGD path, and the sensitivity of the loss function to perturbations to the final output. Recently, Taheri et al. (Taheri et al., 2021) have introduced a "scale regularisation" approach, a general class of regularised least-squares estimators. The primary strategy is to disentangle the parameters of a model into a ''scale'' and a ''direction'' – similar to introducing polar coordinates – which allows data engineers to focus the regularisation on a one-dimensional parameter. The scale-regularised least-squares estimators are then provided with a general statistical guarantee for prediction. The main feature of this guarantee is that it connects neural networks to standard empirical process theory through a quantity called the "effective noise". This connection facilitates the specification of the bound to different types of regularisation. As exemplified in the l1-regularisation, this method guarantees the squared prediction error, decreasing the number of training samples.
6) *Geometry analysis bounds*: in previous studies (Russo and Zou, 2016), it was well known that the generalisation error of supervised learning algorithms can be bounded in terms of the mutual information between their input and the output, given that the loss of any fixed hypothesis has a sub-Gaussian tail. Recently, that aspect was generalised beyond the dependencies between input and output information. Other research interests (Rodríguez Gálvez et al., 2021) have developed bounds based on Wasserstein distance. More specifically, it introduces full-data set, single-letter, and random-subset bounds and their analogous in the randomised subsample setting from Steinke and Zakynthinou (Steinke and Zakynthinou, 2020). Moreover, when the loss

[89] The double descent means that multilayer neural networks can be trained to achieve zero training error, while generalizing well on test data.

function is bounded, and the geometry of the space is ignored by the choice of the metric in the Wasserstein distance, these bounds recover from below (and, thus, are tighter than) current bounds based on the relative entropy. In (Neu and Lugosi, 2022), the mutual information is replaced by a strongly convex function of the joint input-output distribution, with the sub-gaussianity condition on the losses replaced by a bound on an appropriately chosen norm capturing the geometry of the dependence measure.

These theoretical bounds adapted to DL model characteristics could help forecast the model's performance. However, DL models, in general, yield uncontrollable generalisability (Cluzeau et al., 2020) related to theoretical guessing of the generalisation performance. Hence, the generalisation bounds associated with the theoretical methods could be larger than the values in practice. Therefore, practical approaches, such as *Regularisation* (cf. Section 5.5.1) in its different forms (e.g., *batch normalisation* and *early stopping*) and *domain generalisation* (cf. Section 5.4.5) are more adapted than simply splitting the data to leverage the particularities of the different data points. An extensive empirical study (Jiang et al., 2020) has investigated more than 40 complexity measures taken from both theoretical bounds and empirical ones, with over 10,000 trained convolutional networks. By systematically varying commonly used hyperparameters, this study has uncovered potentially causal relationships between each measure and generalisation and showed surprising failures of some measures.

5.4.3 Generalisation in ML

As machine learning becomes widely used in different applications, one of the most relevant concerns is the assessment of confidence in the predictions of a machine learning model (Barbiero et al., 2020). Generalisation guarantees can formalise this. In many real-world cases, it is more important to estimate the capabilities of a machine learning algorithm to provide accurate predictions on unseen data, depending on the characteristics of the target problem.

5.4.3.1 **Function of Data Set Characteristics**

Considering the generalisation aspect as a function of a target data set means we must capture relationships between the conception and design context compared to the target context. In (Barbiero et al., 2020), the generalisation aspect of ML models is addressed as a function of data set characteristics. Hence, a quantitative evaluation of different ML models, analysing 109 classification data sets, demonstrated the relevance of using the concept of the *convex hull* of the training data while assessing machine learning generalisation. In addition to several predictable correlations in different data samples, there are weak associations between the generalisation ability of ML models and metrics related to dimensionality, such as the *curse of dimensionality* that might impair generalisation in machine learning.

The *curse of dimensionality* denotes a variety of phenomena hindering data analysis if many variables need to be considered simultaneously (Bellman, 1966; Bittner, 1962). This aspect prevents ML models from generalising and includes problems like data sparsity, collinearity, and overfitting (Altman and Krzywinski, 2018). In addition to the model capacity, the concept of *extrapolation*, defined as the ability of the model to correctly predict data points that are considerably different from the information provided in the training data, is a part of the generalisation problem. Extrapolation comes from computational geometry. Let's consider data points as points in $\mathbb{R}^d$ where d is the dimension of the features vector representing each point. The convex hull of a data set is the smallest convex polygon that contains all data points. Given the convex hull of a training set, it is then possible to assess whether an unseen test data point would fall inside or outside its convex hull. The hypothesis is that, for points within the convex hull, an ML model will interpolate using known data to obtain a prediction. In contrast, the same model will extrapolate for test points outside the convex hull. An example is presented in Figure 84, where data points of $\mathbb{R}^2$ are presented.

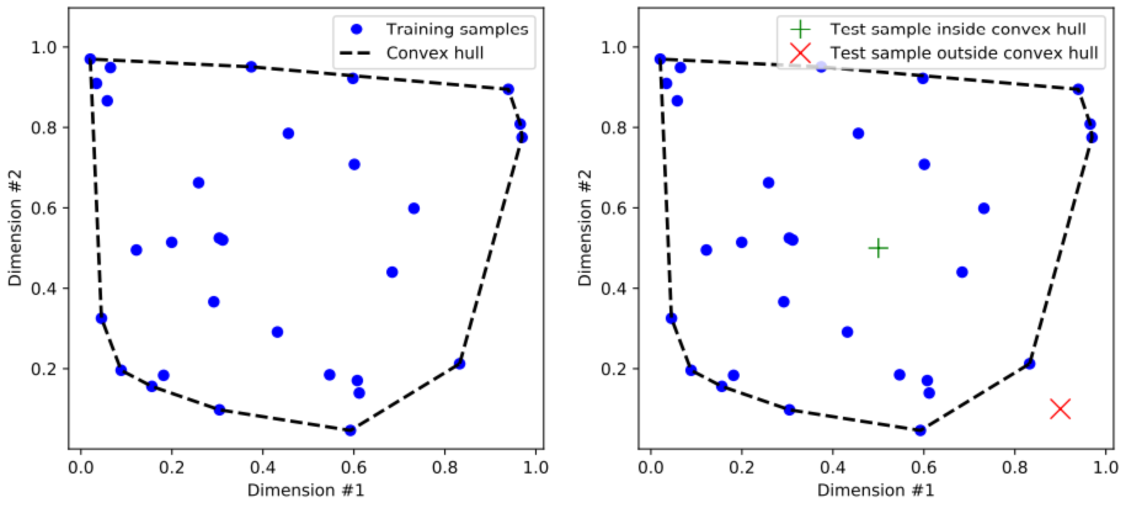


Figure 84. Illustrating the convex hull aspect of intra/extrapolation of training and testing data samples in a two-dimensional space (Barbiero et al., 2020).

5.4.3.2 Function of Model Characteristics

For an ML system to be used effectively in real-world situations, such as autonomous cars (Mohseni et al., 2019) and safety applications (Xu and Saleh, 2021), satisfying the auxiliary criteria related to each application domain is critical. In addition to the optimisation and evaluation of the measures of performance, such as accuracy and precision, the ability of a model to generalise needs to be quantified even when not all possible cases can be listed and tested. For example, we might not be able to enumerate all unit tests required for the safe operation of a semi-autonomous car or all confounds that might cause a credit scoring system to be discriminatory (Doshi-Velez and Kim, 2018). Hence, the generalisation ability guarantees the model's applicability to new unseen data, especially in real-world situations.

As discussed in section 5.4.1, the model's complexity and adequate capacity are two main factors defining its ability to generalise the learned evidence to unseen data samples. The bias-variance trade-off, the model's number of trainable parameters, and data effectiveness are other criteria that considerably impact the robustness of an ML model. Given a trained model $\hat{f}$ (also called a *hypothesis*), to know if the latter can scale up to new data samples, one first learning objective is to minimize the Generalisation Error (true error $\mathcal{L}_{D_{test}}(\hat{f})$) as defined in Section 5.3.2. Other measures like norm-based control and sharpness could also assess the model's generalisation ability.

5.4.4 Generalisation in DL

Deep neural networks (DNNs) trained in practical data sets exhibit good generalisation behaviour, even when the parameters are significantly larger than the training data (Neyshabur et al., 2014; C. Zhang et al., 2021). In those settings, the objective function has multiple global minima[90], all minimising the training error, but not all generalise well (Neyshabur et al., 2017). Picking the wrong global minima can lead to lousy generalisation behaviour, which makes the training insufficient for learning. Different methods are used to minimise the training error for better optimisation, such as the initialisation, update rules, learning rate, and training stopping conditions, all leading to different global minima with varying generalisation abilities. For example, Path-SGD (Neyshabur et al., 2015a) is an optimisation algorithm that is invariant to the rescaling of weights and showed better generalisation behaviour over the classical SGD (stochastic gradient descent) algorithm (Bottou and others, 1991) in the training of different evaluated DNNs. In addition, smaller batch[91] sizes for training with the SGD algorithm generalise better than larger ones (Keskar et al., 2016).

In (Neyshabur et al., 2017), the statistical capacity of a model class is considered in terms of the number of examples required to ensure generalisation, i.e. that the test error is close to the training error, even when minimising the training error. This corresponds to the maximum number of examples with which one can obtain minor training errors even with random labels. Hence, the capacity of a model can be measured using different methods; each candidate gives information about the DL model's ability to generalise.

5.4.4.1 **Network Size**

The evaluation of (Uzair and Jamil, 2020) in a fully connected network has shown that training error decreases as the number of hidden layers increases (increasing the number of parameters). In another study (Brutzkus and Globerson, 2019), a similar phenomenon was observed when learning the MNIST[92] data using an increasing number of channels, as shown in Figure 85.

[90] In gradient descent algorithms, the weights of the NN are initialized, then the gradient of the error is minimized during training and updating the weights. Hence, the error converges until a given minimum value. The latter is called "Local minimum" since the value of the loss function is minimum at that point in a local region. Whereas, a global minimum is called so since the value of the loss function is minimum there, globally across the entire domain of the loss function.
[91] Is a small subset of the training dataset (Le et al., 2011).(Le et al., 2011). Neural networks are trained using different batch size to enable error optimization through smaller amounts of data, rather than the whole training dataset at once.
[92] http://yann.lecun.com/exdb/mnist/

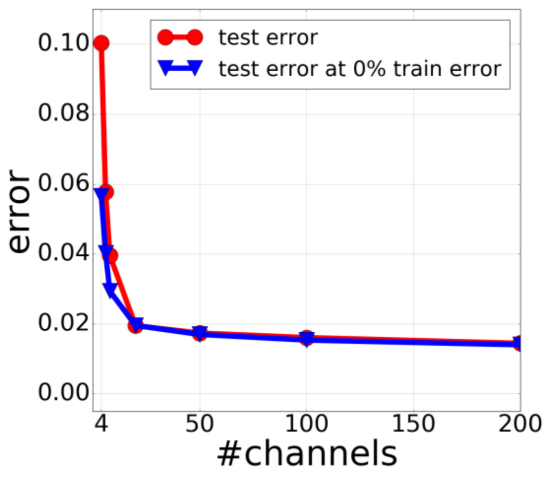


Figure 85. Illustration of the test error evolution depending on the number of channels used in a NN (Brutzkus and Globerson, 2019)

Over-parametrized settings are used in practice, and the number of parameters is more important than the number of training samples. Hence, complexity measures that depend on the total number of parameters are insufficient since NNs having significantly more parameters than samples can perfectly fit even random labels without generalising (Kawaguchi et al., 2017).(Kawaguchi et al., 2017). Moreover, measuring complexity in terms of the number of parameters cannot explain the reduction in generalisation error as the number of hidden units increases (Neyshabur et al., 2015b). Recently, another study (Liu, 2021) has shown that sparse NNs can even generalise better than their dense counterparts and proposed different efficient approaches to yield sparse neural networks with reasonable generalisation bounds.

5.4.4.2 **Norms and Margins of the Network**

For linear models, norms and margin-based measures are commonly used for capacity control (Bartlett and Mendelson, 2002; Evgeniou et al., 2000). Several norm-based complexity measures have been established for feedforward neural networks with the ReLU[93] activation function. Hence, the capacity can be bounded based on the l_1 norm of the weights W_i of hidden units of every layer i in the NN model. This is measured by $\prod_{i=1}^{d} \|W_i\|_{1,\infty}^2$, where $\|W_i\|_{1,\infty}^2$ is the squared value of the maximum, over hidden units in layer i of the l_1 norm of incoming weights to the hidden unit, as defined by (Bartlett and Mendelson, 2002). Based on several norm functions, different capacity measures have been defined to assess the generalisation ability of a DL model:

1) l_2 norm-based capacity

$$\frac{1}{\gamma_{margin}^2} \prod_{i=1}^{d} 4\|W_i\|_F^2$$

[93] The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. This function is used to fix the vanishing gradients problem while training deep NNs.

Where $\gamma_{margin} > 0$ is called the *hard margin* and represents the lowest difference value between the expected output and the computed one, over all training samples.

2) l_2-path norm capacity (Bartlett and Mendelson, 2002; Neyshabur et al., 2015c)

$$\frac{1}{\gamma_{margin}^2}\left|\sum_{j\in \prod_{k=0}^{d}[D_k]}\left|\prod_{i=1}^{d} 2W_i[j_i, j_{i-1}]\right|\right|^2$$

Where $\prod_{k=0}^{d}[D_k]$ is the Cartesian product over all the data sets $[D_k]$.

3) Spectral l_2 norm margin-based capacity (Bartlett et al., 2017)

$$\prod_{i=1}^{d}\|W_i\|_2^2\left(\sum_{j=1}^{d}\left(\frac{\|W_j\|_1}{\|W_j\|_2}\right)^{\frac{2}{3}}\right)^3$$

More capacity-based generalisation bounds can be found in (Neyshabur et al., 2017), where an empirical investigation of the appropriateness of different measures is performed using models trained on true versus random labels. Two main phenomena have been observed: first, the complexity of the models trained on true labels should be substantially lower than those trained on random labels, corresponding to their better generalisation ability. Second, when training on random labels, the capacity is expected to increase almost linearly with the number of training examples since every extra example requires the new capacity to fit its random label. However, when training on true labels one can expect the model to capture the true functional dependence between input and output, and thus, fitting more training examples should only require small increases in the network's capacity. The reported results by (Neyshabur et al., 2017) have shown a gap between the complexity of models learned on natural and random labels for all compared norms, with the difference in the increase in capacity between accurate and random labels being most pronounced for the l_2-norm and l_2-path norm.

5.4.4.3 **Uniform Stability**

Stepping away from complexity measures corresponding to different models of the hypothesis class, another way is to evaluate the generalisation by considering the properties of the training algorithm (C. Zhang et al., 2021). Uniform stability[94] of an algorithm measures how sensitive the algorithm is to replacing a single example. However, it is solely a property of the algorithm, which does not consider details of the data or the distribution of the labels. The weakest stability measure is directly equivalent to bounding generalisation error and does not consider the data.

[94] Stability applies to different notions, both at learning algorithm level the selected model (e.g. local stability when input slightly change). Several definitions are provided in the state of the art. In this document, we refer to the "performance stability" of a model in front of changing data and environment.

Other analyses (C. Zhang et al., 2021) have addressed the generalisation aspect of DL models to make a deep understanding of it, where the main question is *"What practices promote generalisation? And what does it measure?"*. Conducted experiments in (C. Zhang et al., 2021), on both the CIFAR10[95] and ImageNet[96] data sets for image labelling showed how DNNs fit random labels through several randomisation tests. The primary outcome is that the adequate capacity of neural networks is sufficient for memorising the entire data set, and even optimisation on random labels remains easy; the training time increases only by a small constant factor compared with training on the true labels. Furthermore, randomising labels is solely a data transformation, leaving all other properties of the learning problem unchanged.

5.4.5 Domain generalisation

The central assumption in several supervised learning methods is that training and testing data are sampled from the same distribution. However, in real-world applications, this assumption is often violated as conditions for data acquisition may change. In DL, models trained to minimise empirical risk on a single domain usually fail to generalise when applied to other unseen domains due to domain shift (Bayasi et al., 2022).

In the previous sections, we described some widely used methods that can help identify *a priori* the ability of the model to generalise to unseen data samples. Unlike model-driven methods, which focus on the model characteristics, data-driven methods focus on the target data that will be fed into the model. However, none of those methods consider the target application features and the domain characteristics. Regarding *domain generalisation*, the objective is to leverage a model performance in another domain than the training one. To do so, one can train a model on multi-domain source data such that it can directly generalise to target domains with unknown statistics (Dou et al., 2019), or construct a domain-specific model (Han et al., 2020), omitting the domain statistics and its complex characteristics. Hence, extra training and additional data samples can be used within the same application domain for training and target applications (Chung et al., 2018). A more detailed taxonomy of different domain generalisation methods is highlighted in Figure 86.

95 http://www.cs.toronto.edu/~kriz/cifar.html

96 https://www.image-net.org/

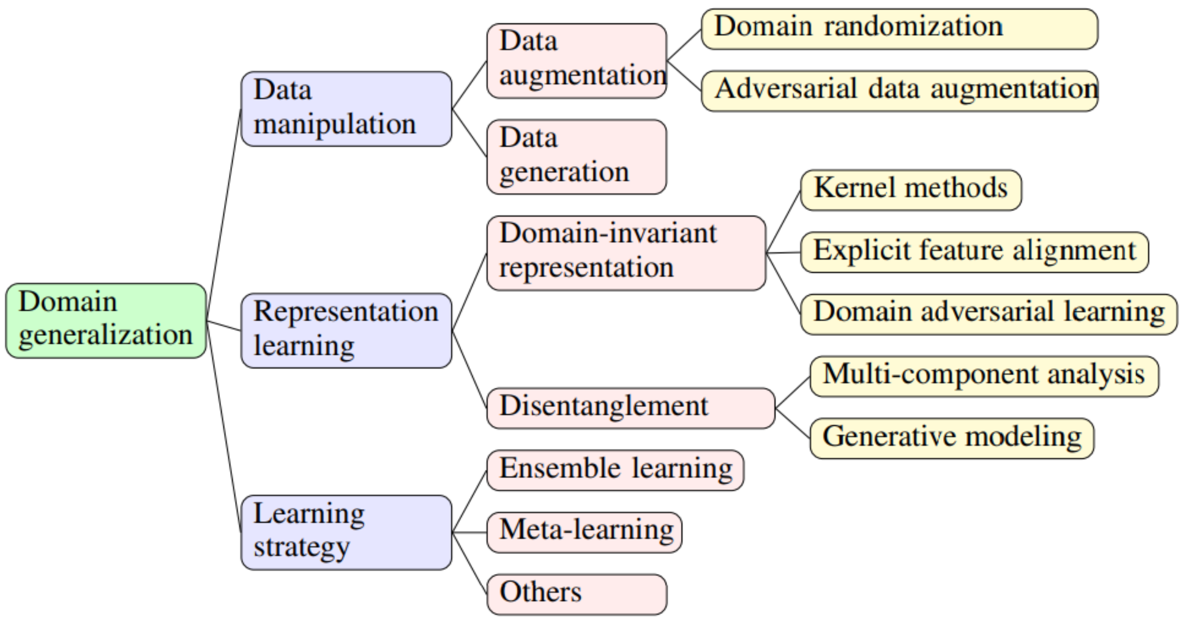


Figure 86. Taxonomy of domain generalisation state of the art (J. Wang et al., 2022).

As shown in Figure 86, several solutions can be adopted to achieve a cross-domain generalisation objective. In this taxonomy, the methods are grouped based on the development phase that is affected by the domain adaptation process. It can be focused on the initial data manipulation through the generation of new instances (Gordon et al., 2020), data augmentation (Volpi and Murino, 2019), or the learning strategies of the representations of the input instances (Ilse et al., 2020), or even the model training process (Zhou et al., 2021b). A complete description of the most recent methods for domain generalisation can be found in (Zhou et al., 2021a).
This document studies the domain generalisation aspect from a different angle. We believe the most important thing to focus on is the target problem to be solved. Hence, another perspective to consider the existing methods focuses on how the ongoing task can be performed while joining the training and the target domain. To this end, the definition of the target domain can be adapted to one of the training domains, and the training process is thus adjusted.

5.4.5.1 Domain adaptation

In this approach, algorithms usually learn to align source and target data in a domain-invariant discriminative feature space. Hence, existing methods have investigated several directions, such as data augmentation by transformation (Y. Shi et al., 2020) and the feature alignment between data from different domains (Z. Wang et al., 2021), where the main idea is to minimise the difference among source domains for learning domain-invariant representations, under the assumption that features that are invariant to the source domain shift should also be robust to any unseen target domain shift. Hence, domain adaptation involves measuring domain distances and learning a representation that reduces them (Motiian et al., 2017). Several statistical distance metrics can be used, such as the simple l_2 distance and *f-divergences.* The most important thing is to know what we need to align between the different domains, and how to align it (Zhou et al., 2021a). To do so, we report the same definition used in (Zhou et al., 2021a) of a domain, as a distribution $P(X, Y)$, with X

being the input (feature) space and Y is the output (label) space. Both the input and output spaces are described with the corresponding distributions $P(X)$ and $P(Y)$, respectively.

$$P(X, Y) = P(X \mid Y)P(Y) = P(Y \mid X)P(X)$$

The domain-alignment occurs when there is a distribution shift in $P(X)$, corresponding to the input space, while the $P(Y \mid X)$ remains the same (same expected outputs by knowing the inputs). Hence, the source domain will undergo some transformations for alignment (Ghifary et al., 2016). When X is the cause of Y, the distribution $P(Y \mid X)$ is also affected, and the class-conditional distribution is aligned instead, assuming $P(Y)$ is unchanged (S. Hu et al., 2020). To perform domain alignment, one can minimise the moments, such as the mean and variance domain, as well as training and testing, by using different mapping functions (Ghifary et al., 2016; X. Jin et al., 2020). Another way is to reduce the distribution mismatch between training and target domains by minimising a dedicated contrastive loss (Yoon et al., 2019). Other distances, such as KL divergence (Z. Wang et al., 2021) and maximum mean discrepancy (MMD) (Gretton et al., 2012) are also investigated in domain alignment studies.
Furthermore, to the domain generalisation objectives, we need to distinguish between *multi-source* and *single-source* domains (Zhou et al., 2021a). The first assumes that multiple distinct but relevant domains are available, and the motivation is to learn representations invariant to different marginal distributions (Blanchard et al., 2011). This process allows a model to discover stable patterns across source domains, generalising better to unseen domains. The *single-source* setting assumes that training data is homogeneous (Hendrycks and Dietterich, 2019), and does not require domain labels for learning thus, they apply to multisource scenarios as well. Independently of the domain specificity, the general Out-of-Distribution (OOD) generalisation problem addresses every challenging setting where the testing distribution is unknown and different from the training. Hence, the OOD entities must be handled carefully to ensure the model's robustness (Shen et al., 2021).

5.4.5.2 **Learning adaptation**

Another way to think about the domain generalisation problem is to focus on exploiting the general learning strategy of ML/DL, to promote domain generalisation capabilities. To do so, the main paradigms[97] so identified are (Shen et al., 2021; J. Wang et al., 2022)

- ***Ensemble learning.*** (Zhou, 2012) model-ensemble learning methods learn sets of multiple specific models for different source domains. Typically, this method learns various instances of the same model with different initialisation weights or using different splits of training data and then uses them together for prediction (Moussa and Owais, 2021; Szegedy et al., 2015). Ensemble-learning-based methods can be grouped into four main approaches:
    - *Exemplar-SVMs* use a collection of Support Vector Machine (SVM) classifiers, each learned using one positive instance and all negative instances (Malisiewicz et al., 2011). Extended to domain generalisation, exemplar-SVMs select the top-K exemplar classifiers that give

[97] Note that not all of these algorithms apply to different applications. In the scope of this report, we are more interested in "offline" supervised learning models. Such that the model is first trained, evaluated, then used for making predictions, which is not the same for "online" learning methods (e.g. *lifelong learning*).

the highest prediction scores (hence more confident), given a test sample for ensemble prediction (Xu et al., 2014).

  - *Domain-specific Neural Networks* aim to learn a set of neural networks, each specialising in a given source domain (Ding and Fu, 2017) or sharing some shallow layers between source domains to capture generic features (Yosinski et al., 2014; Zhou et al., 2021b). Then, in the prediction phase, one can either use the ensemble prediction averaged over all individuals with equal weight (D'Innocente and Caputo, 2018) or adopt a source domain classifier to compute the weights (which determines the most confident candidate) (S. Wang et al., 2020).
  - *Domain-Specific Batch Normalisation. Domain-specific neural Networks* aim to learn a set of neural networks, each specialising in a given source domain (Ding and Fu, 2017) or share between source domains some shallow layers to capture generic features (Yosinski et al., 2014; Zhou et al., 2021b). Then, in the prediction phase, one can either use the ensemble prediction averaged over all individuals with equal weight (D'Innocente and Caputo, 2018) or adopt a source domain classifier to compute the weights (which determines the most confident candidate) (S. Wang et al., 2020).
  - *Domain-Specific Batch Normalisation.* In batch normalisation (Ioffe and Szegedy, 2015), the statistics are computed on the fly during training, and their moving averages are stored in buffers for inference. For domain generalisation purposes, batch normalisation has served to mix statistics of multiple source domains in order to learn generalisable representations (Seo et al., 2020).
  - *Weight Averaging.* (Izmailov et al., 2018) This method aggregates model weights at different time steps during training to form a single model at test time. It improves model robustness under domain shift (Cha et al., 2021). In a runway detection application (Balduzzi et al., 2021), a pointwise average of the density function of the ensemble members is used to combine the predictions corresponding to four classical convolutional neural networks trained similarly.

- ***Meta-learning.*** (Gordon et al., 2020) Also known as learning-to-learn, it consists of learning a general model from multiple tasks by induction. Meta-learning aims to learn from episodes sampled from related tasks to benefit future learning. In domain generalisation, a general strategy is to divide the multi-source domains into a meta-train set and a meta-test set (Rajendran et al., 2020). The motivation behind applying meta-learning to domain generalisation is to expose a model to domain shift during training, hoping that the model can better deal with domain shift in unseen domains. The existing methods (Balaji et al., 2018; Q. Liu et al., 2020) can only be applied to multi-source cases where domain labels are provided. Besides, two main components must be defined: the *episodes* that will use available samples and the *meta-representation* that answers the question of *what to meta-learn*. Hence, the learning objective is to update a model using the meta-source domain(s) so that the test error on the meta-target domain can be reduced, which is often achieved by bi-level optimisation. For example, the model-agnostic meta-learning (MAML) method (Finn et al., 2017) divides training data into meta-train and meta-test sets and trains a model using the meta-train set in such a way to improve the performance on the meta-test set. Rather than producing models that by design generalise well to novel testing domains, the model agnostic training procedure (D. Li et al., 2018) simulates train/test domain shift during training by synthesising virtual testing domains within each mini-batch. Hence, the objective function of the training requires the training steps to improve the testing performances as well. BoostNet (Bayasi et al., 2022) is another recent work on domain generalisation via meta-learning. Designed for image classification of digits and skin lesions, it does not require any changes in the model's architecture or training procedure. It aims at using a measure of feature culpability, which concerns training a model episodically on the most and least culpable data features extracted from critical units in the core network, based on their contribution towards class-specific prediction errors. At inference time, corresponding test image features are extracted from the closest class-specific units, determined by smart gating via a Siamese NN (Bromley et al., 1993), and fed to BoosterNet for improved generalisation.

- ***Self-supervised learning.*** (Jing and Tian, 2020) is often referred to as learning with free labels generated from the data itself. This can be achieved by teaching a model to predict the transformations applied to the inputs. In domain generalisation, self-supervised learning can be applied to single and multi-source scenarios without requiring domain labels. In single pretexts (i.e., single source scenarios), in addition to optimising a classical learning error, models are trained to reconstruct (some) input features. For example, image reconstruction has been investigated (Maniyar et al., 2020) to evaluate the learning ability of an auto-encoder to reconstruct image pixels and features. In multiple pretexts, the models are trained to learn how to solve several (at least two) problems in parallel. For example, in (Bucci et al., 2021), the model is trained to solve Jigsaw puzzles and to predict image rotations at the same time. Overall, using multiple pretext tasks gives a better performance than using a single pretext task (Bucci et al., 2021). In (Zhou et al., 2021a), several self-supervised models for domain generalisation have been reviewed. One of the issues related to self-supervised learning methods is that, in general, none of the existing multiple pretext tasks is universal and that the selection of a pretext task is problem-specific. For instance, when the target domain shift is related to rotations, the model learned with the rotation prediction task will capture more rotation-sensitive information, which is harmful to generalisation.

- ***Transfer learning***[98]***.*** (Zhuang et al., 2020) (TL) aims to transfer the knowledge learned from one (or multiple) problem/domain/task to a different (but related) one. Once a model is trained in a source task, the TL strategy aims to enhance the performance of that model on a target domain/task. To do so, the *pretraining-finetuning* (also used in self-supervised approaches) is the commonly used method, mainly in DL (Tan et al., 2018), where the source and target domains have different tasks: first pre-train deep neural networks on large-scale data sets, such as ImageNet (Deng et al., 2009) for vision models or BooksCorpus (Zhu et al., 2015) for language models; then fine-tune them on downstream tasks (Too et al., 2019; Vrbančič and Podgorelec, 2020). The resulting model is hence able to leverage the knowledge acquired during the pre-training phase to perform the tasks of the fine-tuning phase and perform better on the target

[98] With respect to (EASA and Daedalean, 2024), the certification of TL-based methods is not explicitly considered. However, the traceability and trustworthiness issues related to transfer-learning approaches are discussed in P65 of (EASA and Daedalean, 2024). In the scope of MLEAP, this risk can be mitigated through several experimental evaluations and testing scenarios. Methods that can be used for evaluating non-transfer-learning approaches can be applied. More details in section 5.9.

application. This is due to the extraction of highly transferable features from the first step (pre-training) being transferred to the second one (fine-tuning) (Yosinski et al., 2014). Given these advantages of the TL approach, recent works in domain generalisation (Blanchard et al., 2021; W. Chen et al., 2021) have investigated how to preserve the transferable features learned via large-scale pre-training when learning new knowledge from source synthetic data for synthetic-to-real applications (Inoue et al., 2018). Domain generalisation methods based on TL do not require having the target data for the model fine-tuning phase in the downstream tasks. We assume we have no access to the target data, thus focusing more on model generalisation. In (Zhou et al., 2021a), a theoretical comparison between the domain generalisation objectives and TL intentions is provided. One of the standard features of TL and domain generalisation is that the target distribution in both domains is different from the source distribution; in terms of label space, TL mainly concerns disjoint label space, whereas domain generalisation considers both cases, i.e., same label space for homogeneous domains and disjoint label space for heterogeneous ones. Moreover, in recent years, we have witnessed the rapid development of large-scale pre-training/fine-tuning procedures, such as BERT (Devlin et al., 2018) and GPT-3 (Brown et al., 2020). Pre-training on large-scale data sets and then fine-tuning the model not only improves its performance on downstream tasks but also enables competitive performance on domain adaptation tasks (Laskar et al., 2022; Xu et al., 2021).

- ***Few/Zero-shot learning.*** (Wei Wang et al., 2019; Wang and Yao, 2019) (F/ZSL) is another way to leverage the learned signals and patterns from a set of training data in a given context to solve a different task in another context where few/no data is available. Using prior knowledge, FSL (Wang and Yao, 2019) can rapidly generalise to new tasks containing only a few samples with supervised information. However, the main limitation of this approach is that the empirical risk minimisation is unreliable (Wang and Yao, 2019) due to the low number of training examples for risk minimisation. Other related methods, such as weakly supervised learning (Zhou, 2018) and imbalanced learning (He and Garcia, 2009), have also been used for the same purpose, where incomplete, inexact, inaccurate, or noisy supervised information is used in the former, and rare labels are used in the latter. In (Wei Wang et al., 2019)later. While in ZSL (Wei Wang et al., 2019), the main objective is to identify objects for which labels are unavailable during training. This learning paradigm results in classifiers having the ability to distinguish unseen classes, which is very helpful in practice when acquiring all possible labels is expensive and time-consuming or just impossible (e.g. object recognition in computer vision (Bansal et al., 2018), scene interpretation in video security, or danger detection (Kim et al., 2021)). As there are no available labelled instances belonging to the unseen classes, some auxiliary information from the feature space is necessary to solve the ZSL problem. For instance, an association between a semantic space describing the unlabelled instance and the missing label is performed (X. Li et al., 2020). This space should contain information about all the unseen classes to guarantee they are all provided with corresponding auxiliary information to help the model detect them better. Other methods related to ZSL include cumulative learning (Fei et al., 2016) and class-incremental learning (Rebuffi et al., 2017), in which labelled instances belonging to some previously unseen classes progressively appear after model learning. The learned classifier can be adapted with these newly available labelled instances to classify the classes they cover. This approach is, however, less practical since it still requires complete knowledge of the target domain’s possible courses. This could be impossible in practice. Hence, the open-world recognition methods

(Bendale and Boult, 2015) follow the process of *"unseen classes detection, labelled instances of unseen classes acquisition, and model adaptation"* and adapt the classifier to be able to classify previously unseen classes with the acquired labelled instances belonging to them.

- ***Lifelong learning.*** (Parisi et al., 2019) (LL-learning) refers to the ability to continually acquire, fine-tune, and transfer knowledge and skills throughout the lifespan of a model. It aims to maintain the model's robustness over time in a context where data is evolutionary, and sometimes, the same input instances at time *t* may be considered obsolete at time *t+1*. In such a case, an open problem is the development of incremental learning systems capable of assimilating more and more concepts over time from a data stream. When the acquisition of incrementally available information from non-stationary data distributions is continuous, catastrophic forgetting[99] or interference of information is likely to happen (Parisi et al., 2019). This limitation represents a significant drawback for state-of-the-art DL models that typically learn representations from stationary batches of training data (Zhong et al., 2016), thus without accounting for situations in which information becomes incrementally available over time, e.g. Traffic management applications (Nallaperuma et al., 2019). LL-learning models learn continuously while retaining previously learned experiences, which is different from domain generalisation (that aims to exploit the skills learned from training in one domain to perform tasks in other domains), since it can access the target domain at every time step and does not explicitly handle different distributions across domains (Q. Liu et al., 2020). To bridge the gap between both paradigms, incremental methods (D. Li et al., 2020; Parisi et al., 2019; Rebuffi et al., 2017) leverage the learning continuity and domain diversity aspects. For instance, iCaRL (Rebuffi et al., 2017) is an incremental representation learning algorithm that allows learning in such a class-incremental way. In this setting, only the training data for a small number of classes has to be present at the same time, and new classes can be added progressively. This model learns simultaneously through a set of classifiers and a data representation. In (Rostami, 2021), a continual learning algorithm is proposed to update a model continuously to tackle the challenges of data distribution shifts. The goal is to update a model continuously to learn distributional shifts across sequentially arriving tasks with unlabelled data while retaining the knowledge about past learned tasks. Another approach is sequential learning of several domains (D. Li et al., 2020), inspired by LL-learning, where accumulated experience means that learning the n^{th} thing becomes more accessible than the *first* thing. Applied to domain generalisation, this means that the performance at domain *n* depends on the previous *n−1* learned problems. Thus, backpropagating through the sequence means optimising performance not just for the next domain but for all the subsequent domains.

Furthermore, in the general objective of dealing with OOD cases, causal learning (Yao et al., 2021) and invariant learning (Ilse et al., 2020; Z. Wang et al., 2020) methods stem from causal inference literature and address the OOD generalisation problem differently, aiming to explore causal variables for prediction and became more practical recently (Schölkopf, 2022). Other methods based on stable

[99] Catastrophic *interference*, also known as catastrophic *forgetting* (McCloskey and Cohen, 1989, 1989), is the tendency of an artificial neural network to completely and abruptly forget previously learned information upon learning new information.

learning have been recently used (Cui and Athey, 2022). Compared with domain generalisation and causal learning, stable learning motivates another way of incorporating causal inference with machine learning, which significantly relaxes the requirements for multiple environments. Given a training data set from one environment, the goal of stable learning is to learn a predictive model with uniformly good performance in any possible environment for the target application (X. Zhang et al., 2021). The reader could find more discussions and theorems on these new paradigms for OOD generalisation (Shen et al., 2021).

5.5 Methods to boost generalisation

Several solutions for a better generalisation have been proposed in the literature. Several studies have empirically investigated the impact of each identified generalisation strategy (Ying, 2019), showing how each could help the targeted application. In this section, based on the taxonomy by (Kukačka et al., 2017), we summarise the most used solutions, showing results from some comparative studies, and then discuss their strengths and weaknesses. Several studies have empirically investigated the impact of every generalisation strategy (Ying, 2019), showing how each could help the targeted application. In this section, based on the taxonomy by (Kukačka et al., 2017), we summarise the widely used solutions, showing results from some comparative studies, and then provide a relative discussion about the benefits of each of them.

5.5.1 Regularisation

Regularisation is a method to avoid high variance and overfitting and increase generalisation (Müller, 2012). In deep learning especially, regularisation has a broader definition: *regularisation is a technology aimed at improving the generalisation ability of a model* (Tian and Zhang, 2022). Widely used in deep learning, it allows a better generalisation of unseen data, even when training on a finite and small training data set or with an inappropriate optimisation method. It encompasses any modification made to a learning algorithm intending to reduce its test error but not its training error and, hence, produce better results on test sets (Goodfellow et al., 2016; Kukačka et al., 2017). In the following, the main classes are reviewed.

5.5.1.1 **Data-driven**

A trained model's quality depends on the training data's quality and volume. It is possible to employ regularisation via data by applying some transformation to the training set: (a) some transformations perform feature extraction or pre-processing, modify the feature space or the distribution of the data to some representation, making the learning task simpler (Bishop and others, 1995); (b) other methods allow generating new samples to create a more enormous, possibly infinite, augmented data set (DeVries and Taylor, 2017). Both approaches (a) and (b) are somewhat independent and may be combined. Data-based regularisation relies on transformations with (stochastic) parameters (Kukačka et al., 2017). The latter function can be applied to the network inputs, added to the activations in hidden layers, or applied to targets. The stochasticity of the transformation parameters is responsible for generating new samples, i.e. *data augmentation* (Feng et al., 2021; Shorten and Khoshgoftaar, 2019).

We can categorise the data-based methods according to the properties of the used transformation and the distribution of its parameters:

1) ***Stochasticity of the transformation parameters***. It consists of transformation functions with two types of parameters: (1) deterministic ones, which follow a delta distribution, and the size of the data set remains unchanged (Hoffer et al., 2017); (2) stochastic parameters, where several sampling strategies can be used in the function to allow generation of a more enormous, possibly infinite, data set (DeVries and Taylor, 2017; Loosli et al., 2007).
2) ***Effect on the data representation.*** The data representation can be either preserved or transformed. In the latter case, the objective is to map the data to a different representation, using a different distribution or even a new feature space that may make the learning problem easier (Bengio et al., 2013).
3) ***Data transformation space.*** The transformation functions could be used at different levels of the model pipeline: It can be applied to the input space (enhanced and pre-processed data samples) (Zhu et al., 2021), to the hidden-features space (DeVries and Taylor, 2017), where the transformations are applied to some of the deep-layers corresponding to the input samples (it can use parts of the model weights to map the input into the hidden-feature space). Such transformations act inside the network and thus can be considered part of the architecture or the target output space to help the model's fast learning (in this case, it is used only during the training phase) (Shorten and Khoshgoftaar, 2019).
4) ***Transformation function parameters.*** The parameters of the transformation function can be distributed differently. This can be the same for all samples, specific for each target (class value), dependent on the whole data set particular to every training batch, and so on. For more details about the parameters distribution over the different cases, please refer to (Kukačka et al., 2017).
5) ***Concerned phase by transformation.*** This means that the data transformation function could be applied either to the training set (Morerio et al., 2017) or the test set (Gal and Ghahramani, 2016). For example, multiple augmented sample variants can be classified in the latter, and the result is then aggregated over them.
6) ***Batch Normalisation***[100]***.*** This operator normalises the model responses within each mini-batch. It has been widely adopted in many modern neural network architectures, such as Inception and Residual Networks. Although not explicitly designed for regularisation, batch normalisation is usually found to improve generalisation performance (Santurkar et al., 2018).

Note that the data-driven regularisation methods may affect the ability to meet representativeness and completeness (cf. Chapter 4) objectives from EASA CP (EASA, 2024). Those two objectives should be assessed after data-driven regularisation. Several other data-driven methods for regularisation are cited and classified in (Kukačka et al., 2017); please refer to this reference to learn more about how one can exploit data processing and transformations to let the DL model learn and generalise better.

[100] A complete explanation can be found in: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/

5.5.1.2 **Model-driven**

Another way to construct a suitable DL-based application is to focus on the network architecture and characteristics rather than the training and domain data. Hence, a network architecture f can be built to have specific properties or match certain assumptions to have a regularising effect.
The function $f : (\theta, x) \mapsto y$ defines a data mapping that the architecture of f can do along with the parameters space θ.
Several architecture-based methods can be used to make the model generalise better:

1) ***Assumptions about the mapping.*** This means that the model f_θ needs to implement specific assumptions about the target data set, D_{train} for training and D_{test} for test. These assumptions are meant to make an abstract representation of the reality. Those assumptions may be intractable but can be approximated and then used as guidelines to construct the model structure: choice of the number of units and layers, types of NN layers and architectures (convolutional, recurrent, bidirectional …) (Gulcehre et al., 2016), the processing layers pipeline and the invariances of the mapping, such as locality of some features extraction that can be a layer-specific and defined differently from one layer to another (Zeiler and Fergus, 2013).
2) ***Weight sharing.*** The main objective is to minimise the number of trainable parameters. Reusing a specific trainable parameter in several network parts is called weight sharing (Xie et al., 2021). This usually makes the model less complex than using separately trainable parameters. An example is convolutional networks (Li et al., 2016). Here, weight sharing reduces the number of weights that need to be learned and encodes prior knowledge about the shift-equivariance and locality of feature extraction.
3) ***Activation functions.*** The activation function adjusts the intensity of a signal sent from one node to another between NN layers. Hence, choosing the proper activation function is quite essential. Several activation functions exist, but not all can be applied to all problems (Sharma et al., 2017). There is no rule of thumb for selecting an activation function. In (Onwujekwegn and Yoon, 2020), several functions are analysed, and their impacts on results are highlighted. For instance, in classification problems, sigmoid functions show better loss evolution during training. Due to vanishing gradient problems i.e., gradient tending toward zero, sigmoid and tanh functions are sometimes[101] avoided. ReLU function corrects this behaviour, making it widely used (Agarap, 2018), and performs better than other activation functions in most cases. However, it has to be used only in the hidden layers and not in the outer layer, and if there are dead neurons in the network, then we can use the leaky ReLU function.
4) ***Multi-task learning.*** The objective is to train the same model to learn several tasks simultaneously. The different tasks can help each other to learn mutually useful feature extractors as long as they do not compete for resources (e.g. network capacity) (Ruder, 2017).

[101] Generalized Recurrent Units (GRU) (Irie et al., 2016) use sigmoid for gating and tanh for state due to their output range (0..1 and -1..1 respectively) Generalized Recurrent Units (GRU) (Irie et al., 2016) use sigmoid for gating and tanh for state due to their output range (0..1 and -1..1 respectively)

5.5.1.3 **Based on the training objective**

Another way to generalise a model better is to focus on the training objective. Hence, the error and objective functions can be designed to help the model learn better. The error function $\mathcal{L}_{D_{test/train}}(f)$ of a model f reflects the learning state, during the training and testing phase. Sometimes, it minimises some apparent assumptions about the data distribution. The training objective is to learn the model parameters that minimize the cumulated error values. The common form is:

$$\arg\min_{\theta} \frac{1}{|D_{train}|} \sum\nolimits_{(x_i, y_i) \in D_{train}} \mathcal{L}_{D_{train}}\big(f(x_i, y_i)\big) + \varphi(\ldots)$$

With $\varphi(\ldots)$ is a regularisation term added to control the impact of the accumulated error.
Generalisation strategies based on error functions are proposed to approximate the unit step function for better learning (Guo et al., 2021). Typical examples of error (loss or risk) functions are mean squared error (Allen, 1971) or cross-entropy (Li and Lee, 1993). The error function may also have a regularising effect, thanks to an added term $\varphi(\ldots)$. An example is Dice coefficient optimisation (Milletari et al., 2016) which is robust to class imbalance. Moreover, the overall form of the loss function can be different, in certain loss functions that are robust to class imbalance, the sum is taken over pairwise combinations $D_{train} \times D_{train}$ of training samples (Yan et al., 2003), rather than over training samples.

5.5.1.4 **Based on the optimisation**

While training a DL model, the optimisation algorithm finds the values of the model's parameters θ (weights and bias) that minimise the error when mapping inputs to outputs. The choice of algorithm widely affects the final performance of the deep learning model. It also affects the training speed of the model. Hence, regularisation through optimisation aims to find the optimiser that helps the model converge better and faster to the optimum state.
Stochastic gradient descent (SGD) (Bottou and others, 1991) is one of the most frequently used optimisation algorithms in deep neural networks. Each epoch consists of one forward pass and one backpropagation pass over all of the provided training samples in a whole batch learning process. The actual gradient ∇_θ is obtained by computing the gradient value of each training case independently, then summing together the resulting vectors to further update the model parameters. Hence, SGD is an iterative optimisation algorithm based on the following generic adapted rule:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta R_{D_t}(\theta_t, D_t)$$

Where $\nabla_\theta R_{D_t}(\theta_t, D_t)$ is the gradient of the error amount computed in a mini-batch[102] D_t of the training data set, rather than the whole data set D_{train}, combined with a momentum η_t, to improve the convergence speed (Wilson et al., 2017).
Several optimisation methods are derived from the SGD algorithm to make the model learn more efficiently (fewer data and shorter training time):

[102] Mini-batch learning aims at updating the training weights several times over the course of a single epoch (iteration). In this case, we are no longer computing the true gradient; instead we are computing an approximation of the true gradient, using several training samples in each split of the epoch.

1) ***Initialization, warm-up, and pre-training.*** Methods of this class affect the initial selection of the model's parameters. The most frequently used method is sampling the initial weights from a carefully tuned distribution. The network weights are initialised and then adjusted repeatedly during the training. Several initialisation methods have been developed (Narkhede et al., 2022) with the same aim of keeping the variance of activations in all layers around 1 to prevent vanishing or exploding activations (and gradients) in deep learning models (Tan and Lim, 2019). Several methods can be used:
    1) Random weight initialisation (Sampson, 1987)
    2) Orthogonal weight matrices (Vorontsov et al., 2017)
    3) Data-dependent weight initialisation (Cachi et al., 2020)

    Another (complementary) option is *pre-training* the model with a different objective function and a partially different architecture in a different context (task, data, domain …) where data can be more available, then performing a *fine-tuning* pass in the target context to make the model perform better on the actual objective starts. A critical aspect of this approach is that pre-training a model on a different task of the same domain may lead to learning valuable features, making the primary task easier. Several pre-training methods can be used, such as:
    1) Greedy layer-wise pre-training (Bengio et al., 2006)
    2) Curriculum learning (Bengio et al., 2009)
    3) Spatial contrasting (Hoffer et al., 2016)
    4) Subtask splitting (Gülçehre and Bengio, 2016)

2) ***Update-based.*** It concerns individual weight updates through update rules that modify the form of the update formula, such as:
    1. Momentum, Nesterov's accelerated gradient method, AdaGrad, AdaDelta, RMSProp, Adam (Wilson et al., 2017)
        a. Learning rate schedules (Ge et al., 2018)
        b. Online batch selection [Chaudhari and Soatto 2015]
        c. SGD alternatives (Netrapalli, 2019): L-BFGS, Hessianfree methods, ProxProp.
    2. or by using filters that would affect the value of the gradient or the NN weights, which are used in the update formula, such as injecting noise into the gradient (Wilson et al., 2017):
        a. Annealed Langevin noise (Neelakantan et al., 2015)
        b. AnnealSGD (Chaudhari and Soatto, 2015)

        The Annealed noise on targets can work as noise on gradient but belongs instead to the data-based method.

3) ***Early Stopping.*** This technique stops the training process at an excellent time to avoid the "*learning speed slow-down*" phenomenon. This means that the accuracy of the learning algorithms stops improving after some point, as shown in Figure 65 (right), or even worsens because of the start of noise learning (Raskutti et al., 2014). Also, it has been a widely used regulariser in neural networks since the 1990s .Figure 87 (left) shows the testing error in the blue line and the training error in the green line. If the model continues learning after the red dashed line, the testing error will increase while the training error will continue decreasing.

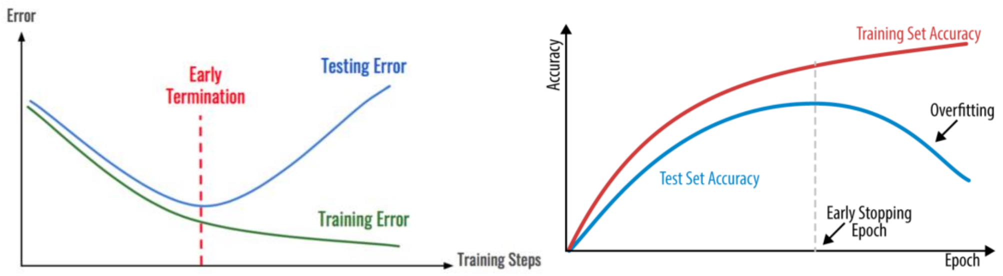


Figure 87. Illustration of the early stopping strategy based on error (left[103]) and accuracy (right[104]) evolution during training in test and training sets.

However, the best training step for stopping is not easy to define. If we stop learning too early, the model could underfit the data; if we stop too late, it may overfit it. Hence, the aim is to find the exact training step to get a perfect fit between underfitting and overfitting. We can track the validation set's accuracy instead of the test set to determine when to stop training. The early stopping method has been used in several effective DL models and has shown its effectiveness in promoting generalisation (Caruana et al., 2000; Wu and Shapiro, 2006).

4) ***Dropout.*** It aims to randomly drop units (along with their connections) from the neural network during training (Srivastava et al., 2014). This prevents units from co-adapting too much. This is one of the most popular methods from the generic group. Still, several variants of Dropout have been proposed that provide additional theoretical motivation and improved empirical results, such as the Random dropout probability (Bouthillier et al., 2015) for training and the Bayesian dropout (Gal and Ghahramani, 2016) at test.

It is not clear which of the methods merely speeds up optimisation and which helps the generalisation. One must perform several optimisation tests and comparative analyses to find the best match between the model complexity and the optimisation algorithm. An empirical study by (C. Zhang et al., 2021) compared different regularisation techniques: data augmentation (aug), weight decay (wd), batch normalisation (BN), and dropout Figure 66 shows the performance evolution of a trained Inception[105] model using different regularisers. Since the Inception architecture uses a lot of batch normalisation layers (Szegedy et al., 2015), a new "*Inception w/o BN*" architecture is used. It is the same as the classical Inception, except with all the batch normalisation layers removed. Figure 88a shows the accuracy of the training and testing of ImageNet. Figure 88b compares the learning curves of the two variants of Inception on CIFAR10, with all the explicit regularisers turned off. Each curve corresponds to one of the regularisation methods, where the shaded areas are the cumulative best

[103]https://medium.com/analytics-vidhya/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting-dce6de4081c5
[104]https://datascience.stackexchange.com/questions/32306/in-which-epoch-should-i-stop-the-training-to-avoid-overfitting
[105] Inception (Szegedy et al., 2015) is a deep convolutional neural network architecture that achieved a new state of the art for classification and detection, in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC2014). The main characteristic of this architecture is the improved use of the computing resources inside the network. By a crafted design that increased the depth and width of the network while keeping the computational cost constant.

test accuracy as an indicator of potential performance gain of early stopping. However, on the CIFAR10 data set, no potential benefit of early stopping can be observed.

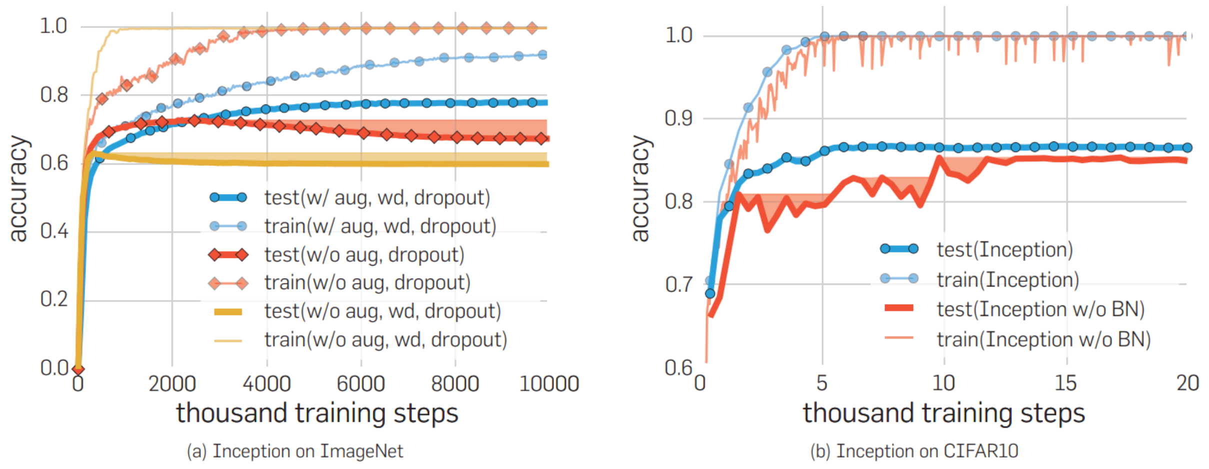


Figure 88. Effects of different regularisers on generalisation performance in terms of accuracy (C. Zhang et al., 2021).

This evaluation shows that a good match between different regularisers can lead to better performances. In this experiment, there are two main observations to retain:

(a) Early stopping could improve generalisation when other regularisers are absent.
(b) Early stopping is sometimes not necessarily helpful (e.g., CIFAR10), while batch normalisation consistently stabilises the training process and improves generalisation.

5.5.1.5 Boost the generalisation with regularisation

All the regularisation methods and generalisation analysis seen so far are based on observations of the training error compared to the test errors. A significant gap between both losses suggests low generalisation, as shown in Figure 89.

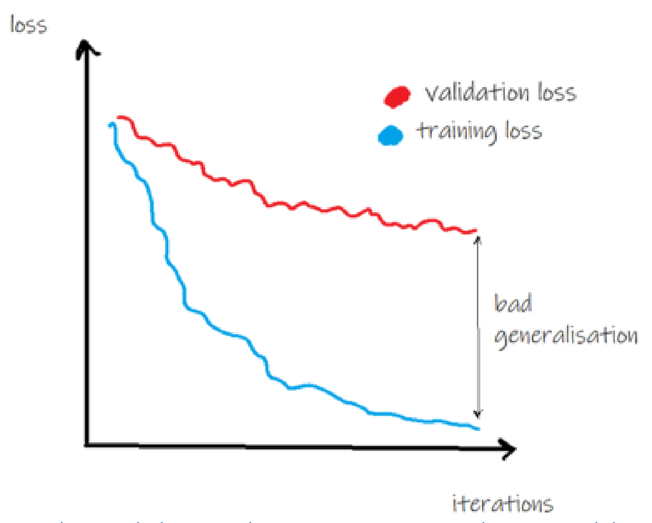


Figure 89. The illustration of a bad generalisation behaviour shows an important gap between validation and training loss errors.

Regularisation can be as simple as shrinking or penalising significant coefficients (cf. Section 5.5.1.3) where an added term $\varphi(\dots)$ is often used to calibrate weights and can be defined differently. Regularisation can also penalise the model's complexity or smoothness (cf. Section 5.5.1), allowing for good generalisation even when training on a finite data set or with an inadequate iteration. Recently, a comparative study (Tian and Zhang, 2022) of different existing regularisation methods for ML/DL applications has discussed choosing a regularisation for a specific task. Such new regularisation techniques can be constructed by extending and combining existing ones.
Even if the methods are different, the objective of regularisation remains the same: improving the generalisation ability of an ML/DL application (Müller, 2012). For instance, in data-driven methods, the optimisation of data representation and the leveraging of latent content on the data, along with the identification of the right location and function for data transformation, are the strengths of regularisation by data processing. The advantage of these methods is that the model will master the data and the target application domain. However, to avoid the opposite effect of these methods on generalisation, we need to pay attention to data biases (e.g., ensure a balanced data distribution over different mini-batches while using batch normalisation), which will harm the data generalisation. Model-driven approaches build the model architecture based on the target application statements and assumptions. Here, independently of the data, the input-output stream is constructed by highlighting transitions and shared links between intermediate results and representations. The most crucial attention point is to correctly formulate the abstract observations to avoid associating wrong or inadequate tasks or features together, in the multi-task performance or during the weight-sharing to ensure a smooth transition of the information contained in the initial data. Finally, the methods based on the optimisation algorithms can also be leveraged to promote the models' generalisation. These methods, such as early-stopping and weight-update functions, rely on the training process rather than the model's architecture or inputs. They enable the preservation of the optimal state of the model that is more likely to generalise.

5.5.2 Penalty Methods

An overfitted model tends to memorise all the data features while training, even if some are noisy. To limit these cases, two possible solutions can be adopted:

1) Select only the valuable features and remove the useless ones from the model (Javed et al., 2020; Khalid et al., 2014; Mas' ud et al., 2014), where methods such as Dimensionality Reduction, as a pre-processing step to a machine learning model and an objective of the first layers of a deep learning model, is effective in removing irrelevant and redundant features, while keeping the vital information unchanged. In this case, a trade-off between dimensionality and informativeness of the features needs to be found to guarantee some degree of efficiency and effectiveness of the model;
2) Minimize the weights of the features which have little influence on the final classification. In other words, we need to limit the effect of those useless features. However, we do not always know which features are worthless. In a context where features are automatically computed (e.g., deep NN where representations are latent), DL models could not effectively filter out the redundant features from the original data. Besides, DL-based approaches usually obey the rule of feature engineering first and algorithm hyper-parameter tuning later to build the machine learning pipeline, which could lead to sub-optimal outcomes (Bai et al., 2022).

To better combine both solutions (1) and (2), optimal features need to acquire more attention from the model, though sometimes noisy content can bring guidelines that could help the model's learning if leveraged correctly (Song et al., 2022). The standard approach is to limit the effect of noise by minimising the cost function of the model. To do this, a "*penalty term*", called regulariser, can be added to the cost function used during training, as shown in the formula of section 5.5.1. Hence, regularisation can be achieved by adding the term $\varphi(\dots)$ into the loss function. Unlike the error function $R_{D_{train}}$ (which expresses the consistency of outputs with targets), the regularisation term is independent of the targets. Instead, it is used to encode other properties of the desired model, to provide inductive bias (i.e., assumptions about the mapping other than the consistency of outputs with targets). The value of φ can thus be computed for an unlabelled test sample. The regularisation term φ is generally used to express a given assumption about the elements of the data set and the target application domain. For example, one of the classical regularisers is weight decay (Goodfellow et al., 2016)

$$\varphi(\theta) = \lambda \frac{1}{2} \|\theta\|_2^2$$

With λ is a weighting term controlling the importance of the regularisation over the consistency. θ represents the weights of the model being trained.
Several other error regularisers can be used; a brief classification of several penalty methods is provided in (Kukačka et al., 2017). The following dependencies can be observed:

1) Dependence on the weights θ
2) Dependence on the network output $y = f_\theta(x)$
3) Dependence on the derivative $\frac{\partial y}{\partial \theta}$ of the output y w.r.t the weights
4) Dependence on the derivative $\frac{\partial y}{\partial x}$ of the output y w.r.t the input x

5.5.3 Network reduction

Network reduction is a strategy to reduce the number of trainable parameters of the DL model without losing performance on the target task and have the same good results. This regularisation approach is inspired by noise reduction to prevent overfitting. Known also as pruning (Ying, 2019), network reduction is proposed to reduce the high inference cost of deep models in low-resource settings. Pruning is a theory that reduces classification complexity by eliminating less meaningful or irrelevant data, preventing overfitting, and improving classification accuracy. During pruning, according to a specific criterion (rules), redundant weights (inter-neuron connections) are removed, and necessary weights are kept to preserve the accuracy of the model best (Liu et al., 2018); a fine-tuning pass is then performed to optimise the resulting reduced network.

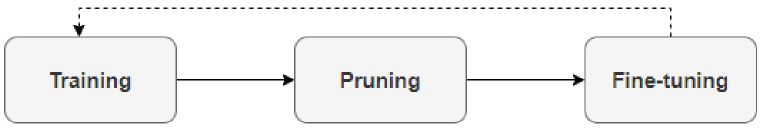


Figure 90. The three stages of the NN pruning pipeline.

Two methods have been used to make the deep learning model simpler:

- pre-pruning that functions during the learning process, and where a stopping criterion is used to determine when to stop adding conditions to a pruning rule or adding a rule to a model description, such as encoding length restriction;
- Post-pruning splits the training set into two subsets: the growing set and the pruning set. It prevents overfitting by deleting conditions and rules from the model generated during learning.

Formally, given a neural network model $f_\theta\,(x)$, pruning involves producing a new model $f_{\theta' \odot M}\,(x)$. Here $\theta' \odot M$ is the elementwise product of the new parameters θ' with the binary mask $M \in \{0,1\}^{|\theta'|}$. Algorithm 1 (Blalock et al., 2020) describes a generic format of the pruning strategies for the NN reduction models in the literature.

Algorithm 1: Generic pruning strategy

Input: *N the number of iterations of pruning, X the data set on which to train and fine-tune*
Output: *the new parameters distribution $\theta \odot M$*

```
1. $\theta \leftarrow initialize()$
2. $\theta \leftarrow trainToConvergence\big(f_\theta\,(X)\big)$
3. $M \leftarrow 1^{|\theta|}$
4. For $i \in \{0, \ldots N\}$ do
    a. $M \leftarrow prune\big(M, score(\theta)\big)$
    b. $\theta' \leftarrow fineTune\left(f_{\theta \odot M}\,(X)\right)$
    c. $\theta \leftarrow \theta'$
5. End for
6. Return $M, \theta$
```

In this algorithm, the network is first trained to converge. Afterwards, a score is computed for each parameter or structural element in the network, and the network is pruned based on these scores. Pruning and fine-tuning are often iterated several times to gradually reduce the network's size without losing performance. This pipeline is shown in Figure 68:

1. **Pruning**: where the objective is to reduce the number of trainable parameters in the network gradually. In this step, the target model is designed to have as little complexity as possible. Several methods can be used:
    1) ***Sparsity structure***. When individual parameters are pruned separately, the algorithm produces a sparse NN (Molchanov et al., 2019). Some other methods (Wang et al., 2017) consider group parameters pruning, by removing entire neurons, filters, or channels for a better-optimized computation.
    2) ***Scoring***. Different score values can be computed to assess the probability that the parameter, the node or the layer will be dropped. Scores like absolute values, trained importance coefficients, or contributions to network activations or gradients can be used. Hence, some methods (Tanaka et al., 2020) prune parameters with the lowest scores within each structural subcomponent of the network (node or layer). At the same time, other methods (You et al., 2019) are based on global scores. They aim to compare the scores of a group of parameters to different parts of the network.
    3) ***Scheduling***. Aims to prune different amounts of parameters at each step. Some methods prune all desired weights at once in a single step (Liu et al., 2018), some other techniques

prune a fixed fraction of the network iteratively over several steps (Hubens et al., 2021), or vary the rate of pruning according to a more complex function (N. Liu et al., 2020).

2. **Fine-tuning**[106]**.** Since the pruning reduces the accuracy of the network, this latter is trained further (fine-tuned) to recover. This step generally aims to adapt an already trained model in a given context, to a target context, where data and tasks can be different (Vrbančič and Podgorelec, 2020)—pruning methods involving fine-tuning commonly restart training the network using the weights trained before pruning. Alternative proposals include rewinding the network to an earlier state (Frankle et al., 2019) or reinitialising the network entirely (Zhang et al., 2022).

3. **Truncation.** (Nevzorov et al., 2022) is another method for network reduction which is similar to the classical dropout (cf. section 5.5.1.1). According to a position-based probability, some neurons are excluded during training. By defining a training direction in the network, neurons positioned at the beginning of each layer tend to be better connected (have higher weights), so they are not excluded by the Truncation (Nevzorov et al., 2022). They make the main contribution to the result. Meanwhile, neurons at the end of a layer have weaker connections to the next layer and can be excluded without making a significant impact. Hence, the dependence of network accuracy on the number of neurons in a layer has been implemented after one cycle of training while performing some regularisation, without losing accuracy.

5.5.4 Data Expansion

5.5.4.1 Data quality and volume qualification

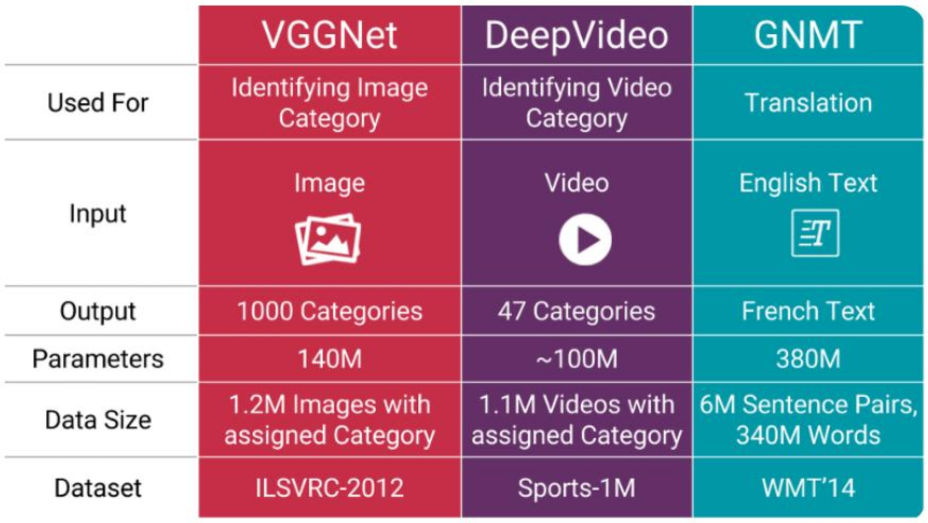
A crucial issue in machine learning projects is determining how much training data is needed to achieve a specific performance goal (w.r.t industrial requirements and theoretical evaluation metrics. Cf. Section 2.1), and how to qualify this data set (identify characteristics that describe the target application). Qualified data acquisition becomes more critical in a supervised setting, where DL models should be trained on a sufficient and qualified data set (i.e., required amount, less noise, balanced labels). To figure out if the data size and characteristics fit the target application and the models' requirements, several studies have provided data scientists with measures that help with assessing the data quality and volume:

5.5.4.2 Importance of data

Data choice and preparation are two critical steps in developing DL applications. However, a significant amount of data from the target domain is needed when the target application requires a complex model. In particular, deep learning applications require considerable amounts of data for training[107]. Figure 91 shows some examples of DL applications w.r.t the amount of training data sizes used to train

[106] http://d2l.ai/chapter_computer-vision/fine-tuning.html
[107] https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/

all the model parameters for applications of different models: VGGNet (Simonyan and Zisserman, 2014) for image recognition, DeepVideo (Karpathy et al., 2014) for videos classification, and GNMT (Wu et al., 2016b) for machine translation.



	VGGNet	DeepVideo	GNMT
Used For	Identifying Image Category	Identifying Video Category	Translation
Input	Image	Video	English Text
Output	1000 Categories	47 Categories	French Text
Parameters	140M	~100M	380M
Data Size	1.2M Images with assigned Category	1.1M Videos with assigned Category	6M Sentence Pairs, 340M Words
Dataset	ILSVRC-2012	Sports-1M	WMT'14

Figure 91. Three example applications show the complexity of the DL model used, the data type, and the sizes of the training sets[108].

Collecting such a volume of data in real-world applications is complicated and sometimes impossible. In some domains, collecting new data is either not feasible or requires comparably many resources, due to the need for experts to validate and label the data samples, or simply because it is costly to perform (Bansal et al., 2021). Sometimes, large data sets can be found for real applications. However, raw primary data often suffers from over-representation of one (or more) class/label over others, as in the case of computer vision, information security, marketing, and medical science (Kaur et al., 2019). This is the data imbalance problem, where some parts of the desired labels for training are not as available as others in the same data set, which can lead to a data bias problem.

For most[109] of the problems above, data augmentation (expansion) is one of the standard solutions to reach the required data amount for DL model training. In addition, the performance of most ML models, and deep learning applications in particular, depends on the quality, quantity and relevance of training data (Redman, 2018). Data augmentation is a set of techniques to artificially increase the amount of data by generating new data points from existing data. This includes making minor changes to the original qualified data, or using deep learning models to create new data points for the training (cf. section 4). Formally, given a data set $D = \{(x_i, y_i), i = 0, \dots, |D|\}$, a transformation function tf produces a new data example associated to every input sample $x_i \in X$: $tf(x_i, \omega) = x_i^t$, hence (x_i^t, y_i) will be the new training sample, where y_i is the label corresponding to the original input x_i, and ω is a set of transformation parameters corresponding to tf.

Several data augmentation methods have been used to cope with the data availability barrier in ML and DL applications. These methods can be either online or offline. In online augmentation, data is

[108] https://www.datarobot.com/blog/introduction-to-dataset-augmentation-and-expansion/

[109] Recently (C. Wang et al., 2021), there has been a vast interest in self-supervised learning where the model is pre-trained on large scale unlabelled data and then fine-tuned on a small labelled dataset. The objective of these approaches is to help developing models for applications where few labelled data is provided.

augmented at training time, so there is no need to store the augmented data (Lemley et al., 2017). In offline augmentation, data is augmented in the pre-processing phase and stored for later use (Lei et al., 2017). Online and offline methods can be either learnable (based on trained models to perform data augmentation) or non-learnable (heuristic methods that make a series of data transformations).

5.5.4.3 Non-learnable methods

This class relies on a set of human-based heuristics. It includes manual data transformation methods where the tf function is designed manually, and ω includes handcrafted rules and features. These methods are based on simple transformation functions (TFs) that are defined and tuned by experts, where TFs make the same changes (e.g., image rotation or flip) to all the original data entities.—the definition of tf and ω depends on the target application. Most of the existing methods can be described by the same heuristic pipeline (Sharon Y. Li, n.d.), as shown in Figure 92. Where a series of transformation functions (TF) are defined and applied to different data entities under the supervision of a human expert, then the qualified augmented data set is used for DL model training.

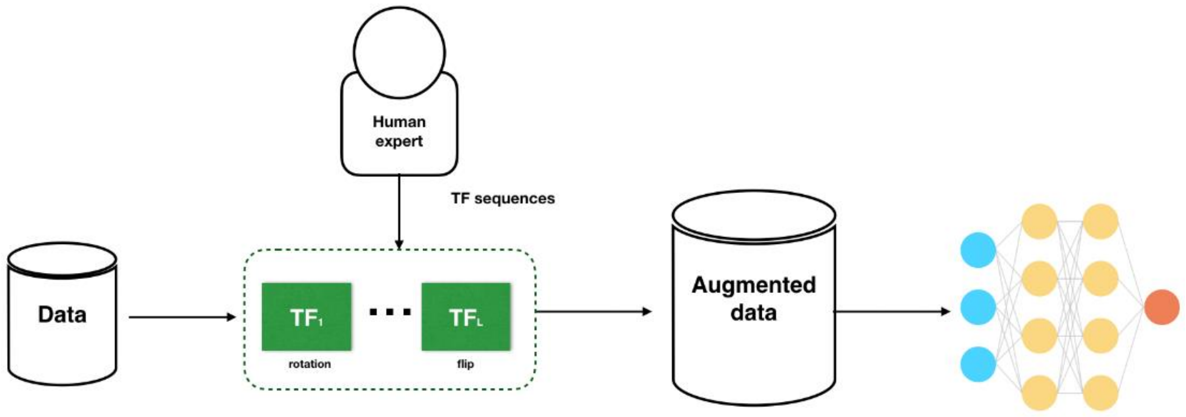


Figure 92. Description of the heuristic data augmentation pipeline,[110] which applies a deterministic sequence of transformation functions tuned by human experts.

In image processing (Shorten and Khoshgoftaar, 2019), one of the most used heuristics is *geometric transformation* (Goodman, 2022), where the primary TFs are *Flipping*, which is a mirror effect, done by reversing the pixels of an image horizontally or vertically; *Rotation*, which is done by simply rotating the image at a certain angle; *Cropping*, which is used to create image data with mixed width and height dimensions; and *Translating*, which aims to shift the original image in a given direction (i.e., right, left, up or down) and is very useful to preserve the label. Another common heuristic is colour augmentation based on the colour space transformation. It is also known as photometric transformation, it can be made by transferring colours between images (Xiao et al., 2019) or applying colour perturbations to a given input image (Khosla and Saini, 2020). Noise-based methods, such as Gaussian noise (Lopes et al., 2019) and random erasing (Zhong et al., 2020), are also used to produce new images by noise injection in different patches of the input image, in the former or by randomly erasing parts/areas of every image, in the latter. Other than heuristics, the style transfer method is an artistic approach that

[110] https://salmenzouari.medium.com/automating-data-augmentation-f2bdf1f1c0da

is widely used to produce new images by transferring the style of one image to another. This method defines three images: a content image *C* (the image for which we want to transfer a style), a style reference image *S* (the image we want to transfer the style from, such as an artwork by a famous painter), and the input (generated) image *G*. It blends them such that the image *G* is transformed to look like the image *C*, in the style of the image *S*. Figure 93 shows an example of a style transfer result:

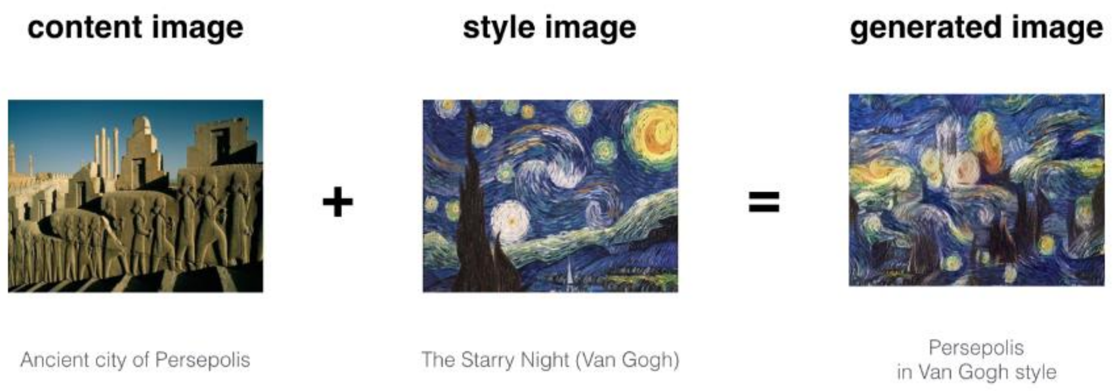


Figure 93. Example of style transfer data augmentation in image processing[111].

Apart from computer vision, several natural language processing (NLP) data augmentation methods have also been proposed to deal with the same data scarcity problem. A review of several non-learnable text data augmentations, known as *"symbolic augmentations"*, can be found in (Shorten et al., 2021). Where rule-based methods like Easy Data Augmentation (EDA) (Wei and Zou, 2019), based on random operations (swap, insertion, deletion, and synonym replacement) as shown in Table 35, the rule-based attack (D. Jin et al., 2020), namely TextFooler which selects the words that most significantly change the outputs for the synonym replacement; or Regular Expression Augmentations (Spasic et al., 2020) that find common forms of language and generate extensions that align with a graph-structured grammar; aim to generate accurate textual data to enhance DL models for NLP applications. Graph-structured augmentation (Ding et al., 2022) aims to construct graph representations of text inputs. This includes relation and entity encodings in knowledge graphs and grammatical structures in syntax trees. These augmentations add explicit structural information that can help find label-preserving transformations and representation analysis and add prior knowledge to the data set or application. A key benefit of symbolic augmentation is the interpretability of the human designer. Symbolic augmentations also work better with short transformations, such as replacing words or phrases to construct augmented examples. This entails if-else programs for augmentation and symbolic templates to insert and re-arrange existing data.

[111] https://towardsdatascience.com/neural-style-transfer-on-real-time-video-with-full-implementable-code-ac2dbc0e9822

Easy Data Augmentation	Short Example
Random Swap	I am jogging → I tiger jogging
Random Insertion	I am jogging → I am salad jogging
Random Deletion	I am jogging → I jogging
Random Synonym Replacement	I am jogging → I am running

Table 35. Examples of Easy Data Augmentation Techniques

The main limitation of non-learnable data augmentation methods is that humans are too involved in the whole pipeline, starting from the TFs design and choice to the transformation and validation of the resulting samples. Therefore, in a context where large data sets are needed (DL training), this approach becomes infeasible due to the associated cost and time. To deal with these limitations, learnable methods can be more efficient for some use cases, where a huge amount of training data is needed and human expertise is unavailable.

5.5.4.4 Learnable methods

Learnable data augmentation is promising, in that it enables to search for more powerful parameterisations and compositions of transformations. Perhaps the most significant difficulty with automating data augmentation is how to search over the space of transformations. This can be expensive due to the search space's many transformation functions and associated parameters. The learnable data augmentation methods are based on advanced models trained to learn how to generate new data examples.
One of the learnable methods for image data augmentation is AutoAugment (Cubuk et al., 2019), which uses learned augmentation policies, where a TFs sequence generator learns to optimise for validation accuracy on the end model directly. Feature Space Augmentation methods (Chu et al., 2020; B. Li et al., 2021) describe augmenting data in the intermediate representation space of Deep Neural Networks. Nearly all Deep Neural Networks follow a sequential processing structure where input data is progressively transformed into distributed representations and, eventually, task-specific predictions. Feature Space Augmentations isolate intermediate features and apply noise to form new data instances. This noise could be sampled from standard uniform or Gaussian distributions or designed with adversarial controllers. Another approach is Generative Data Augmentation, which is one of the most emerging ideas in Deep Learning. This includes generating photorealistic facial images (Karras et al., 2019) or high-level text passages indistinguishable from those written by humans (Brown et al., 2020). Text generation approaches are based on language modelling to perform text data augmentation. This is useful to produce models for different tasks in different languages (especially in the case of languages with few resources). One of the most popular strategies for training a language model for Generative Data Augmentation is Conditional BERT (C-BERT) (X. Wu et al., 2019). C-BERT augments data by replacing masked tokens of the original instance. The key novelty is that it takes a vector representation of the class label as input, such as to preserve the semantic label when replacing masked tokens. This targets the label-preserving property of Data Augmentation. Based on the same idea, the Transformation Adversarial Networks for Data Augmentations (TANDA) (Ratner et al., 2017) models data augmentations as sequences of TFs provided by users. TANDA is a framework that consists of three main components: (1) the learnable TF sequences, (2) the generator using the

sequence to create new images, and (3) the discriminator that should distinguish the augmented images from real ones. Another text data augmentation case study is training sentence classifiers in a few labelled data. In (Anaby-Tavor et al., 2020), a method, referred to as *language-model-based data augmentation* (LAMBADA), involves fine-tuning a state-of-the-art language generator to a specific task through an initial training phase on the existing (usually small) labelled data. New sentences for the class are generated using the fine-tuned model and given a class label. The sentence classifier then performed better after training on the generated data.

5.5.4.1 **Discussion**

As described by the different evaluation metrics, confidence in the performance of an ML model heavily relies on actual observations and the values computed by evaluation measures. As a result, the classical methods relating to the software engineering discipline (e.g. behaviour correctness verification by unit tests) continue to be limited in size and suffer from a lack of extensiveness and coverage (Naser and Alavi, 2021). This is often related to limitations in conducting full-scale tests, the need for specialised equipment, and a wide variety of tested samples. For instance, an empirical study has shown how the atmosphere pressure computed by ML approaches can significantly vary from one study to another (Priyadarshini and Puri, 2021) due to some context differences. Therefore, several metrics, such as MAE, MSE, RMSE and R-squared coefficient (cf. Chapter 2), are required to validate the model's effectiveness.
Furthermore, adopting a set of metrics does not negate specific common issues, namely overfitting and bias. As such, an analysis that uses ML should also consider additional techniques, such as independent test data sets and threshold values (Ji et al., 2019) and varying degrees of cross-validation folds (Refaeilzadeh et al., 2009). Hence, a robust ML model should not only provide reasonable values based on the performance evaluation metrics but should also be capable of capturing the underlying physical and semantic aspects that govern the investigated system (Brennan et al., 2020; Peters and Kriegeskorte, 2021). This aspect is more developed in Chapter 4. An essential approach to verify the robustness of the ML model is to perform parametric and sensitivity analyses (Moussa and Owais, 2021; Razavi et al., 2021; Xu et al., 2012). It is the study of how the outputs of a system are related to and are influenced by its 'inputs. These analyses provide indicators of the model's generalisation ability and ensure that the system's behaviour after deployment will be less affected by other phenomena and processes. Hence, this excludes the fact that the model's performances are simply a combination of the variables with the best fit on the data during training. This is thanks to the inputs/outputs special definition in the *sensitivity analysis modelling* (Razavi et al., 2021): *Inputs of interest*, commonly referred to as '*factors*', may include model parameters, forcing variables, boundary and initial conditions, choices of model structural configurations, assumptions and constraints; *Outputs* may consist of any functions of model responses, including those that may vary over a spatiotemporal domain, objective functions such as a production or cost function in cost-benefit analysis, or an error function in model calibration.

5.6 Limitations of existing methods and discussion

In this chapter, we have reviewed several existing methods for generalisation evaluation, mentioning their strengths and weaknesses. In this part, we analyse the main limitations of the existing techniques and pipelines for ML/DL model analysis and generalisation evaluation benchmarks. This analysis will provide the primary resources to develop a relatively compact methodology for analysing and leveraging the different elements (data preparation, training pipeline, evaluation, and analysis), allowing a rigorous workflow for developing a certifiable, trustworthy AI.

5.6.1 Misunderstanding of the generalisation bounds

To generalise well beyond the training set, ML/DL models undergo analysis based on many hypotheses concerning what the triplet of a *model*, *optimisation algorithm,* and *data properties* could generate. Combining this triplet's elements could help provide a better-generalised AI solution. To do so, hypotheses about the model's complexity, adequate capacity and data properties are made. This yields a set of theoretical and empirical analyses of the generalisation ability. However, despite the prominent role of complexity measures in studying generalisation, the estimation functions and their effectiveness evaluation (cf. Sections 5.4.2 and 5.6.1) are usually limited to a few models, often on toy problems. A measure can only be considered reliable as a predictor of the generalisation gap if tested extensively on many models at a realistic scale (Jiang et al., 2020). The findings of this study are surprising. For instance, it has been verified that one can easily capture false correlations between generalisation and complexity and that do not reflect more causal insights about generalisation using some complexity measures. Hence, an empirical verification of the use case needs to be performed. Another limitation is that many norm-based measures (Bartlett et al., 2017) could negatively correlate with generalisation, precisely when the optimisation procedure injects some stochasticity.

An experimental study (C. Zhang et al., 2021) has shown that conventional generalisation bounds, based on uniform convergence or uniform stability, are inadequate for over-parameterized deep neural networks. Hence, several empirical studies have shown how generalisation could be evaluated in deep neural networks. This results in a set of empirical methods that could be computed to estimate how well the trained model will generalise to unseen data, mainly the target domain. However, despite extensive development spanning many decades (Anthony et al., 1999), there is growing concern that these bounds are not only weak (Dziugaite and Roy, 2017) but that they do not correlate with the underlying phenomena (approximating the correct generalisation performance). As an explicit demonstration of the looseness of the existing bounds, several previous studies (D. Hsu et al., 2021; Jiang et al., 2020) have shown that calculated bounds for a feed-forward NN architecture, achieving a small test error represent an essential gap in different observations.

Despite these limitations, the extensive analysis by (Jiang et al., 2020) have shown that measures related to the optimisation procedures, such as the gradient noise and the speed of the optimisation, can be predictive of generalisation. Besides, sharpness-based measures like PAC-Bayesian bounds (McAllester, 2003) perform the best overall and seem promising candidates for further research.

5.6.2 Practical Issues Preventing Generalisation

In the previous sections, we have reviewed the importance of the generalisability of an ML/DL model in AI applications. We have seen that some problems can prevent an AI application from performing well on unseen data during training. This issue is the main limitation that harms ML/DL models' generalisation and could prevent them from being released. There are two significant sources of performance limitations: (1) overfitting resulting from too much learning of a model or a complex model with insufficient data. The model will then perform well on the training data set and poorly on new data; (2) underfitting results from too little learning or training a model with too small a capacity to learn the task. The model will then fail to understand how to solve the problem, perform poorly on a training data set, and not perform well on test samples. In addition to the overfitting and underfitting issues, several studies have investigated the main problems that can lead ML/DL models to under/overfit the data. In the following, we review the common mistakes and pitfalls that lead to poor performance in ML/DL applications. This list is not exhaustive:

- ***Inappropriate training objective.*** The loss function is essential in the ML/DL model training. It calculates how good the model is at making predictions using a given set of values (input representations, model weights and biases). Hence, choosing a relevant way to drive the model training is essential. Not all existing loss functions defined in the literature (Q. Wang et al., 2022) could fit a given targeted application. Moreover, one can build a custom loss function that closely matches specific solution objectives and performance criteria.
- ***Ignoring outliers***[112]***.*** In statistics, an *outlier* is an observation point distant from other observations. In training data, the presence of several notable cases in the data set should be handled carefully. One outlier could skew the distribution of results if it is ignored during training. In observing the results, the outlier may be due to just variability in the measurement or may indicate experimental errors. In this case, an additional evaluation or results investigation should be considered (Carlini et al., 2019).
- ***Regularisation without standardisation.*** Standardisation[113] is a data pre-processing technique. Regularisation improves model performance and aims to tune the function by adding a penalty term in the error function. However, when the data is not standardised or normalised, the regularisation alone could not boost the model's generalisation ability. To better help the regularisation, it is necessary to rescale the features using standardisation and normalisation, which will, together with regularisation, make the parameters in learning algorithms more likely to converge to smaller values, resulting in better performances (Dauphin and Cubuk, 2020).
- ***Inappropriate data representation.*** ML/DL, models are built by observing and interpreting the data. The correct application of data preparation will transform raw data into a representation that allows learning algorithms to get the most out of the data and make correct predictions (Zhong et al., 2016). Hence, it is highly important to feed the data in a way that important information is preserved. However, the wrong representation function can result in a lack of features, miss formatted inputs, or high sparsity.

[112] Another essential element to be considered is the impact of edge and corner cases. An edge case takes only one parameter into account and analyses the impact of its various values (e.g. If one feature of the data samples could impact the output related to that sample, then it can be considered an edge case). A corner case is defined as one with two or more parameters working together and significantly impacting the system's behaviour. The chapter related to Task 3 of the MLEAP project provides more analysis of the corner and edge cases.

[113] Standardization (Ali et al., 2014) typically means rescaling data with a mean of 0 and a standard deviation of 1 (unit variance). Feature scaling is one of the most critical data preprocessing steps in machine learning. Algorithms that compute the distance between the features are biased towards numerically more significant values if the data is not scaled.

- ***Inappropriate data in training and testing.*** As discussed in this chapter, the data quality used in developing any ML/DL application is essential to produce performant models (Gupta et al., 2021). Nevertheless, the following(Gupta et al., 2021). Nonetheless, these issues can lead to a poor generalisation ability of the model:
  - Training and target domain data are very different and distinct (different domains, different types, inputs format, length…), and the mismanagement of the domain shift characteristic will lead to poor performances.
  - Lack of Quality Data. (e.g., imbalanced data having more samples of a given class compared to others).
  - Noisy data. Data that contain a large amount of conflicting or misleading information (contradictory classes for similar entries or differences between the entries are not easy to capture by the model).
  - Dirty data: data that contains missing values, categorical and character features with many levels and inconsistent and erroneous values.
  - Sparse data. Data containing very few actual values comprises zeros, undefined, or missing values.
  - Inadequate data. Data that is either incomplete or biased.

  To cope with data completeness, representativeness, and volume adequacy, the reader can refer to Chapter 4 for a more detailed analysis of the state-of-the-art in data qualification and recommendations to enhance the data preparation pipelines.
- ***Inappropriate model complexity to perform the task***. As explained in the previous sections, the model's generalisation ability is highly correlated with its complexity and data qualification. More complex models will perform better on extensive training data than simpler ones. These models have more parameters that can be adjusted during training to fit the desired result well. Therefore, their error rate will usually be lower when measured on the training data set. However, a model that is too complex can end up overfitting to random effects that are present only in the training data set (Barr et al., 2013). The model will perform poorly if these random effects are absent in evaluation and test data sets. To figure out if this is the case, the model's error rate on a validation data set is the best indicator:
  - A high error rate on training and validation data indicates that the model may be too simple to capture any relationships in the data faithfully. In this case, the model seems to have a high bias (underfitting).
  - If the error rate is low on training data but high on validation data, then the model may be too complex and hence suffers from high variance (overfitting).

  To overcome this problem, one may regularise the complexity measure (Jiang et al., 2020) by adding a regulariser and directly optimising it, but this could fail due to: (1) The complexity measure that could change the loss landscape in non-trivial ways and make the optimisation more difficult; (2) The existence of implicit regularisation of the optimisation algorithm. This makes it hard to run a controlled experiment since one cannot simply turn off the implicit regularisation. The consequences should not be ignored; in (1), if the optimisation fails to optimise the measure, no conclusion can be made about the causality; in (2), if optimising a measure does not improve generalisation, it could be simply because it is regularising the model in the same way as the optimisation is regularising it implicitly.
  - ***Inappropriate evaluation metrics***. Evaluation metrics assess the model's evolution during training and in the validation set. Hence, the metrics to be used must correspond to the

objective of the application (classification or regression). In addition, the mean performance of the evaluation needs to be handled carefully during the selection of metrics. For instance, in the case of recommending safe city locations to tourists, true positives should be prioritised. In this case, a True Positive Rate or Precision should be preferred to Recall. Besides, not all the commonly used metrics are appropriate to reflect the targeted performance under evaluation; some combinations and testing practices can be needed (cf. Section 2.1) to ensure a more rigorous evaluation and verification.

- ***Bad global minima*[114]*.*** Picking the bad global minima can lead to lousy generalisation behaviour, especially for models trained with SGD (S. Liu et al., 2020). To avoid the wrong global minima, regularisation plays a central role, where complex models do not simply "*get lucky*" with the training data, but the regularisation makes simple models fit the data and the global optima. It also clarifies how to make them discoverable through local methods, such as SGD. According to (H. Li et al., 2018), the training parameters, including the batch size, the learning rate, and the used optimiser, highly affect the shape of minimisers. They have shown that when networks are too deep, the loss landscapes quickly go from nearly convex to highly chaotic, coinciding with a drop in generalisation error.
- ***Misuse of the model-driven regularisers***. Misunderstanding the assumptions about the input-output mappings that could apply only to the training data set as a limited sample or mismatching different tasks in the multi-task learning-based methods could lead to a competition-based learning process.
- ***Bad match between the model complexity and the optimisation algorithm***. As for the evaluation metrics, not all the optimisers can achieve better results. For example, early stopping is useless when the error does not stabilise, and the wrong data augmentation function will harm the model's performance instead of providing a more extensive data set for training. In this case, note that not all the data domains can be augmented easily; some use cases (e.g., air-traffic control data processing) could not benefit from trainable or non-trainable augmentation methods.
- ***The violation of data (training-validation) independence assumption.*** Inaccurate data oversampling[115], data augmentation before splitting into training, validation, and test sets, or performing feature selection before splitting data are all procedural errors that may lead to the violation of the train-test data sets independence assumption (Maleki et al., 2022). Hence, this would result in a performance measure not representative of the entire input domain defined by the ODD.
- ***Adapting an ample hypothesis space.*** When the training algorithm is not bounded well, the problem to be solved becomes more and more complex. To deal with this issue, one can use statistical learning theory[116]. This latter considers constructing approximations that

[114] If the loss function is convex, standard optimization techniques like gradient descent will find parameters that converge towards global minima, otherwise, the optimization might lead towards local minima convergence, where loss value is higher than the value at global minima.

[115] It aims to duplicate some examples from the minority class (data samples) in the original dataset, to overcome the data imbalance problem.

[116] Statistical learning theory (de Mello and Ponti, 2018) is a framework for machine learning that draws from statistics and functional analysis. It deals with finding a predictive function based on the data presented. The main idea in statistical learning theory is to build a model that can draw conclusions from data and make predictions.

converge to the desired function with increasing observations. Recently, (Vapnik and Izmailov, 2020) have explored the Hilbert space of L_2-norm functions and studied the convergence in the space of functions to reduce the set of admissible functions in a learning process. The authors have discussed learning methods that allow reducing the learning space by selecting an admissible (appropriate) subset of functions and finding the desired approximation in this admissible subset. This method is called the *"complete statistical theory of learning"* and uses weak and robust convergences of the Hilbert space (Soenjaya, 2013).

5.6.3 Expectations from evaluation vs reality

As discussed before, the evaluation metrics of different machine learning applications, such as *MSE*, *precision*, and *recall*, are used to measure only the technical performance of the ML/DL component. Through additional testing of the corresponding module, an analysis of how the system would interact with the user can be made, and an overall idea about the performances after the release is then drawn.

In the industry, some AI technologies can replace (or highly assist) a human being in several tasks and decision-making. Hence, these technologies are supposed to behave as closely as possible to that of an expert in the concerned industrial domain regarding decision-making and choice of the different actions necessary to perform a task. This is why the evaluation phase is so crucial in the certification[117] process of an AI that is intended to replace or complement a human effort. However, several empirical analyses have shown a significant gap between the behaviour of an ML/DL-based system and that of a human. For instance, a comparison of object recognition performance by humans and deep convolutional neural networks in image manipulation (van Dyck and Gruber, 2020) and object recognition (Ho-Phuoc, 2018) has shown that the best DL model trained for this task still cannot recognise correctly several images that are otherwise clear to a human eye. On the other hand, human-computer interactions imply that evaluation of human-facing systems, such as comment toxicity, misinformation detection and real-time video surveillance analysis, should account for people's reactions to the system (Gordon et al., 2021).

Despite the need to bring the man-machine links closer together, there is a gap between the ML/DL model performances and the efficiency needed in practice. In fact, in many instances, researchers assessed the validity of a specific ML model by reporting its performance against traditional evaluation metrics, only to be later identified that such a model does not correctly represent actual observations – despite having a good fit w.r.t the evaluation process. This can be avoided by adopting a rigorous validation procedure (Otles et al., 2021). Unfortunately, many of the published studies in the area of ML application in engineering do not include multi-criteria/additional validation phases and simply rely on conventional performance metrics such as R or R^2 of the derived models (Naser and Alavi, 2021).

Another procedural detail is the gap in performance between the empirical evaluation metrics and targeted performance in practice. The model evaluation must involve appropriate performance indicators. More precisely, the chosen metrics and performance thresholds must adequately reflect

[117] Note that the certification process of AI technologies is not directly considered in the scope of the MLEAP project. This latter focusses on complementing the W-shaped process steps for ML/DL development and evaluation even more implementation, in order to make conclusions that could help the certification process.

human judgement. Standard metrics, like Mean Absolute Error (MAE) and accuracy, are sometimes not enough to reflect the model's ability to generalise or the domain-specific (business) key performance indicators (KPIs). For example,[118], the enhanced use of email cannot be enough to measure the effectiveness of spam filters until proper isolation is done. Comments toxicity and misinformation detection can score highly on the evaluation pipeline but perform poorly in practice (Gordon et al., 2021). Thus, it is recommended to use AI/ML technical metrics in conjunction with business value metrics/KPIs to measure the effectiveness of ML solutions, such as the responsiveness[119] of the model answers to the business questions. Hence, when the model capabilities in the product are more pronounced, their influence on operational KPIs becomes greater. Moreover, with the rise of commercial, open-source, and user-oriented AI/ML tools, a critical question must be answered: *what constitutes a good AI/ML model?* A proper answer to this question depends on various factors, including that a good model optimally performs and describes the phenomenon being addressed. One possible option (Naser and Alavi, 2021) is using multi-fitness criteria (where a series of metrics are checked on one problem) to ensure the validity of ML models, as their combination may overcome the limitations of each metric. Furthermore, several fundamental engineering safety principles need to be handled, along with ML safety issues due to dependability[120] limitations (Mohseni et al., 2021), such as generalisation errors. Several research activities are launched in ML safety investigation and implementation, mainly with the perspective of enhancing performance and robustness by adapting robust network architectures (Djolonga et al., 2021) or robust training processes (Mohseni et al., 2021) and the run-time error detection for uncertainty resolving (Geifman and El-Yaniv, 2019, 2017).

Finally, aligning evaluation metrics with reality is essential to promote AI adoption in the industry (Gordon et al., 2021). The relation between ML models' performance and user-facing performance indicates a more extensive disconnection between researchers working on machine learning and human-computer interaction (HCI), while evaluating their work (Muller et al., 2021). In ML, a large set of technical metrics is used, primarily based on generalisation errors. On the other hand, HCI systems (Sinha et al., 2010) report user-facing experience, developing metrics (Ledo et al., 2018) that measure direct user response or opinions such as agreement rates (Hertzum and Jacobsen, 2001), Likert scales (Nataraja and Raju, 2013), and behavioural outcomes (Kostakos and Musolesi, 2017). To assess the acceptability of the technical performances of a system, Kay et al. (Kay et al., 2015) have introduced a new measure called *Acceptability of Accuracy* based on measurements of classifier accuracy. The objective is to assess the user tolerance to the uncertainty degree of a given classifier (i.e., is the 85% accuracy is acceptable? And how the remaining 15% of uncertainty would affect the user experience and tolerance?). The proposed tool enables the systematic selection of an objective function to optimise during classifier evaluation and insights into designing user feedback in interactive classification systems.

In the same line as the technical performance evaluation of ML/DL models, the safety of the systems integrating these models needs to be evaluated in the most critical applications of AI, including the vast area of aeronautics. In this case, AI systems are submitted to a consistent evaluation and

[118] https://vitalflux.com/different-success-metrics-for-ai-ml-initiatives/

[119] Ability of the model to bring an answer and which is relevant to the input question.

[120] It refers to the model's ability to minimize prediction risk on a given test set [Mohseni et al. 2021]. Unlike code-based algorithms, the dependability of ML algorithms is bounded to the model's learning capacity and statistical assumptions such as independent and identically distributed on relation of source and target domains.

validation pipeline (Brunton et al., 2021). Recently, the outcomes (Alecu et al., 2022) of one of the most promising projects in aeronautics, called DEEL[121] (DEpendable Explainable Learning), have provided an overview of several analyses required for the assessment of ML models' capacity to comply with existing safety standards. Various methods can be considered (and even combined) to attempt to fill the gap identified between ML performances and safety requirements, such as the conformity of predictions, the OOD and robustness monitoring, the availability, etc. Two practical examples from the railway and automotive industries have been explored, showing that ML performances are far from those required by safety objectives. Several techniques for reducing the error rate of ML components have been provided, such as model diversification, monitoring, classification with a reject option, conformal prediction, and temporal redundancy.

5.6.4 ML/DL testing limitation and challenges

As described in Section 5.6.3.1, a massive testing[122] approach can be adopted to ensure that the trained model and the module that includes a trained AI model have the expected behaviour in a defined scenario. To make sure the ML/DL component testing gets as close as possible to the reality of the targeted application, we need to verify that:

1) We have enough "*qualified*" and labelled data as needed for the whole testing scenario;
2) The testing scenarios should include system failures related to the implemented software as much as possible. Hence, continuous and resource-consuming testing is required;
3) The testing schemes should include "*performance evaluation*" indicators and scenarios to test the correct behaviour of the ML/DL model and the system module that embeds it and modules that are related to it as well;

Testing in software development explicitly identifies which parts of the source code fail and provides a relatively coherent coverage measure (e.g., lines of code covered). It is not that simple regarding the ML/DL model, where other criteria related to "*explainability*" (Arrieta et al., 2020) can be involved.

Despite the difficulties related to the massive testing of ML/DL components, this approach still represents essential elements and attention points adopted from software engineering. In particular, it helps in two ways: (a) *quality assurance, whether the software works according to requirements, and (b) identifying defects and flaws during development and* production. Both verifications (a) and (b) could apply to ML/DL modules meant for critical systems. To do so, several challenges need to be solved:

1) The lack of transparency due to the "*black box*" model consideration (mainly in deep learning);
2) The indeterminate modelling/results are due to stochastic training algorithms (e.g., SGD). This latter can provide two different models for two training times at the same data set due to random local/global minima and random initialisation of the same network connections. Hence, a slight difference in the predicted scores can have a significant impact on the results (e.g., predicted probability for a given class), which prevents models from reproducing the same results (or output scores) after (re)training;

[121] https://www.deel.ai/

[122] Like in software engineering, in order to validate the system, a big bang testing needs to be implemented and run. This can include different types of testing, such as functional testing (unit-testing, integration…) and non-functional testing (security, performance, usability…).

3) The unclear/insufficient test coverage is due to a lack of well-defined methods and tools to define testing coverage for machine learning models. Since in ML/DL, the "*Coverage*" does not refer to lines of code as it does in software development. Instead, it might relate to concepts like input data and model output distribution;

These issues make it challenging to understand the reasons behind a model's low performance, interpret the results, and ensure that the model will work, even when there is a change in the input data distribution (data drift) or in the relationship between our input and output variables (concept drift). These concepts relate more to the trained model's robustness (cf. Section 5.2.4) and generalisability (cf. Section 5.2.5). Unless combined with a well-defined evaluation pipeline, we believe the ML/DL testing is insufficient to build a safe AI module properly.

5.7 Analysis of generalisation assessment methods – Which impact on the development?

The main objective of this section is to draw up a summary of existing methods, highlighting any shortcomings in the state of the art and any limitations noted concerning the generalisation assessment techniques, ML/DL models development and training. So far, we have analysed several methods for generalisation assessment and the standard practices allowing, on the one hand, optimal model training and, on the other hand, a consistent evaluation by emphasising the expectations of the target application. We identified several limitations of existing approaches preventing the produced models from generalising their performance to unseen data sets and from being released. In the following, we will explain some research questions being addressed and how to deal with each issue, providing the identified ways to lead to answers in the industry. A set of selected methods for generalisation assessment and the main drivers of the selection process are then discussed. Finally, an analysis of the applicability and the expectations toward the guarantees and upstream patterns that can be drawn from these methods are provided.

5.7.1 Research questions and roadmap

How do we deal with overfitting/underfitting in industry?
Overfitting is a common problem in several supervised machine-learning tasks. It is noticed when a learning algorithm fits all the training data samples so well that noise and the particularities of the training data are memorised. This results in a drop in performance when the model is tested on an unknown data set (Jabbar and Khan, 2015). On the other hand, underfitting is noticed when the model cannot capture the variability of the data, resulting from using a model with a low capacity to describe a given problem. The common point is that overfitting and underfitting lead to the deterioration of the generalisation properties of a resulting ML/DL solution (Ying, 2019).

To deal with this problem, several techniques have been developed to help the ML/DL models generalise better. Although the original model may be too large to generalise well, regularisation techniques help limit learning to a subset of the hypothesis space, where resulting models will have manageable complexity (C. Zhang et al., 2021). By combining different methods, it makes the model generalise better, independently of the generalisation type (domain-based, multi-tasking, OOD-based):

- The regularisation methods could be data-driven (transformations and representation adaptation) or model-driven (making assumptions about the input-output mapping and using specific activations), even based on the training process through assumptions on the objective function, or based on the optimisation algorithm that aims to converge to the best global minima. Hence, specific initialisation and warm-up or pre-training options could be adopted, or specific functions in the weight update (e.g., weight sharing) could be used, as well as stopping the training at the best moment to preserve performance. Even if pre-training can help boost model performance (e.g., using a multi-lingual model pre-trained to solve a question-answering task in a specific language), pre-trained models are also often misused to deal with problems where training from scratch or using thorough domain adaptation, transfer learning, or multi-task learning methods would be worth trying. Other methods based on modifications of the network architecture (reduction methods) or the volume of the data set (expansion) have also been discussed, and their advantages and applicability depend highly on the target application. All these methods can be used to boost the performance of an ML/DL model that cannot be generalised to unseen instances during training.

How do we bridge the gap between experimentation and industrial expectations?

While developing AI solutions for the industrial domains, one typical objective is to put the models into production. To do this, the operational and behavioural expectations of the designed system are expressed via precise performance indicators (KPI) and acceptability criteria with a margin of error that must be very small. However, expectations are difficult to meet while developing the ML/DL modules. Indeed, in most AI development pipelines, several performance evaluation metrics are used for regression and classification applications to know if the model generalises effectively. So far, much work has been done in ML/DL evaluation metrics (Roelofs, 2019). All the existing evaluation metrics are based on the model predictions. These existing evaluation metrics are based on the model results. Others from (ISO/IEC, 2022b) described robustness metrics based on the model performance over part of the input domain. The expected outputs are compared to the ones produced by the model, and the evaluation metric is built on this observation. The number of accurate classifications or ranks is also assessed to determine whether the model is performant. An industrial development pipeline focuses on performance and robustness indicators that are not always expressed by classical ML/DL evaluation pipelines. In (De Prado, 2018), most of the ten reasons why AI applications fail in industry are derived from poor practices related to the experimental pipeline and which are not adapted to the target domain. We understand that infinite evaluation metrics could be used based only on the model results and behaviour during and after the training. However, none of the existing methods have focused on the reasons that make the model's generalisability weak, hence an adverse (limited) user experience.

To bridge the gap between the empirical and industrial processes, we need to leverage the evaluation metrics to reflect the targeted performances and integrate the KPIs in the training objectives and the evaluation pipeline. Hence:

- To evaluate the model results more rigorously, one exciting conclusion by Caruana et al. (Caruana and Niculescu-Mizil, 2004) is that *"Different performance metrics yield different trade-offs that are appropriate in different settings. No one metric does it all, and the metric optimised to or used for model selection does matter."* With this in mind, recent works in ML/DL models evaluation recommend the utilisation of multi-fitness criteria (Naser and Alavi, 2020; Yates et al., 2020), where a series of metrics are checked on one problem to assess the validity of AI models, as these metrics may overcome some of the limitations of individual metrics. Finally, the industrial KPIs must be considered seriously to prevent adverse user experiences after the ML/DL application release.

How do we cope with common data processing and evaluation mistakes?
The previous section 5.7 reviews pitfalls that could lead to weak model performances. This includes several practices during model evaluation and data preparation for training and testing, in addition to the limitations of typical evaluation schemes based on a giant map of the performances and the errors a trained model can make. The evaluation process of ML/DL applications needs to dive deeply into the model error distribution on the test dataset and understand what this means and how it could be reflected in reality.

To ensure a more rigorous evaluation pipeline, we suggest that the complete ML development workflow, from the data preparation and qualification step to the model validation and release, benefits from some software engineering best practices, such as the scenarios building for the test and the iteration on the process of data set improvement, evaluation benchmarks as well as the verification and validation process of the final trained model. Then, a rigorous error analysis is needed to provide a significant interpretation of a single metric computed on the whole dataset. When carried out correctly, both actions enable a better return on investment for the entire development process. This will prevent the IA-based project from being wholly doomed to failure because the objectives were not achieved due to an error in the model that was not understood, which would have a decisive impact on the acceptability of the final product. To do so:

- For better monitoring and understanding of the model outputs in case of a failure, an error analysis is needed. Error analysis, sometimes called *ML debugging* (Chakarov et al., 2016), aims to investigate unexpected model outputs in case of failure. For example, classification tasks determine where errors in training data lead to misclassifications in test points and propose ways to find the root causes. To do so, several methods have been developed in the state-of-the-art. For instance, the TIDE framework (Bolya et al., 2020) provides a set of tools for analysing the errors of a computer vision model while detecting their sources. Precisely, based on the bounding boxes predicted by a trained object-detection model, an error category is assigned, and the negative impact that each error category has on the averaged precision is calculated. This measures the importance of the different error types that the model makes on the test set. This error analysis can help focus on the most detrimental errors to performance. Besides, according to the industrial target objective, the error distribution through the different samples on the test dataset will, including the monitoring at the system level, help identify the acceptance/rejection of specific errors and limitations on performances.

5.7.2 Generalisation Guarantees Selection

The objective is to ensure the trained model performs well on unseen data. The generalisation performance assessment aims to:

- Ensure a minimum gap between the generalisation error and the test error;

- Provide generalisation guarantees on the entire domain of the application;

This concern must be considered during the full development of the machine-learning solution. Each step of the model development will have to demonstrate the performance level according to the target application's different criteria. The model performance should be validated according to requirements that are verified in each phase of the development.

Any hypothesis that correctly classifies all the training examples or gives the most optimal approximation of the expected value is considered consistent. However, some issues should be handled carefully, including:

- The training data may be noisy, so no consistent hypothesis exists.
- The target function may be outside the hypothesis space and must be approximated.
- A rote[123] learner and the overfitting learner that simply output y for every x such that $(x, y) \in D$ are consistent but fail to classify or predict a correct scoring value any x not in D (out of distribution samples).

As discussed previously, generalisation performance is affected by:

- The adequacy between the targeted function complexity and the algorithm capacity;
- The assumption taken on the data distribution and the data quality, such as the completeness and the representativeness;
- The selected hyper parameters w.r.t data complexity;
- The training process and the loss function selection;
- The trained model's robustness and stability indicate how the generalisation can be achieved. Hence, the target data and algorithm must be considered to choose the suitable bounds.

5.7.2.1 **Data Impact**

A strong assumption made at the beginning is that the data sets available are issued from the same distribution as the real world. Chapter 4 details elements that enable the provision of recommendations that help verify that the datasets adequately cover the operational domain concerned by the model. Besides, the volume of data needed to train the model properly is also crucial. *A priori* approach can help estimate this volume according to the task's complexity (cf. section 5.4.1), in addition to the aimed model capacity and the number of parameters involved. However, in a high-complexity setting, it is difficult to assess the volume needed as many different hypotheses can be used to draw the relationship between input and output data (for example, the groups of symmetries) are unknown.

[123] A rote learning (Foote, 2022) corresponds to a learning based on instructions. In this kind of learning, the model is trained to memorize a set of rules to make a correct correspondences between inputs and outputs (Rong et al., 2021), while its memorization capacity is evaluated using metrics such as Rademacher complexity and Vapnik-Chervonenkis (VC) dimension.

5.7.2.2 Model impact

At this stage, we aim to leverage some functions to help select potential models for the target task learning, considering its generalisation ability. Theoretical generalisation bounds can support the identify the correct hypothesis to be chosen. A summary table can be found in (X. Li et al., 2018).
Here, the adequacy between the targeted function and the algorithm's complexity is an important step and will assess the risk for the model to overfit or underfit. In this step, the models' hyperparameters are defined using the validation data set, for example, bias-variance trade-off or cross-validation techniques. Whatever the adapted method is, the objective is the same and should ensure that:

- The model will have high stability regarding the data selected for the training phase (cf. section 6.2);
- The volume of data of D_{train} is sufficient to train the model and answer performance requirements, such as robustness under adversarial attacks (B. Li et al., 2022) or generalisation error;
- The metrics are correctly identified to measure the performance of the model;
- The loss function is selected correctly and can help to minimise the cost;
- The training strategy (optimisation algorithm) identifies generalisation bounds that minimisation can support.

Regarding the theoretical evaluation of generalisation, we have reviewed different methods to correctly assess the model's generalisation error (cf. Section 5.4.2). For example, Rademacher complexity (Bartlett and Mendelson, 2002) is commonly used to determine how well a model can fit a random assignment of class labels. The VC dimension (V. Vapnik, 1999) measures the capacity/complexity of a learnable function set. To ensure the chosen method can correctly reflect the model's performance, a common practice is to adapt the one recommended to the model's architecture. Bounds based on uniform convergence provide guarantees that hold for any hypothesis and bounds (with probability $1 - \delta$) the actual risk by its empirical risk plus a penalty term that depends on the number of training examples, the complexity and the value of δ. Bounds based on performance stability deal with the dependence of the model trained by a learning algorithm by considering the stability of the algorithm concerning different training datasets. The main difference with the bounds based on uniform convergence is the incorporation of regularisation term and removal of the complexity argument; it can be used when hypothesis classes are challenging to analyse with classic complexity argument (such as support vector machines where VC Dimension is infinite). The robustness-based generalisation bounds are based on the capability of the model to keep its performance under several conditions (Gonen and Shalev-Shwartz, 2017; Hardt et al., 2016; Kuzborskij and Lampert, 2018); those bounds deal with a more extensive class of regularisers than stability, and its geometric interpretation adapts to non-standard settings possible such as non-IID data. For instance, (Rohlfs, 2022) proposed to categorise generalisation under several levels of abstraction, including sample generalisation, where test cases are drawn from the same population as the training data set; distribution generalisation, where test cases are drawn from new populations; and domain generalisation where the input-output relationship has changed. Another criterion is the confidence dimension (R. Wang et al., 2022) which is used to measure the deep model's generalisation ability and to theoretically calculate the upper bound of generalisation based on the conventional VC-dimension and Hoeffding's inequality.

As a result, Table 36 shows a selection of commonly used bounds identified for several model architectures, their applicability analysis (cf. sections 5.8.2.1 and 5.8.2.2), and the assumptions required to apply theorems.

5.7.2.3 Generalisation evaluation

Notation	Description
N_0	*The size of the dataset $\mathcal{S}_0$. $N_0 = \|\mathcal{S}_0\|$*
$\mathfrak{D}_0$	*The distribution of the dataset $\mathcal{S}_0$*
$\mathcal{H}$	*Hypothesis space*
h	*Single function from $\mathcal{H}$*
h_θ	*Single function from $\mathcal{H}$ identified by the parameters θ*
$\hat{h}_\theta$	*Single function for which the error on training data is minimized*
$\mathcal{R}^{\mathcal{D}}(h)$	*True risk or generalisation error : $\mathcal{R}^{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim P(X,Y)=\mathcal{D}}[L(y,h(x))]$*
$\mathcal{R}^{\mathcal{S}}_{emp}(h)$	*Empirical risk : $\mathcal{R}^{\mathcal{S}}_{emp}(h) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, h(x_i))$, $S = \{(x_i, y_i)\}_{i=1}^{N}$*
Θ	*Parameters' space ($\theta \in \Theta$)*
$L(y, f(x))$	*The loss function measuring the discrepancy between y and $f(x)$*
$\mathcal{A}$	*A learning algorithm*
$\mathbb{E}[.]$	*Expectation of a variable*
$\mathcal{X}$	*Input space*
$\mathcal{Y}$	*Output space*
$\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$	*Full space*
$KL(\mu\|\|\nu)$	*Kullback-Leibler divergence between μ and ν*
$\mathcal{S}$	*An arbitrary set*
$\|\mathcal{S}\|$	*Number of elements in $\mathcal{S}$*
$\mathcal{S}^n$	*A set of n elements from $\mathcal{S}$*
$\mathcal{P}$	*A probability distribution*
$x \sim \mathcal{P}$	*x is drawn IID. from probability distribution $\mathcal{P}$*
$z = (x, y) \in \mathcal{X} \times \mathcal{Y}$	*An arbitrarily labelled instance*

Table 36. Notations

In supervised learning, a loss function $L: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$, $L(y, \hat{h}_\theta(x))$ measures the discrepancy of the value predicted $\hat{h}_\theta(x)$ versus the expected one *y*. Loss functions satisfies the following properties:

$$\forall(\mathrm{y}, \hat{\mathrm{y}}) \in \mathcal{Y}^2, L\left(y, \hat{h}_\theta(x)\right) \geq 0 \text{ and } y = \hat{y} \Longrightarrow L(y, \hat{y}) = 0$$

Given a loss function L, the risk of a model h_θ is defined as the expectation over the full domain of the loss function applied to the output of the model:

$$\mathcal{R}(h) = \mathbb{E}_{(x,y)\sim P(X,Y)}[L(y, h(x))]$$

Whit $P(X, Y)$ is the probability distribution of the full domain.
However, we cannot access the complete domain distribution in a machine-learning context. Still, instead of an observation set of samples drawn from the joint distribution of $\mathcal{X}$ and $\mathcal{Y}$. This is the empirical risk on the dataset $\mathcal{S}$ defined as:

$$\mathcal{R}^{\mathcal{S}}_{emp}(h) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, h(x_i)),\ S = \{(x_i, y_i)\}_{i=1}^{N}$$

This metric can be applied on $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ during model finetuning and optimisation, or on $\mathcal{D}_{test}$ to estimate the generalisation error. Indeed, as long as the trained model does not depend on the test set and $\mathcal{D}_{train}, \mathcal{D}_{test}$ are drawn IID. from $\mathcal{D}$ (IID. assumption between training/validation/test dataset enable having un-bias estimate of model generalisation error), standard concentration results (Roelofs, 2019) show that $\mathcal{R}_{emp}^{\mathcal{D}_{test}}(\hat{h}_\theta)$ is an approximation of the true performance given by the population loss $\mathcal{R}^{\mathcal{D}}(\hat{h}_\theta)$. From statistical theory, we can say that the correctness of this approximation is strongly related to the volume and the quality of data in D_{test}, as we need to take into account the associated confidence interval (which is correlated to the distribution and the volume of test data). As far as the observations are complete and representative of the targeted distribution and we can demonstrate the data from D_{train} and D_{test} are issue from the observations distribution, the conclusions that are done using this approach are generalizable to the full domain and can be used to guarantee the average results on unseen data defined by the ML ODD[124].
Note that when $\hat{h}_\theta$ depends of $\mathcal{D}_{test}$, (Roelofs, 2019) propose methodologies to provide upper bound on generalisation gap based on $\mathcal{R}_{emp}^{\mathcal{D}_{test}}(\hat{h}_\theta)$.
The test-set method is only one way to estimate generalisation error. Other methods include re-substitution, cross-validation, bootstrap, posterior probability, and bolstered estimators.

We can apply this approach for the different metrics which are relevant versus the targeted function to provide for the selected metrics the guarantees that the probability of a metric m_i, to achieve an expected performance $perfo_i$ is greater than the expected probability Req:

$$P(m_i > perfo_i/x) > Req \text{ }, \forall\, x \in X$$

Generalisation is the average performance on the entire domain, and the previous formulation assures that with a high probability, model performance is higher than a certain threshold, relying a lot on the completeness and representativeness of the datasets within the operational domain.

It does not prevent the model from performing poorly in some regions of the domain, such as those close to decision boundaries or with corner cases, which is an important concern for model behaviour management.

This aspect is further developed in sections 6.1 and 6.2, where the model's robustness and stability are analysed in depth, and methods and tools for model evaluation are presented. Evaluating the model's behaviour under various perturbations or challenges to its input will help identify potential weaknesses and vulnerabilities.

Figure 94 illustrates the machine learning pipeline and some practical activities for the generalisation concerns.

[124] Note that we made several assumptions; in particular, we assume new data samples to be independent and drawn from the same distribution as the training data. However, in many real-world scenarios data samples are not entirely independent, and data used during training does not always come from the same distribution as that encountered later during deployment. (Lust and Condurache, 2020) is analysing several methods to detect out of distribution samples

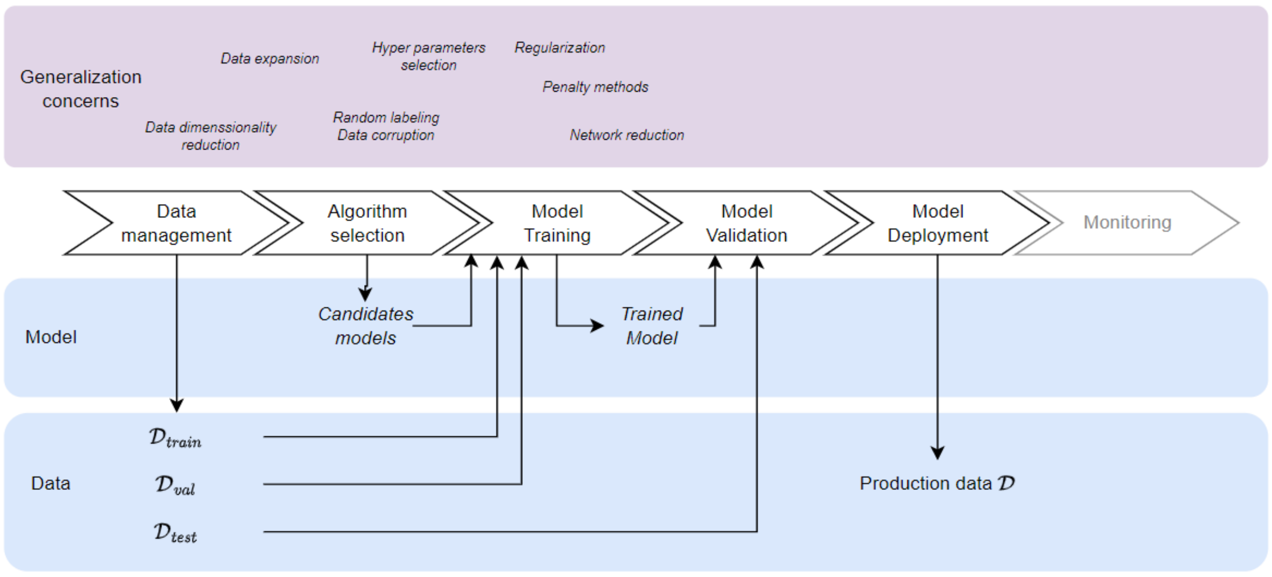


Figure 94 Illustration of the machine learning model development[125]

5.7.3 Methods Applicability Analysis

Generalisation guarantees aim to provide evidence that the trained model will produce correct outputs on average (with a certain level of accuracy) for all inputs of the entire ODD. Hence, we:

- Need to show a distribution of the observations that is close enough to the one of the actual target domain and adapted to the expectation purpose (cf. Chapter 4);
- Ensure that the fitting adequacy between the hypothesis space selected and the targeted function and the data complexity;
- Ensure that the trained model is correctly optimised;
- Ensure the absence of overfitting, underfitting and bias of the trained model by introducing metrics of performance evolution during the training process;
- Analyse the stability of the model w.r.t the training data set content during the training process;
- Assess the generalisation performance of the trained model using statistical theory results such as Chernoff Bound to estimate the Generalisation error based on the test error (Chernoff, 1952):

If $|D_{test}| \geq \frac{1}{2\epsilon^2}\log\left(\frac{2}{\delta}\right)$ *then with the probability* $1-\delta$ *the difference between the generalisation error and the test error is at most* ϵ*.*

Note that this measure is based on the data set volume, which makes the measurement more complicated (cf. section 2.3.2). Indeed, as this bound is growing exponentially with the tightness of the difference and the guarantee[126] *around* (for example for $\delta = 10^{-2}$ and $\epsilon = 10^{-4}$ it

[125] Based on the ML pipeline in a nutshell: https://www.altexsoft.com/blog/machine-learning-metrics/

[126] for example *"we need closely almost 50 000 data to statistically guarantee a difference of* 10^{-2} *with a 90% probability"*

requires $|D_{test}| \geq 2.65.10^8$), we will use bounds identified in the literature and potentially applicable to the use cases;

- Dive into the local generalisability of the trained model using data set margin calculations to assess the thickness of decision boundaries and area to focus on the local generalisation performance assessment. This analysis will potentially provide guarantees depending on the input (using what is presented in Chapters 4 and 6);
- Investigate information bottleneck framework and multi-view to bound generalisation error through input compression bound (Shwartz-Ziv and Tishby, 2017; Yan et al., 2023).

5.8 Experimental evaluation

After the generalisation properties analysis and the selection process of the methods that can be used to assess the generalisation bounds of a model, these aspects are put into practice using several application use cases. In this section, we first study the selected generalisation bounds in terms of their applicability and interpretability of the results using a toy use case. Then, the findings of the aviation-chosen use cases will be verified (cf. chapter 3). Evaluating the model's performances and analysing the existing pipelines would reveal several aspects of the consistency between the meant objective and the implemented models. The generalisation bounds will then assess the models' performances and recommend how their learning could be enhanced to boost their performances and better meet the fundamental industrial objectives.

We plan to follow and assess the selected bounds (difficulty to calculate, applicability limitations, assumptions validation…, etc.) for the subsequent evaluation steps of this project through different use cases. The selected bounds mentioned in Table 37 will be used to do so.

Algo.	Ref.	Bound
CNN	(Lin and Zhang, 2019)	$R_{\mathcal{D}}(F_C) \leq \hat{R}_{S,l_\eta}(F_C) + \tilde{O}\left(\frac{s^{\frac{L-1}{4}} L^{\frac{3}{4}} a^{\frac{1}{4}} c^{\frac{1}{2}} m^{\frac{1}{8}} r^{\frac{1}{2}}}{\sqrt{N}}\right) + \tilde{O}\left(\frac{s^{\frac{L-1}{4}} L^{\frac{3}{4}} a^{\frac{1}{4}} d}{\sqrt{N}}\right)$
NN for classification	(P. Jin et al., 2020)	$\mathbb{P}_{\mathcal{D}}\left[\forall h_\theta \in \mathcal{H}, \mathbb{E}_{x,y\sim\mathbb{P}(X,Y)}\left[\mathcal{R}(h_\theta)\right] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}^{\mathcal{D}}_{emp}(h_\theta)\right] + \frac{\sqrt{d}.CD(\mathcal{D})}{\min\left(\delta_0, \kappa\delta_{\mathcal{D}}\right)}\right] \geq 1-\delta$
NN	(Alquier, 2021)	Cantoni's bound (PAC Bayes) $\mathbb{P}_{\mathcal{D}_0}\left[\forall \rho \in \mathcal{P}(\Theta), \quad \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}(h_\theta)\right] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}^{\mathcal{D}_0}_{emp}(h_\theta)\right] + \frac{\lambda C^2}{8N_0} + \frac{KL(\rho\|\pi) + log\frac{1}{\varepsilon}}{\lambda}\right] \geq 1-\varepsilon$
	(Alquier, 2021) (McAllester, 1998)	Mc Allester's bound $\mathbb{P}_{\mathcal{D}_0}\left[\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}(\theta)\right] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}^{\mathcal{D}_0}_{emp}(\theta)\right] + \sqrt{\frac{KL(\rho\|\pi) + log\frac{1}{\varepsilon} + \frac{5}{2}\log\left(N_{\mathcal{D}_0}\right) + 8}{2N_{\mathcal{D}_0} - 1}}\right] \geq 1-\varepsilon$
	(Alquier, 2021) (Seeger, 2002)	Seeger's bound

Algo.	Ref.	Bound
		$\mathbb{P}_{\mathcal{D}}\left[\forall\rho\in\mathcal{P}(\Theta),\mathbb{E}_{\theta\sim\rho}[\mathcal{R}^{\mathcal{D}}(h_\theta)]\leq kl^{-1}\left(\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_\theta)\right]\,\Big\vert\,\frac{KL(\rho\Vert\pi)+\log\frac{2\sqrt{N_{\mathcal{D}}}}{\epsilon}}{N_{\mathcal{D}}}\right)\right]\geq 1-\epsilon$
	(Alquier, 2021) (Tolstikhin and Seldin, 2013)	Tolstikhin and Seldin's bound $\mathbb{P}_{\mathcal{D}}\left[\begin{array}{c}\forall\rho\in\mathcal{P}(\Theta),\mathbb{E}_{\theta\sim\rho}[\mathcal{R}(h_\theta)]\leq\ \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_\theta)\right]+\\ \sqrt{2\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_\theta)\right]\frac{KL(\rho\Vert\pi)+\log\frac{2\sqrt{N_{\mathcal{D}}}}{\epsilon}}{2N_{\mathcal{D}}}}+2\frac{KL(\rho\Vert\pi)+\log\frac{2\sqrt{nN_{\mathcal{D}}}}{\epsilon}}{2N_{\mathcal{D}}}\end{array}\right]\geq 1-\epsilon$
Fully connected NN & CNN	(Arora et al., 2018)	$\mathbb{P}_{\mathcal{D}_0}\left[\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}(h_\theta)\right]\leq\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}_0}(\hat{h}_\theta)\right]+\tilde{\mathcal{O}}\left(\sqrt{\frac{c^2d^2\max_{x\in\mathcal{D}_0}\Vert h_\theta(x)\Vert_2^2\sum_{i=1}^{d}\frac{1}{\mu_i^2\mu_{i\to}^2}}{\gamma^2N_{\mathcal{D}_0}}}\right)\right]\geq 1-\delta$
Two class classifier	(Anthony, 2004)	$\mathbb{P}_{\mathcal{D}}\left[\mathbb{E}_{\theta\sim\rho}[\mathcal{R}(h_\theta)]<\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_\theta)\right]+\sqrt{\frac{8}{N_{\mathcal{D}}}\left((n+k-1)\log\left(\frac{2eN_{\mathcal{D}}k}{n+k-1}\right)+\log\left(\frac{4}{\epsilon}\right)\right)}\right]\geq 1-\varepsilon$
Supervised learning	(Neu and Lugosi, 2022)	$\lvert\mathbb{E}[\mathrm{gen}(W_n,S_n)]\rvert\leq\sqrt{\frac{4H(P_n)\mathbb{E}\left[\left\Vert\bar{l}(.,Z)\right\Vert_*^2\right]}{\alpha n}}$
γ-uniformly stable learning algorithm	(Feldman and Vondrak, 2018)	$\mathbb{P}_{\mathcal{D}}\left[\mathbb{E}_{\theta\sim\rho}[\mathcal{R}(h_\theta)]\leq\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_\theta)\right]+8\sqrt{\left(2\gamma+\frac{1}{N_{\mathcal{D}}}\right).\log\frac{8}{\mathcal{E}}}\right]\leq 1-\varepsilon$
DNN	(Arora et al., 2018)	$\mathbb{P}_{\mathcal{D}_0}\left[\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}(h_\theta)\right]\leq\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}_0}\left(\widehat{h}_\theta\right)\right]+\widetilde{\mathcal{O}}\left(\sqrt{\frac{c^2d^2\max_{x\in\mathcal{D}_0}\Vert h_\theta(x)\Vert_2^2\sum_{i=1}^{d}\frac{1}{\mu_i^2\mu_{i\to}^2}}{\gamma^2N_{\mathcal{D}_0}}}\right)\right]\geq 1-\delta$
	(Hardt et al., 2016)	$\mathbb{E}\left[\mathcal{R}\left(h_{\theta_T}\right)\right]\leq\mathbb{E}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_{\hat{\theta}})\right]+\frac{M.L_{max}}{\sqrt{N_{\mathcal{D}_0}}}\sqrt{\frac{N_{\mathcal{D}_0}+2.T}{T}}$
	(Lei et al., 2022)	$\mathbb{P}_{\mathcal{D}}\left[\mathbb{E}_{\mathcal{D}}[\mathcal{R}(h_\theta)]\leq\mathbb{E}_{\mathcal{D}}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_\theta)\right]+\Omega+\sqrt{4\Omega\Delta}+8\Delta+\delta\right]\leq 1-\varepsilon$ *where* • $\Omega=\delta+\sqrt{\frac{1}{2(1-\eta)N_{\mathcal{D}}}\log\frac{1}{\varepsilon}}$, ● $\Delta=\eta\log\frac{e}{\eta}+\frac{1}{N_{\mathcal{D}}}\log\frac{2}{\varepsilon}$
CNN	(Arora et al., 2018)	$\mathbb{P}_{\mathcal{D}_0}\left[\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}(h_\theta)\right]\leq\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}_0}\left(\widehat{h}_\theta\right)\right]+\widetilde{\mathcal{O}}\left(\sqrt{\frac{c^2d^2\max_{x\in\mathcal{D}_0}\Vert h_\theta(x)\Vert_2^2\sum_{i=1}^{d}\frac{\beta^2([\kappa_i/s_i])^2}{\mu_i^2\mu_{i\to}^2}}{\gamma^2N_{\mathcal{D}_0}}}\right)\right]$ $\geq 1-\delta$
General NN	(Kawaguchi et al., 2023)	$\mathbb{P}_{\mathcal{D}}\left[\forall h_\theta\in\mathcal{H}_{val},\mathbb{E}_{x,y\sim\mathbb{P}(X,Y)}\left[\mathcal{R}(h_\theta)\right]\leq\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}_{val}}(h_\theta)\right]+\frac{2C\log\frac{\lvert\mathcal{H}_{val}\rvert}{\delta}}{3N_{val}}+\sqrt{\frac{2\gamma^2\log\frac{\lvert\mathcal{H}_{val}\rvert}{\delta}}{N_{val}}}\right]$ $\geq 1-\delta$

Table 37. List of selected generalisation bounds

5.8.1 Selected bounds exploration

Based on the generalisation assessment above and referring to the set of selected bounds for our experiments, this section highlights their main components and differences to help understand the subsequent section.

5.8.1.1 **Bound 001: (Lin and Zhang, 2019) – CNN**

Theorem: *(Generalisation Bound for Convolutional Neural Networks) Given a training sample* $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$ *of size* N*, each* (x_i, y_i) *is IID. according to some unknown distribution* $\mathcal{D}$*. Let* $X = (x_1, \dots, x_N)^T \in \mathbb{R}^{d \times N}$ *be the* N *inputs. For a convolutional neural network* $\mathcal{F}_{\mathcal{C}}$ *defined, let* $(\sigma_1, \dots, \sigma_l)$ *be some fixed functions (e.g., rectified linear unit (ReLU) and max pooling function) where* σ_i *is ρ*$_i$*-Lipschitz satisfying* $\sigma_i(0) = 0$ *. Then with probability at least 1 − δ, we have*

$$R_{\mathcal{D}}(F_C) \leq \hat{R}_{S,l_\eta}(F_C) + \tilde{O}\left(\left(\frac{\|X\|_F \Re_C}{\eta}\right)^{\frac{1}{4}} N^{-\frac{5}{8}} + \sqrt{\frac{\ln(1/\delta)}{N}}\right)$$

Where $\Re_C$ represents the corresponding complexity.
After simplification using bounds on weights norms, we have the following inequality:

$$R_{\mathcal{D}}(F_C) \leq \hat{R}_{S,l_\eta}(F_C) + \tilde{O}\left(\frac{s^{\frac{L-1}{4}} L^{\frac{3}{4}} a^{\frac{1}{4}} c^{\frac{1}{2}} m^{\frac{1}{8}} r^{\frac{1}{2}}}{\sqrt{N}}\right) + \tilde{O}\left(\frac{s^{\frac{L-1}{4}} L^{\frac{3}{4}} a^{\frac{1}{4}} d}{\sqrt{N}}\right)$$

Where:

- *L* is the number of layers
- *s* is a bound of the spectral norm of the corresponding fully connected weight matrix
- *a* is defined by: $||W||_F < a$; $W = (w_1, \dots, w_c) \in \mathbb{R}^{c \times r}$ is the convolutional weight matrix
- *c* is the number of convolutional filters
- *m* is the number of outputs generated by each convolutional filter $w_i \in W$
- *r* is the maximum size of the convolutional layers
- *d* is the maximum output dimension of fully connected layers

This bound is data-independent, algorithm-independent and based on a uniform convergence approach.

5.8.1.2 **Bound 002: (P. Jin et al., 2020)**

This bound is for classification, data-dependent, and trained model-dependent. It focuses on generalisation error without considering approximation and optimisation errors (respectively, depending on the network size and correlated with the loss value); it uses *cover complexity* to measure the difficulty of learning a data set and the *inverse of the modulus of continuity* to quantify neural network smoothness.

$$\mathbb{P}_{\mathcal{D}}\left[\forall h_\theta \in \mathcal{H}, \mathbb{E}_{x,y \sim \mathbb{P}(X,Y)}\left[\mathcal{R}(h_\theta)\right] \leq \mathbb{E}_{\theta \sim \rho}\left[\mathcal{R}^{\mathcal{D}}_{emp}(h_\theta)\right] + \frac{\sqrt{d}.CD(\mathcal{D})}{\min(\delta_0, \kappa\delta_{\mathcal{D}})}\right] \geq 1 - \delta$$

- d is the input dimension
- $\mathrm{CD}(\mathcal{D})$ is the cover difference of the dataset $\mathcal{D}$
- δ_0 represents the supremum separation gap between the different categories (the smallest distance between two different single label points)
- κ is a constant (0.1 for example) such that $\kappa\delta_0 \leq \kappa\delta_\tau \leq \delta_\tau/2$ holds for any single label training set
- $\delta_{\mathcal{D}}$ is the empirical separation gap for h_θ

5.8.1.3 Bound 003: Cantoni's bound (Alquier, 2021)

This algorithm-dependent bound is part of the PAC-Bayes bounds based on uniform convergence and has been optimised for the prior distribution.
Theorem:
For any $\lambda > 0$, for any $\varepsilon \in [0,1]$,

$$\mathbb{P}_{\mathcal{D}_0}\left[\forall \rho \in \mathcal{P}(\Theta), \qquad \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}(h_\theta)\right] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}_0}(h_\theta)\right] + \frac{\lambda C^2}{8N_0} + \frac{KL(\rho||\pi) + log\frac{1}{\varepsilon}}{\lambda}\right] \geq 1-\varepsilon$$

Where $\pi \in \mathcal{P}(\Theta)$ a given probability measure, C is the upper bound of the loss function ($0 \leq L \leq C$) and ε is the probability of being misled by the training set in the frame of probably approximately correct theory. h_θ is assumed to be a measurable and bounded function.

The main difficulty with this bound is choosing the prior distribution. This type of bound depends on the distance between prior and posterior distributions through the KL divergence term and is algorithm-dependent.

5.8.1.4 Bound 004: McAllester's bound (McAllester, 1998)

This algorithm-dependent bound deals with distributions over the hypothesis class and is part of the PAC Bayes Bounds.
Theorem: For finite and measurable parameters space Θ,

$$\mathbb{P}_{\mathcal{D}_0}\left[\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}(\theta)\right] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}_0}(\theta)\right] + \sqrt{\frac{KL(\rho||\pi) + log\frac{1}{\varepsilon} + \frac{5}{2}\log\left(N_{\mathcal{D}_0}\right) + 8}{2N_{\mathcal{D}_0} - 1}}\right] \geq 1-\varepsilon$$

Where $\pi \in \mathcal{P}(\Theta)$ a given probability measure, and ε is the probability of being misled by the training set in the frame of probably approximately correct theory. h_θ is assumed to be a measurable and bounded function.

5.8.1.5 Bound 005: Seeger's bound (Seeger, 2002)

Theorem: for any $\varepsilon > 0$,

$$\mathbb{P}_{\mathcal{D}}\left[\forall\rho \in \mathcal{P}(\Theta), \mathbb{E}_{\theta\sim\rho}[\mathcal{R}^{\mathcal{D}}(h_\theta)] \leq kl^{-1}\left(\mathbb{E}_{\theta\sim\rho}[\mathcal{R}^{\mathcal{D}}_{emp}(h_\theta)] \middle| \frac{KL(\rho||\pi) + \log\frac{2\sqrt{N_{\mathcal{D}}}}{\epsilon}}{N_{\mathcal{D}}}\right)\right] \geq 1-\epsilon$$

Where $\pi \in \mathcal{P}(\Theta)$ a given probability measure

This algorithm-dependent PAC Bayes is difficult to calculate, and we will use the Tolstikhin and Seldin's bound which is using an explicit bound for the function kl^{-1}

5.8.1.6 Bound 006: Tolstikhin and Seldin's bound (Tolstikhin and Seldin, 2013)

This proposed bound is algorithm-dependent and based on PAC Bayes framework.

Theorem based on PAC Bayes approach: for any $\varepsilon > 0$,

$$\mathbb{P}_{\mathcal{D}}\left[\forall\rho \in \mathcal{P}(\Theta), \mathbb{E}_{\theta\sim\rho}[\mathcal{R}(h_\theta)] \leq \mathbb{E}_{\theta\sim\rho}[\mathcal{R}^{\mathcal{D}}_{emp}(h_\theta)] + \sqrt{2\mathbb{E}_{\theta\sim\rho}[\mathcal{R}^{\mathcal{D}}_{emp}(h_\theta)]\frac{KL(\rho||\pi) + \log\frac{2\sqrt{N_{\mathcal{D}}}}{\epsilon}}{2N_{\mathcal{D}}}} + 2\frac{KL(\rho||\pi) + \log\frac{2\sqrt{nN_{\mathcal{D}}}}{\epsilon}}{2N_{\mathcal{D}}}\right] \geq 1-\epsilon$$

Where $\pi \in \mathcal{P}(\Theta)$ a given probability measure

5.8.1.7 Bound 007: "Arora" bound (Arora et al., 2018)

This bound algorithm is based on a compression-based framework. It is data-independent and can be applied to fully connected neural networks or convolutional neural network architectures.

Fully connected neural network:

For any fully connected network h_θ with $\rho_\delta \geq 3d$, any probability $0 < \delta \leq 1$ and any margin γ, the Algorithm generates weights for the network $\hat{h}_\theta$ such that over the training set $\mathcal{D}_0$, we have:

$$\mathbb{P}_{\mathcal{D}_0}\left[\mathbb{E}_{\theta\sim\rho}\,[\mathcal{R}(h_\theta)] \leq \mathbb{E}_{\theta\sim\rho}\,\left[\mathcal{R}^{\mathcal{D}_0}_{emp}\left(\hat{h}_\theta\right)\right] + \tilde{\mathcal{O}}\left(\sqrt{\frac{c^2 d^2 \max_{x\in\mathcal{D}_0}||h_\theta(x)||_2^2 \sum_{i=1}^{d}\frac{1}{\mu_i^2\mu_{i\rightarrow}^2}}{\gamma^2 N_{\mathcal{D}_0}}}\right)\right] \geq 1-\delta$$

Where:

- d is the number of layers in the network
- μ_i represents the layer cushion ($^1/_{\mu_i^2}$ is equal to the noise sensitivity of matrix Ai with respect to Gaussian noise)
- $\mu_{i\rightarrow}$ is the minimal inter layer cushion, $\mu_{i\rightarrow} = \min\left\{\frac{1}{\sqrt{d_i}}, \min_{i<j<d}\mu_{i,j}\right\}$, d_i is the number of hidden units in layer i
- c is the activation contraction

- ρ_δ is the interlayer smoothness defined as the smallest number such that for any layer i<j and any $x \in \mathcal{D}_0$, with probability $1-\delta$ over noise η $||M^{i,j}(x^i+\eta) - J^{i,j}_{x^i}(x^i+\eta)|| \leq ||\eta||\ ||x^j||/\rho_\delta||x^i||$
- $||.||_2$ denotes spectral norm

With this bound, we can see that

- the more the margin is essential, the more the bound will converge
- regularisation is also a driver (minimising the norm) by bounding the cushions

Convolutional neural network:
For any convolutional neural network h_θ with $\rho_\delta \geq 3d$, any probability $0 < \delta \leq 1$ and any margin γ, the Algorithm generates weights for the network $\hat{h}_\theta$ such that with probability $1-\delta$ over the training set $\mathcal{D}_0$, we have:

$$\mathbb{P}_{\mathcal{D}_0}\left[\mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}(h_\theta)\right] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}^{\mathcal{D}_0}_{emp}(\hat{h}_\theta)\right] + \tilde{\mathcal{O}}\left(\sqrt{\frac{c^2 d^2 \max_{x\in\mathcal{D}_0}||h_\theta(x)||_2^2 \sum_{i=1}^{d} \frac{\beta^2([\kappa_i/s_i])^2}{\mu_i^2 \mu_{i\rightarrow}^2}}{\gamma^2 N_{\mathcal{D}_0}}}\right)\right] \geq 1-\delta$$

Where:

- d is the number of layers in the network
- μ_i is the layer cushion
- $\mu_{i\rightarrow}$ is the minimal inter layer cushion
- d_i the number of hidden units in layer i
- c is the activation contraction
- ρ_δ is the interlayer smoothness,
- $||.||_2$ denotes spectral norm
- β is the well-distributed Jacobian
- s_i and κ_i are stride and filter width in layer i.

5.8.1.8 Bound 008: Anthony's bound (Anthony, 2004)

This PAC Bayes bound is valid for 2 classes only and will not be used in this report. It is algorithm-dependent.

$$\mathbb{P}_{\mathcal{D}}\left[\mathbb{E}_{\theta\sim\rho}[\mathcal{R}(h_\theta)] < \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}^{\mathcal{D}}_{emp}(h_\theta)\right] + \sqrt{\frac{8}{N_{\mathcal{D}}}\left((n+k-1)\log\left(\frac{2eN_{\mathcal{D}}k}{n+k-1}\right) + \log\left(\frac{4}{\epsilon}\right)\right)}\right] \geq 1-\varepsilon$$

- *n* is the input dimension
- k is the number of terms of $\mathcal{H}$ been a set of decision lists on $\mathbb{R}^n$

5.8.1.9 **Bound 009: Neu and Lugosi's bound (Neu and Lugosi, 2022)**

This bound is valid when the functions $h \in \mathcal{H}$ are α-strongly convex with respect to the norm $||.||$. This bound is similar with the Arora one (it is part of the compression bounds giving generalisation bound of the compressed model) and will not be calculated as it will give the same results and will drive same conclusions.

$$\mathbb{P}_{\mathcal{D}}\left[\mathbb{E}_{\theta\sim\rho}[\mathcal{R}(h_\theta)] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_\theta)\right] + \sqrt{\frac{4KL(\rho||\pi)E[||l||_\infty^2]}{n}}\right] \leq 1-\varepsilon$$

5.8.1.10 **Bound 010: Feldman's bound (Feldman and Vondrak, 2018)**

This algorithm dependent bound has a small dependence regarding the data through the stability of the trained model w.r.t train dataset; It is part of the stability bounds.
$\mathcal{H}$ represents a space of data dependent functions with uniform stability γ, then for an ε ∈ $[0,1]$ we have

$$\mathbb{P}_{\mathcal{D}}\left[\mathbb{E}_{\theta\sim\rho}[\mathcal{R}(h_\theta)] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}}(h_\theta)\right] + 8\sqrt{\left(2\gamma+\frac{1}{N_{\mathcal{D}}}\right).\log\frac{8}{\varepsilon}}\right] \leq 1-\varepsilon$$

Definition: Let $\mathcal{A}: \mathcal{Z}^{\mathrm{d}} \to \mathcal{H}$ be a learning algorithm mapping a dataset $\mathcal{D}_{\mathrm{train}}$ to a model in $\mathcal{H}$ and $\mathrm{L}: \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$ be a function such that L measures the loss of model h_θ on point z. Then $\mathcal{A}$ is said to have uniform stability γ with respect to L if for any pair of datasets $\mathcal{D}_{\mathrm{train}}$, $\mathcal{D}'_{\mathrm{train}}$ that differ in a single element and every $\mathrm{z} \in \mathcal{Z}$, $|\mathrm{L}(\mathcal{A}(\mathcal{D}_{\mathrm{train}})) - \mathrm{L}(\mathcal{A}(\mathcal{D}'_{\mathrm{train}}))| \leq \gamma$.

5.8.1.11 **Bound 011: Hardt's bound (Hardt et al., 2016)**

Let $\mathcal{D}_0 = (\mathrm{z}_1, \mathrm{z}_2, \ldots, \mathrm{z}_{\mathrm{N}_{\mathcal{D}_0}})$ be a set of $\mathrm{N}_{\mathcal{D}_0}$ samples. Let L be a β-smooth convex loss function satisfying $||\nabla\mathrm{L}(\mathrm{h}_\theta(\mathrm{z})|| \leq \mathrm{L}_{\max}$ and let $\hat{\theta}$ be a minimizer of the empirical risk $\mathcal{R}_{\mathrm{emp}}^{\mathcal{D}_0}(\widehat{\mathrm{h}_\theta})$. Suppose we run T steps of SGM (stochastic gradient method) with suitably chosen step size from a starting point θ_0 that satisfies $\|\theta_0 - \hat{\theta}\| \leq \mathrm{M}$. Then, the average θ_{T} over the iterates satisfies

$$\mathbb{E}[\mathcal{R}(h_{\theta_T})] \leq \mathbb{E}[\mathcal{R}_{emp}^{\mathcal{D}}(\widehat{\mathrm{h}_\theta})] + \frac{M.L_{max}}{\sqrt{N_{\mathcal{D}_0}}}\sqrt{\frac{N_{\mathcal{D}_0}+2.T}{T}}$$

This bound is based on stability properties and is algorithm-dependent.

5.8.1.12 **Bound 012: Lei's bound (Lei et al., 2022)**

This margin bound (data dependent, algorithm dependent) is based on the decision boundary of a classifier and is then data dependent.
Definition: *(decision boundary). Let $h_\theta(x): \mathbb{R}^{d_{in}} \to \mathbb{R}^k$ be a neural network for classification*

parameterized by ϑ, where d_{in} and k are the dimensions of input and output, respectively. Then, the decision boundary of h_θ is defined by

$$\left\{x \in \mathbb{R}^{d_{in}} \mid \exists i \neq j \in [k], h_\theta^i(x) = h_\theta^j(x) = \max_{q=1\dots k} h_\theta^i(x)\right\}$$

***Theorem**: (data DB variability-based upper bound on expected risk). If the decision boundaries of h_θ possess a (δ,η)-data Decision Boundary variability on the data generating distribution $\mathcal{D}$, and assume η ≤ 0.5, then, over a sample of size $N_\mathcal{D}$, we have*

$$\mathbb{P}_\mathcal{D}\left[\mathbb{E}_\mathcal{D}[\mathcal{R}(h_\theta)] \leq \mathbb{E}_\mathcal{D}\left[\mathcal{R}_{emp}^\mathcal{D}(h_\theta)\right] + \Omega + \sqrt{4\Omega\Delta} + 8\Delta + \delta\right] \leq 1 - \varepsilon$$

where

- $\Omega = \delta + \sqrt{\frac{1}{2(1-\eta)N_\mathcal{D}} \log\frac{1}{\varepsilon}}$,
- $\Delta = \eta \log\frac{e}{\eta} + \frac{1}{N_\mathcal{D}}\log\frac{2}{\varepsilon}$

(δ,η)-data DB variability is proposed to characterise the decision boundary from the perspective of the randomness in training data. Given a neural network, if its decision boundary can be "reconstructed" by training a network with the same architecture from scratch on a smaller η subset (which contains η% examples of the source training set), while the "error" of the reconstruction is not larger than δ, we call the model has (δ, η)-data DB variability.
In contrast to previous generalisation bounds based on hypothesis complexity, this bound only needs the predictions, meaning that it can be used only after the training phase.

5.8.1.13 Bound 013: Kawaguchi's bound (Kawaguchi et al., 2022)

Theorem: Assume that S_{val} is generated by IID. draws according to a true distribution P(X,Y) . Let $\kappa_{h_\theta,i} = \mathcal{R}(h_\theta) - L(h_\theta(x_i), y_i)$ for $(x_i, y_i) \in S_{val}$. Suppose that $\mathbb{E}\left[\kappa_{h_\theta,i}^2\right] \leq \gamma^2$ and $\left|\kappa_{h_\theta,i}\right| \leq C$ almost surely, for all $(h_\theta, i) \in \mathcal{H}_{val} \times \{1, \dots, N_{val}\}$. Then, for any δ > 0, with probability at least 1 − δ, the following holds for all $h_\theta \in \mathcal{H}_{val}$:

$$\mathbb{P}_\mathcal{D}\left[\forall h_\theta \in \mathcal{H}_{val}, \mathbb{E}_{x,y\sim\mathbb{P}(X,Y)}\left[\mathcal{R}(h_\theta)\right] \leq \mathbb{E}_{\theta\sim\rho}\left[\mathcal{R}_{emp}^{\mathcal{D}_{val}}(h_\theta)\right] + \frac{2C\log\frac{|\mathcal{H}_{val}|}{\delta}}{3N_{val}} + \sqrt{\frac{2\gamma^2\log\frac{|\mathcal{H}_{val}|}{\delta}}{N_{val}}}\right] \geq 1 - \delta$$

$\mathcal{H}_{val}$ is defined as a set of models h_θ that is independent of a held-out validation dataset, but can depend on the training dataset.
The central assumption of this bound is IID. draws according to $\mathbb{P}(X, Y)$. As $|\mathcal{H}_{val}|$ is enormous (it contains all functions defined by some hyperparameters constraints), this bound is not useful in practical application and provides vacuous results. For example, considering $\mathcal{H}_{val}$ as a set of models such that each element h_θ is constrained by some hyperparameters and training accuracy, it exists a huge number of possible parameters θ for h_θ especially in deep learning (for example it can be

possible to add a small epsilon (in the range of 10^{-9}-10^{-7}) to specific parameter without changing the output of the network meaning that $|\mathcal{H}_{val}|$ is becoming infinite).

In the next section will analyse the application of those bounds at different steps of the development process:
A priori evaluation (before training activities):

- To provide quantifiable generalisation guarantees before training the model (LM-04)
- To support model selection, model design and adaptation

A posteriori evaluation (a naïve application of theorems on trained models):

- To analyse the quantifiable generalisation guarantees we can obtain on a trained model
- To verify that empirical risk measured on test data is properly bounded by the computed bound (LM-14)

5.8.2 Toy Use case: Classification on FashionMNIST

A relatively simple use case facilitates the analysis and ensures the right understanding of the results, enabling comparisons between generalisation bounds and their significations.

5.8.2.1 **Dataset**

5.8.2.1.1 Dataset description.

Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000. Each example is a standardised 28x28 grayscale image, associated with a label from 10 classes (0: "T-shirt/Top", 1: "Trouser", 2: "Pullover", 3: "Dress", 4: "Coat", 5: "Sandal", 6: "Shirt", 7: "Sneaker", 8: "Bag", 9: "Ankle Boot").
Zalando created this dataset as a compatible replacement for the original MNIST dataset of handwritten digits.

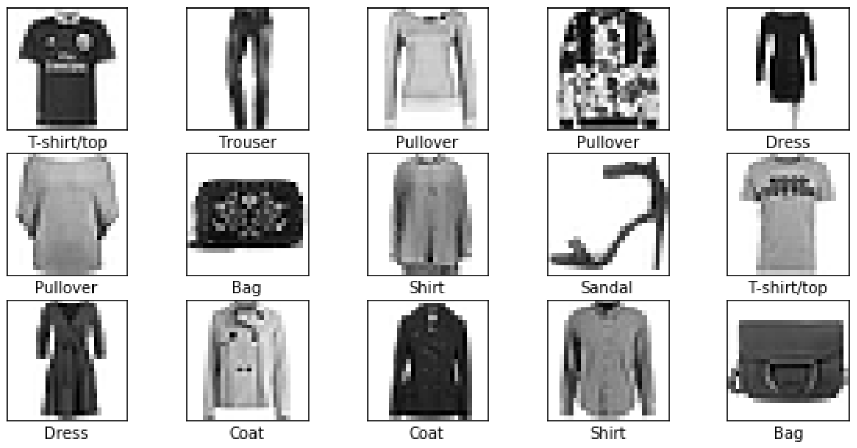


Figure 95 Example of FashionMNIST images

5.8.2.1.2 PCA Analysis (Principal Component Analysis) (Jolliffe and Cadima, 2016)

The PCA approach is a statistical procedure that reduces the number of dimensions in a dataset and retains the most information. It will be helpful to identify the complexity of the classification task. In (Hein and Audibert, 2005) the authors show that it is possible to estimate the intrinsic dimensionality of a submanifold from random samples (for example, intrinsic dimensionality of digit 4 in MNIST is estimated at 13).
PCA computes new variables called principal components through linear combinations of the original variables in a data set. This operation is equivalent to performing the singular value decomposition on the original data set, which results in an eigen decomposition of the covariance or correlation matrix.

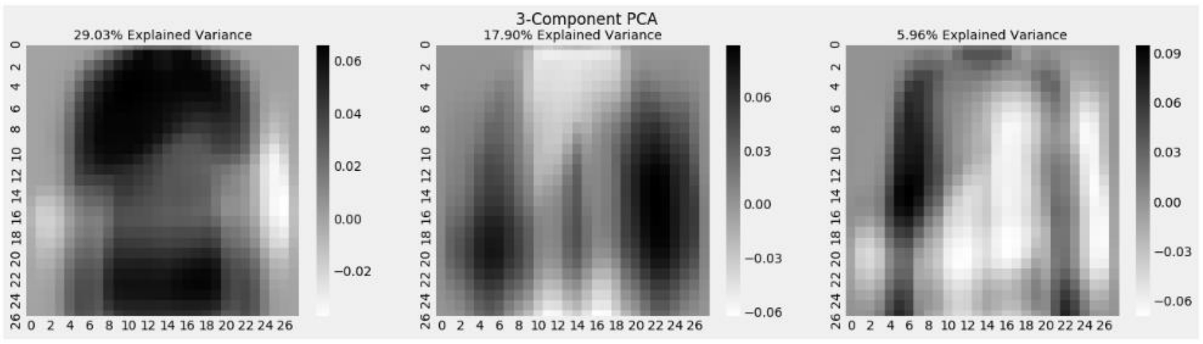


Figure 96. FashionMNIST PCA analysis results

Each component is an eigenvector. The first component looks like a T-shirt and a shoe. The second component is a trouser and a pullover. The third component looks like a pullover and an ankle boot.
The first dimension explains 29% of the pixel data, the second dimension explains approximately 17%, and the third dimension explains only 6%. Each additional dimension added to the PCA explains less variance of the data. If the number of components is 784, the cumulative explained variance would be 100.

5.8.2.1.3 t-SNE analysis (t-distributed Stochastic Neighbour Embedding) (Van der Maaten and Hinton, 2008)

t-SNE is an unsupervised non-linear dimensionality reduction technique for data exploration and visualising high-dimensional data.

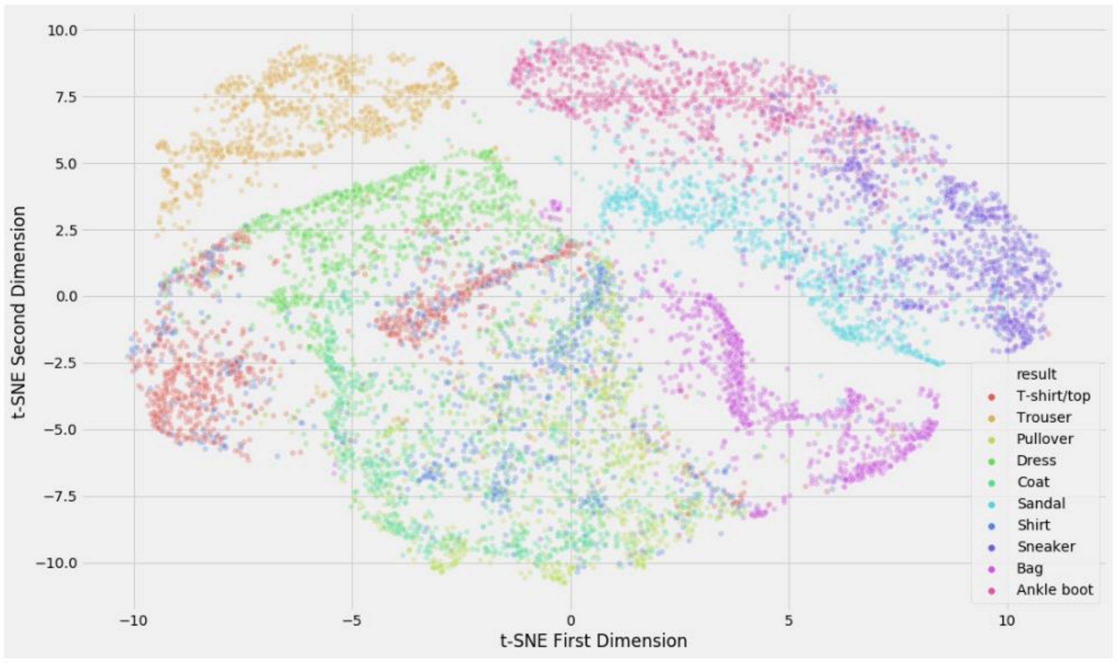


Figure 97 FashionMNIST t-sne analysis results

t-SNE is a technique for visualising high-dimensional data by giving each data point a location in a two or three-dimensional map. Suppose the data points from different classes are not separated in the t-SNE plot. In that case, the classes may not be easily distinguishable in the original high-dimensional space.

This result shows that for some classes (dress and coat, for example), the distance between input is minimal, and we can anticipate the point of attention in assessing the trained model.

PCA is a suitable method for dimensionality reduction, but it is not good if you want to reuse the result to make a decision boundary. T-SNE is a better approach for decision boundary, as we wish to point closer to the actual data to be closer to the decision surface.

5.8.2.2 **Model Architecture**

To analyse model architecture selection w.r.t generalisation performance, two families of network architecture, CNN and FCNN, have been identified (some comparisons on classification performance are available in (Kayed et al., 2020).

5.8.2.2.1 CNN architecture:

Convolutional neural networks generally comprise three main parts: convolutional layers, pooling layers and fully connected layers.

- The convolution layer uses filters that perform convolution operations as it scans the input concerning its dimensions. Its hyperparameters include the filter size and stride. It is used for feature extraction
- The pooling layer is a down-sampling operation typically applied after a convolution layer, which provides some spatial invariance. In particular, max and average pooling are unique

kinds of pooling where the maximum and average values are taken, respectively. Pooling layers avoid overfitting by pooling the convolved feature map, decreasing its dimensionality, reducing sampling, and retaining essential features.

- Finally, fully connected layers operate on a flattened input where each input is connected to all neurons. The fully connected layer is similar to the traditional ANN in which the neurons in each layer are connected to all the neurons in the next layer, and the final output layer outputs the final image classification based on the classifier.

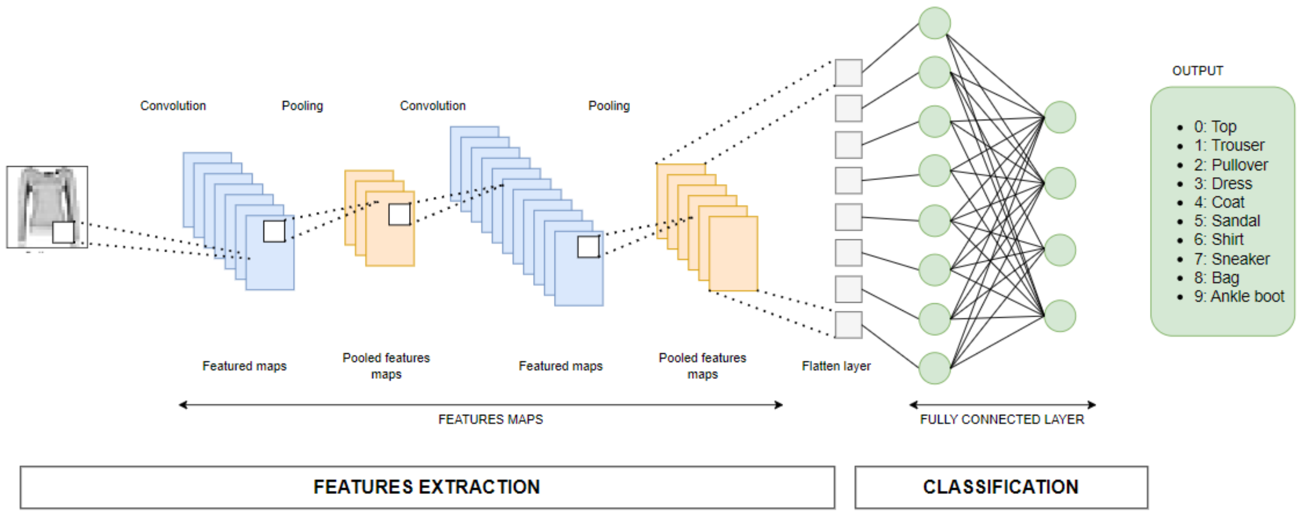
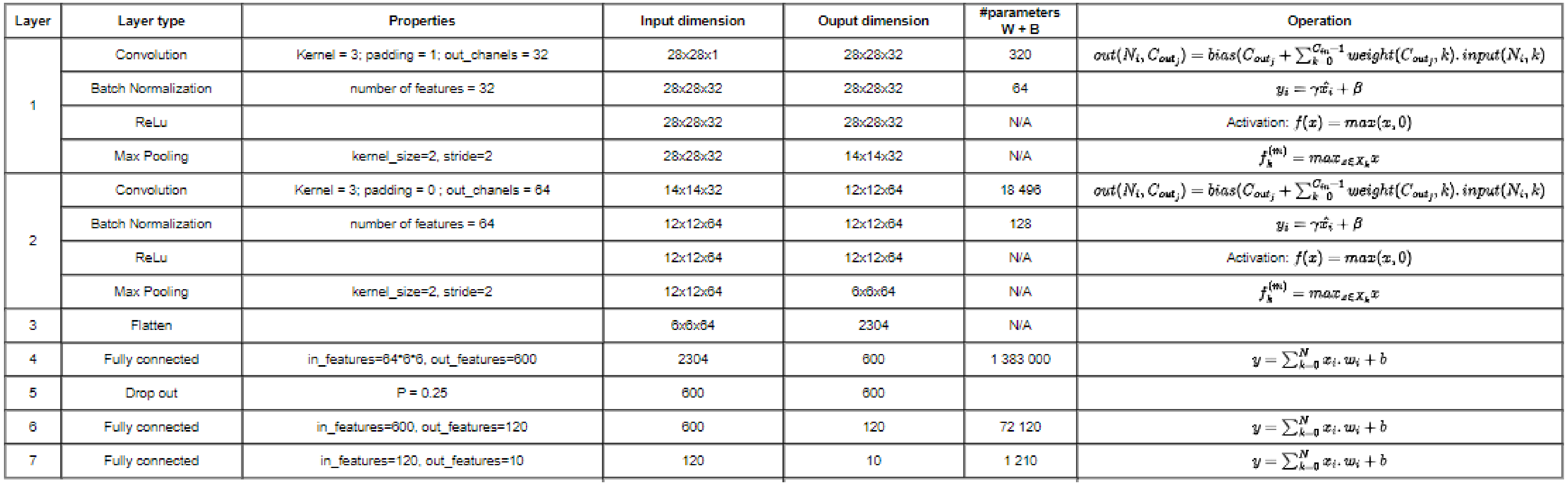


Figure 98. Representation of typical convolutional neural network

To illustrate detail architecture, details of one of CNN we are using have the following layers (cf. Table 38 for more information):

- Two Sequential layers, each consisting of the following layers
- Convolution layer that has kernel size of 3 * 3, padding = 1 (zero_padding) in 1st layer and padding = 0 in second one. The stride of 1 in both layers.
- Batch Normalisation layer.
- Activation function: ReLU.
- Max Pooling layer with kernel size of 2 * 2 and stride 2.
- Flatten out the output for the dense layer.
- 3 Fully connected layer with different in/out features including1 Dropout layer that has class probability p = 0.25.



Layer	Layer type	Properties	Input dimension	Ouput dimension	#parameters W + B	Operation
1	Convolution	Kernel = 3; padding = 1; out_chanels = 32	28x28x1	28x28x32	320	$out(N_i, C_{out_j}) = bias(C_{out_j} + \sum_{k\ 0}^{C_{in}-1} weight(C_{out_j}, k).input(N_i, k)$
	Batch Normalization	number of features = 32	28x28x32	28x28x32	64	$y_i = \gamma\hat{x}_i + \beta$
	ReLu		28x28x32	28x28x32	N/A	Activation: $f(x) = max(x, 0)$
	Max Pooling	kernel_size=2, stride=2	28x28x32	14x14x32	N/A	$f_k^{(m)} = max_{x \in X_k} x$
2	Convolution	Kernel = 3; padding = 0 ; out_chanels = 64	14x14x32	12x12x64	18 496	$out(N_i, C_{out_j}) = bias(C_{out_j} + \sum_{k\ 0}^{C_{in}-1} weight(C_{out_j}, k).input(N_i, k)$
	Batch Normalization	number of features = 64	12x12x64	12x12x64	128	$y_i = \gamma\hat{x}_i + \beta$
	ReLu		12x12x64	12x12x64	N/A	Activation: $f(x) = max(x, 0)$
	Max Pooling	kernel_size=2, stride=2	12x12x64	6x6x64	N/A	$f_k^{(m)} = max_{x \in X_k} x$
3	Flatten		6x6x64	2304	N/A	
4	Fully connected	in_features=64*6*6, out_features=600	2304	600	1 383 000	$y = \sum_{k=0}^{N} x_i.w_i + b$
5	Drop out	P = 0.25	600	600		
6	Fully connected	in_features=600, out_features=120	600	120	72 120	$y = \sum_{k=0}^{N} x_i.w_i + b$
7	Fully connected	in_features=120, out_features=10	120	10	1 210	$y = \sum_{k=0}^{N} x_i.w_i + b$

Table 38. Example of Convolutional neural network layers details

The following 3 architectures have been identified and will be tested w.r.t their generalisation ability with FashionMNIST classification task:

1. Three convolution 3x3 layers with 32, 64, and 128 filters and four fully connected layers with 1200, 600, 120, and 10 neurons. We have added a dropout layer between fully connected layers 1 and 2. ReLU is used as the activation function. The SoftMax function applied on the last layer gives the class prediction. The loss function is cross entropy. Here, we have 1 502 650 trainable parameters.
2. 4 convolution 3x3 layers with 32, 32, 64 and 128 filters and 4 fully connected layers with 1200, 600, 120 and 10 neurons respectively. Dropout layers were added between each convolutional layer, and before the last layer, 25% and 50% were added (the dropout acts as a regulariser; this should boost generalisation based on generalisation bounds theories). ReLU is used as an activation function, prediction of the class is given by the SoftMax function applied on the last layer. The loss function is cross entropy. Here, we have 1 226 090 trainable parameters.
3. There is only 1 convolutional layer with 32 filters, followed by 4 fully connected layers (1200, 600, 120, and 10 neurons). Dropout layers were added before the fully connected layers were added. ReLU is used as the activation function. The SoftMax function applied on the last layer gives the class prediction. The loss function is cross entropy. Here, we have 8 312 914 trainable parameters.

5.8.2.2.2 *Fully Connected Neural Network Architecture*

The neuron applies a linear transformation to the input vector through a weight matrix in fully connected layers. A non-linear transformation is then applied to the product through a non-linear activation function. Each pixel of the input picture is connected to the first layer of our neural network.

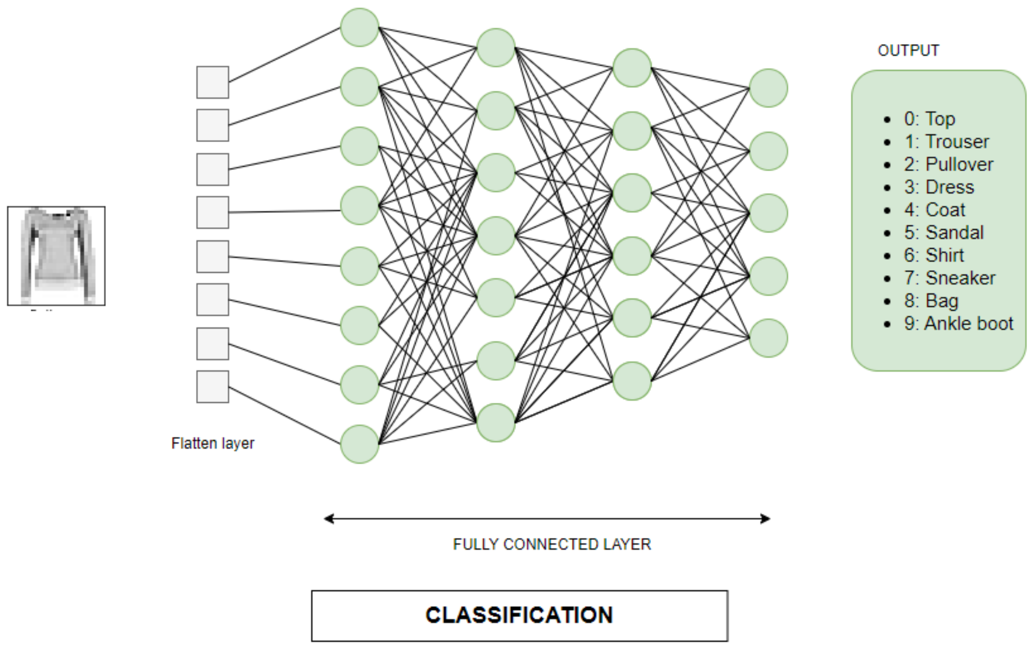


Figure 99. Representation of a fully connected neural network

An example of the architecture selected is detailed in Table 39.

Layer	Layer type	Properties	Input dimension	Ouput dimension	#parameters W + B	Operation
1	Fully Connected	in_features=784, out_features=256	784	256	320	$y = \sum_{k=0}^{N} x_i . w_i + b$
	Relu		256	256	64	Activation: $f(x) = max(x, 0)$
2	Fully Connected	in_features=512, out_features=512	256	128	N/A	$y = \sum_{k=0}^{N} x_i . w_i + b$
	Relu		128	128	N/A	Activation: $f(x) = max(x, 0)$
3	Fully Connected	in_features=512, out_features=10	128	64	18 496	$y = \sum_{k=0}^{N} x_i . w_i + b$
	Relu		64	64	N/A	Activation: $f(x) = max(x, 0)$
4	Fully Connected	in_features=512, out_features=10	64	32	18 496	$y = \sum_{k=0}^{N} x_i . w_i + b$
	Relu		32	32	N/A	Activation: $f(x) = max(x, 0)$
5	Fully Connected	in_features=512, out_features=10	32	10	18 496	$y = \sum_{k=0}^{N} x_i . w_i + b$
	log_softmax		10	10	18 496	$f(x_i) = \log\left(\frac{\exp(x_i)}{\sum_{j=0}^{k_{layer}} \exp(x_j)} \right)$

Table 39. Fully connected neural network layers details

The following 3 architectures have been identified and will be tested w.r.t their generalisation ability with FashionMNIST classification task:

1. 5 fully connected layers (neurones: 784, 512, 256, 128 and 10) using ReLU activation function. A dropout layer is added after the 1st layer. The SoftMax function applied on the last layer gives the class prediction. The loss function is cross entropy. We have here 567 434 trainable parameters.
2. 11 fully connected layers (neurones: 784, 1024, 2048, 4096, 2048, 1024, 512, drop, 256, drop, 128, 10) using the ReLU activation function. Dropout layers are identified with drop. The SoftMax function applied on the last layer gives the class prediction. The loss function is cross entropy. We have 22 474 890 trainable parameters here.
3. 11 fully connected layers (neurones: 784, 784, 784, 784, 784, 784, 512, drop, 256, drop, 128, 10) using the ReLU activation function. Dropout layers are identified with drop. The SoftMax function applied on the last layer gives the class prediction. The loss function is cross entropy. This architecture represents 3 644 634 trainable parameters.

The following table summarises the different architectures selected:

Type	Archi ID	# layers	Details
FCNN	1	5	784/drop/512/256/128/10
FCNN	2	11	784/1024/2048/4096/2048/1024/512/drop/256/drop/128/10
FCNN	3	11	784/784/784/784/784/784/512/drop/256/drop/128/10
CNN	1	9	3 Conv: 32, 64, 128 4 FC: 512, drop, 1200, 600, 120, 10
CNN	2	14	4 Conv: 32, 32, drop, 64, drop, 128, drop 4 FC: 512, drop, 1200, 600, 120, drop2, 10
CNN	3	6	1 Conv: 32 4 FC: 6272, drop, 1200, 600, 120, 10

Table 40. selected architectures, with same configuration: (Classifier = SoftMax), (Loss = cross-entropy), (Activation = ReLu)

5.8.2.3 Model Comparison

5.8.2.3.1 A priori evaluation

The following table summarises the A priori evaluation of the bounds applied to the selected architecture. The aim is to use generalisation bounds to provide quantifiable generalisation guarantees (LM-04).

Results are presented in Table 41. As mentioned previously, for deep neural networks, the values obtained drive vacuous bounds; in our case, with SoftMax multiclass cross entropy, we are expecting an average loss lower than 1 to secure an acceptable predicted probability of the correct class (in our use case, the average loss should be much lower than 2.3 as it corresponds to the case where the prediction is the same for all classes).

Vacuous bounds do not guarantee generalisation (LM-04), as we expect that the gap between the average loss function and the average loss on the full ODD remains small enough to ensure similar performance.

In classification tasks, the cross-entropy loss compares the predicted probability distribution (after applying log SoftMax) with the actual distribution. It measures the difference between the prediction and the correct label.

In a priori evaluation, we give a pessimistic value for the bound as it applies to all functions contained in the hypothesis class. Still, analysing the results could support identifying qualitative phenomena or drive architecture selection.

A priori generalisation bounds / epsilon = 0.05 (95% confidence)							
		CNN	CNN	CNN	FCNN	FCNN	FCNN
	Assumptions for A priori evaluation	1	2	3	1	2	3
Lin's Bound	spectral norm lower than 10 for FC layers Convolutional weights lower than 10	172	202	153	136	62218	11909
Jin's bound	Cover difference of the dataset						
Cantoni's bound	KL divergence upper bounded by a function of the number of parameters	55	45	306	21	829	134
McAllester's bound	KL divergence upper bounded by a function of the number of parameters	7	6	17	4	28	11
Seeger's bound							
Tolstikhin and Seldin's bound	KL divergence upper bounded by a function of the number of parameters	1664	1503	3918	1023	6438	2592
"Arora" bound	cushion is lower than 1/sqrt(#param)	9	21	4	3	13	13
Anthony's bound							
Neu and Lugosi's bound							
Feldman's bound	Stability w.r.t D_train is 0.2	11	11	11	11	11	11
Hardt's bound	gradient of the loss function over iterations is lower than 1, Norm of parameters is lower than 1, and the number of iterations is 30	1.8	1.8	1.8	1.8	1.8	1.8
Lei's bound	delta (data Decision Boundary variability) is lower than 0.5 and delta is less than 1	10	10	10	10	10	10
Kawaguchi's bound							

Table 41. A priori evaluation of generalisation bounds

Most bounds favour minimising the number of trainable parameters in our models a priori. Some bounds are not dependent on the architecture in this evaluation due to the dependence regarding the training phase results and the norms of the weights (in the a priori evaluation, we take a conservative assumption driving non-discriminative results).
Based on this table, the recommendation should be to select the first FCNN architecture followed by CNN 2 and 3.

5.8.2.3.2 Hyperparameters analysis

Before analysing the a posteriori results and the gap between training loss and test loss, we are presenting an analysis of the model's performance on the train and validation datasets. This step is usually known as the hyperparameters finetuning phase.
Hyperparameters, distinct from model parameters, aren't learned during the training phase; they're predetermined. Their precise configuration can profoundly impact the model's behaviour and performance. Fine-tuning models is key to better aligning with a specific task.

Hyperparameters (learning rate, number of layers, percentage of dropout, learning rate, batch size, …) determine how our model is structured. The fine-tuning phase aims to find the right combination of their values, which can help us find either the minimum (e.g., loss) or the maximum (e.g., accuracy). In this phase, we are using the train and validation dataset.

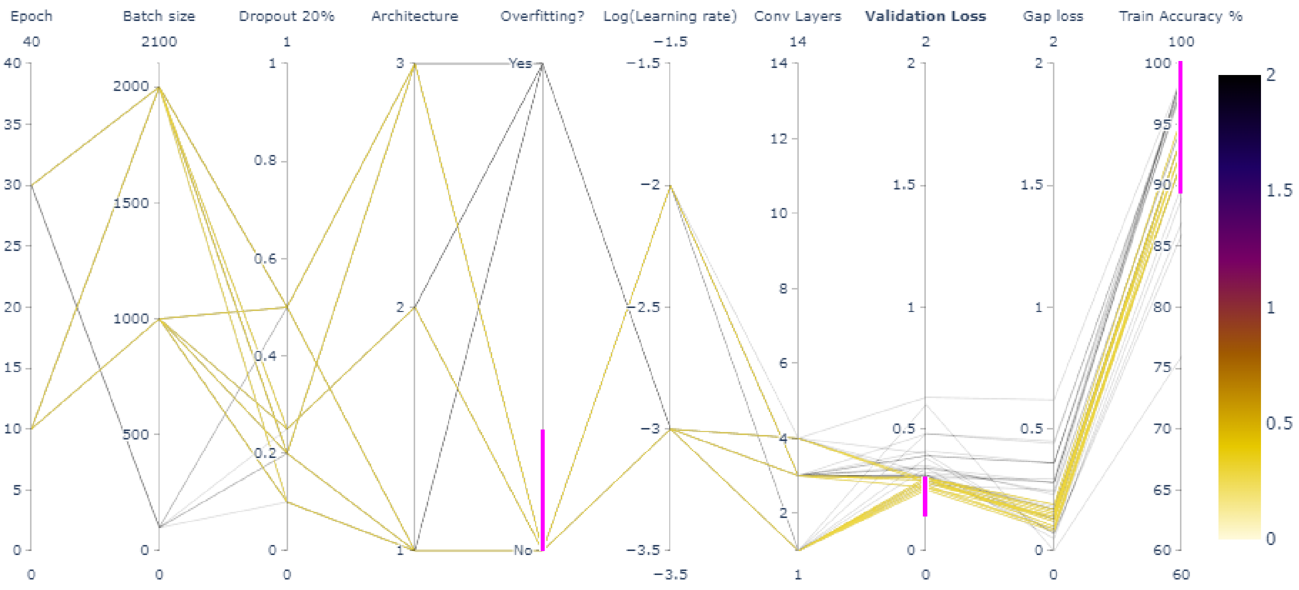


Figure 100. Evolution of validation loss for different hyperparameter combinations for convolutional neural networks identified

Figure 100 is a parallel coordinates plot showing the evolution of validation loss for different hyperparameter combinations for convolutional neural networks after the training phase. This shows that to achieve a minimum gap between training and validation loss, we should use a batch size larger than 200 to limit overfitting risk. Dropout has to be adapted regarding the number of convolutional layers, and we have an acceptable sensitivity of the architectures regarding performance. The three architectures seem to be capable of good generalisation.
Based on this analysis and regarding the generalisation gap estimation based on the test dataset, the third architecture, with a higher batch size, seems to be the best compromise.

Fully Connected Neural Network (FCNN)

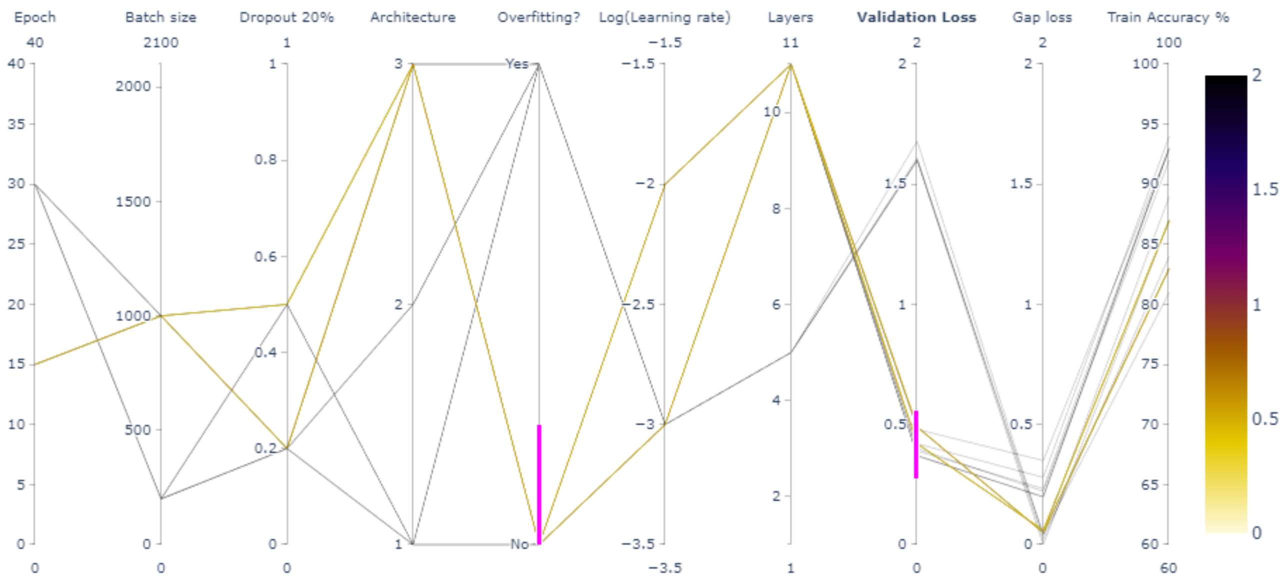


Figure 101: Evolution of validation loss for different hyperparameter combinations for fully connected neural networks identified

Figure 101 is a parallel coordinates plot showing the evolution loss computed on the validation dataset after neural network training. The coloured lines have been selected under two main constraints, which are no overfitting (it is vital to check learning curves evolution to identify potential overfitting securing low dependency of the trained model with regards to the sample selection. See (Dziugaite et al., 2020)) and validation loss lower than 0,5. It suggests that the trained model that will perform correctly on unseen data uses batch size 1000, 11 layers with architecture 3, running the optimisation to epoch 15 with 10^{-3} or 10^{-2} as the learning rate.
The generalisation gap estimated using the test dataset confirms that this configuration performs correctly on unseen data. Nevertheless, we can see we are not achieving the best accuracy.
In this case, the hyperparameter analysis is not aligned with the direction suggested by the bounds, where the first architecture (minimum layers) is supposed to promote generalisation.

The following section focuses on evaluating a posteriori generalisation bounds. This should provide tighter bounds and guarantees on the average loss value over the full domain.

5.8.2.3.3 *A posteriori evaluation*

As mentioned in the a-priori generalisation bounds evaluation, some depend on the algorithm and are linked with the convergence process. In this section, we will apply the theorems to the trained model, moving from a conservative assumption made on norms, for example, to values calculated after the training phase.

A posteriori generalisation bounds / epsilon = 0.05 (95% confidence)						
	CNN	**CNN**	**CNN**	**FCNN**	**FCNN**	**FCNN**
	1	2	3	1	2	3

Lin's Bound	11	19	101	1.77	147	2.17
Jin's bound	2.56	2.45	2.18	2.47	2.84	2.21
Cantoni's bound	14.4	14	27.8	9.8	66.8	20.3
McAllester's bound	1.8	1.8	2.9	1.2	4.9	2.4
Tolstikhin and Seldin's bound	6.7	6.5	17.4	3	48.6	11.4
"Arora" bound	9	21	4	3	13	13
Feldman's bound	11	11	11	11	11	11
Hardt's bound	1.62	1.54	1.59	1.74	1.67	1.6
Lei's bound	10	10	10	10	10	10

Table 42. A posteriori evaluation of generalisation bounds

All calculated bounds remain above a value which could secure the correct behaviour of our model on unseen data. The empirical generalisation gap measured on properly trained models (learning curves stable and no overfitting) is at most 0.4 (in our case, on average, we want the loss to be close to 0 and accurate class prediction probability high – 0.4 has been selected as a target to maximise the actual predicted class probability – this value depends on the loss function and the targeted function); it is a good indication that our models are not overfitting and are keeping good performance on unseen data.

In the context of cross-entropy loss, a lower value means that the predicted probabilities align well with the proper distribution of classes. It also can be insightful in understanding the type of errors the model makes. For example, a high loss might be associated with instances where the model is confident but wrong, potentially indicating areas for improvement.

Most of the bounds seem to orientate our choice of architecture: CNN1 and 3 and FCNN 1. Benchmarks on image classification models promote convolutional architectures, not only for their capacity to generalise but also for their faster convergence.

Regarding the upper bound values for the generalisation gap, it is difficult to conclude the capacity of the models to keep the same level of performance on the entire domain. We can only say that with a high probability, the generalisation gap is lower than a value that is not sufficiently small to conclude on generalisation performance. The objective LM-14 will be verified, but as the values are not tight enough, it does not provide confidence on performance similarity between train and test datasets.

Regularisation is used, models are not overfitting (LM-07), and if the robustness and stability (w.r.t. Training dataset (k-folds cross-validation) and input noise) are demonstrated, we have some indicators that the models will generalise adequately. This means that what we have measured in average on the different datasets will also be measured on the full ODD as our datasets are complete and representative of the entire domain.

It is important to note that while theoretical analysis generalisation bounds provide valuable insights, the practical success of deep learning models often relies on empirical performance and thorough validation of specific datasets and applications. Research continues to explore theoretical foundations to enhance our understanding of how and why deep neural networks generalise well in practice.

5.8.2.4 Architecture adaptation to decrease generalisation bounds

This section is dedicated to improving the model’s architecture regarding generalisation bounds theories to improve quantifiable generalisation guarantees (LM-04) during the generic pipeline's model design and adaptation step. The main drivers are:

- The number of trainable parameters
- The volume of data
- The output dimension of the different layers
- The spectral norms of layers and maximum values of the weight matrix

To improve the architecture, we have focused on reducing the number of parameters and minimising the number of fully connected layers and the convolution filters. This new architecture is called Fmnist, which will be improved in the rest of this section.

Thanks to these adaptations, the number of trainable parameters has been drastically reduced from 8,321,914 to 424,810 with the architecture detailed in Table 43

Table 43. This has been achieved by limiting the number of filters in the second convolution layer and reducing the number of fully connected layers.

Layer	Layer type	Input dimension	Output dimension	#parameters (W+B)
1	Convolution	28x28x1	28x28x32	320
	ReLU	28x28x1	28x28x32	
	Max Pooling		14x14x32	
2	Convolution	14x14x32	12x12x32	9248
	ReLu	12x12x32	12x12x32	
	Max Pooling		5x5x32	
3	Flatten	5x5x32	800	
4	Fully connected	800	512	410112
5	Fully connected	512	10	5130

Table 43. Updated convolutional neural network architecture layers details

The a priori and a posteriori (before and after training) are generalisation bounds applied to the improved architecture are summarised in Table 44. Limiting the number of parameters enables an apparent decrease in the different bounds. Some of the bounds calculated a posteriori are lower than one, which is a vast improvement. Nevertheless, considering the upper generalisation bound at 0.8 and with a training risk at 0.2, the theorem for bound number 3 says that with a probability of 95%, the true risk is lower than 1. In our case, normalised categorical cross entropy it means that the correct class probability is predicted with an average score of 37%. This allows other classes to be expected with higher probability, but it provides more confidence in the model behaviour on the entire domain. Improvements in the neural architecture to achieve a tighter bound have driven a lower optimisation and an empirical risk measurement moving from 0,23 to 0,29, meaning that the updated model predicts the correct class with lower confidence.

Finally, adapting a model’s architecture is a compromise between the trained model's performance and the expected level of statistical guarantee.

		Fmnist ref		Fmnist Improved	
		A priori evaluation	**A posteriori evaluation**	**A priori evaluation**	**A posteriori evaluation**
BOUND	001	172	11	20,2	6,4
	002		2,56		1,6
	003	55	14,4	4,4	0,8
	004	7	1,8	3,9	1,3
	006	1664	6,7	31	3,6
	007	9	9	4,4	4,4
	010	11	11	11,4	8,8
	011	1,8	1,62	1,8	0,54
	012	10	10	3,6	3,6
Loss	**Train**		0,14		0,24
	Test		0,23		0,29
Acc %	**Test**		91		89

Table 44. Comparison of bounds for the convolutional neural network before and after architecture optimisation regarding generalisation bounds minimisation

Fashion MNIST - Data augmentation

Another way to improve generalisation bounds results is to increase the volume of data used to train the model. This section presents the results obtained using data augmentation on the FashionMNIST use case. We use the first convolutional model as a reference to see the improvement in the generalisation bounds computation and the model behaviour on the test dataset.

For each image in the dataset, we have generated five new images with random modifications using rotation with a maximum angle of 10° and small translation. The results can be viewed in Figure 102.

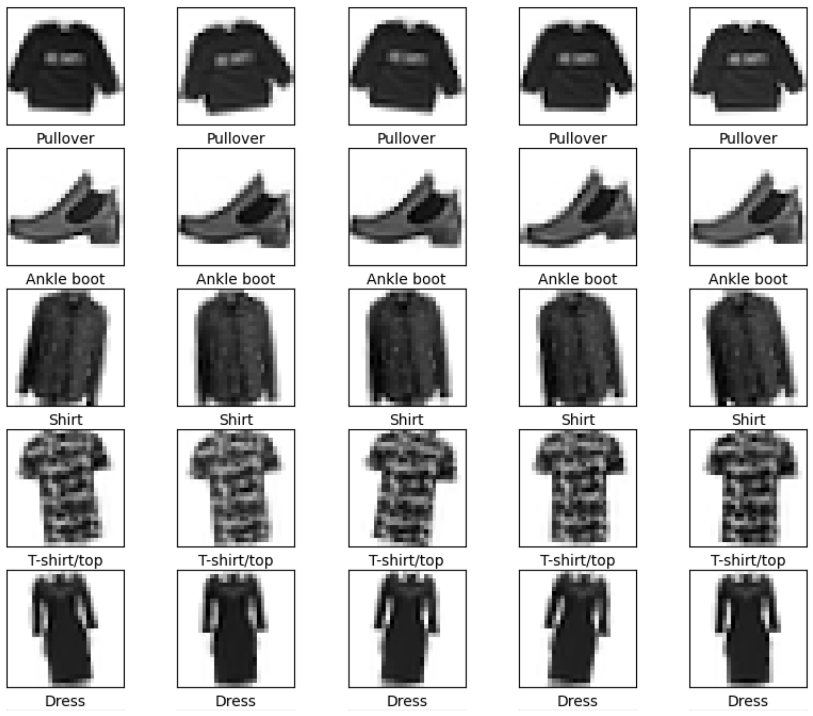


Figure 102. Results of data augmentation using the library ImageDataGenerator from Keras

The model was trained with data from the initial train dataset, which included up to 60,000 images. Starting from this limit, the training dataset has been enriched with augmented images. During this experiment, the balance between the different classes was constant to remove all unbalanced dataset effects from the results. The t-sne results can be visualised in Figure 103.

The objective here is to see the influence of the dataset's completeness (w.r.t. a wider ODD than in the initial expectations) on the generalisation capacity of a trained model (LM-04).

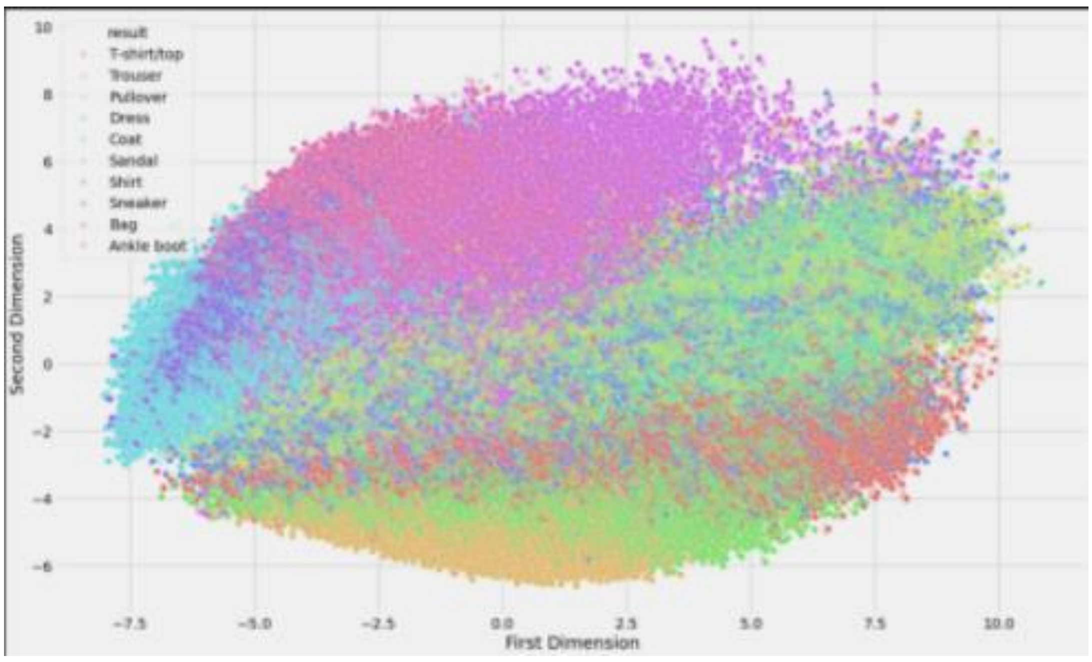


Figure 103 FashionMNIST t-sne analysis results

In Figure 104, we plot the evolution of the empirical risk measured on the test dataset, which has not been modified for a training dataset volume in the range of 10,000 images to 1,000,000 images.

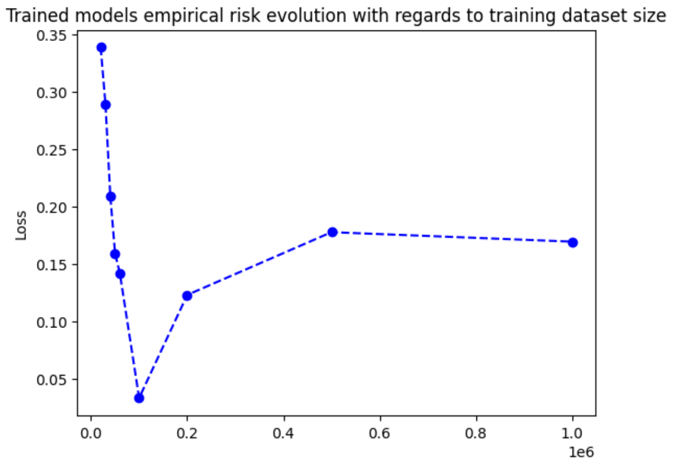


Figure 104: Trained model empirical risk evolution with regards to training dataset size

The empirical risk decreases initially, reflecting the expected convergence effect to the true risk when the data volume increases. A minimum of 100,000 images is achieved thanks to the 40,000 images added from data augmentation. After this value, we have more images augmented in the training dataset than the original one, the IID. The assumption is violated between the train and test set (the train dataset no longer represents the test dataset), explaining why the empirical risk is increasing again.
In this case, we can see that more than 100% of augmented images are degrading performance measured on the test dataset. This is mainly due to the IID assumption violation, which demonstrates that representativeness has to be a concern and should drive the way new data are generated.

5.8.2.5 **Main Takeaways**

As mentioned in (Dziugaite et al., 2020) it is difficult to find efficient bounds for all architectures and all framework cases. Nevertheless, this exercise confirms that CNN architecture is converging quicker than the FCNN which on top seems more to overfit.
We have not been able to use generalisation bounds to provide an acceptable performance level regarding trained model behaviour on unseen data. The overfitting detection was done through the bias-variance trade-off analysis. Most of the bounds selected are not data- and algorithm-dependent, but at the same time, they drive conservative assumptions and generate a high value for upper-bounding generalisation errors. In that context, the data management and the IID. The assumption is a key (Watson and Wright, 2021). We rely then on a bundle of clues regarding generalisation more than on simple measures providing tight statistical guarantees.
Note that, on the analysis performed so far, the toy use case of classification on FashionMNIST data shows only one part of the bound's applicability analysis. Based on one data type, our conclusions cannot apply straightforwardly to other kinds of data. To do so, further study on another small dataset of another type can be considered, in addition to the analyses on the more complex use cases selected for the project, for which the following section shows the initial analyses of the corresponding data

and models, before applying the same process analysing the generalisation bounds (by the final phase of the project).

5.8.3 Aviation use-cases exploration

In this section, we first analyse the models and data within the different use cases: ATC-STT, AVI and ACAS Xu. The objective is to understand how these use cases are built, analyse the provided models, determine how these have been designed with the target application objectives, and confront them with other alternatives that can be used to help achieve better results. In the second step, the selected generalisation bounds are applied to the aviation use cases to provide an apriori and a posteriori assessment of the generalisability of the developed models per use case. This analysis targets the means to address the EASA's Concept paper objectives (cf. section 1.5.1.2) concerning the learning management (LM): LM-03, LM-04, LM-07, LM-10 and LM-14. Hence, the results of the different experimentations are discussed in line with verifications that are needed during the learning management steps. Further, the conclusions will help refine the *"generalisation"* verification procedure and its definition.

5.8.3.1 **Use cases evaluation – Pipeline Analysis & Application Objectives**

After the bounds applicability and behaviour analysis on the toy-use cases described above, similar scenarios are applied for the selected use cases of the MLEAP project. The goal of the experiments is three folds:

a) Set up the ML evaluation pipeline using different models and data sources. This will help us to compare the performances of the trained models on various data sets while performing the same task to understand their differences and how different architectures would behave to solve the same problem;
b) The whole ML/DL development and evaluation pipeline used to develop each of the use cases will be analysed, focussing on the different research questions previously described (cf. section 5.8.1), where the entire process (from model design and choice of performance to evaluation and acceptability analysis) will be dissected to identify the common errors that have been repeated in these UCs and that have resulted in a gap between experimentation and the industrial objective in terms model abilities to generalise;
c) Finally, to help achieve better results, explore some alternatives based on the state-of-the-art and experimentally verify their applicability for each use case.

5.8.3.1.1 ATC-STT:

As part of the MLEAP project, this use case aims to correctly transcribe the exchanges, whatever the speakers' accents. Subsequently, the detection of the callsigns (used to identify aircraft and correspond to their registration) will be further analysed to more easily recognise the speakers and their roles.

5.8.3.1.1.1 Dataset's analysis

Various data with multiple accents is used to analyse the different models. As described in section 3.3, AIRBUS has an actual database of air traffic control (ATC) exchanges as well as Automatic Terminal Information Service (ATIS)[127] messages (Delpech et al., 2018). In addition to this, we used several open-source datasets for experimentation and to compare the models' behaviour in several accents. The statistics related to the complete datasets are provided in Table 45.

Dataset	Reference	# samples	Duration	Accent(s)
ATCO2[128]	(Kocour et al., 2021)	560	1 h 06 min	Czech, Slovak, German, French, Australian
UWB[129]	(Šmídl, 2011)	2 657	20 h 35 min	Czech
ATCOSIM[130]	(Hofbauer et al., 2008)	10 078	10 h 42 min	German, French,
AIRBUS	-	6826	5h	French

Table 45 Statistics of the ATC-STT datasets, used so far, in the experimental part for the generalisation properties analysis.

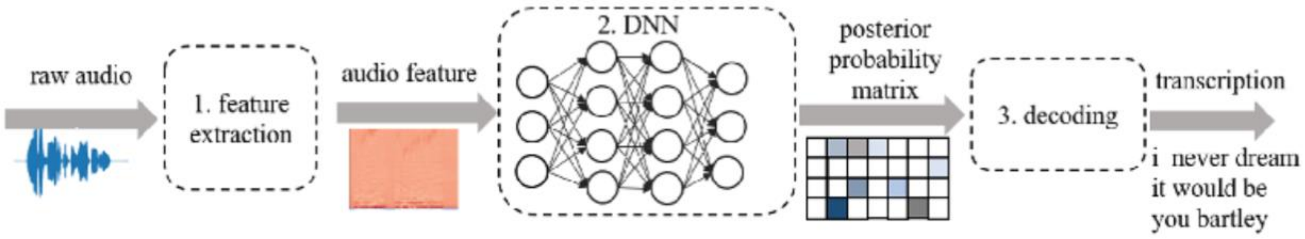
The datasets in the table above are used as follows:

- AIRBUS: This dataset is used for fine-tuning the open-source models (transformer-based described in Figure 107). The samples are split into 10% for testing, 10% for validation, and 80% for training.
- Open-source data (ATCO, UWB, ATCOSIM) were used in the same test data to evaluate Airbus's model (Kaldi-based described in Figure 106).

5.8.3.1.1.2 Models' description

For the transcription task, the model provided by Airbus uses ML and DL with KALDI's architectures as shown in Figure 105. This model is made of three elements:

1) An extractor which creates a vector for each time step of the audio, using the MFCCs (Mel-Frequency Cepstral Coefficients)[131], providing concatenated vectors of audio features ;
2) These vectors are then fed into a deep neural network (DNN), which returns a probability matrix ;
3) This matrix is then used by a decoder which returns the most probable transcription. This decoder is a Hidden Markov Model (HMM) that reconstructs a sequence of words. It contains a total of 810,224 different states and 2,278,755 transitions (Upadhyaya et al., 2017).



[127] ATIS messages are broadcast on airport frequencies and contain useful information for pilots, including meteorological data
[128] https://www.atco2.org/
[129] https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0001-CCA1-0
[130] https://www.spsc.tugraz.at/databases-and-tools/atcosim-air-traffic-control-simulation-speech-corpus.html
[131] The MFCC extraction method approximates the way the human auditory system works (Prabakaran and Sriuppili, 2021)

Figure 105 Architecture of an STT pipeline in KALDI (also used by the Airbus model).

In addition to the models from the Airbus internal project, we use two different DL-based models. Namely *transformers*[132], these models are available for download in the HuggingFace[133] API. The transformer architecture was first introduced by (Vaswani et al., 2017). It was initially designed to perform translation tasks. It consists of an encoder representing the input text sequences and then a decoder providing the output corresponding to this representation. In this project, we used two transformers trained to transcribe air traffic control exchanges. The aim is to compare the behaviour of different ML architectures while solving the same problem to tackle their generalisation limitations further. The used transformers are based on the Wav2Vec2 architecture (Baevski et al., 2020), described in Figure 106. Trained either on the UWB dataset or on the UWB and ATCOSIM datasets, they have been proposed by Facebook and designed initially for conventional automatic speech recognition.
As described in Figure 106, the Wave2Vec2 method extracts a vector representation of the audio using several convolution blocks (CNN) applied to the raw waveform input. A transformer layer is then used, which returns the exchange transcript. Hence, the transformer architecture is used for automatic speech recognition, assuming the task aims to translate an audio file into a corresponding text.

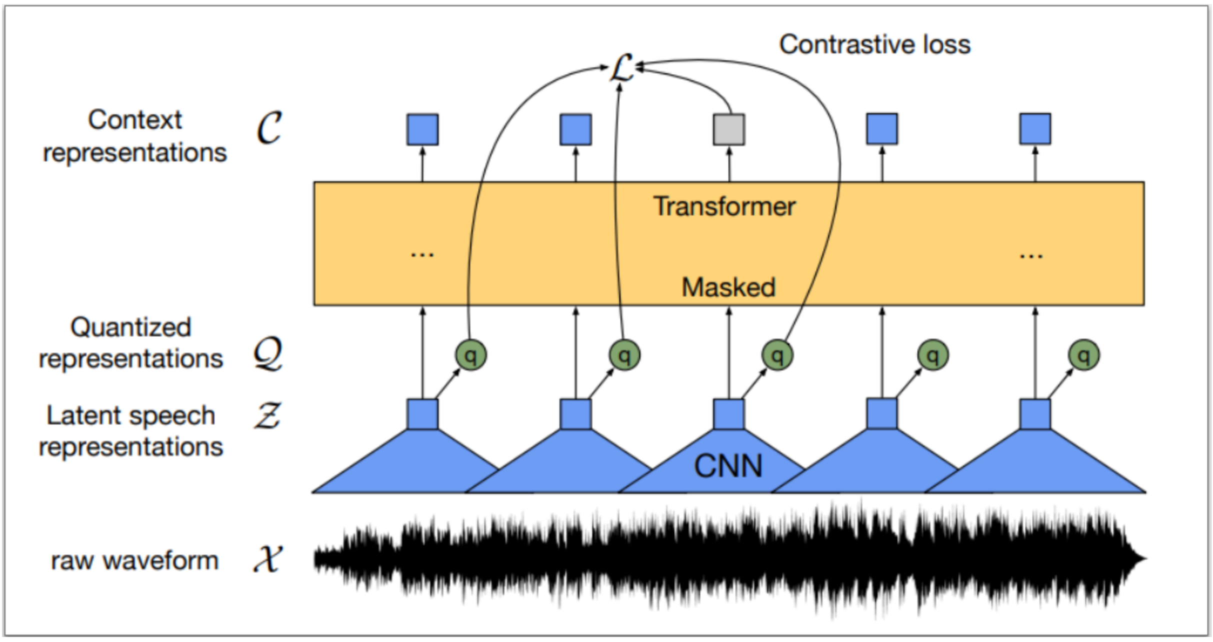


Figure 106 Architecture of the Wav2Vec2 model (Baevski et al., 2020)

Another DL model Airbus uses is based on Whisper[134](Radford et al., 2022). This model is designed using transformers to transcribe audio exchanges. However, it was not considered as an option, as it did not meet the objective of being able to be used in a real-time framework[135], and its results were limited compared to those of the KALDI-based architecture (cf. Figure 105).

[132] https://huggingface.co/docs/transformers/index
[133] https://huggingface.co/models?pipeline_tag=automatic-speech-recognition&sort=downloads&search=atc
[134] https://openai.com/research/whisper
[135] Whisper is based on an Attention-based Encoder-Decoder (AED) architecture. In the scope of the ATC-STT, it was trained to perform the task, however, due to its latency in inference time, this has not met the objectives of the project.

For both architecture (KALDI and Wave2Vec, the a priori evaluation based on bounds theories applied to the DNN model provides a vacuous statistical guarantee concerning the generalisation gap. This is mainly driven by the vast number of parameters in the model, which could not be counterbalanced by a reasonable volume of features extracted from the training dataset.
The generalisation bounds approach is then inefficient in this case in providing a priori (before training the model) statistical guarantees on the generalisation gap we can expect when the model is trained.

5.8.3.1.1.3 Evaluation measures

In the automatic speech recognition state-of-the-art (Errattahi et al., 2018), several evaluation metrics can be used. The most common one is the Word Error Rate (WER), which measures the percentage of incorrect words over the entire transcription, also called the Word Information Lost (WIL), which approximates the dependency between the actual exchange (target sentence) and its transcription (predicted by the STT model).

$$WER = \frac{n_S + n_D + n_I}{n_T}$$

Where: n_S, n_D, n_I, and n_T, are respectively, the number of substituted words, number of deleted words, number of inserted words, and the output sentence length in terms of number of words.
In the scope of the MLEAP project, and concerning the target objective of the ATC-STT use case in the Airbus project, we rely on a WER $\leq 10\ \%$, corresponding to an error rate that is likely to result in more wide-spread acceptance among users (Green et al., 2021). Hence, the objective is to achieve a minimum value such that closer the WER is to 0, the more similar the prediction is to the expected sentence (label). We use this metric in the rest of these experiments, as it allows us to correctly evaluate the performance of an STT model in correlation with the performance objective of the use case defined in the Airbus project. Three other variants of this metric will also be used:

- *The match error rate:* $MER = \frac{n_S+n_D+n_I}{n_T+n_I}$
- *The word information preserved:* $WIP = \frac{{n_C}^2}{(n_C+n_S+n_D)(n_C+n_S+n_I)}$
- *The word information lost:* $WIL = 1 - WIP$

5.8.3.1.1.4 Results

After a preliminary evaluation in which seven different transformer-based models were compared (cf. Table), we selected two baselines for the remainder of the experimental assessment. In addition to the Airbus model, a KALDI-based model, Table 46 shows the selected models for the preliminary analysis.

Model	Approach	Source	Training Dataset
AIRBUS	KALDI		AIRBUS dataset
DL 1	Transformers	HuggingFace[136]	UWB and ATCOSIM

With an inference time longer than what can be accepted, since it causes a time delay that prevents the system from finishing transcribing a sentence before another one be spoken.
[136] https://huggingface.co/Jzuluaga/wav2vec2-xls-r-300m-en-atc-uwb-atcc-and-atcosim

DL 2	Transformers	HuggingFace[137]	UWB
DL 3	Transformers	HuggingFace[138]	UWB and ATCOSIM
FT 3.1	Transformers	Finetuned DL 3 during 10 epochs	UWB, ATCOSIM and AIRBUS dataset
FT 3.2	Transformers	Finetuned DL 3 during 50 epochs	UWB, ATCOSIM and AIRBUS dataset
DL 4	Transformers	HuggingFace[139]	UWB
FT 4	Transformers	Finetuned DL 4 during 50 epochs	UWB and AIRBUS dataset

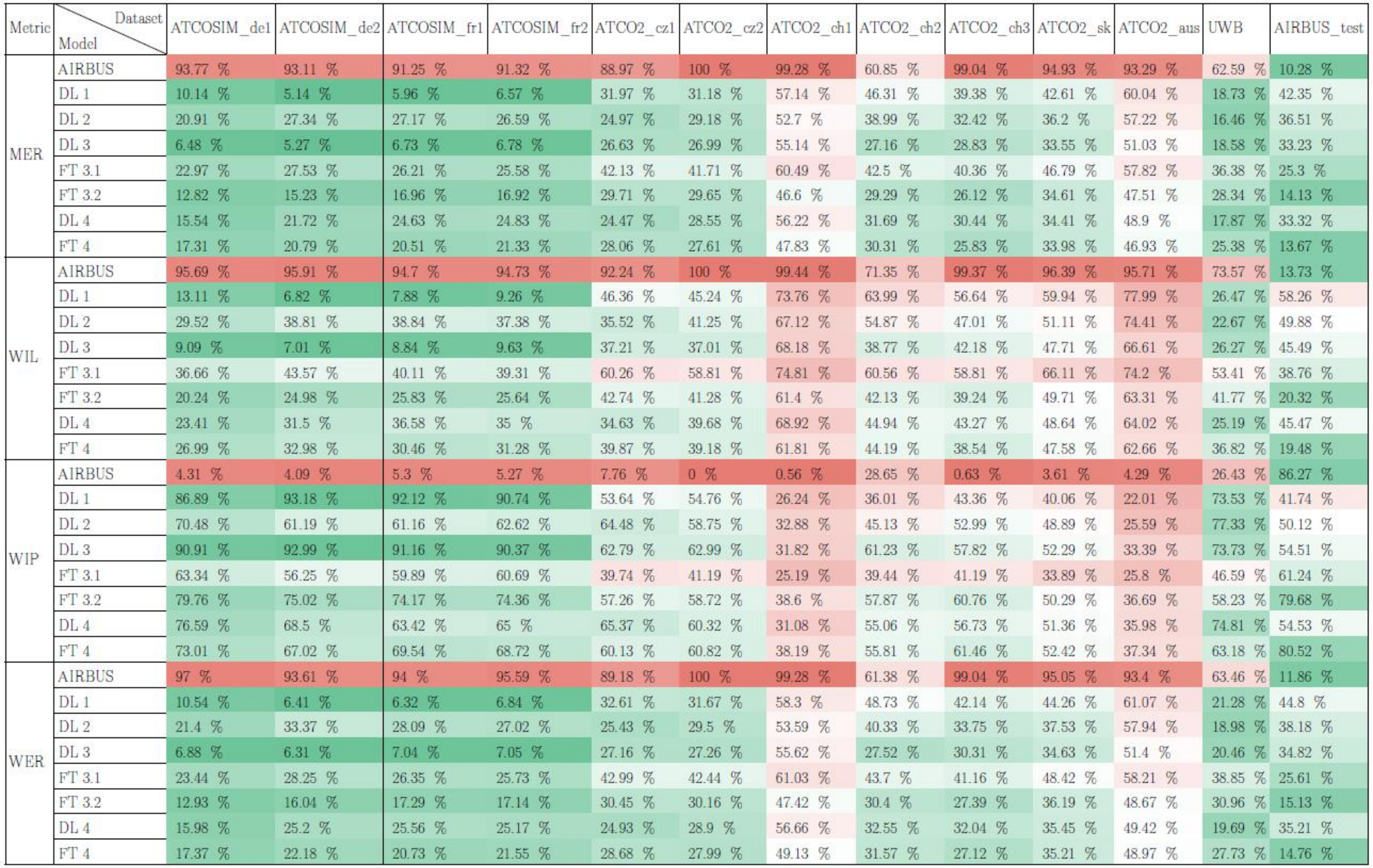
Table 46 Details of the different baselines used to select best candidates from the ATC-STT state-of-the-art

To select better candidates, these models have been evaluated on different parts of the open-source datasets (cf. Table 45) and the AIRBUS data. To do so, the transformer models were pre-trained using the corresponding open-source datasets, such as UWB and ATCOSIM; these models were then used for fine-tuning using the Airbus data sets. The objective is to select models that perform best on the majority of test data (cf. Table 47) based on the analysis of the WER (model with minimum value), while performing as closely as possible to the AIRBUS model on the AIRBUS test data partition, w.r.t different evaluation measures, and different accents.

[137] https://huggingface.co/Jzuluaga/wav2vec2-xls-r-300m-en-atc-uwb-atcc

[138] https://huggingface.co/Jzuluaga/wav2vec2-large-960h-lv60-self-en-atc-uwb-atcc-and-atcosim

[139] https://huggingface.co/Jzuluaga/wav2vec2-large-960h-lv60-self-en-atc-uwb-atcc



Metric	Model \ Dataset	ATCOSIM_de1	ATCOSIM_de2	ATCOSIM_fr1	ATCOSIM_fr2	ATCO2_cz1	ATCO2_cz2	ATCO2_ch1	ATCO2_ch2	ATCO2_ch3	ATCO2_sk	ATCO2_aus	UWB	AIRBUS_test
MER	AIRBUS	93.77 %	93.11 %	91.25 %	91.32 %	88.97 %	100 %	99.28 %	60.85 %	99.04 %	94.93 %	93.29 %	62.59 %	10.28 %
	DL 1	10.14 %	5.14 %	5.96 %	6.57 %	31.97 %	31.18 %	57.14 %	46.31 %	39.38 %	42.61 %	60.04 %	18.73 %	42.35 %
	DL 2	20.91 %	27.34 %	27.17 %	26.59 %	24.97 %	29.18 %	52.7 %	38.99 %	32.42 %	36.2 %	57.22 %	16.46 %	36.51 %
	DL 3	6.48 %	5.27 %	6.73 %	6.78 %	26.63 %	26.99 %	55.14 %	27.16 %	28.83 %	33.55 %	51.03 %	18.58 %	33.23 %
	FT 3.1	22.97 %	27.53 %	26.21 %	25.58 %	42.13 %	41.71 %	60.49 %	42.5 %	40.36 %	46.79 %	57.82 %	36.38 %	25.3 %
	FT 3.2	12.82 %	15.23 %	16.96 %	16.92 %	29.71 %	29.65 %	46.6 %	29.29 %	26.12 %	34.61 %	47.51 %	28.34 %	14.13 %
	DL 4	15.54 %	21.72 %	24.63 %	24.83 %	24.47 %	28.55 %	56.22 %	31.69 %	30.44 %	34.41 %	48.9 %	17.87 %	33.32 %
	FT 4	17.31 %	20.79 %	20.51 %	21.33 %	28.06 %	27.61 %	47.83 %	30.31 %	25.83 %	33.98 %	46.93 %	25.38 %	13.67 %
WIL	AIRBUS	95.69 %	95.91 %	94.7 %	94.73 %	92.24 %	100 %	99.44 %	71.35 %	99.37 %	96.39 %	95.71 %	73.57 %	13.73 %
	DL 1	13.11 %	6.82 %	7.88 %	9.26 %	46.36 %	45.24 %	73.76 %	63.99 %	56.64 %	59.94 %	77.99 %	26.47 %	58.26 %
	DL 2	29.52 %	38.81 %	38.84 %	37.38 %	35.52 %	41.25 %	67.12 %	54.87 %	47.01 %	51.11 %	74.41 %	22.67 %	49.88 %
	DL 3	9.09 %	7.01 %	8.84 %	9.63 %	37.21 %	37.01 %	68.18 %	38.77 %	42.18 %	47.71 %	66.61 %	26.27 %	45.49 %
	FT 3.1	36.66 %	43.57 %	40.11 %	39.31 %	60.26 %	58.81 %	74.81 %	60.56 %	58.81 %	66.11 %	74.2 %	53.41 %	38.76 %
	FT 3.2	20.24 %	24.98 %	25.83 %	25.64 %	42.74 %	41.28 %	61.4 %	42.13 %	39.24 %	49.71 %	63.31 %	41.77 %	20.32 %
	DL 4	23.41 %	31.5 %	36.58 %	35 %	34.63 %	39.68 %	68.92 %	44.94 %	43.27 %	48.64 %	64.02 %	25.19 %	45.47 %
	FT 4	26.99 %	32.98 %	30.46 %	31.28 %	39.87 %	39.18 %	61.81 %	44.19 %	38.54 %	47.58 %	62.66 %	36.82 %	19.48 %
WIP	AIRBUS	4.31 %	4.09 %	5.3 %	5.27 %	7.76 %	0 %	0.56 %	28.65 %	0.63 %	3.61 %	4.29 %	26.43 %	86.27 %
	DL 1	86.89 %	93.18 %	92.12 %	90.74 %	53.64 %	54.76 %	26.24 %	36.01 %	43.36 %	40.06 %	22.01 %	73.53 %	41.74 %
	DL 2	70.48 %	61.19 %	61.16 %	62.62 %	64.48 %	58.75 %	32.88 %	45.13 %	52.99 %	48.89 %	25.59 %	77.33 %	50.12 %
	DL 3	90.91 %	92.99 %	91.16 %	90.37 %	62.79 %	62.99 %	31.82 %	61.23 %	57.82 %	52.29 %	33.39 %	73.73 %	54.51 %
	FT 3.1	63.34 %	56.25 %	59.89 %	60.69 %	39.74 %	41.19 %	25.19 %	39.44 %	41.19 %	33.89 %	25.8 %	46.59 %	61.24 %
	FT 3.2	79.76 %	75.02 %	74.17 %	74.36 %	57.26 %	58.72 %	38.6 %	57.87 %	60.76 %	50.29 %	36.69 %	58.23 %	79.68 %
	DL 4	76.59 %	68.5 %	63.42 %	65 %	65.37 %	60.32 %	31.08 %	55.06 %	56.73 %	51.36 %	35.98 %	74.81 %	54.53 %
	FT 4	73.01 %	67.02 %	69.54 %	68.72 %	60.13 %	60.82 %	38.19 %	55.81 %	61.46 %	52.42 %	37.34 %	63.18 %	80.52 %
WER	AIRBUS	97 %	93.61 %	94 %	95.59 %	89.18 %	100 %	99.28 %	61.38 %	99.04 %	95.05 %	93.4 %	63.46 %	11.86 %
	DL 1	10.54 %	6.41 %	6.32 %	6.84 %	32.61 %	31.67 %	58.3 %	48.73 %	42.14 %	44.26 %	61.07 %	21.28 %	44.8 %
	DL 2	21.4 %	33.37 %	28.09 %	27.02 %	25.43 %	29.5 %	53.59 %	40.33 %	33.75 %	37.53 %	57.94 %	18.98 %	38.18 %
	DL 3	6.88 %	6.31 %	7.04 %	7.05 %	27.16 %	27.26 %	55.62 %	27.52 %	30.31 %	34.63 %	51.4 %	20.46 %	34.82 %
	FT 3.1	23.44 %	28.25 %	26.35 %	25.73 %	42.99 %	42.44 %	61.03 %	43.7 %	41.16 %	48.42 %	58.21 %	38.85 %	25.61 %
	FT 3.2	12.93 %	16.04 %	17.29 %	17.14 %	30.45 %	30.16 %	47.42 %	30.4 %	27.39 %	36.19 %	48.67 %	30.96 %	15.13 %
	DL 4	15.98 %	25.2 %	25.56 %	25.17 %	24.93 %	28.9 %	56.66 %	32.55 %	32.04 %	35.45 %	49.42 %	19.69 %	35.21 %
	FT 4	17.37 %	22.18 %	20.73 %	21.55 %	28.68 %	27.99 %	49.13 %	31.57 %	27.12 %	35.21 %	48.97 %	27.73 %	14.76 %

Table 47 Results of the seven state-of-the-art models based on transformers, and evaluated on different datasets to select best candidates for the rest of the ATC-STT experiments

These include two best transformer-based models, both based on the Wav2Vec2 architecture (Baevski et al., 2020) and for which we refer as transformer-based (1) and transformer-based (2), where the (1) is pre-trained on both UWB and ATCOSIM datasets and the (2) is trained only on the UWB dataset. Table 48 shows the KALDI-based model's performance and the transformer-based baselines (1) and (2).

Datasets / Model	AIRBUS	ATCO2
KALDI-based	11.43 %	91.05 %
transformer-based (1)	43.70 %	45.54 %
transformer-based (2)	34.63 %	36.27 %

Table 48 Average WER values of the different models on both AIRBUS and ATCO2 datasets.

The results of the KALDI-based model are better on the AIRBUS test data, but the same model performs poorly on the ATCO2 test data. On the other hand, the transformer-based models seem to perform well and in the same order on the different datasets. We can notice a significant difference in performance between the transformer-based and KALDI-based models. This is due to several factors. First, the AIRBUS dataset was collected in French airports only, and the materials used for this end should be the same for all the airports where the utterances were recorded but different from those used for the ATCO2 exchanges recording. This difference, which is often imperceptible to a human ear, is a significant detail and could be a bias to the trained model. Therefore, there is an essential gap in the performance of the same KALDI-based model evaluated in two different datasets from different sources. The second unneglectable factor is the very different architecture of the models. The KALDI-based model uses a hybrid approach where a combination of non-ML and ML components are optimised together (HMM & DNN), using a statistic method to compute the features matrix of the input data. At the same time, both transformer-based models use a significant Wav2Vec2 architecture, which is a DL model including complex layers of over 314M trainable parameters.
In addition, the KALDI-based model was trained by Airbus on English exchanges, including only a French accent; other accents are present in the ATCO2 data (cf. Table 45). Therefore, the model is not robust enough to handle accent variation, and its use is limited. Another source of limitation is due to the many proper nouns present in the exchanges of ATCO2 and AIRBUS datasets. The KALDI-based model knows the names of several French airports. Hence, it performs better than the transformer-based models. However, when it comes to the ATCO2 dataset, several entities, such as airport names, waypoints, and airlines, are unknown to the KALDI-based model, which explains, in part, the drop in performance.

Comparing both transformer-based models, we can see that (2) is better than (1) in the AIRBUS and ATCO2 datasets. Note that the transformer-based (2) has not been trained on the ATCOSIM dataset, which contains simulated air traffic control exchanges. The audio is, therefore, very noiseless, unlike real exchanges. To improve the results of both transformer-based models, we performed some fine-tuning on the same training set of the AIRBUS dataset (on which the KALDI-based model was trained).

The aim is to add contextual details related to the French airport and the collected recordings (accent, noise, … etc.).

For both transformer-based models (1) and (2), a batch size of 16 is used to fine-tune the models during 50 epochs (i.e., a total of 21,300 training steps). Table 49 shows the performances of the different models, including the fine-tuned transformers.

Dataset / **Model**		**AIRBUS**	**ATCO2**
KALDI-based		11.43 %	91.05 %
transformer-based (1)	Original	43.70 %	45.54 %
	Fine-tuned	15.13 %	28.75 %
transformer-based (2)	Original	34.63 %	36.27 %
	Fine-tuned	14.76 %	29.85 %

Table 49 Comparison of the transformer-based models' performances, in terms of WER measure, before and after fine-tuning on the AIRBUS training dataset. The evaluation is then performed on both the AIRBUS and ATCO2 datasets.

In Table 49, we can notice that the fine-tuning of the transformer-based models has considerably improved the performances, reducing the error rate in both test datasets AIRBUS and ATCO2, approaching the performances of the KALDI-based model by Airbus, and that is closer to the targeted objective of $\mathrm{WER} \leq 10$ by the internal project, referring to the state-of-the-art best performances.
A posteriori (applied after the model training step) evaluation of the generalisation bounds also provides vacuous statistical guarantees concerning the maximum gap expected in average on the full ODD loss functions and the one obtained on the training dataset. We can verify that the anticipated generalisation bounds are valid regarding the loss obtained on the test dataset (LM-14) and the KALDI-based model, which is not generalised correctly. It demonstrates that generalisation bounds obtained to answer LM-04 need to be tight enough and cannot be used in this case to guarantee the generalisation of a trained model.

5.8.3.1.1.5 Pipeline Analysis

In the context of automatic speech transcription for ATC, we have mentioned that the target performance objective is $\mathrm{WER} \leq 10$, in concordance with the ordinary state of the art. Besides, the Airbus project has set for air traffic control purposes, at the system level, the four main objectives:

- The system must be able to be used in real time;
- The system must be robust to different accents ;
- The system must be able to identify callsigns ;
- The system must be able to determine the role of the speakers.

At this stage of the MLEAP project, we focus on the generalisation-related objectives[140]. Thus, the objectives above can be split into sub-objectives to be achieved gradually at the ML/DL component. Starting with the performance of the textual transcriptions produced by the trained model, with a target $\mathrm{WER} \leq 10$. To achieve these preliminary goals, in the following paragraphs, we compare the development pipelines implemented for the Airbus (KALDI-based) model, as well as for the open source (transformer-based) models.

KALDI-based model

In the Airbus project, the models were developed to be embedded rapidly in the aircraft as a dedicated option for pilots to facilitate exchanges with air traffic controllers. Hence, the most important criterion is to carry out the task in real time. To achieve this, the time taken by the system to transcribe an exchange must be shorter than the exchange itself (the duration of the input utterance). Therefore, a KALDI-based approach was adopted, as it makes predictions more rapidly. The model is then trained to minimise the WER over words. The training objective also considers the same granularity level of information to be learned if the datasets associate one or more sentences per exchange.
However, there are several limitations concerning the model predictions. First, the transcription made by the KALDI-based model does not meet the targeted objective perfectly. More precisely, the current error rate of 11.43% is higher than the targeted 10%. The AIRBUS dataset comes merely from French airports. Hence, the model's domain does not correspond to that of the ATCO2 dataset. Ultimately, Airbus' ATC-STT use case aims to have a system that performs well regardless of the speaker's accent. So far, only French and Chinese accents have been included in the analysis. More accents should be included, and a dedicated analysis of each accent needs to be performed to determine where the model is more prone to errors. Furthermore, during the study, we noticed that the model struggles with homophone transcription. For example, when the two words "*there"* and *"they're"*, the model cannot find which of them will be used. This could be due to the poor performance of the decoder HMM used in the kaldi-based model. A language model can help predict correct sentences in the model's output. Another solution is to include more examples of the training sets where homophones are more likely present. The same applies to managing moments of silence, which could be indicated in the actual transcriptions by an initial and a final instant.

Transformer-based models

The transformer-based models (1) and (2) were developed to make accurate transcriptions of ATC exchanges, whatever the context. However, they have not been tested for a real-time[141] task. The time delay related to their predictions. Therefore, it needs to be investigated.
The Wav2Vec2 architecture, used by the transformer-based models, has the advantage of mostly making good predictions and is relatively quick to train. Hence, it remains a relevant axis that needs

[140] Besides to the generalization objectives, the ATC-STT models are expected to be robust to different accents of spoken English. This will be analysed latter in a coordination with the findings of task 3 (cf. chapter 6) where the robustness of all the models will be investigated more deeply.
[141] In the scope of the MLEAP project, this is not an issue but it is not neglected during the analysis since the Airbus source project of the ATC-STT use case considers this as a key element during the development.

to be explored. Being already trained on datasets containing more accents and being recorded in different contexts (cf. Table 45), using a training objective correlated to the one of this use-case (minimise the WER), these models are capable of absorbing more information than the KALDI models can handle, thanks to their complexity (more than 314M trainable parameters) and adequate capacity to record information.

The comparison between the KALDI-based model obtained with the Airbus test dataset and ATCO2 illustrates the impact of the datasets' lack of representativeness and generalisation. This trained model cannot maintain good performance on ATCO2 data, which contains more accents and covers more airports than the Airbus one.

However, even if they showed a significant gain in performance after being fine-tuned in Airbus's dataset, these models also struggle with correctly recognising homophones and handling silence times. Even if we don't know currently whether the transformer-based models' architecture is suitable for real-time applications, due to their complexity, the performance of the models is still lower than the objective of 10 % WER by the target application. Hence, further training and optimisations are needed, and the results after fine-tuning these models align with our hypothesis concerning the ability of these models to learn more and reach the targeted performances.

5.8.3.1.1.6 Alternatives' Exploration

After analysing the results of both KALDI-based and transformer-based approaches, in addition to the analysis of the pipeline construction, one common conclusion is that both approaches face the same problem related to the WER measure optimisation. Concerning the target application requirements, one of the leading performance indicators is to have a model that makes fewer errors in the predicted words per transcription. The WER measures this optimised to less than 10%. Besides, the optimised model should be able to run in a real-time application, having the prediction duration shorter than the utterance itself. While the transformer-based models are more performant in the different accents of spoken English (using open-source datasets), they take longer to perform the transcription of a given utterance.

Focusing on several parts of the STT process is necessary to understand the main characteristics and differences between several models and tools for automatic speech recognition. Figure 107 shows the main steps and components for a complete STT pipeline.

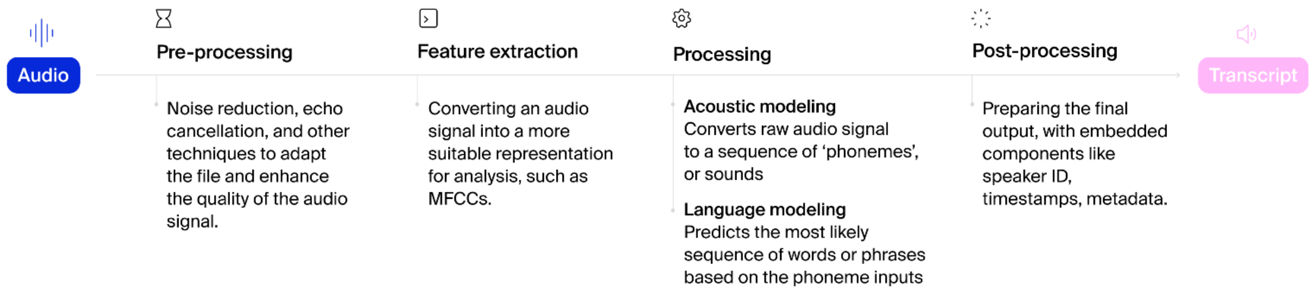

Audio
Pre-processing
Noise reduction, echo cancellation, and other techniques to adapt the file and enhance the quality of the audio signal.
Feature extraction
Converting an audio signal into a more suitable representation for analysis, such as MFCCs.
Processing
Acoustic modeling
Converts raw audio signal to a sequence of 'phonemes', or sounds
Language modeling
Predicts the most likely sequence of words or phrases based on the phoneme inputs
Post-processing
Preparing the final output, with embedded components like speaker ID, timestamps, metadata.
Transcript

Figure 107. Complete pipeline for STT applications and tools.

Data pre-processing is a crucial step, the most common reason several STT models cannot produce good results. The data is fed directly to the model in the models evaluated in this chapter, specifically the KALDI-based model by Airbus and the transformer-based models (Wav2Vec). The reason is to have a model that can produce good results under different operating conditions of the STT-ATC system.

Audio pre-processing (Sharma et al., 2020) is essential for training models and during deployment. To this end, the same work of cleaning up the data received during training must be reproduced even during deployment. Furthermore, an important detail to consider is the non-modification of the information conveyed in the input data for both phases (training and deployment). For STT, there is a set of methods, such as spectrogram pre-processing (Knight et al., 2020), for better-preparing inputs to be supported by AI models while respecting the information in them. During the experiments on this project, we did not consider the specific pre-processing of the audio recordings to reproduce the same experiments carried out by the Airbus project teams and carry out a more in-depth analysis to identify why the objective of WER<10 was not achieved. One possibility to enhance the results is to consider the pre-processing techniques of the input audio to help the models, KALDI-based and the transformers-based, learn more significant and less noisy information. However, selecting the pre-processing techniques should be driven by the system-level requirements, such as applicability and real-time transcription. Indeed, introducing the audio pre-processing layer must not add to the prediction time, particularly for transformer models. In addition, the pre-processing result must be analysed using the data qualification process to check its representativeness and relevance criteria.

The other two layers, feature extraction and processing, depend highly on the STT model and architecture. For instance, in the Wav2Vec model, these three functions are performed in the core of the transformer architecture, as shown in Figure 106. The KALDI-based model represents These two layers differently (cf. Figure 105). As shown in the previous results (cf. Table 48), the KALDI-based architecture struggles with the datasets, including different accents and where the data is collected in other airports. Two options can be considered separately or together to enhance the results: (1) train the acoustic model, a neural network that can be fine-tuned in different data sets, including different accents and for a longer training time. Note that, in our experiments, we trained the Wav2Vec model for 50 epochs (due to the long time that it takes to finish the training), while for deep neural networks training, the longer the model is trained, the better performances will be ; (2) train and optimise the language model that is responsible for building the output sentences (transcriptions) based on the recognised phonemes. Both options (1) and (2) can be performed together or separately, using mixed audio data recorded from different airports, to make the KALDI-based model achieve the WER < 10.
Finally, for end-to-end speech recognition, several models and tools exist that can also be explored to achieve better performances in terms of WER and ensure short-term prediction time to be adapted to the target objective of the STT-ATC use-case. For instance, Whisper is rated as the best ASR state-of-the-art model (Radford et al., 2022). This model is given a broader complexity[142] and built over several encoder blocks, as shown in Figure 108.

[142] Whisper includes from 39M to 1.55B training parameters, depending on the version that can go from tiny to large one.

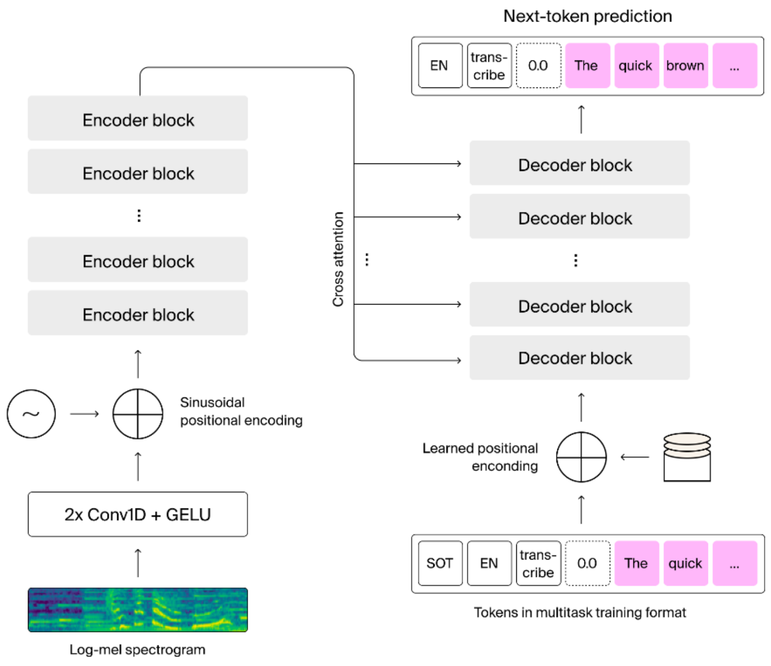


Figure 108. Whisper architecture showing the multitask training setup.

The model is open-sourced by OpenAI[143] and trained in 680000 hours of labelled data. However, the vanilla Whisper often comes with insufficient speed and accuracy in several use cases. In STT applications, it is widespread that an increase in transcription quality tends to go with a corresponding decrease in speed. Besides, the different versions of Whisper tend to hallucinate, as reported in recent studies by (Koenecke et al., 2024)[144]. The hallucination behaviour of ASR models aims at outputting transcriptions containing entire hallucinated phrases or sentences, which did not exist in any form in the underlying audio. This behaviour could be dangerous when using these technologies in safety-critical use cases, such as the STT application for air traffic control.

Another alternative to meet the ATC-STT use-case objectives regarding speed and performance is to consider tools that provide different models that are included and ready for use. For instance, SpeechBrain[145] is one of the tools worth exploring to see what it can bring to the use case regarding calculation efficiency, precision and performance. Moreover, this tool can be used to perform, among other things, voice recognition tasks that are also within the scope of the Airbus project, in particular Speech and Speaker Recognition, to help identify the profile of the speaker and Speech Enhancement to help understand the spoken instructions.

5.8.3.1.2 AVI

In the visual inspection use case, MLEAP aims to detect defects in aircraft fuselage based on taken pictures and then classify them into two classes depending on their cause: *lightning strikes* and *dents*.

[143] https://www.gladia.io/blog/what-is-openai-whisper
[144] To overcome the hallucinations, a new version of Whisper is recently released by GladIA[144], namely Whisper-zero[144].
[145] https://speechbrain.github.io/

The experimental analysis focuses on the generalisation capabilities of the models while evaluating the detection of different damages and the correct classification of the detected defects. In the Airbus source project, the aim is to estimate the severity of further defects to help operators identify the areas where it is necessary to take action as a priority during maintenance, thereby optimising the process. In MLEAP, this is out of the scope of the experimentation.

5.8.3.1.2.1 Dataset Analysis

In this case, we only used datasets from Airbus's internal projects. No open-source data has been found so far. Airlines and manufacturers are not keen on sharing their datasets of images of defects on their aircraft due to the emerging competition in this field, which explains the lack of publicly available datasets that can be used for this case study. However, similar datasets, such as the COCO-car dataset, can be used for automatic visual inspection of cars for scratches and surface damage[146].
In our experimentation, we used the Airbus dataset (cf. section 3.4), where each image can contain several defects and each of them is represented by a surrounding frame, representing:

- *Lightning strike*: where 28 images of 440 lightning strikes were used for training. The impacts vary in size and shape and were photographed at different distances.
- *Dents*: 3,659 images containing 1,225 dents are used for training. They were taken at different distances and include dent_al, which refers to those visible through ambient light, and dent_lb, which refers to those visible only using an LED lamp.

Using a lamp makes it easier to detect defects in the fuselage. Its shape is modified when it is projected onto a dent[147].
To run the experiments, a split of 10%, 10%, and 80% is used for testing, validating, and training the models, respectively.

5.8.3.1.2.2 Models' Description

This use case aims to detect lightning strikes and dents on aircraft fuselages. To this end, a binary classifier is trained for each class within the Airbus project, providing the data. The models are both based on a fine-tuning of the YOLOv5[148] model for object recognition model[149]. In MLEAP experimentations, we trained two models based on YOLOv5, one per damage type: *Lightning strike* and *dents* detection, using the datasets described above. The overall architecture of the model uses a Convolutional Neural Network (CNN), and is trained with the objective of minimising a loss function defined (Adhikari et al., 2022) as:

$$loss = l_{box} + l_{obj} + l_{cls}$$

With l_{box} is the location and dimension error associated with the frames containing the detected defects, and is defined as follows:

[146] https://www.kaggle.com/datasets/lplenka/coco-car-damage-detection-dataset
[147] Examples can be found in Annexe.
[148] https://ui.adsabs.harvard.edu/abs/2020zndo...3983579J/abstract
[149] https://github.com/ultralytics/yolov5

$$l_{box} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} (2 - w_i \times h_i) \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + \left(h_i - \hat{h}_i\right)^2 \right]$$

l_{obj} represents the prediction confidence-related error, computed as follows :

$$l_{obj} = \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} \left[\left(C_i - \hat{C}_i\right)^2 \right] + \lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[\left(C_i - \hat{C}_i\right)^2 \right]$$

l_{cls} represents the classification error of the defect, defined as:

$$l_{cls} = \lambda_{class} \sum_{i=0}^{S^2} \mathbb{I}_{i}^{obj} \sum_{c \in classes} p_i(c) \log\left(\hat{p}_i(c)\right)$$

With λ_{coord}, λ_{noobj}, λ_{obj}, and λ_{class} are coefficients used to focuss on the different error parameters of the $loss$ function.
The table below shows the description of the remaining parameters.

Parameter	Description
λ_{coord}	Coefficient on the location of the prediction
λ_{noobj}	Coefficient on the confidence of the prediction if no object is detected
λ_{obj}	Coefficient on the confidence of the prediction if an object is detected
λ_{class}	Coefficient on the class of the prediction
S	To calculate the $loss$ the image is divided into S^2 cells using a grid.
B	Number of predictions made on the image by the model
$\mathbb{I}_{ij}^{obj}$	Function equal to 1 if an object is present in cell i and the prediction j detects it, otherwise 0
$\mathbb{I}_{ij}^{noobj}$	Function equal to 1 if there is no object in box i, otherwise 0
$\mathbb{I}_{i}^{obj}$	Function equal to 1 if there is an object in box i, otherwise 0
x_i	x coordinate of the reference frame in the box i
y_i	y coordinate of the reference frame in the box i
w_i	Width of the reference frame in the box i
h_i	Height of the reference frame in box i
$\hat{x}_i$	x coordinate of the prediction in box i
$\hat{y}_i$	y coordinate of the prediction in box i
$\hat{w}_i$	Width of prediction in box i
$\hat{h}_i$	Height of the prediction in box i
C_i	Coefficient on the confidence of the reference
$\hat{C}_i$	Coefficient on the confidence of the prediction
$classes$	Set of classes that can be detected: lightning strikes or dent impacts
$p_i(c)$	Conditional probability that an object of class c will appear in box i
$\hat{p}_i(c)$	Predicted conditional probability that an object of class c will appear in box i

Based on this loss function, the models are then trained to predict either of lightning strikes or dent impacts. Similar to the data availability issue, no pre-trained models are available on the literature.

Even if several research work in aviation maintenance have been launched recently (Shailaja and Padmanabhan, 2022), there is no sharing of either the implemented models or the data used for training. Hence, based on the model description above, we optimised two models for each class.
A priori generalisation bounds evaluation provides results that are not tight enough to guarantee that the model will maintain the same level of performance on unseen data.

5.8.3.1.2.3 Evaluation measures

To evaluate the generalisation capabilities of the model, we focus on the trained model's analysis of the predicted areas. The predictions are bounding boxes, locating where the model detects damage. To determine whether its assumptions are correct, a joint verification function is the *Intersection over Union* (IoU), which is the intersection area between the prediction and the actual fault location divided by the area of their union, as shown in Figure 109. This technique is used in the following for model evaluation. It could also be used as a loss function, but the architecture of the models we used (YOLOv5) is based on a more complex one, which will be detailed further.

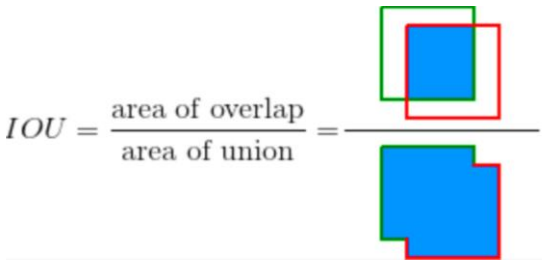


Figure 109 Illustration of the IoU measure for objects recognition (Padilla et al., 2020). The red box corresponds to the model prediction, the green corresponds to the ground truth.

A threshold value IoU_{th} is then defined. This value defines a level of requirement for the intersection of the two frames. The closer the value is to 1, the higher the level of requirement, so the aim is to locate the object in the image as accurately as possible. So, to avoid penalising the predictions too much, the most common threshold is $IoU_{th} = 0.5$, allowing a slight margin of error between the prediction and the reference while still requiring the object to be correctly located.
For the defects' detection, this corresponds to locating the defect on the input image correctly. Based on the threshold value, two possibilities arise:

- if $IoU \geq IoU_{th}$: the prediction is considered as a *True Positive* (TP), and the defect is deemed to be correctly detected;
- if $IoU < IoU_{th}$: the prediction is considered as a *False Positive* (FP), which means that there is no defect on the predicted frame.

The predictions can then be ranked by decreasing confidence and the number of accumulated TPs (*Acc TP*) and accumulated FPs (*Acc FP*) can be calculated. For each prediction, *Precision* and *Recall* will then be:

$$Precision = \frac{Acc\ TP}{Acc\ TP + Acc\ FP}$$

$$Recall = \frac{Acc\ TP}{N_{defects}}$$

Where $N_{defects}$ is the number of all existing defects in the dataset.
The *Accuracy* versus *Recall* curve can be plotted, and the area under this curve gives an average *Accuracy* value (Padilla et al., 2020).

5.8.3.1.2.4 Results

For the AVI task, the models are built using a *transfer learning* approach[150], where only some layers have been trained. As mentioned in section 3.4, a pre-trained YOLOv5 model is used. This one is pre-trained in large datasets for image processing, such as COCO and ImageNet. This model is then used in our experiments for fine-tuning in the AVI use case.
The corpus-level analysis in section 4.8's experiments concluded that the data seemed reasonably representative for this task, in the sense of no specific imbalance, but in clearly insufficient amounts. Thus, the expected result is that the system should not have any particular problematic bias but may exhibit low performance compared to expectations.

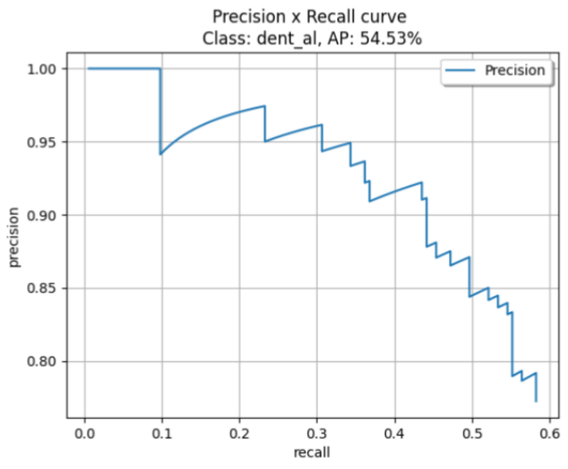
- **Lightning strike detection**

For this task, the provided data set is too small and not representative of the ODD coverage. For example, not all light conditions, painting colours, or camera angles are covered by images in the dataset. The dataset used for testing consists of only eleven (11) lightning strikes, among which seven (7) are detected by the model. The average accuracy measured is 41% with these few images.

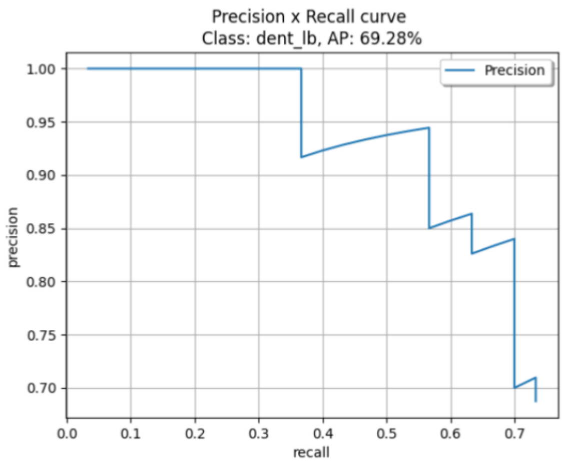
- **Dent detection**

The dents detection test dataset consists of 522 images and 193 defects. Figure 110 shows the *precision* versus *recall* curves with $IoU_{th} = 0.5$, where the average precision is 61.91% over the different classes. However, a better precision on the set of defects illuminated by a LED lamp (*dent_lb*) is observed: 69.28 % average precision compared with 54.53% for non-illuminated defects (*dent_al*). This is due to the added features of the LED light in the image. The lamp beam takes on a different shape when projected onto a surface defect, which helped the model recognise them more easily.

[150] https://docs.ultralytics.com/yolov5/tutorials/transfer_learning_with_frozen_layers/



(a) Classe dent_al



(b) Classe dent_lb

Figure 110 Accuracy *versus* Recall *curves, with $IoU_{th} = 0.5$, corresponding to a trained YoloV5 model for detection of two types of dent instances.*

Equivalent average accuracies with $IoU_{th} = 0.8$ are observed on the two classes *dent_al* and *dent_lb*, but for $IoU_{th} = 0.9$, it drops to 1.17%. Figure 111 shows the performance variation w.r.t the IoU_{th} values.

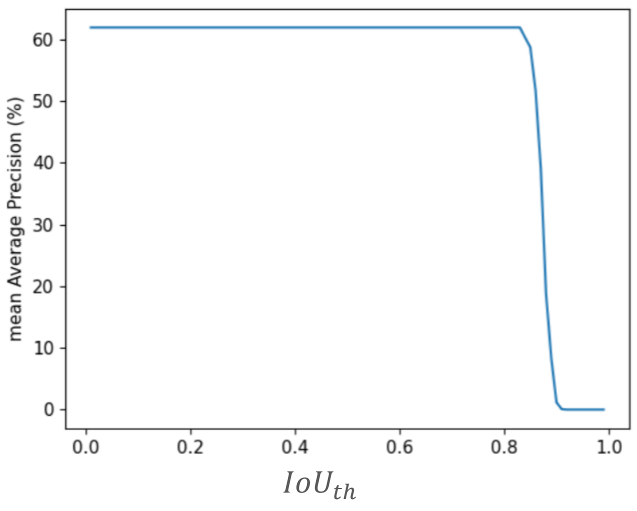


Figure 111 Average precision variation w.r.t values of the IoU_{th}

For $IoU_{th} \in [0, 0.8]$, the average precision remains constant, which means that no fault is detected with $IoU \leq 0.8$. This curve shows that the IoU of the references vs predictions is always higher than 0.8, which means that when a fault is detected, it is correctly located and no damage is identified by mistake. However, the average accuracy is 61.91% for $IoU \leq 0.8$. Unfortunately, this performance is inadequate compared with the expectations of the target application (source project in Airbus). Indeed, the objective is to have models with an average accuracy of 95%.

5.8.3.1.2.5 Pipeline Analysis

The study of the challenges associated with the AVI use case aims to correctly locate fuselage damages of two types (dents and lightning strikes), with a detection accuracy of 95%, to meet the requirements of the target application. Hence, two models have been designed to minimise the loss function based on the differences between the detected boxes and the reference ones, corresponding to the detection error definition (cf. loss function above). As a result, the IoU measure is optimised. The trained models are based on YOLOv5[151] architecture. This model is trained initially for object detection and classification, and in this use case, it is fine-tuned for defect detection.

As mentioned before, due to limited data amounts, especially for lightning strike detection, the obtained performances seem limited: 41% on lightning strikes and 61.91% on dents. This does not meet the target objective of 95% accuracy. One possibility to deal with this issue is to use augmented data by generating similar data. Several techniques can be used (Shorten and Khoshgoftaar, 2019). However, their impact needs to be investigated deeper regarding their correlation with real data and their performance with the models.

The objective function optimised is based on the defects' location (comparing predicted and reference boxes surrounding each damage class). This objective is consistent with the target objective definition (*Accuracy* versus *Recall* based on IoU values. Cf. section 5.9.2.2.3). Thus, the design model aligns with the objective set by its use. Furthermore, one model per damage class was used for defect classification. This architecture, therefore, assumes that the defect class is known in advance. However, the implementation phase would reveal the model integrity on the target domain as part of the target objective; the model is expected to recognise each defect type and provide a corresponding severity score. Hence, the model architecture needs to be adapted accordingly. To do so, segmentation methods[152] can be used to calculate the severity of the defect based on the analysis of the surface or volume of the detected contour. Precisely, given information on the shape and size of the damage, the severity score will be computed outside the model. So far, the used YOLOv5 model can only give these defects' locations without knowing their size. However, the latest version YOLOv8[153] has been released and allows segmentation. Hence, this will be a relevant option to explore further in our experimentations.

Finally, the loss function used by YOLOv5 is not bounded (cf. section 5.9.2.2.2). Indeed, it is made of three main parts including λ_{coord}, λ_{noobj}, λ_{obj}, and λ_{class} that are coefficients corresponding to each part of the loss function ($loss = l_{box} + l_{obj} + l_{cls}$). This depends on the number of defects detected per image. Given $\lambda_{coord} + \lambda_{noobj} + \lambda_{obj} + \lambda_{class} \neq 1$ the resulting loss is not standardised. Without knowing the boundaries of the loss, we cannot interpret it more easily (e.g., an error between 0 and 1 makes it easy to understand the model learning behaviour and makes it suitable to compare with other models, all having normalised loss functions).

[151] https://github.com/ultralytics/yolov5

[152] Image segmentation aims at detecting contours of an object in an input image https://docs.ultralytics.com/tasks/segment

[153] https://docs.ultralytics.com/models/yolov8/

5.8.3.1.2.6 Alternatives' Proposal

To achieve the target objectives for the AVI use case in terms of damage detection performance, we explored different alternatives for the model. The purpose is to analyse the other options to help meet the industrial objective of 95% precision. To do so, we considered the Ultralytics'[154] model YoloV8[155] to perform the damage detection tasks. YoloV8, or You Only Look Once (Redmon et al., 2016) version 8, is a deep learning object detection model. The different versions of YOLO are known for their speed, accuracy, and scalability improvements. YOLOv8, an evolution of the YOLO family, aims to enhance these aspects further while maintaining robust object detection capabilities. In (Jocher et al., 2023), an experimental analysis compared the different versions of YOLO and highlighted that YoloV8 outperforms its predecessors in prediction tasks using COCO[156] dataset, as shown in Figure 112.

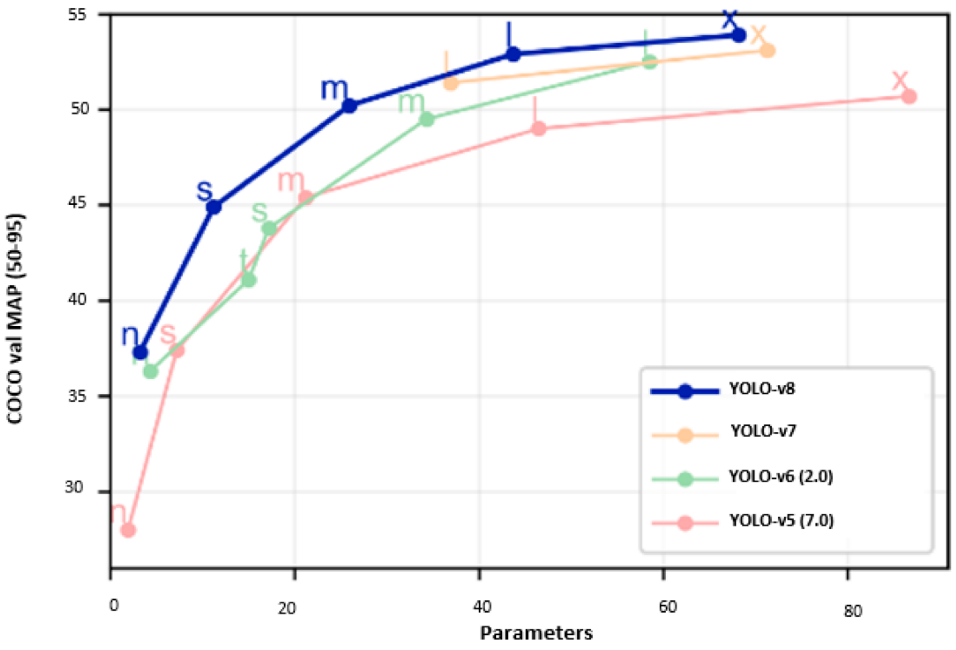


Figure 112 Comparison of YoloV8 with its predecessors, in COCO dataset, in terms of MAP (mean average precision).

In addition to its performance in different versions, one of the primary objectives of YoloV8 is to achieve real-time performance, enabling fast and efficient object detection in video streams or on embedded devices.

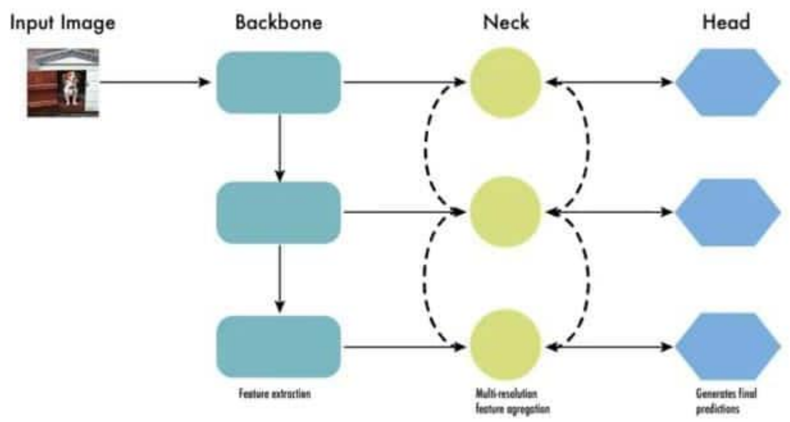
A priori evaluation of generalisation bounds applied to the two models provides high values that are not useful for comparing the generalisation capacity. The values are so large that they don't guarantee that the model will maintain the same level of performance on unseen data.

All YOLO architectures are based on three main significant parts, as shown in Figure 113.

[154] https://www.ultralytics.com/

[155] https://yolov8.com/

[156] https://docs.ultralytics.com/datasets/detect/coco/


Input Image
Backbone
Neck
Head
Feature extraction
Multi-resolution feature agregation
Generates final predictions
Figure 113. Architecture of YOLO models made of three big parts: backbone, neck and head.

The backbone is crucial in extracting significant and relevant features from input images using CNNs trained on large-scale image classification tasks. It captures hierarchical features at varying scales. Lower-level features (edges and textures) are extracted in the previous layers, and higher-level features (like object parts and semantic information) are removed in the deeper layers. Finally, the neck computes intermediate features, and it connects the backbone to the head outputs, providing the final predictions. This same architecture is used in YoloV8. This later employs an improved backbone such as Darknet-53 (Redmon and Farhadi, 2018), which provides a robust feature extraction layer. A feature pyramid network (FPN) is then used to capture multi-scale features. This is crucial for detecting objects of varying sizes in the input image. Finally, the detection head comprises multiple convolutional layers responsible for predicting bounding boxes and prediction scores. It typically predicts a fixed number of bounding boxes per grid cell across different scales.

YoloV8 represents an excellent alternative for the AVI use case for all these reasons. In our experiments, for both datasets, Dents and lightning-strikes images, we used the same YoloV8 configuration in the following essential values:

- Training epochs: 100
- Batch size: 16
- Images size: 640
- Learning rate: 0.01
- Momentum: 0.937
- Weight decay: 0.0005

The performances of the different models are compared in terms of three main measures:

- **mAP50:** Mean Average Precision at 50% Intersection over Union (IoU).
- **Precision:** This indicator, which is key to understanding prediction accuracy, indicates the ratio of correctly predicted positive observations (correctly detected impacts and classified correctly).
- **Recall:** Important for models where missing a detection is significant, this metric measures the ability to detect all relevant instances (detect all impacts).

Results are shown in Table 50. In this table, different instances of YoloV8 have been compared with YoloV5s, with 7.5M parameters. For YoloV8, “s” stands for “small” and refers to a light version of the model’s architecture, including up to 30M trainable parameters. The other indices, “m” and “l,” stand for “medium” and “large” and represent bigger versions, with 70M and 120M parameters, respectively.

Metric	Model	Dents (1044 images, 316 labels)	Lightning strikes (6 images, 13 labels)
Precision %	Yolov5s	69.4	69.9
	Yolov8s	86.3	**98.9**
	Yolov8m	85.9	39.8
	Yolov8l	**88.5**	90.1
Recall %	Yolov5s	64.3	**50**
	Yolov8s	**84.9**	38.5
	Yolov8m	82.1	46.2
	Yolov8l	79.7	15.4
mAP@50 %	Yolov5s	64.4	**54.5**
	Yolov8s	**89.2**	44.8
	Yolov8m	88.6	26.8
	Yolov8l	86.6	28.3

Table 50: Performance’s comparison of different Yolo architectures, trained in original and augmented datasets for AVI use case. The performances are % values of three main measures: precision, recall and mAP@50.

As shown in Table 50, the YoloV8 architecture seems to have better results than YoloV5, in both datasets for Dents and Lightning-strikes detection. Specifically, the YoloV8l outperforms the different models in terms of precision in Dents detection while having comparative results with the smaller versions YoloV8s and YoloV8m. Besides, in Lightning-strikes detection, the YoloV8 outperforms YoloV5 with up to 98% accuracy using YoloV8s. Based on the results of Table 50, let’s analyse in more detail the results of the best model, YoloV8l, for Dents and Lightning-strikes detection.

- **Dents detection**

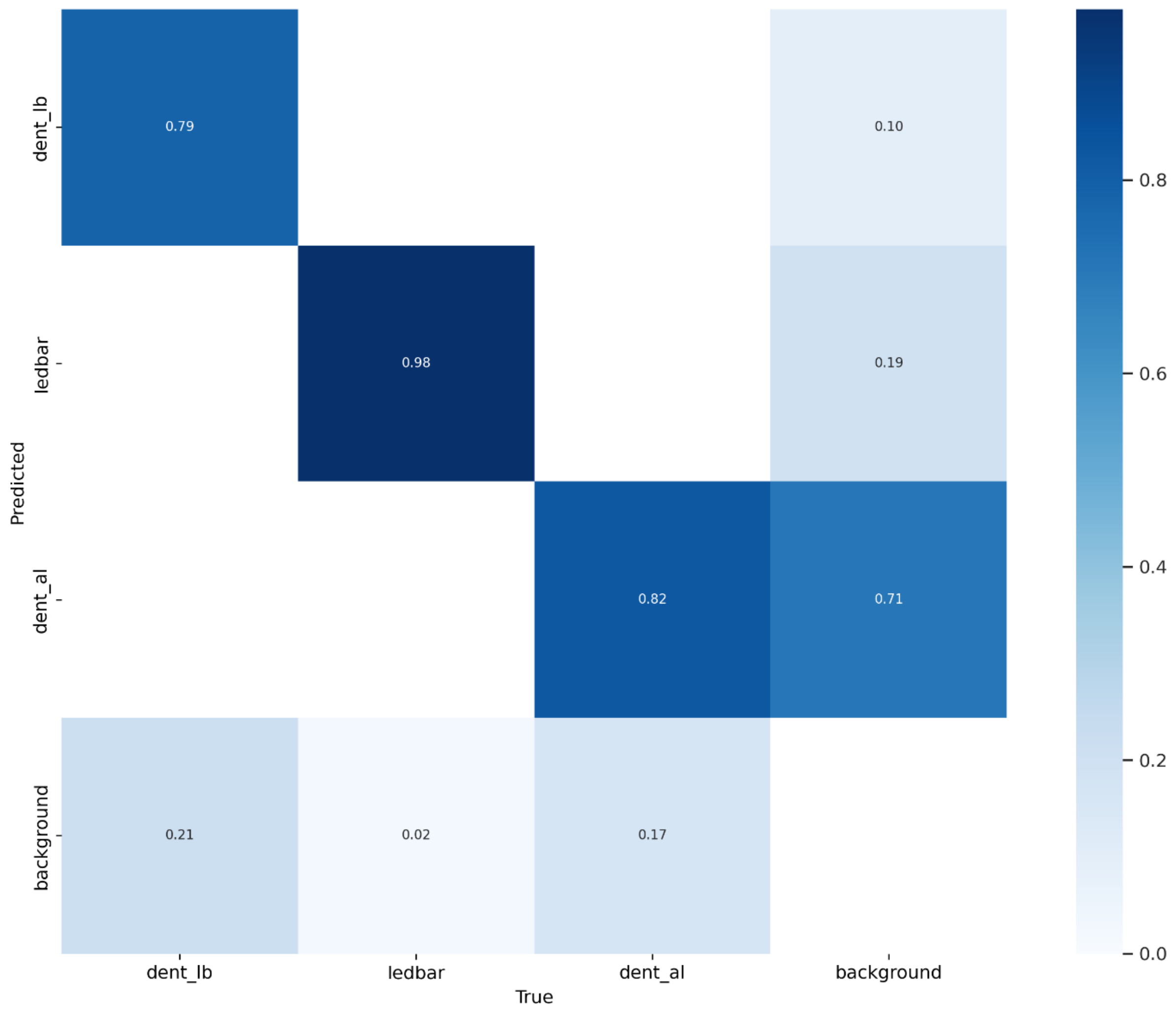
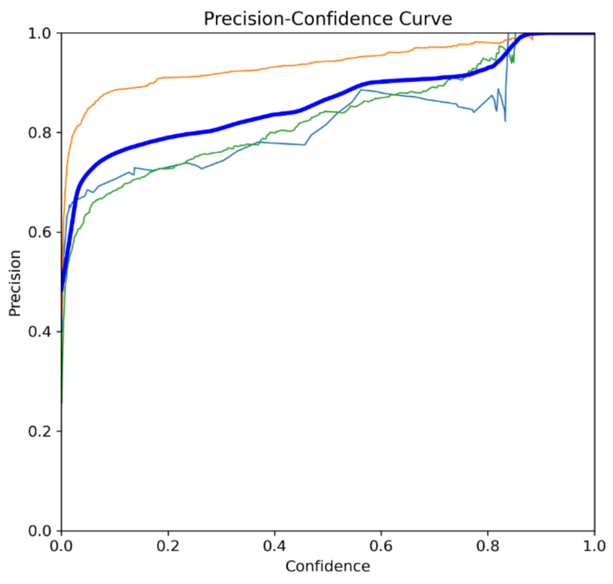
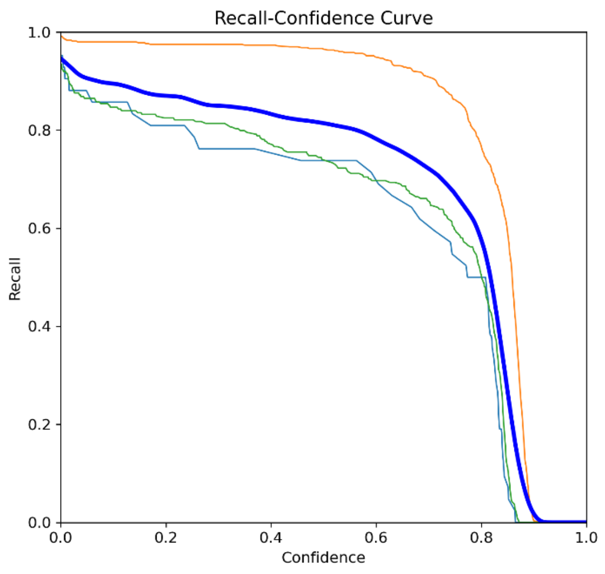


Figure 114. Normalized confusion matrix for Dents, of the best model Yolov8l for the detection and classification of the different damages.





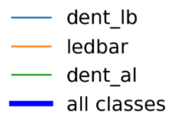


Figure 115. Evolution of precision and recall performances w.r.t confidence scores, for YoloV8l in dents classification

Based on the precision and recall evolution curves analysis, we can notice that the precision of the model increases with confidence while the recall decreases. Indeed, the precision measures the proportion of true positive predictions among all positive predictions made by the model. When precision increases with confidence, it means that as the model becomes more confident in its predictions (i.e., higher confidence threshold), it tends to make fewer positive predictions overall. Still, a higher proportion of these predictions are correct. In other words, the model becomes more selective in its predictions, making them only when highly confident. This leads to a decrease in false positive predictions, thus increasing precision. This is highlighted in the confusion matrix of Figure 114, where most predictions are correctly classified. To see where the model is getting confused, we analysed some of the predictions to understand where the model seems to have bad predictions[157]. While analysing the predicted impacts, we noticed that most of the damages were correctly detected, with high scores for the correct predictions. While still confusing, some shadows are predicted as dent impacts. One possible solution to reduce this confusion is to have more training examples of images, including light reflections and shadows where *dent_al* impacts are also tagged. This will allow the model to distinguish between a light shadow/reflection and a real impact.

- **Lightning strikes detection**

As for the dent's detection, we analyse the YoloV8.

[157] Prediction examples can be found in Annexe.

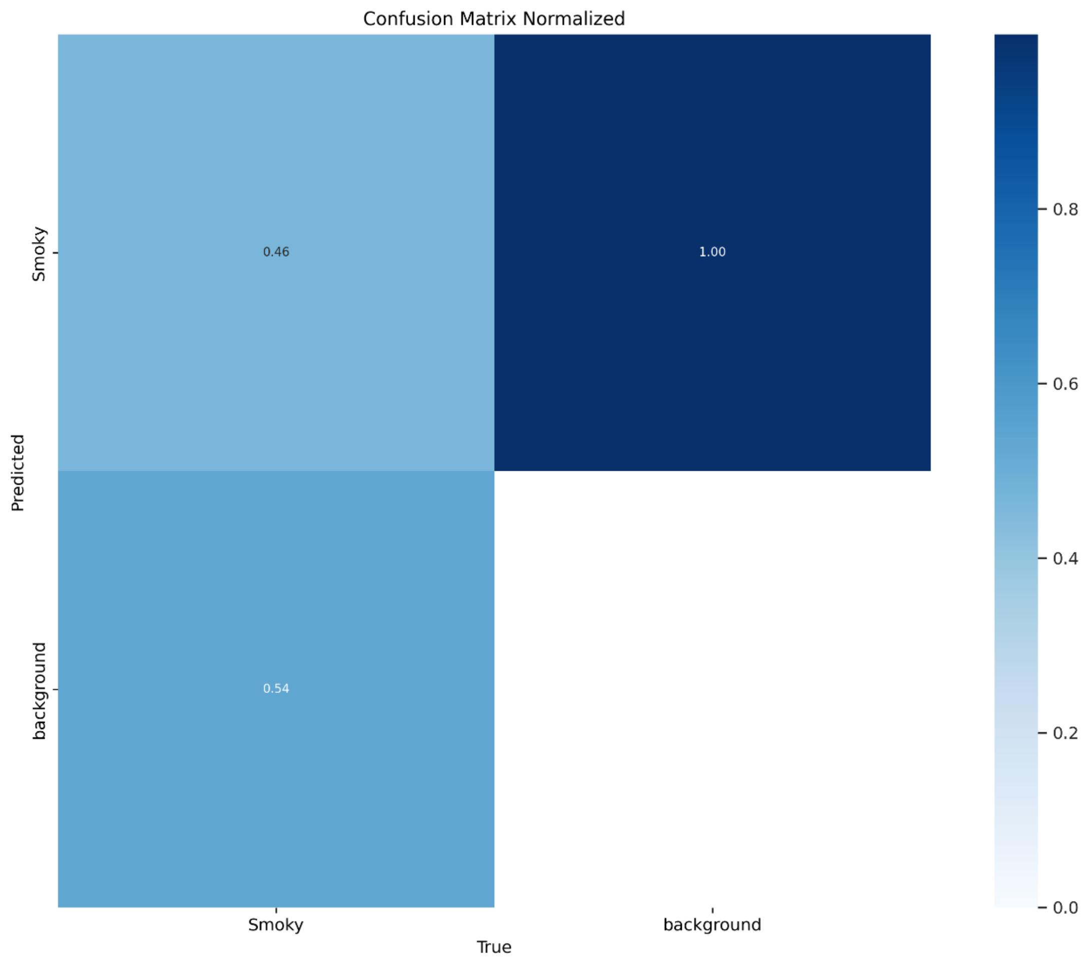


Figure 116. Normalized confusion matrix for Lightning Strikes detection (smoky label), of the best model Yolov8s for the detection and classification of the different damages

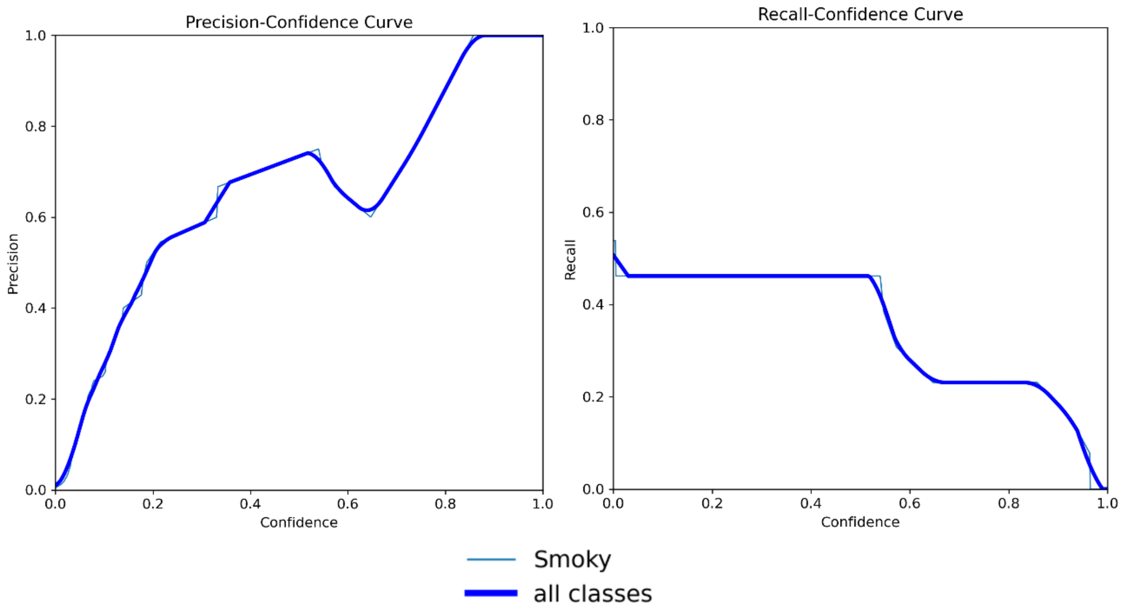


Figure 117. Evolution of precision and recall performances w.r.t confidence scores, for YoloV8s in lightning strikes detection. classification

Based on the results of the precision and recall curves and the analysis of the prediction examples[158], we can notice that the model struggles to find all the lightning strikes while the predicted ones are highly scored. Sometimes, when the image's impact is poorly defined, the model cannot predict it correctly. Note that the model has an accuracy of 90% (YoloV8l) and 98% (YoloV8s), outperforming the baseline YoloV5 and achieving the target application objective. However, the model performs well in correctly classifying the impacts as "smoky" but still struggles to identify all positive instances correctly (finding all the impacts correctly).
Achieving very high accuracy despite low recall indicates that the model is excellent at correctly classifying instances when it makes a prediction. Still, it may not capture all positive instances present in the data. This situation often arises when the dataset is imbalanced, meaning significantly more negative than positive instances exist. Indeed, in the lightning strikes dataset, there are few training examples with an essential variation of the smoky areas that represent impacts. Hence, the model learned to prioritise accuracy by focusing more on correctly classifying the abundant negative instances while sacrificing recall for the minority positive instances.

In both damages detection tasks, *Lightning strikes* and *Dents*, the impact of the volume of data is also present while training and evaluating YoloV8. Even if this later has more excellent performances (with the different versions, s, m, and l), it still struggles with the prediction of all the damages (low *recall* performances), due to the small number of examples in the training data sets, which makes the model confuse some classes. To overcome this problem, the data augmentation can be a suitable option. To do so, we first analysed a set of methods used to produce an augmented dataset for lightning strikes[159] detection.
Data augmentation applied to images involves a sequence of functions used randomly. Among these functions, we find translation, rotation, zooming, scaling, flipping, blurring, and noise application functions. Additionally, perspective transformation and brightness adjustment functions are also applied. Each of these functions is used with a randomly defined intensity.

Metric	Model	Lightning strikes (6 images, 13 labels)
Precision %	Yolov5s	**69.9**
	Yolov5s finetuned on augmented data (100 epochs)	54
Recall %	Yolov5s	**50**
	Yolov5s finetuned on augmented data (100 epochs)	46.2
mAP@50 %	Yolov5s	**54.5**
	Yolov5s finetuned on augmented data (100 epochs)	39.9

Table 51. Comparison of the YoloV5 model trained in original data and the one trained in augmented data.

As shown in Table 51, the performances of the model trained in the augmented dataset are much lower than the same model trained in the original data. Therefore, exploring another alternative,

[158] Examples can be found in Annexe.
[159] In our analysis, we focused more on the lightning strikes for the data augmentation, since it is the task that is less provided with data samples.

which consists of acquiring enough data for damage detection, is better. To do so, one can explore the use of open-source datasets for damage detection. For instance, the YoloV8 trained[160] in car parts damage detection dataset[161] performs excellent results in the identification of 8 different damages. Figure 118 shows some examples of damages that are present in the dataset.

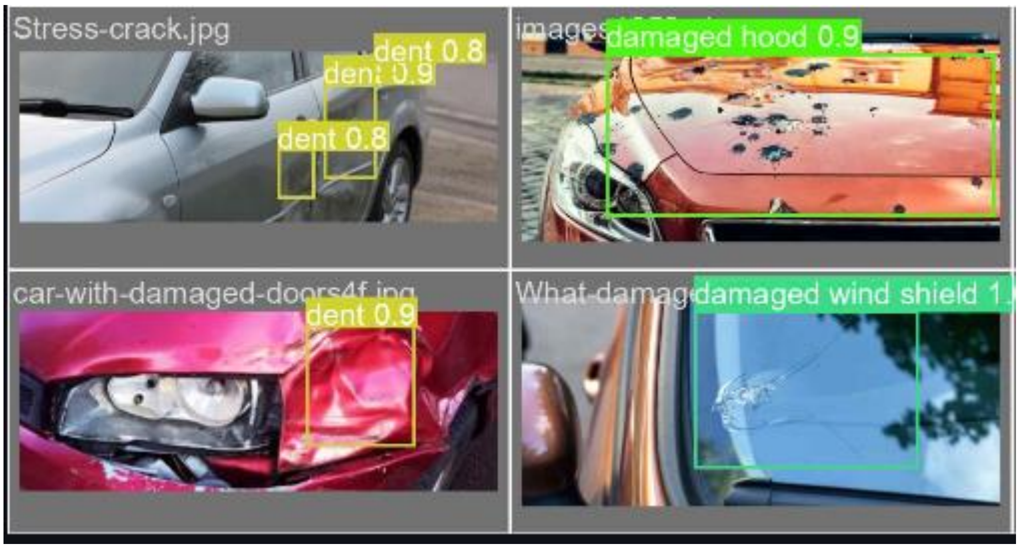


Figure 118. Example of some damages detected with the trained model showing the kind of damages present in the dataset

As shown in Figure 118, there could be some similarities with the AVI use case for aircraft damage detection. The model YoloV8n trained on this dataset can be used, after fine-tuning in the minor Dents and Lightning Strikes data, to leverage its best performances for damage detection in cars. Figure 119 shows the evolution of the performance of the YoloV8n trained for damage detection in cars. We can notice that for the different classes, the model achieves excellent scores in terms of accuracy, with very high confidence values, making it an excellent alternative to explore to achieve the 95% accuracy target for aircraft fuselage damage detection.

[160] https://github.com/suryaremanan/Damaged-Car-parts-prediction-using-YOLOv8
[161] https://github.com/suryaremanan/Damaged-Car-parts-prediction-using-YOLOv8/tree/main/data

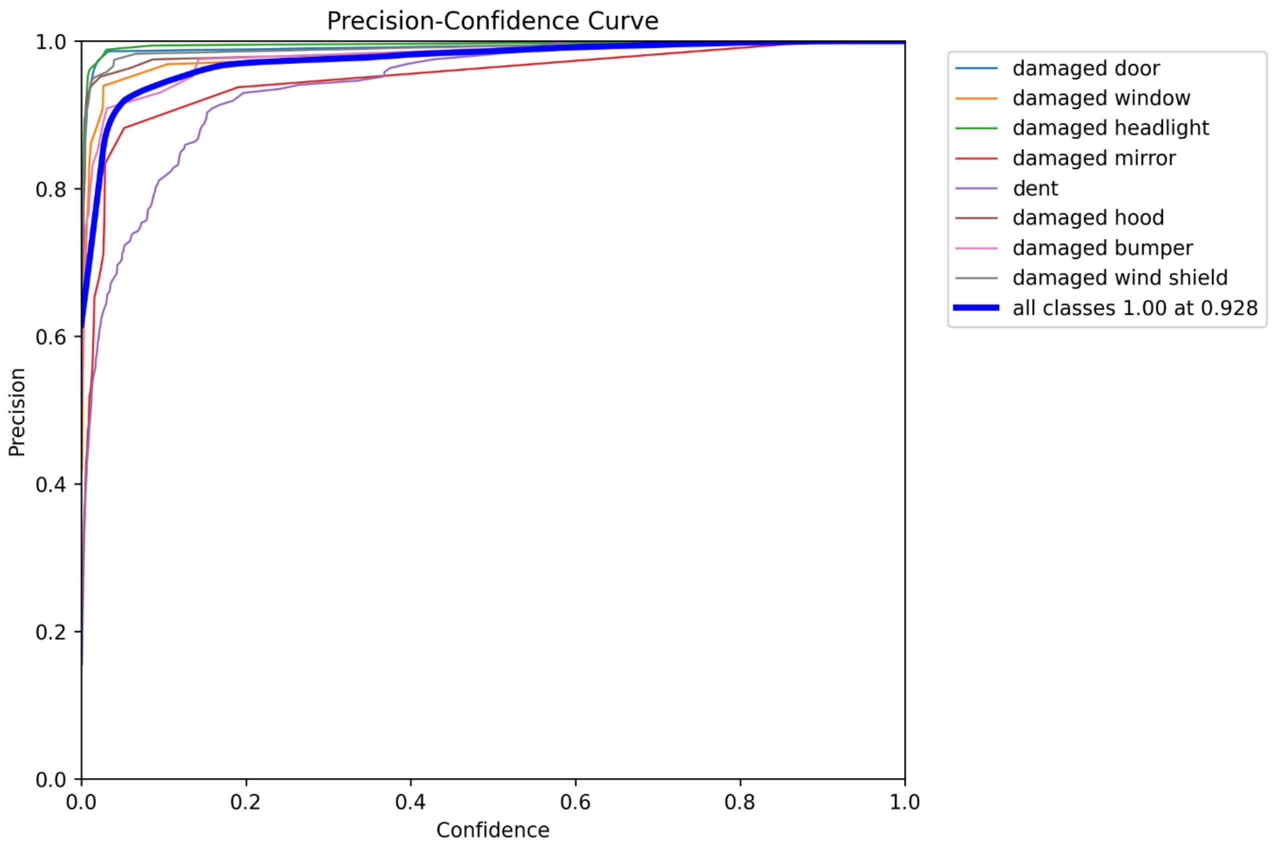


Figure 119. Precision-confidence curve of YoloV8n for 5000 epochs.

5.8.3.1.3 ACAS Xu

5.8.3.1.3.1 Dataset Analysis

The datasets[162] are taken from the Minimum Operational Performance Standards (MOPS) provided by Radio Technical Commission for Aeronautics (RTCA) Special Committee 147 working on the Traffic Alert and Collision Avoidance System (TCAS). The variable definition domains are also well-defined (cf. section 3.5). Based on these variables (cf. Table 8), different instructions are provided to avoid the collision between the unmanned aircraft.

5.8.3.1.3.2 Models' description

The model[163] is made of 45 neural networks (Julian et al., 2019). There is one NN for each pair (τ, a_{prev}). The system therefore chooses the suitable NN accordingly to make its prediction. For example, the neural network $N_{0,COC}$ will be used if the two aircrafts are at the same flight level ($\tau = 0\ s$) and the last instruction was to do nothing (COC = *Clear Of Conflict*). Once the network has been correctly selected, it takes the five remaining variables $(\rho, \theta, \psi, v_{own}, v_{int})$ as parameters. The output

[162] https://my.rtca.org/productdetails?id=a1B1R00000LoYKtUAN
[163] https://github.com/mldiego/AcasXu/tree/master/networks

is a score for each possible instruction (COC, WL, WR, SL, SR), representing the probability of obtaining a conflict when applying the associated instruction.
Each model is a feedforward neural network (FNN) with six hidden layers and 13,000 trainable parameters (cf. Figure 120). The table contains 600 million pieces of data, so each model has been trained on over 13 million inputs.

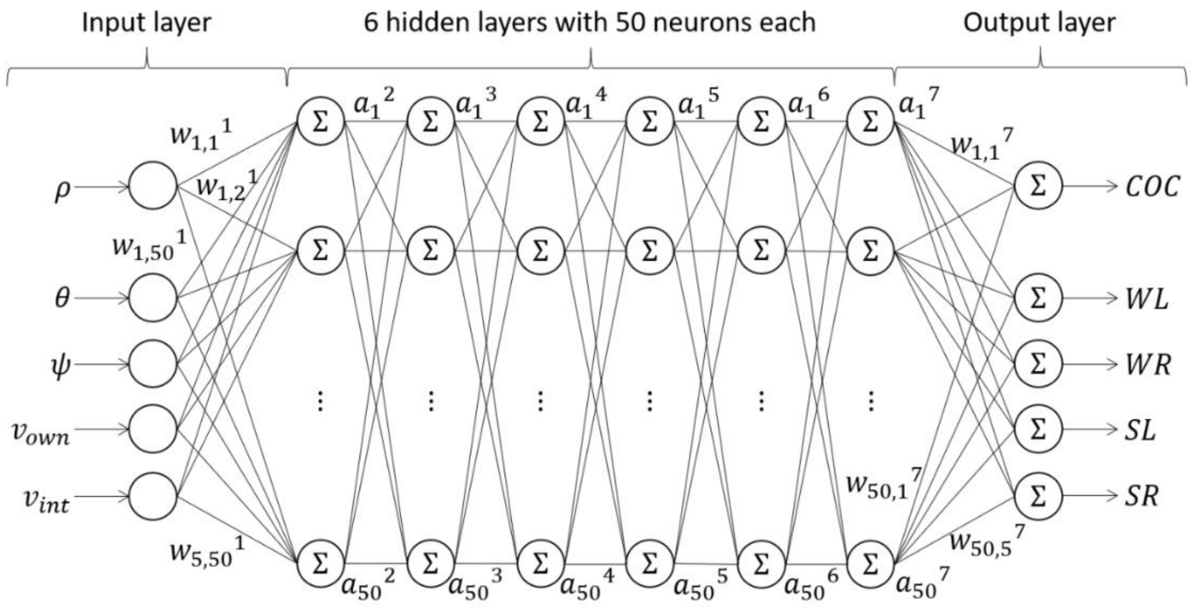


Figure 120 Illustration of the used NN architecture for one pair (τ, a_{prev}) (Lopez et al., 2021)

In the following, we will assume that there can only be one intruder at a time. This assumption was also made when designing the models presented above.
The evaluation of generalisation bounds for this use case provides tight values. We can anticipate good alignment between the metrics obtained on the train and test dataset (LM-04 & LM-14).

5.8.3.1.3.3 Results

To observe the model's predictions, a simulation for two drones at the same flight level ($\tau = 0\ s$) is used, while going in opposite directions ($\psi = -\pi$) at 430 *ft/s*. The ownship is at (0,0) and heading in the positive x direction. The results are shown in Figure 121. The colours correspond to the instructions given to the ownship w.r.t the position of the intruder, which is moving along the negative x axis.

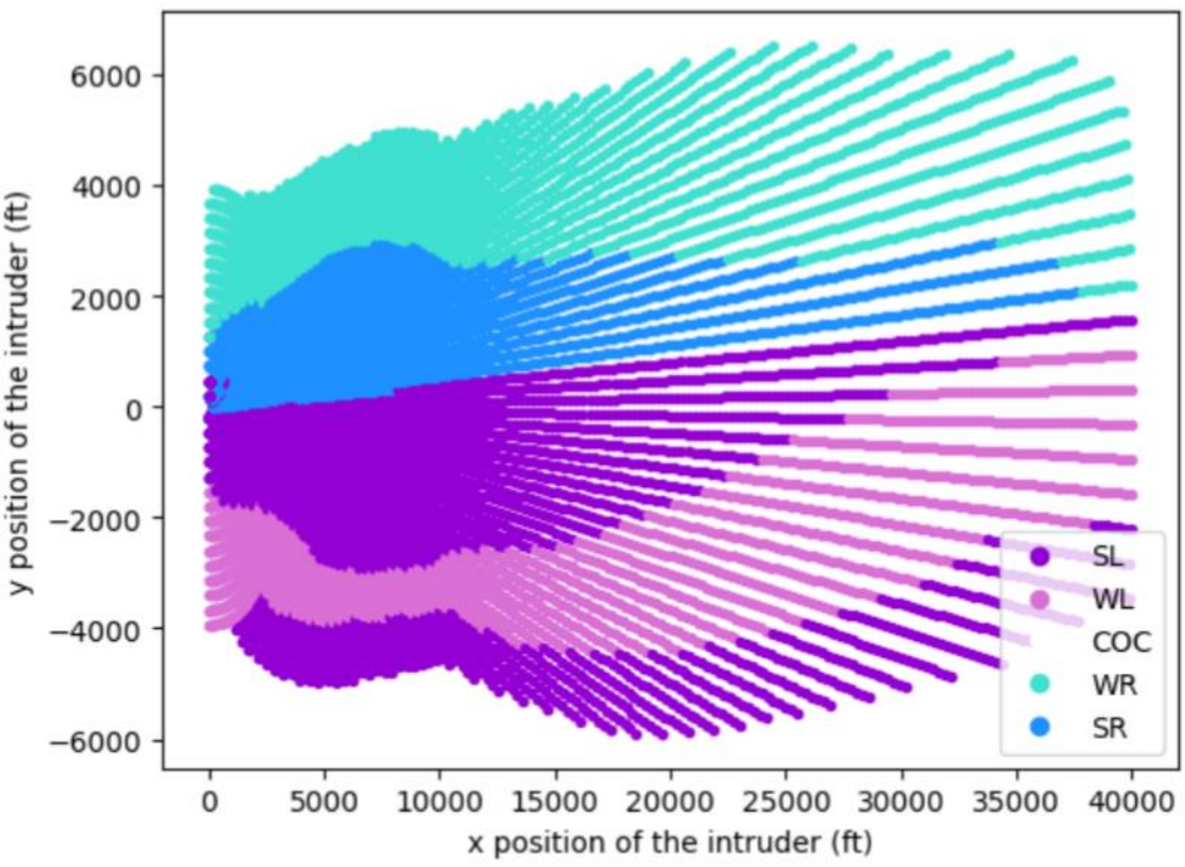


Figure 121 Instruction of the neural network $N_{0,COC}$ as w.r.t: the position of the intruder, the ownship being at (0,0) and moving along the x axis. With $(\psi = -\pi)$ at and $v_{own} = v_{int} = 430$ ft/s.

The instructions obtained are straightforward. If the intruder is on the right side of the ownship, the latter is asked to turn left, and vice versa. Furthermore, the closer the two aircraft are and the more they face each other, the more their ownship will be asked to turn severely (*SL* or *SR*). However, some strange predictions exist for a possible intruder far away and on the right side of the ownship. The recommendation was to turn sharply when the two aircraft were far apart. According to the analysis of (Julian et al., 2019), with the same configuration, the instruction given by the table is to turn slightly to the left (*WL*). Hence, further analysis is needed to understand where the difference comes from.
Another way to deal with the different instructions is to focus the evaluation of generalisation capability by interpolating some points close to the boundary decision layer that are not in the current LUT and assessing whether the model output results in an unsafe situation (i.e., lousy generalisation) or is similar to the points used for the interpolation (i.e., good generalisation).

5.8.3.1.3.4 Pipeline analysis

The main objective of this use case is to reduce the storage space required to run ACAS Xu systems. To achieve this, using neural networks was the option selected by users to compress the contents of the data table. The current state-of-the-art system uses a table to give correct avoidance instructions. The models used to replace this table are trained to reduce the difference between the model's predictions and the actual instructions of the input data. Accuracy optimisation enables the model to

learn and compress the input tables perfectly. 45 neural networks have been used, each corresponding to a possible instruction (τ, a_{prev}). Therefore, one NN only has to know the parameters of one table as a function, not the entire correspondence table.
Differently from the generalisation analysis objectives, the aim is to make the model provide predictions only for inputs that belong to the system's current table. Hence, the problem focuses on learning the tables by several neural networks to compress them and make fast predictions. To do so, the NNs have been trained to solve a classification problem, where the output class corresponds to one of the five possible instructions setpoints (COC, WR, WL, SR or SL) that should be given to the ownship, avoiding the collision. Therefore, The training objective was to minimise the cross-entropy, a loss function used in classification problems, and to maximise the model accuracy. Hence, the learning management in this use case aims to make the model better learn the input data and enhance the learning process. A priori evaluation of generalisation bounds has provided tight values; with high probability, trained model results on the full ODD will keep the same performance level as the one obtained after the training phase.
Another way to enhance the learning process (Julian et al., 2019) is to use decision trees where the models are almost as big as the LU tables. Hence, the use of neural networks is more recommended. However, they can show some limitations in handling the data specificities, such as discontinuities (Bak and Tran, 2022). For instance, the command switch from weak-left to strong-right, only a few seconds before the collision, corresponds to the relative position angle θ wrapping from −π to π. This discontinuity in the network input between successive steps is a substantial candidate root cause of the eventual near-mid-air collision. Hence, it makes it more difficult for the NN to regress the problem, causing errors at the boundaries between the predicted instructions.

Regardless of the model architecture, the data needs to be analysed and may be represented differently by the NN. Indeed, when this task is viewed as a classification problem based on the parameters' configurations (i.e., Model features), an instruction will correspond (i.e., An output class). However, in specific scenarios, several instructions may be relevant. For example, if the intruder arrives opposite the ownship, turning right or left will resolve the conflict in both cases. Thus, in specific configurations, the model may give a suitable solution but is different from that of the table, and consequently, its performance will be underestimated by the metrics used. For this reason, one solution may be to define, in each scenario, optimal instruction and secondary instructions, if any. This would avoid penalising the model too much if its predictions are consistent but not optimal. In (Julian et al., 2019), a similar approach was evaluated using an asymmetric MSE as a training objective.
Finally, the ACAS Xu LU tables are not balanced for a classification-like ML problem. In fact, most of the configurations provide the same instruction: "*to do nothing*" (COC).

More analysis of task 1 of the MLEAP project can be found in Chapter 4. They are confirming that this imbalance would have an impact on model training. This imbalance can be handled differently by using weighting approaches (cf. section 5.8.3.1.3.5) to balance the effect of the classes. Another way can be to reduce the number of configurations whose output instruction is COC, or to increase the number of those corresponding to the other instructions (WL, WR, SL and SR). However, reducing the size of the dataset means that the model will not learn the entire ODD (as meant for the ACAS Xu models). So, this cannot be an option since the learning would no longer be consistent with knowing the tables perfectly. The non-COC cases can be increased by adding a small δ to one or more variables in existing configurations, while ensuring that the original data and the synthetic data give the instructions. For example, considering only one variable, rather than having only one point for $v_{int} = 200$ *ft/s* and for r $v_{int} = 201$ *ft/s*, if these two configurations give the instruction WL, another data point could be created for $v_{int} = 200.5$ *ft/s* whose instruction would also be WL. This way, the number of points WL, WR, SL and SR could be increased to rebalance the dataset.

Note that achieving balance in the ACAS Xu dataset aims to diminish the challenges related to unbalanced data, which often impede machine learning performance. In ACAS Xu, this endeavour should not impact the system's safety and enables preserving its ability to execute manoeuvres while mitigating the impact of data distribution biases.

5.8.3.1.3.5 Unbalanced dataset analysis: ACAS Xu – COC action

As mentioned in Chapter 4, the ACAS X lookup table drives the COC, resulting in manoeuvre in most cases. The trained model is prone to output the COC command and could have difficulties making good predictions for other classes. In this section, we will test two approaches to limit the impact of the unbalanced dataset on the model performances.

- The first one uses a weighted loss function, where we weigh the loss function to minimise the COC class effect on it. This will balance the impact of the COC output versus the other one.
- The second approach is more upstream, improving the dataset regarding the WR, WL, R, and L outputs. To do this, we interpolate the inputs when the resulting manoeuvre is not COC and remains the same for two input neighbours.

The experimentation (models have been retrained) was done on a single network corresponding to the previous action COC with a vertical distance of 0. The training phase is illustrated in Figure 122.

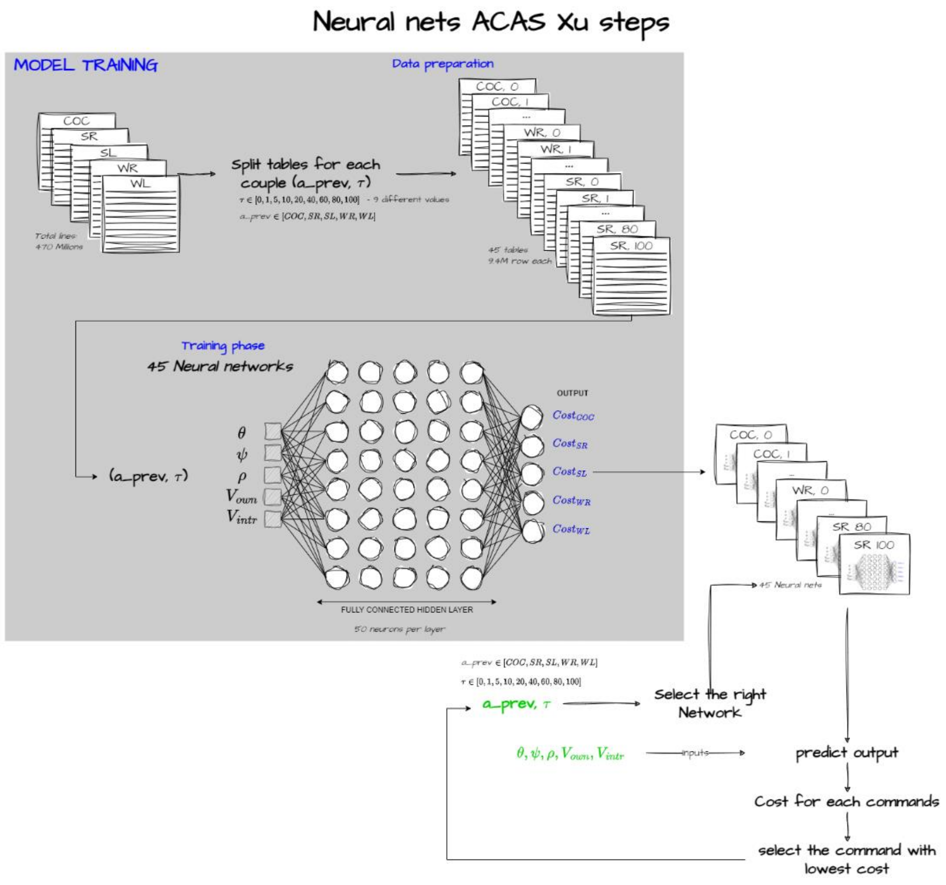


Figure 122: ACAS Xu neural network approach illustration

To ensure a fair comparison, the reference we used is a model rebuilt entirely, securing comparison with the same environment with the same hyperparameters. The effect of data augmentation or the weighted loss function has no effect as we can see in Table 52, except for bound 001. We have kept the same neural network architecture and training dataset volume A priori evaluation remains constant. Bound 1 depends on layers' parameter norms, which have been modified in the two approaches. It explains changes in a posteriori evaluation. It is not the case for the others, a non-significant modification on parameter distributions, gradients during the convergence phase, or data DB variability.

The data augmentation has not been used to increase the volume of data in the training dataset (which was already large enough; we have kept the same training dataset size in both cases), so we were not expecting any impacts on the generalisation bounds values.

The weighted loss function also minimised the bounds computation and balanced the influence of COC risk costs during the training phase.

The positive effect could have been on the training error, which was already small. So, finally, it is difficult to conclude whether both approaches have a positive influence on generalisation. The benefits should be more focused on the stability and robustness of the models. It is essential to notice that in "data augmentation," we have changed the data distribution, which could lead to bias introduction and would have to be considered in the validation phase.

		Reference		w/ data augmentation		w/ weighted loss function	
		A priori evaluation	A posteriori evaluation	A priori evaluation	A posteriori evaluation	A priori evaluation	A posteriori evaluation
BOUND	001	41,9	2,2	41,9	5,2	41,9	2,5
	002		1,6		1,6		1,6
	003	1,23	0,014	1,23	0,014	1,23	0,014
	004	0,17	0,06	0,17	0,06	0,17	0,06
	006	0,06	0,008	0,06	0,008	0,06	0,008
	007	2,5	2,5	2,5	2,5	2,5	2,5
	010	8	3,6	8	3,6	8	3,6
	011	0,6	0,05	0,6	0,05	0,6	0,05
	012	3,6	3,6	3,6	3,6	3,6	3,6

Table 52. Generalisation bounds comparison for ACAS Xu use case with data augmentation or weighted loss function

5.8.3.1.3.6 Alternatives

5.8.3.1.3.6.1 Uncertainty Quantification

Uncertainty quantification involves providing a range of possible outcomes or beliefs due to imperfect information. Quantifying the uncertainty in a trained model relative to a given input example requires first identifying the different sources of error and then combining those sources into an uncertainty associated with the predicted quantity.
With generalisation bounds, we assess the true risk in the full domain, which is difficult to calculate because our ODD's probability distribution is unknown.

$$\mathcal{R}(h) = \mathbb{E}_{(x,y)\sim P(X,Y)}[L(y, h(x))]$$

In the case where we can bound the gap between the model prediction and the correct value for all inputs of the ODD, the loss function is bounded on the entire domain, and the empirical risk is also lower than this limit. But, in general, this uncertainty depends on the input, and we are back to our issue with the unknown distribution of the entire domain.
Nevertheless, uncertainty quantification could support it by guaranteeing model performance on unseen data.

- Support model evaluation, providing information on the trained model's ability to generalise and identify hidden biases or failure scenarios.
- Identification of predictions that are not trustworthy or have significant uncertainty and support risk management.

5.8.3.1.3.6.2 Conformal prediction

Conformal prediction offers distribution-free marginal guarantees of coverage at any user-defined level. In the article (Zecchin et al., 2024), the authors are proposing « an upper bound is derived on the expected size of the conformal prediction set predictor that builds on generalisation error bounds for the base predictor ». This approach enables the identification of a reliable set of predictors and relies on calibration dataset size. The average prediction set size is more significant when the

generalisation error of the base predictor is more significant. So, the knowledge of the prediction set size could support assessing the model's generalisation ability.
Those two methods were briefly assessed as part of MLEAP but would require further investigation to link them with generalisation guarantees.

5.8.3.2 Application of the selected bounds to aeronautical use cases

This section is dedicated to application of selected generalisation bounds in a naive way (applied without consideration of potential model adaptation), that is as a measure applied on model. As some of our selected bounds required information from the training phase such as the initial value of the parameters or gradient calculation during each epoch, it has not been calculated for bounds 002, 007, 010, 011, 012. For the bound 013 (Kawaguchi's bound), the bound is vacuous as the size of $|\mathcal{H}_{val}|$ is infinite (we have an infinite number of functions h_θ that will fit the assumptions on $\kappa_{h_\theta,i} = \mathcal{R}(h_\theta) - L(h_\theta(x_i), y_i)$).
Automated Visual Inspection and ATC STT models use complex architecture. We approximate the neural networks using the equivalence between a fully connected network and convolution or transformer layers to calculate the bounds.
The AVI and ATC STT networks contain many trainable parameters (resp. 7M and 310M), significantly affecting the bounds. So, with the limited volume of data available for AVI and STT, it is tough to achieve tight generalisation bounds.
Based on the generalisation bounds theory, the volume of train data to achieve tight bound should be larger than 10^9 ; currently, this is a scientific challenge explaining the generalisation of deep learning models with a reasonable volume of data as theories are unable to explain empirical performance on unseen data for deep neural networks.
Table 53 summarises the results obtained, showing a massive gap between the statistical guarantee expected and the best we can get with a single « measure » done for the AVI and ATC STT use cases (the bounds are vacuous) in a priori and a posteriori evaluation. For those use cases, using generalisation bounds to provide quantifiable generalisation guarantees (LM-04) on the full ODD is inefficient as it says that with a high probability, we will have the performance gap lower than a large upper bound. This statement is true before and after having trained. Guarantees obtained are verified regarding the test dataset (LM-14) but do not provide an acceptable assurance level regarding the model performance on unseen data.

		BOUND			
		001	**003**	**004**	**006**
AVI dents	A priori evaluation	219999	2250	28372	1609964
	A posteriori evaluation	4642	52	15	468
ATC STT	A priori evaluation	810	4.10^7	1.106	3.10^9
	A posteriori evaluation	7	2255	90	16425
ACAS Xu	A priori evaluation	0.9	1.2	0.11	0.02
	A posteriori evaluation	0.1	0.014	0.06	0.008

Table 53. Generalisation bounds computation on a trained model for the aeronautical use cases

In the use case ACAS Xu, the number of parameters is relatively small (around 13000), and the volume of data to train the model is huge, driving a small value for the generalisation bounds. With high probability, the true risk is close to the empirical one calculated on the train dataset. Numerically, this means that the average prediction error on the full domain is lower than a few per cent.
A priori evaluation provides good insight into the model architecture selected, which can maintain good performance on unseen data (LM-04). For a posteriori evaluation, having tighter bound values and verifying that the gap between loss functions on train and test datasets is lower than the obtained bound provide confidence that the optimum obtained is a good solution regarding generalisation.
In this use case, the minimum cost associated with the command is the primary outcome, and the difference between the costs (as illustrated in the bottom right area of Figure 122) for the same input is slight. Then, achieving perfect prediction is not enough to ensure good results in predicting the correct output. For example, for a given input, we could have 1% difference between output costs, so if the error on each cost is higher the 0.2%, the command associated with the minimum cost could be changed, dealing with a risk of wrong command output. It highlights that generalisation bounds provide an upper bound on the gaps between generalisation error and empirical error, which is not directly linked to the accuracy of the trained models.
Those results are aligned with the analysis performed on the Fashion MNIST toy use case. It shows that when neural networks are too deep, this measure cannot provide tight enough statistical guarantees regarding the average performance of a trained model on unseen data.

5.8.3.3 Main Takeaways

This section's main objective was to analyse the existing pipelines, including the built data and models w.r.t the target objectives of each use case. To do so, the datasets and model performances have been investigated regarding the models' selected performance measures and training objectives. The primary outcomes have shown an alignment between both use cases ATC-STT and AVI, regarding the remarkable gaps between the target objective and the current state of model performances and behaviour. Concerning the ACAS Xu, this one has some specificities, as detailed in section 5.9.3.1.3 regarding the generalisation assessment and verification, since ACAS Xu aims to have AI models that can reproduce the same outputs as presented in input tables.

In the ATC-STT analysis, the AIRBUS models were designed with the main criterion of being able to run in a real-time embedded system, unlike the transformer-based ones (Wav2Vec) models, which were developed to produce the best possible transcriptions, independently of the real-time issue. Based on these objectives, different design choices are made. However, in both cases, models were trained with the same purpose to minimise the WER values only, which was not enough to meet the industrial objective of a maximum 10% WER, in different English-spoken accents. The data used for training the AIRBUS model comes from French airports only, which makes the trained models (KALDI-based model) less robust while tested on a more significant number of accents. Meanwhile, for the Wav2Vec models, despite the diversity of accents in the training data, there is an essential gap between the obtained performance and the targeted ones.

The analysis of the AVI case revealed several limitations in the generalisation study. Indeed, despite the consistency with the target objectives in terms of targeted performances and the evaluation measures and training objective, the models performed poorly on the test datasets. Several elements have impacted the performances, among which the lack of data required for optimal training of the models, compared to the complexity of the used model architecture and the number of trainable parameters. In addition, the model developed in the Airbus's use-case is based on YOLOv5, which is not the latest one, YOLOv8[164] has been released and performs better. Note that YOLOv8 allows segmentation[165] which can be used for determining the severity of defects. Hence, it can be more suitable for meeting the target system's objectives.

The analysis of ACAS Xu is different from the two previous case studies. Here, the objective is not focused on a generalisation guarantee but rather on monitoring the training and learning process of the model. A-priori and a-posteriori evaluation of the generalisation bounds is performed in the experiments run during this project. The main takeaway is that the two methods employed to mitigate the impact on the trained model's performance in unbalanced datasets, specifically the data augmentation and the weighted COC class, did not yield distinct positive outcomes regarding generalisation assessment. However, there is potential for these approaches to enhance model stability and robustness, offering valuable advantages despite the lack of explicit improvement in generalisation. Besides, uncertainty quantification and conformal prediction analysis represent alternatives and complementary tools to generalisation bounds, and they may provide insights into the model's behaviour, contributing to a more reliable and resilient ML development.
Several existing state-of-the-art approaches can be adopted based on the model's optimisation. For instance, in (Bak and Tran, 2022; Lopez et al., 2021) closed-loop methods are proposed to ensure, or not, that neural networks provide instructions that preserve aircraft safety during a predefined period. Another study (Julian et al., 2019) showed that using neural networks made it possible to correctly learn the behaviour of the table, where for some classes of setpoint, an accuracy of 94% is obtained. The study of system design also showed that considering the problem as a simple classification task gives the best performance. One perspective of this use case is to consider its integration into drones. Intruders' heading, speed and position must be known to be used. This requires the use of sensors, which introduces measurement errors, such as a poor estimate of the position or speed of the intruder. It may, therefore, be a good idea to try to introduce a representation of this bias created by the sensors when learning the model.

Finally, to achieve better performance, the alternatives considered for each use case had different impacts, and there were other lessons to be learned. For instance, in the AVI use case, the analysis of a more complex model (YoloV8) has shown better performances, while the data augmentation did not boost the results as expected. One typical result of the toy and AVI use cases is that exploring different architectures has clarified the correlation between task complexity and the model's adequate capacity. Furthermore, applying selected generalisation bounds on aeronautical use cases confirms that applying on relatively small networks could provide tight bounds for the generalisation gap. This provides confidence that the trained model will keep behaviour noticed with the test

[164] https://docs.ultralytics.com/
[165] https://docs.ultralytics.com/tasks/segment/

dataset. We cannot conclude deep neural networks as the calculated bounds are vacuous. Besides, the two approaches we used to limit the unbalanced dataset effect on the performance of the ACAS Xu trained model have not delivered precise positive results regarding generalisation; nevertheless, the benefits could be on model stability and robustness.

5.9 Conclusion and Perspectives

5.9.1 Outcomes of Generalisation Analysis

In machine learning and deep learning, the key to the success of any approach is to provide a robust enough model that could generalise well on unseen data during training without a drop in performance when moving from one environment to another. In this chapter, we have comprehensively analysed the state-of-the-art ML/DL generalisation assessment using bounding functions and evaluated several metrics that could give an average value of the model performance on the test dataset. Several issues and limitations have been identified regarding the standard practices that are less likely to improve results. For instance, the weak data processing when some hypotheses are violated (e.g. IID hypothesis in test, train and validation datasets), the miss-use of generalisation bounds (these are tools that could help to understand the model behaviour and the task complexity, but they could be misleading when they are misunderstood), a less rigorous evaluation (e.g. when inaccurate evaluation measures are used), and the resulting gap between the evaluation objectives and the industrial needs (i.e. KPIs are not reflected in the experimental benchmark), as well as the non-adapted testing schemes and scenarios w.r.t ML-based systems (e.g. when massive testing is widespread on system development, when scenarios do not include enough data and testing workflows regarding the ML component, this later will not be evaluated thoroughly).

The experimentations run in this chapter were two folds: first, the set of selected generalisation bounds have been analysed in a toy use-case, FashionMNIST (cf. section 5.8.2), before being applied to the aviation use cases (cf. Chapter 3 to know more about the selected aviation use-cases) that are more complex with a higher data dimensionality. This analysis aims to understand their applicability conditions and analyse the gap between the theoretical application and the empirical results when small data and models are used. To do so, a simple image classification task has been explored. The primary outcomes have shown that the bounds do not provide accurate and self-sufficient means to guarantee the generalisability of the used models. Further analysis has been performed using other data types (e.g., open-source STT-ATC datasets of section 3.3.2) to verify these findings.
Besides, some differences were noticed in the generalisation bounds' behaviour when applied in a-priori and a-posteriori model training. This one has shown that when the generalisation bounds are applied before model training, their values seem larger than a-posteriori. Besides, in this latter, all evaluated bounds consistently exceed a threshold that ensures the accurate behaviour of the trained model on unseen data. The empirical generalisation gap, when computed on adequately trained models, if it exhibits stable learning curves, means there is no overfitting, indicating that the models are not overfitting. They are maintaining strong performance on unseen data.

The second experimental part has studied the selected aviation use-cases, ATC-STT, AVI and ACAS Xu, in terms of the data and models provided by proprietary Airbus projects, compared to other open-source data and models. The objective is to investigate the alignment of the industrial objectives targeted by these applications with the construction of the data and model development pipelines that have been implemented, including the quality and volume of the datasets, the choice of model architectures and their configurations, as well as the training objectives and performance evaluation measures. The analysis revealed a considerable gap between the target objective and the existing state of model performances and behaviour. For instance, in the ATC-STT analysis, the perfect model would have been trained on various accents and collected in different airports to avoid airport-related biases. The Wav2Vec architecture also seems suitable for the automatic transcription task. In fact, after fine-tuning, the transformers have shown results that are not far off the performance of the AIRBUS KALDI-based model, while more improvements are needed to meet the target WER objective (less than 10%).

The second use-case AVI analysis was affected by the fact that less data was used to train a large model. As mentioned before (cf. section 5.5.4), the volume of data should always be greater than the number of model parameters and the task complexity. However, estimating the data needed for training is not straightforward; it can be estimated empirically. Besides, the used model enables the detection and classification of the surface defects but does not allow the determination of the severity level, which is one of the most important objectives of the target system. To do so, a segmentation model is explored. Namely, YOLOV8 outperforms the YoloV5 model with an accuracy of around 95% and is the closest to the objective targeted by the AVI application. The segmentation aims to identify individual objects in an image and segment them from the rest. Hence, the output of an instance segmentation model is a set of masks or contours that outline each object in the image, along with class labels and confidence scores for each object.

Finally, in the ACAS Xu analysis, the models are more performant and able to predict the suitable actions for collision avoidance, w.r.t the LUT. However, at the system level, the only consideration of a good model is that this one should be able to reproduce the same data in output as provided in input. Even if the predicted actions are correct, in terms of collision avoidance, they are considered errors if they do not match the LUT content. Another limitation of excellent model training is the balance regarding the COC class (precise of conflict) that suggests the ownship do nothing. In terms of data balance, this class is much more present than the other actions (classes), while this imbalance is necessary since the COC decision is preferred in terms of system safety. To deal with this, a weighting function is used to limit the impact of unbalanced data, as well as the augmentation of the other class instances. However, experts must analyse and validate the drone data after this augmentation.

Finally, the work done about generalisation in this project came with several alternative explorations; instead of ML models design and development (in models setting and evaluation), the use of different approaches to meet the target KPIs for each use-case (explore new models and methods to implement the tasks), or even to ensure that the model will perform good results after training, meaning that the generalisation capabilities are assured. To do so, the model's different properties and generalisability have been explored empirically and joined to the generalisation assessment using the statistical tools, namely the generalisation bounds and reminding that the learning management, as part of the W-shaped learning assurance, allocates essential steps for the learning management (LM) and verification tasks, among which several LM objectives have been targeted (cf. section 1.5.1.2) in this project to provide means to guaranty models' generalisability. As a result of this task, a comprehensive way forward to perform *"Generalisation"* verification is provided, leveraging the different outcomes of task 2, in correlation with other elements related to EASA's guidance for AI systems of levels 1 & 2 (EASA, 2024).

5.9.2 Generalisation – Revision and Assurance

As a result of MLEAP, we introduce the generalisation assurance aspect that, when verified, enables more trust in the target AI-based system. Besides, in addition to ML/DL model's generalisation verification and assurance, the aviation regulation in the certification specification CS25.1309[166] and its acceptance means of compliance (AMC) for large transport aircraft recognise that in safety assessment, there are some uncertainties and that uncertainty shall be accounted in a way that does not compromise safety. Hence, for the product to be certified, conservative assumptions should be made when there are uncertainties. For ML-based systems, this can be addressed by providing accurate or conservative measurements about the generalisation.

The generalisation assessment of machine learning models is based on the ability of the model to maintain good performance when face with unseen data. However, providing guarantees on this aspect is not straightforward, as explained in the different results and model analyses. The generalisation bounds are statistical tools that provide a wide margin for risk assessment related to the model being designed, the data, and the task being addressed. Nevertheless, the generalisation bounds theory relies on strong assumptions concerning the data representativity, and they are trusted to verify the hypothesis being assumed concerning the data. However, in real life, this is not the case. Hence, it is necessary to complete this analysis with different quantitative analyses of the model's performance to have some assurance about its generalisability.

Quantitative criteria must be provided to ensure the model's generalisation, considering various aspects of model performance, uncertainty, error analysis, and data characteristics. For instance:

- Apply statistical hypothesis testing to rigorously compare the performance of different models or configurations. Techniques such as hypothesis testing for mean performance differences or non-parametric tests can provide statistical evidence of performance disparities;
- Simulate[167] model performance under different scenarios or environmental conditions to assess its generalisation capabilities, including already-known operating conditions. Monte Carlo simulations or sensitivity analysis techniques can provide insights into how the model behaves in diverse real-world contexts;
- Apply conformal prediction techniques to provide valid confidence intervals for individual predictions. Conformal prediction constructs prediction regions/intervals around each prediction, allowing for probabilistic correctness guarantees at specified confidence levels. To

[166] Check the certification specifications for more details: [CS-25 Amendment 28 | EASA (europa.eu)]()
[167] Note that the performances simulation is based on assumptions that are done on the environmental perturbations that could occur after the model's implementation.

calculate the uncertainty in a prediction, a prediction interval is computed to estimate the interval the observation will fall in with a certain confidence level. However, this will not be enough to assess the model's ability to generalise, but it gives an estimation of it;

- Loss optimisation and learning behaviour analysis. This will measure how well the model fits the training data. Lower training loss indicates better optimisation. Besides, validation loss computed on a separate validation set helps monitor model performance during training and detect overfitting or underfitting;
- Have a wide range of performance measures, selected carefully, aligning experimental and industrial objectives. Accuracy, precision, and recall can be relevant but insufficient to represent the real KPIs. These measures will then be computed and values verified and validated with the domain experts;
- Conduct a rigorous data analysis and understand the dataset's distribution of features and labels. This helps quantify the proportion of instances in each class to identify imbalanced courses that may affect model performance. In addition to assessing the importance of different features in predicting the target variable. This can be measured using techniques like feature importance scores or SHAP values (S. M. Lundberg, 2017);
- Error analysis of the trained mode. This is a vital process in diagnosing errors an ML/DL model makes during its training and testing steps. Especially after the training phase, this step will help evaluate the models' performance and identify areas for improvement. The objective is to provide a comprehensive explanation of 5% or 1% error margins for experts and decide about the acceptability of the model's performances, mainly in critical use cases, where a 1% error margin cannot be tolerated.

By incorporating these – among other – quantitative criteria and other verifications and evaluations to be added depending on the target use case into the definition of "*generalisation assurance"*, we can have a kind of *"guarantee"* about the model performances, with a comprehensive evaluation of ML models, focusing not only on predictive accuracy but also on uncertainty, error patterns, and data characteristics. This holistic approach enables a deeper understanding of model behaviour and enhances trust in model predictions across various applications.

6. Model evaluation: robustness and stability

6.1 Introduction

6.1.1 Background

Methods to evaluate ML models' robustness and stability are rushing from academic prototypes to more industrial tools. These methods can be classified roughly into three categories: statistical, formal and empirical. They can also be combined. Statistical approaches usually rely on a mathematical testing process and can illustrate a certain confidence level in the results. Formal methods rely on formal proofs to demonstrate a mathematical property over an ODD. Empirical methods rely on experimentation, observation and expert judgement.
In the chapter, we will study the approaches developed by the available tools and methods, the target properties, and their domain validity.

6.1.2 Scope of the chapter

This chapter aims to study the relationship between concepts developed in the EASA CP and ISO/IEC concepts regarding the robustness and stability of machine learning models.
First, the chapter relies on the terminology associated with the robustness and stability concepts detailed in Section 1.2.4.1, specifically between the EASA CP, the CoDANN documents and ISO/IEC literature. The chapter studies how to align related concepts from all sources to support the MLEAP program with the ongoing standardisation work. Then, this chapter aims to analyse within each type of method which one is mature enough and what kind of robustness properties it allows to obtain. For each, a study of the maturity of tools associated (if any) with the method will be done.

6.1.3 Robustness and stability

6.1.4 Robustness and stability concepts

Section 1.2.4 introduced robustness and stability concepts as components of an ML system's trustworthiness. These two concepts are obviously related to the model performance, how it has been trained (see Chapter 5 for more), and the data used (see Chapter 4 for more). In this chapter, we aim to detail what is currently available to assess robustness and stability beyond the initial performance metrics (see section 2.1 for more).

6.1.5 Robustness and stability properties already available

From the literature, several properties can be tied to robustness and stability. This section lists (non-exhaustively) available properties and their associated criteria[168]. For the whole chapter, we use the following terms (from the CoDANN documents):

- X is the input space;

[168] The document uses the word criterion in the ISO/IEC sense as *rules on which a judgment or decision can be based, or by which a product, service, result, or process can be evaluated* (ISO/IEC/IEEE 15289:2019).

- n is the size of a data set;
- $x \sim \mathcal{X}$ denotes a random variable sampled from the probability space $\mathcal{X} = (X, P)$ with a probability distribution P;
- $\mathbb{E}$ denotes the expected value of a random variable;
- $\mathcal{F}$ represents the training algorithm, and by a slight abuse of notion is also the set of possible models produced by this learning algorithm. $\hat{f}: X \to Y$ is a function resulting from the training algorithm on a training set $D \subset X$ which approximate the function $f: X \to Y$;
- An average model $\overline{f_n} = \mathbb{E}_{D \sim \mathcal{X}^n}[\mathcal{F}(\mathrm{D})(\mathrm{x})]$. $\overline{f_n}$ is mostly a theoretical tool which is not possible to compute in most cases. Intuitively, the average model can be interpreted as a "theoretically idealized" model one could produce by training algorithm $\mathcal{F}$ on all possible data sets of size n.

6.1.5.1 **Local vs. Global Properties**

The current state-of-the-art methods to assess the robustness or the stability of machine learning models tends to be focusing on local properties (e.g., (Z. Zhong, 2010), (Singh et al., 2018), (K. Leino, 2021)) and not often on global ones (e.g., (Leino K., 2021), (W. Ruan, 2019), (Z. Wang, n.d.)).
It is more common to verify local robustness properties than global robustness properties, as the former is more straightforward to specify and verify. Local robustness properties are determined by comparing a sample input from the test data set. For example, given an image classified with a given outcome, the local robustness property can specify that all images generated by applying some transformation are still classified with the same result. Since local stability or robustness does not entail local performance, monitoring an AI component at the system level should not just be determined by its local stability. However, this assumption can be nuanced since, for some systems, some local properties can be a (statistical) discriminator of performance. Also, a drawback of verifying local stability or robustness properties is that the guarantees are local to the provided test sample and do not extend to other samples in the data set.
In contrast, global robustness properties define guarantees that hold deterministically over all possible inputs (Katz G., 2017). For ODD, where input features have semantic meanings, such as air traffic collision avoidance systems, the global properties can be specified by defining valid input values for the input features expected in a real-world deployment. Defining meaningful input values is more challenging in settings where the individual features have no semantic meaning (e.g., the input space of all possible images).

6.1.5.2 **[Model] Bias (CoDANN-1 and EASA CP)**

Model Bias is defined as an error caused by erroneous assumptions in the learning process. High bias can cause an algorithm to miss the relevant relations between attributes and target outputs (underfitting; for more, see Section 5.3).
The quantity bias $(\mathcal{F}, n)$ is the average over all points $x \in X$ on the difference between the average model and the target function, that is:

$$bias^2(\mathcal{F}, n) = \mathbb{E}_{x \sim \mathcal{X}}\left[(\overline{f_n}(x) - f(x))^2\right]$$

Intuitively, the bias of a learning algorithm can be interpreted as a measure of how well the average model deviates from the true one and thus is a measure of model quality. One wants to make the bias small to have the average model close to the true function f.

6.1.5.3 [Model] Variance (CoDANN-1)

It is defined as an error caused by sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data rather than the intended outputs (overfitting; for more, see Section 5.3).
First, for every fixed $x \sim \mathcal{X}$, the variance of $\mathcal{F}: D \longmapsto \hat{f}^{(D)}$ is the average distance to the value of the average model $\overline{f_n}$:

$$var(\mathcal{F}, n, x) = \mathbb{E}_{D \sim \mathcal{X}^n}\left[\left(\hat{f}^{(D)} - \overline{f_n}(x)\right)^2\right]$$

Second, averaging this quantity over all $x \sim \mathcal{X}$ gives the variance of the learning algorithm:

$$var(\mathcal{F}, n) = \mathbb{E}_{x \sim \mathcal{X}}[var(\mathcal{F}, n, x)]$$

Intuitively, a learning algorithm's variance can be interpreted as a measure of its fluctuations around the average model and thus reflects its stability.

6.1.5.4 Learning Algorithm Stability (EASA CP)

The LM-11 objective from EASA CP refers to learning algorithm stability by pointing to CoDANN-1. This latter refers to it as the type of robustness ensuring that the produced model keeps a defined level of performance under perturbations of the training data set. Learning algorithm stability ensures that the trained model is not too sensitive to the training data and will not deviate much in case the training set changes.
The evaluation of the distance between two machine learning models is not defined here. It cannot be directly measured through the difference in weights and biases of the two models because of the random nature that can occur during training. Instead, it would be defined using the gap in performance between the two evaluated models by using metrics such as bias, variance, robustness, and relevance. For that evaluation, a threshold should be defined.

Following the definition used in the CoDANN-1 we propose the following formalization for the learning algorithm stability. Given two training data sets $D_{train} \subseteq X$ and $D'_{train} \subseteq X$ such as $|D_{train} \cap D'_{train}| > N, N \in \mathbb{N}$, a training function $\mathcal{F}$ and a performance function P evaluating the performance of a trained machine learning model over a test set $T \subseteq X$. A learning algorithm stability correspond to:
$|D_{train} \cap D'_{train}| > N \Rightarrow \|P(\mathcal{F}(D_{train}), T) - P(\mathcal{F}(D'_{train}), T)\| < \varepsilon$, where $\varepsilon \in \mathbb{R}_+$.

6.1.5.5 Stability (ISO/IEC)

6.1.5.5.1 Trained model or inference model stability properties

The property is first introduced in ISO/IEC 24029-1 (ISO/IEC, n.d.) as “stability property”. This notion corresponds to the EASA CP property of the trained model stability property.
A trained model stability property expresses the extent to which a neural network (and here, by extension, a machine learning model) output remains the same when its inputs vary over a specific ODD. Checking the stability over an ODD where the behaviour is supposed to hold allows for checking whether or not the performance will have too. A trained model stability property can be expressed either in a closed-end form (e.g., “is the variation under this threshold?”) or an open-ended form (e.g., “what is the largest stable space of the input space?”).

A stability property allows checking if a machine learning model remains performant in noisy inputs. To be usable, a trained model stability property requires that the input space presents some regularity properties. For example, it is not adapted for a chaotic system since its results will not be relevant. When the regularity of the function to approximate over the ODD is not easy to affirm (e.g., chaotic function), it can still be helpful to use the trained model stability property to compare different machine learning models. For example, measuring each system's maximum stable space around the same input is possible.
The property is then being implicitly extended to the inference model stability by ISO/IEC 24029-2 (ISO/IEC, 2022b) by the requirements expressed by the document in the Deployment phase. Indeed, at this phase of the life cycle the system should be verified to have the same performance on the targeted hardware. This notion is, therefore, equivalent to the inference model stability of the EASA CP.

6.1.5.5.2 Trained model or inference model stability criterion

A trained or inference model stability criterion establishes whether a trained or inference model stability property holds within a specific ODD, in contrast to a specific set of examples or a subset in the ODD, such as training or validation data sets. The inference model is obtained by transforming the trained model to be adapted to the targeted platform. A trained or inference model stability criterion can be checked using formal methods described in 6.2.2.1.
To be precise, according to the ISO/IEC 24029 standards (ISO/IEC, n.d.) and (ISO/IEC, 2022b):

- A trained or inference model stability criterion defines at least the ODD, the output value space on which it has been measured, and the trained or inference model stability property expected.
- A trained or inference model stability criterion may be used as one of the criteria to compare models.

To be a fair comparison, the machine learning models must have performed the same tasks, the criterion must have been used on the same ODD, and the criterion needs to have the same objective to be proven.
For example, for a machine learning model performing classification, a trained or inference model stability criterion assesses whether a particular decision holds for every input in the ODD. For a machine learning model performing regression, a trained or inference model stability criterion assesses whether the regression remains stable on the ODD on which the inferences are made.
To be applicable, a trained or inference model stability criterion relies on pre-existing information of the expected output of the machine learning model. The user can know this information or can determine it by other means (using simulation or solver systems). It is well-suited to assess the stability over an ODD where the expected answer is known to be similar. For this reason, a trained or inference model stability criterion is especially recommended for any decision-making process handled by a machine learning model (e.g., classification, identification).
Following the definition used in the CoDANN-1 a stability criterion would be defined over the ODD noted D as:
Given $x \in D$ and $\delta \in R_+$ then $\|x' - x\| < \delta \Rightarrow \hat{f}(x') = \hat{f}(x)$, where $x' \in X$.

6.1.5.6 **Sensitivity (ISO/IEC)**

6.1.5.6.1 Trained model or inference model sensitivity property

The property is first introduced in ISO/IEC 24029-1 (ISO/IEC, n.d.). A sensitivity property on a neural network (and here, by extension, on a machine learning model) expresses the extent to which the output of a machine learning model varies when its inputs are changed. To assess the robustness of an ODD, it is sometimes necessary to check the variation of the outputs of a system. A sensitivity analysis can be carried out to determine how much the system varies and the inputs that can influence the output variation. This analysis is then compared to a pre-existing understanding of the system's expected performance. The sensitivity concept is quite close to stability, except it does not require that the output does not change *at all* (as for a classification task) but that it does not change *too much* (as for an interpolation task).
Sensitivity analysis over an ODD proves that a machine learning system stays bounded (Hess D. E., 2007). As is the case for the trained or the inference model stability property, sensitivity analysis can be more suited for ODD where the function to approximate presents some regularity properties. As for the stability property, by following the requirement in ISO/IEC 24029-2 (ISO/IEC, 2022b) in the Deployment requirements, the sensitivity property can be applied to the inference model and not only to the trained model.

6.1.5.6.2 Trained model or inference model sensitivity criterion

Since a sensitivity criterion expresses a property over an ODD (and not just a specific set of examples), it can be checked using formal methods described in 6.2.2.1.
To be precise, according to standard 24029-1 (ISO/IEC, n.d.):

- A sensitivity criterion describes at least the ODD on which it has been measured and the sensitivity thresholds to be checked.
- A sensitivity criterion may be used to compare different machine learning model architectures or trained models. To be a fair comparison, the machine learning models have to perform the same task, and the criterion has to be used on the same ODD and have the same objective to be proven.

A sensitivity criterion is especially well-suited for machine learning models performing interpolation or regression tasks. These kinds of tasks often allow direct proof against a ground truth that can hold over the ODD.
A sensitivity criterion is usually expressed in a closed form as a threshold of variation over a specific ODD noted D.
Following the definition used in the CoDANN-1 a sensitivity criterion would be defined as:
Given $x \in D$ and $\delta \in R_+$ then $\|x' - x\| < \delta \Rightarrow \left\|\hat{f}(x') - \hat{f}(x)\right\| < \varepsilon$, where $x' \in X$ and $\varepsilon \in R_+$.

6.1.5.7 **Relevance (ISO/IEC)**

6.1.5.7.1 Relevance property

The relevance property is first introduced in ISO/IEC 24029-1 (ISO/IEC, n.d.). This property helps to evaluate a model's decision, contributing to its development explainability, stability and robustness evaluation. A relevance property on a neural network (and here, by extension, to a machine learning model) expresses an order of the impact of the inputs on the outputs. For each production, a relevance property expresses the individual effect of each input on the result obtained for this output. For each production, the personal impact of each input can be sorted. A relevance property checks if the ordering obtained satisfies a user-required order. A relevance property can be checked using various methods to evaluate each input's impact. Contrary to stability and sensitivity properties, a relevance property can lead to a debate between the experts in charge of its evaluation. Indeed, two machine

learning models can have very different relevant property results, which could be considered acceptable depending on the experts. This case should be considered in the comparison protocol to be resolved. For example, a protocol may use an evaluation system (as for an inter-annotator agreement (Mathet et al., 2015)) to resolve the situation. This stems from the fact that interpretability methods used to build a relevant property give debatable results (or even contradictory results, depending on the method used).
A relevance property should be used in cases where the machine learning model performs a task that a human can do. For (ISO/IEC, n.d.), in these cases, the justification of the output of the machine learning model should be understood and checked. In contrast, in EASA CP, the objectives of explainability are applicable in every case (i.e. not only by tasks done by a human). A relevant property is essential to assert whether the system's performance can be assured for the correct reasons. If that is the case, then the system's robustness can be justified and not just asserted. A human operator may do this verification manually or automatically using references that have been checked before.

6.1.5.7.2 Relevance criterion

A relevance criterion expresses a relevance property over an ODD, which requires demonstrating a link between each input and the output. For that, a method is needed to quantify the influence of each input. Formal methods relying on symbolic calculus, logical calculus or computational techniques can be used to achieve such a goal. Examples of formal methods available to check a relevance criterion are provided in 6.2.2.1.
To be precise, according to standard 24029-2 (ISO/IEC, n.d.):

- A relevance criterion presents the ODD on which it has been measured, the expected results, or the methodology used to evaluate the results if they cannot be defined *a priori*.
- A relevance criterion allows for comparing different machine learning architectures or training outputs. In order to be a fair comparison, the machine learning models have to perform the same task, the criterion has to be used on the same ODD, and the criterion has to have the same objective to be proven.

For example, on a machine learning model performing a classification task, a relevance criterion can be used to check if the most relevant pixels are located on a specific part of the object to be identified (e.g., the wheels to determine a vehicle). For a machine learning model performing predictive analysis of a time series, a relevance criterion can be used to check if the predicted event matches a consequential logic acceptable for the user (e.g., a predictive maintenance alert for an engine can be triggered by an over-heating alarm).
A relevance criterion can be expressed for various tasks as long as a user can analyse the result. For example, a relevance criterion can be used for classification, detection, interpolation, or regression tasks. Checking a relevance criterion can be automated or can rely on human assessment. When the checking relies on human assessment, the decision can be transferred as a new requirement to automate tests as much as possible. The goal of using the relevance output as a stability indicator is to help determine the dimensions used and investigate what changes in the input can affect the output.
Following the definition used in the CoDANN-1 we propose the following formalization for a relevance criterion. Given a relevance function R that can analyse the function f on a subset $S \subseteq X$ and returns a vector $[r_1, r_2, \dots r_n]$ where $n \in \mathbb{N}$ is the number of dimensions of the space X and $r_i \in \mathbb{R}$. Each r_i is the influence of the i^{th} dimension of the input over the output of $f(x), x \in S$. Therefore, $R(f, S) =$

$[r_1, r_2, \dots r_n]$. From the vector $[r_1, r_2, \dots r_n]$ we considered the fully ordered vector $[r_1', r_2', \dots r_n']$ which is a permutation of the initial vector. A relevance criterion is an evaluation of acceptability of the vector $[r_1', r_2', \dots r_n']$ regarding an expected vector $[r_1^*, r_2^*, \dots r_n^*]$. This acceptability can be formalized in several ways, for example by using the Hamming distance or the transposition distance.

6.1.5.8 **Reachability (ISO/IEC)**

6.1.5.8.1 Reachability property

The reachability property was first introduced in ISO/IEC 24029-2 (ISO/IEC, 2022b). A reachability property on a machine learning model expresses the model's multi-step performance in conjunction with its operating environment. It checks whether an agent can reach a set of states when using the machine learning model to self-check in a given environment. A reachability property can specify either a set of failure states the agent shall avoid or a set of goal states that the agent shall reach.
Expressing this type of property requires defining an environment model that describes the effect of an agent's action on its next state. The environment can evolve either deterministically or stochastically. For a deterministic environment, the reachability property expresses whether the agent can reach a particular set of states.

6.1.5.8.2 Reachability criterion

A reachability criterion expresses a reachability property over a given set of initial states. In a deterministic environment, it can be checked using methods described in Section 6.2.2.1.3. In a stochastic environment, the criterion expresses a probability of reaching a set of states. This probability can be determined using methods described in Section 6.2.2.1.4.
A reachability criterion should be satisfied for a given set of initial states. The set of initial states can be specified as part of the criterion. Alternatively, formal methods can determine the initial states for which the system satisfies the criterion. An advantage of using a reachability criterion to evaluate a learned system is that it provides a metric on the network's performance in a closed-loop environment. Therefore, it can be used to express high-level safety properties that go beyond input-output properties.
For example, in the case of an aircraft collision avoidance task, the reachability criterion can express a requirement to avoid reaching a set of collision states given a particular environment model. In this type of use case, the stability or sensitivity at each step is crucial to computing the next steps and verifying a reachability property. In a sense, reachability tends to be viewed as a form of stability over time for a safety property (such as the one expressed by the LM12 requirement).
Following the definition used in the CoDANN-1 we propose the following formalization for reachability criterion. Given a subset $Z \subseteq X$ of forbidden states that the function f should not reach, $S \subseteq X$ a subset of starting set, and an environment $\mathcal{E}: X, Y \longrightarrow X$ which can transform a state of X impacted by the outcome of $\hat{f}$ into another, such as $\mathcal{E}(x, f(x)) = x'$. One can build a sequence of composition of the functions $\hat{f}$ and $\mathcal{E}$ where at each step both are composed as such: $\mathcal{E}(x, \hat{f}(x))$. To ease the notation, we define $\mathcal{E}^2(\hat{f}^2) = \mathcal{E}\left(\hat{f}\left(\mathcal{E}(\hat{f})\right)\right)$.
Given $S \subseteq X$ a reachability criteria are met if:

$$\forall x \in S, \qquad n \in \mathbb{N}, \qquad \mathcal{E}^n\left(x, \hat{f}^n(x)\right) \notin Z$$

6.1.5.9 Discussion

The EASA CP and ISO/IEC concepts overlap in part but are not entirely aligned. EASA CP considers a “fine-grained” notion of robustness by assessing the stability of the training algorithm, as well as the stability and robustness of the trained model and the inference model, while the ISO/IEC is almost only focused on the trained and inference models.

The CoDANN-1 document would use the notions of bias and variance, for example, to evaluate a model's stability and robustness. These concepts can also be partially found in the ISO/IEC literature through the trained model stability and sensitivity concepts to assess the robustness of a trained model or an inference model. The notion of reachability is also present and equivalent in both literatures.

However, the concept of relevance introduced in the ISO/IEC standards is not present in the EASA CP (both EASA CP and CoDANN-2 IPC report focus on the related notion of explainability) and could help expand the notions around robustness.

In conclusion, while the concepts from both sources are not completely aligned, they can be used complementarily quite easily since they do not contradict themselves and are not mutually exclusive.

EASA CP	ISO/IEC
[learning algorithm] Stability	-
[trained model] Stability	Robustness during stages before Deployment
[inference model] Stability	Robustness during stages after Deployment
[trained model] Robustness	Robustness during stages before Deployment
[inference model] Robustness	Robustness during stages after Deployment
Bias	-
Variance	Trained model stability (stability in 24029-1) Sensitivity
Relevance/explainability	Relevance
Reachability (from CoDANN-1)	Reachability

Table 54 – table of relevant concepts from either EASA CP or ISO/IEC

6.1.6 General framework of methods to assess robustness and stability

6.1.6.1 **EASA CP framework**

Depending on the system level being assessed, robustness and stability are verified through different requirements. Stability is viewed as a property stemming from the learning process, which implies that the outcome of the process has to handle the trade-off between bias and variance properly. Different methods can be deployed to assess bias and variance at this stage. Robustness is a property of the model, the ML component, and the system that protects against adverse conditions in the ODD. On each level, different approaches to assess or monitor robustness are possible. Edge cases, corner cases and out-of-distribution cases can be used to test the ability of the system to perform under these varying conditions at different levels of system design.

Learning algorithm stability	**ML-trained model stability and robustness**	ML inference model robustness	AI-based system robustness
Assessed through LM11, using performance metrics on the different versions of the trained models (e.g., bias and variance)	Assessed through LM-12 and LM13 with requirement-based robustness testing, including input perturbations and edge cases tests cases	Assessed through IMP-08 requirement through ODD monitoring, outliers, infeasible corner cases and novelty	Performance monitoring and fallback mechanism

Table 55 – table describing the different concepts of robustness and stability and the actions related to their assessment

6.1.6.2 **ISO/IEC framework**

6.1.6.2.1 AI Life cycle

In ISO/IEC literature, the life cycle of AI is organized as described in Figure 124. It is composed of several phases, starting with the Inception of the AI project and ending with the Retirement of the AI system. CoDANN documents mostly focus on phases starting at the Inception phase up to the Operation and monitoring phase. The EASA CP has a wider spectrum, which overlaps most of the life cycle described in Figure 123.

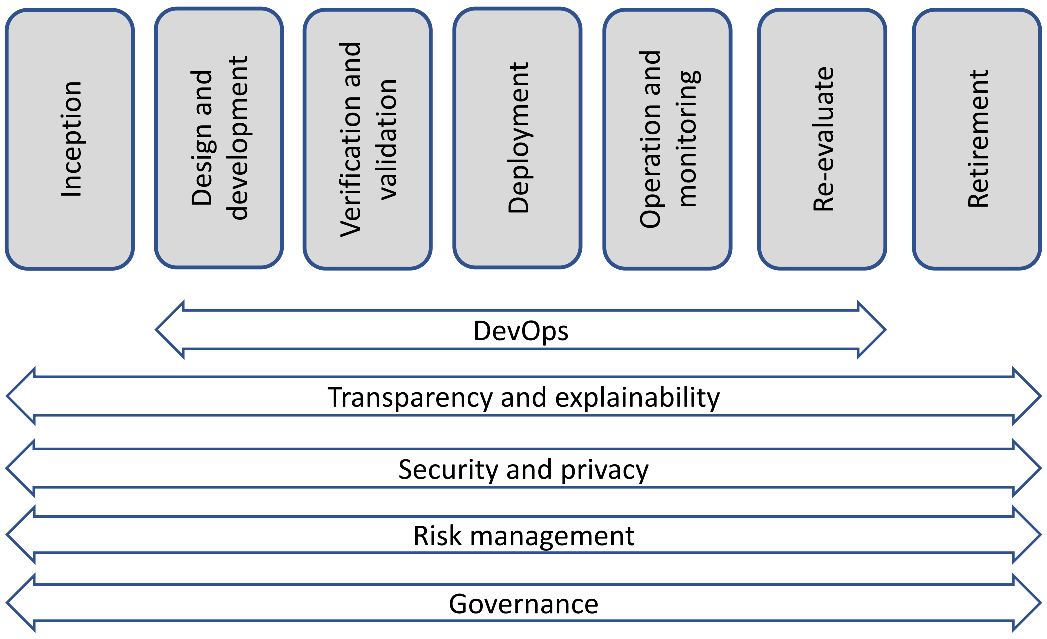


Figure 123 – ISO/IEC AI life cycle available in ISO/IEC 22989 (ISO/IEC, 2022a)

6.1.6.2.2 Correspondence between AI life cycle frameworks regarding robustness

Following the description given in 6.1.6.1 EASA CP and ISO/IEC 22989 (ISO/IEC, 2022a) life cycle and the requirement expressed along this life cycle in ISO/IEC 24029-2 (ISO/IEC, 2022b), it is possible to detect some overlap.

In ISO/IEC, the Design and development phase regroups a trained model's training and internal selection. The verification and validation steps are more oriented towards the trained model and the requirement checking. The ML constituent robustness and AI-based system robustness can be regrouped in the Operation and monitoring step. At the same time, the Deployment phase does not seem to be explicitly covered in CoDANN-1. CoDANN-2 (EASA and Daedalean, 2024) considers in Chapter 3 the impact of the transformation of a deep neural network toward specific hardware and implies the fact that it would probably require retraining the model. This is also considered in the notion of inference model used by the EASA CP. CoDANN-2 advocates an approach where the trained model adapted to hardware is a separate model that must go through every verification step. The EASA CP aims to generalise the objectives to avoid being prescriptive on one approach, which considers the inference model as derived from the trained model and defines a set of verification objectives. One of those objectives (IMP-04) aims at verifying that trained model properties are preserved. The evaluation protocol should also be applied to the inference model to ensure it respects the same properties. ISO/IEC is more aligned with EASA CP, considering the transformation of a trained model as a new step in deploying a pre-validated model.

Table 56 matches the corresponding components between the two frameworks (EASA CP and ISO/IEC). The main difficulty in this comparison is that ISO/IEC does not cover the architectural aspects. In contrast, in the EASA CP, this notion appears through the distinction between the ML model, the ML constituent, and the robustness of the AI-based system. However, this progression of concepts from the EASA CP is somewhat found in the ISO/IEC AI life cycle. Indeed, this life cycle covers the design of an ML component to deploy and monitor the model in an AI system.

Learning algorithm stability	**ML-trained model stability and robustness**	**ML inference model robustness**	**AI-based system robustness**
Design and development	Verification and validation Note: the ML inference model refers more to the Deployment phase in ISO/IEC terminology.	Operation and monitoring (when referring to ops aspects)	Operation and monitoring (when referring to Exp-15)

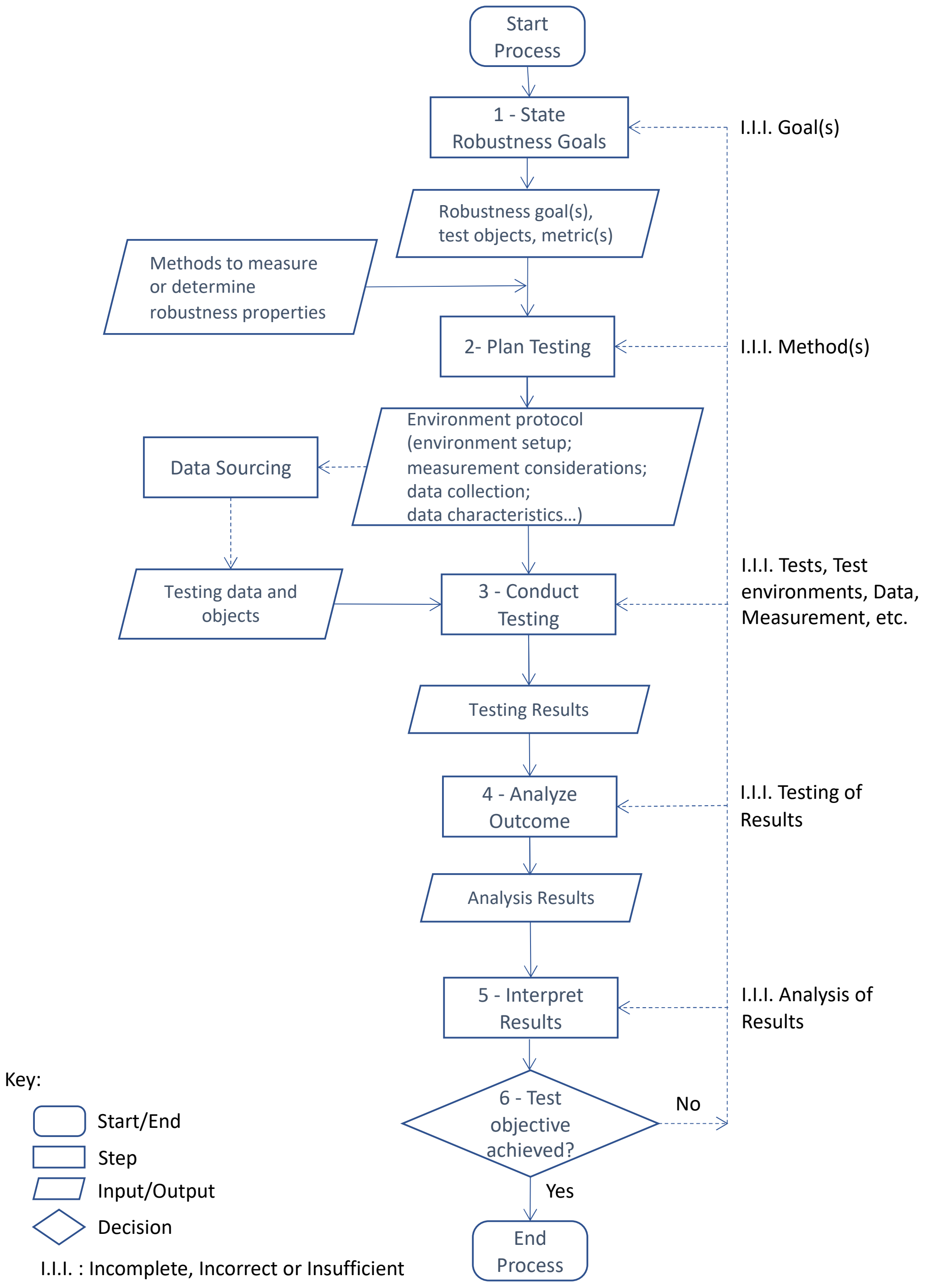
Table 56 – Correspondence between the EASA CP robustness concepts and the ISO/IEC life cycle phase.

6.1.6.2.3 Robustness assessment

In ISO/IEC TR 24029-1, robustness is assessed after the training phase. This process is not necessarily done a single time because it could fail at any step and require rolling back to a previous one. This framework does not consider the stability of the learning algorithm and only focuses on the robustness of the outcome. The process is decomposed into six steps.

1) The first step is a statement of the robustness goals. The targets to be tested for robustness are identified during this initial step. The metrics to quantify the objects that demonstrate the achievement of robustness are subsequently identified. This constitutes the set of decision criteria on robustness properties that can be subject to further approval by relevant stakeholders (see ISO/IEC 22989 – 5.19 (ISO/IEC, 2022a)). The choice of these criteria would be documented to be justified if necessary (for example, in the avionic context). The retained criteria can imply to employ one or several methods to verify them. In practice, several metrics and processes are combined to capture a better overall picture of the robustness of the model. This constitutes the set of decision criteria on robustness properties that can be subject to further approval by relevant stakeholders
2) The second step describes the methodology to demonstrate the robustness properties expected of the machine learning model. The methodology can rely on different tools. In planning the testing, the environment setup needs to be identified, data collection planned, and data characteristics defined (that is, which data element ranges and data types will be used, which edge cases will be specified to test robustness, etc.). The output of Step 2 is a testing protocol that comprises a document stating the rationale, objectives, design and proposed analysis, methodology, monitoring, conduct and record-keeping of the tests.
3) The third step conducts testing according to the defined testing protocol. Tests can be performed using a real-world experiment, a simulation, or a combination of these approaches.

4) After completion, test outcomes are analysed using the metrics chosen in Step 1.
5) The analysis results are then interpreted to inform the decision to relevant evaluators or stakeholders.
6) A decision on system robustness is then formulated given the criteria identified earlier and the resulting interpretation of the analysis results.



Start Process
1 - State Robustness Goals
I.I.I. Goal(s)
Robustness goal(s), test objects, metric(s)
Methods to measure or determine robustness properties
2- Plan Testing
I.I.I. Method(s)
Environment protocol (environment setup; measurement considerations; data collection; data characteristics...)
Data Sourcing
Testing data and objects
3 - Conduct Testing
I.I.I. Tests, Test environments, Data, Measurement, etc.
Testing Results
4 - Analyze Outcome
I.I.I. Testing of Results
Analysis Results
5 - Interpret Results
I.I.I. Analysis of Results
6 - Test objective achieved?
No
Yes
End Process
Key:
Start/End
Step
Input/Output
Decision
I.I.I. : Incomplete, Incorrect or Insufficient

Figure 124 – ISO/IEC 24029-1 generic workflow to assess robustness from (ISO/IEC, n.d.)

When the final test objectives are not met, the process is analysed, and the process returns to the appropriate preceding step to alleviate deficiencies, e.g., adding robustness goals, modifying or adding metrics, considering different aspects to measure, re-planning tests, etc.

One crucial step that is not overly detailed in the 24029-1 is Data sourcing. It is a process of selecting, producing and generating the testing data and objects needed for the testing. It can sometimes include legal or other regulatory considerations, as well as practical or technical issues. This step could constitute a severe problem in some industries since the availability of the data could be a challenge. This is, for example, the case for prediction machine learning models, which have to predict (feared) occurrence of failure of the system from data where the failure does not occur thanks to other built-in failsafes preventing them. In any case, the testing protocol contains the requirements and criteria for data sourcing. Data sourcing issues and methods are covered extensively in Chapter 4.

6.2 Review of methods and tools

6.2.1 Statistical methods

6.2.1.1 **Method**

Statistical methodology can be applied to evaluate the robustness or stability of an ML system. In short, the general method consists of choosing the data set to assess the ML system and the metrics that will be calculated. For that, the general process will select one or several metrics presented in Section 2.1 to consider them together. These metrics will then be applied to the system using the testing data to assess the robustness of the stability. This section describes the available statistical methodology to perform steps 2 and 3, described in Section 6.1.6.2 as to plan and conduct the testing. Performing a testing protocol is not unique to machine learning models, and considerations include the testing environment setup, what and how to measure it, and data sourcing and characteristics. The difference in machine learning model robustness test planning is a deeper consideration of the data sourcing (e.g., quality, granularity, train/test/validation data sets, etc.). While conducting the testing, planned data sourcing and availability of computational resources are important considerations due to the sometimes massive amounts of data and computational resources required by machine learning models.
Statistical measures of performance are applied first to a reference data set and then to one or several data sets representative of the targeted changes in circumstances. If the performance drop on the reference test set is sufficiently low for each of those, the system is deemed robust. Section 2.1 presents a detailed list of available metrics for the purpose of applying statistical methods.

6.2.1.2 **Corpus amplification**

Corpus amplification, or augmentation, is a statistical method used to ensure better coverage of testing of the system by creating new test cases using techniques such as:

- Changing input data through transformations that are not supposed to change the result, called “Metamorphic relationships.” Sometimes, the reference has to be transformed alongside the input. For instance, the detection of objects in an overhead satellite view should maintain identical performance when rotating both image and reference.

- Adding different types of noise or other forms of input transformation or degradation[169] that are reasonable in the considered use case should make the system robust.

It should be noted that these methods can also be used at the system training level, but that is outside the scope of this chapter. See Chapter 5 for the use of those methods at training time.
These approaches tend to be applied in perception cases where the input is in a continuous context (e.g., vision or audio processing, sensor measurements). The metamorphic approach tends to be more challenging to implement because it requires the presence of transformations that are supposed to be invariant for the model, which is not often the case. Some examples are:

- Mirroring of images
- Scaling/resolution changes of images
- Rotation of images
- Phase inversion on audio
- Removal or addition of frequencies outside of the relevant audio range
- Unit scaling within sane limits (e.g., multiplying distances and speeds by the same values on Doppler radars)

Not all of those are always applicable. For instance, rotation can move an input outside of the defined ODD. A fixed camera, for example, would not have the sky at the bottom of the image.

Noise addition is used when a moral argument can be made for the existence of those noises, or transformations in general, in the real world. Numerous categories exist, some examples are:

- Additive (usually Gaussian) or multiplicative (usually Poisson) noise on images, representing thermal and electromagnetic noise
- Blur due to movement, incorrect focal distance, or optical aberrations
- Loss of pixels or lines due to sensor failures
- Luminosity variations
- Environmental variations (fog, snow, rain, dirt)
- In audio, filtering and reverberation (due to acoustic path issues), volume variations
- In sensors, periodic or spiking noises and general drift

In evaluation, the approach is usually to plot the system's performance as a function of the noise level and, where a level of performance is deemed to be inadequate, have an expert examine examples of noisy inputs to determine whether they are still realistic (e.g., in the ODD) or outside of reasonable bounds.
This transformation approach is also sometimes used in a monitoring setup to detect whether the system is nearing a zone of instability in its decisions, which are expected to indicate inputs for which its results are untrustworthy. The issue with the method is that it partially relies on chance: the number of transformed inputs that can be generated and tested is limited. Formal methods presented later in the document analytically cover the whole noise space to ensure every potential input is tested.

[169] This part focuses on the evaluation of the system, not its training. These perturbations do not cover the concept of data corruption which concerns mostly the training data set.

6.2.1.3 **Corner and edge case detection using test-based methods**

Corner and edge cases are more complex cases that can be hard to test exhaustively because they are located on the limit of the ODD (see Section 1.2.4.2 for more). However, corner case detection is not limited to ML systems; in classical software validation, their handling is necessary. Software test methods directly inspired some test methods for DNNs corner case detection. In classical software testing, one can find two primary sorts of methods (Ahamed, 2010):

- White-box test methods (Williams, 2006), for which one must own the source code. In this case, tests mainly focus on code coverage;
- Black-box test methods are based on the executable and do not require knowledge of the source code. In this case, tests mainly focus on functional properties.

These methods coming from software engineering have been adapted to DNNs as developed hereafter. Section 6.2.1.3.1 deals with the white-box approach for neural networks, while the techniques related to black-box testing are discussed in Section 6.2.1.3.2.

6.2.1.3.1 White Box Testing

In (Pei et al., 2017), the authors state that at a conceptual level, erroneous corner-case behaviours in DNN-based software are analogous to logic bugs in traditional software. Similar to the bug detection and patching cycle in traditional software development, the erroneous behaviours of DNNs, once detected, can be fixed by adding the error-inducing inputs to the training data set and possibly changing the model architecture/parameters. However, traditional software testing techniques for systematically exploring different parts of the program logic by maximizing branch/code coverage are not very useful for DNN-based software as the logic is not encoded using control flow. DNNs are built differently than traditional software by automatically learning their logic from the data rather than expressing the human intent through a control flow graph. They also operate differently since they do not rely on the control flow to change behaviour but instead on weight and bias through activation functions. In that regard testing DNNs is fundamentally different from testing traditional programs. Unlike traditional models, finding inputs that will result in high model coverage in a DNN is significantly more challenging due to the non-linearity of the functions modelled by DNNs. Moreover, the Satisfiability Modulo Theory (SMT) solvers (Biere et al., 2009) that have been quite successful at generating high-coverage test inputs for traditional software are known to have trouble with formulas involving floating-point arithmetic and highly nonlinear constraints, which are commonly used in DNNs.

In (Cheng, 2013), Pei et al. propose DeepXplore[170], a white box differential testing algorithm for systematically finding inputs that can trigger inconsistencies between multiple DNNs. They introduced neuron coverage as a systematic metric for measuring how much of the internal logic of a DNN has been tested.

In (Tian et al., 2017), the authors address the issues mentioned in the former paragraph and design a systematic testing methodology for automatically detecting erroneous behaviours of DNN-based software of self-driving cars. They present a systematic technique to automatically synthesise test cases that maximise neuron coverage in DNN-based systems like autonomous cars. They also demonstrate that different realistic image transformations like changes in contrast, presence of fog,

[170] https://github.com/peikexin9/deepxplore

etc. can be used to generate synthetic tests that increase neuron coverage. This approach is implemented into the DeepTest[171] tool.
In (Yu et al., 2022) the authors propose a differential testing strategy to verify the inconsistent behaviour of DNNs automatically. The strategy can track inactive neurons and trigger them to maximise neuron coverage. Then, they use an optimisation technique to derive the predictions from the initial input and construct new test data.
In (S. Lee et al., 2020), the authors present a white-box testing method using an adaptive neuron-selection strategy without using the gradient of the selected internal neurons for the selection. This is done to improve the coverage and to find adversarial inputs. This approach is implemented into the Adapt[172] tool.

6.2.1.3.2 Black Box Testing

In (Tian et al., 2017) an overview of the black box testing approaches is provided, which include the three following approaches detailed in this section.
Usual approaches in evaluating machine learning systems primarily measure their accuracy on randomly drawn test inputs from manually labelled datasets and ad hoc simulations (Google, 2016; Madrigal, 2017; Witten et al., 2016). However, without knowledge of the model's internals, such black box testing paradigms cannot find different corner cases that may induce unexpected behaviours (Goodfellow and Papernot, 2017; Pei et al., 2017). In (Aghababaeyan et al., 2023), the authors showed a comparative analysis with the coverage-based approach using diversity metrics, to use them then to construct corner case detection cases.
Another recent work has explored verifying DNNs against different functional safety properties (Huang X., 2016; Katz et al., 2017; Pulina and Tacchella, 2010). However, none of these techniques can verify a rich set of properties for real-world-sized DNNs.
Metamorphic testing (Chen et al., 2020; Zhou et al., 2004) creates test oracles in settings where manual specifications are unavailable. Metamorphic testing has been used for testing both supervised and unsupervised machine learning classifiers (Murphy et al., 2008; Xie X, 2009).
Finally, as described in Section 1.2.4.2.3, surprise adequacy methods can also be considered black box testing methods.
In this section, the academic papers reference only prototypes in their work. These prototypes are not always available to be tested, but some are.

6.2.1.4 **Applicability analysis**

In this section, we study the applicability of the methods presented in the previous sections to the properties detailed in Section 6.1.5. Corner case and edge case detection are not considered here since they relate more to performance properties (see Section 2.1 for more) than stability and robustness properties. The choice of properties reflects both the current literature on the topic from the standardisation effort (e.g. (ISO/IEC, 2022a)) and the EASA CP (EASA, 2024). We grade by "+" or "++" the ability of the methods to deliver results to assess each property and by N/A if the property cannot be evaluated by this method to the best of our knowledge.

[171] https://github.com/ARiSE-Lab/deepTest
[172] https://github.com/kupl/adapt

Method	Stability of the learning algorithm	Stability of the trained model	Robustness of the inference model	Bias	Variance	Relevance	Reachability
RMSE	++	+	+	++	++	N/A	N/A
MAE	+	+	+	+	+	N/A	N/A
Max error	+	+	+	+	+	N/A	N/A
Actual/predicted correlation	+	+	++	+	+	N/A	N/A
Precision-recall	+	++	++	++	++	N/A	N/A
ROC	++	++	++	+	+	N/A	N/A
Lift	++	+	+	+	+	N/A	N/A
AUC	++	++	++	+	+	N/A	N/A
Balanced_accuracy	+	+	+	+	+	N/A	N/A
Micro-average	+	+	+	+	+	N/A	N/A
MCC	+	++	++	+	+	N/A	N/A
Confusion matrix	++	++	++	++	++	N/A	N/A
Hinge loss	++	++	++	+	+	N/A	N/A
Cohen's kappa	++	++	++	+	+	N/A	N/A
Corner case testing							
• White box	N/A	++	++	N/A	N/A	N/A	N/A
• Black box	N/A	+	+	N/A	N/A	N/A	N/A

Table 57 – Applicability of the different metrics and methods on the robustness and stability property

6.2.1.5 **Tools maturity and scalability**

The statistical methods presented in 6.2.1 use various tools to implement metrics. They can also be executed directly in any programming language.
Here are some examples of standard tools available to ease the work of a data scientist in charge of a statistical evaluation of the robustness and stability property:

- Using R, with the following packages:
  - Package « metrics »[173]: AUC, ROC, rmse, mae.
  - Package « bmrm »[174] hinge loss
  - Package « vcd »[175] cohen's kappa
  - Package « caret »[176] confusion matrix

[173] https://cran.r-project.org/web/packages/Metrics/Metrics.pdf
[174] https://cran.r-project.org/web/packages/bmrm/bmrm.pdf
[175] https://cran.r-project.org/web/packages/vcd/vcd.pdf
[176] http://cran.nexr.com/web/packages/caret/caret.pdf

- Using Python, with the library scikit-learn,[177] which already contains almost all metrics
- Using Matics[178] (LNE)
- Proprietary solutions also exist, such for example as the ones provided by:
    - Tibco (Data Science[179])
    - Microsoft (Azure [180])
    - Salesforce (Tableau[181])
    - SAS (Model Studio[182])
    - …

Concerning the evaluation of corner cases, most of the most advanced tool available rely on a white-box testing approach, such as:

- DeepXplore[183]
- DeepTest[184]
- Adapt[185]

6.2.2 Formal methods

6.2.2.1 **Methods**

This section uses the description provided by ISO/IEC 24029-1 (ISO/IEC, n.d.) and 24029-2 (ISO/IEC, 2022b) to introduce the different methods available to apply formal methods to neural networks. Indeed, most of the techniques described in these documents were designed primarily for neural networks. Here, we consider machine learning models in a broader sense. While for many of the methods described, there is no major problem with adapting them to other machine learning techniques, this work has mostly not yet been done.

6.2.2.1.1 Solver

Neural network models are often large, nonlinear, and nonconvex, and they are beyond the reach of general-purpose tools such as linear programming solvers or existing satisfiability modulo theories (SMT). However, some advances have been made in using solver technologies to prove properties over machine learning models. For example, (Katz G., 2017) presents an approach to efficiently prove properties over some classes of neural networks (using ReLU activation functions which is defined as $ReLu(x) = max(0, x)$) by using a variation of a Simplex algorithm.
Mixed-integer linear programming (MILP) solvers (V. Tjeng, n.d.) and satisfiability modulo theories (SMT) solvers(Katz G., 2017) are deterministic, white-box and typically complete verification methods. They encode all computations of a given machine learning model as a collection of constraints and

[177] https://scikit-learn.org/stable/search.html?q=metrics
[178] https://www.lne.fr/fr/logiciels/lne-matics
[179] https://www.tibco.com/fr/products/data-science
[180] https://azure.microsoft.com/fr-fr/
[181] https://www.tableau.com
[182] https://support.sas.com/en/software/model-studio-support.html
[183] https://github.com/peikexin9/deepxplore
[184] https://github.com/ARiSE-Lab/deepTest
[185] https://github.com/kupl/adapt

then use them to prove robustness properties. Depending on the machine learning model's architecture, these methods can be complete or incomplete. In the case of neural networks, specific non-linear activations (such as hyperbolic functions, including sigmoid and tanh) are too complex to encode precisely. For example, the same issues would be identified for SVM with non-linear kernels (polynomial, radial, etc). Therefore, solvers approximate them with sound abstractions. Other non-linear activations (such as ReLU linear kernel) can be precisely encoded.
To prove a given robustness property, the machine learning model and constraints on the input are encoded as a MILP problem, which can then be used to optimise the robustness constraint. The property is proven if the bounds on the robustness constraint satisfy the requirements. SMT solvers pose the verification problem as a constraint satisfiability question that either holds or does not.
Some techniques include symbolic linear relaxation that computes tighter bounds on the machine learning model outputs by keeping track of relaxed dependencies across inputs and directed constraint refinement (refining the output relaxation by splitting the set of initial or intermediate neurons) to verify safety properties (S. Wang, 2018) (of neural networks). Other techniques propose a satisfiability modulo convex (SMC)-based algorithm combined with an SMC-based pre-processing to compute finite abstractions of machine learning model (here also neural network) controlling autonomous systems (X. Sun, 2019).

6.2.2.1.2 Abstract interpretation

Abstract interpretation is a formal method that relies on a theory that constructs controlled approximations. It is often used to prove complex program properties (R. and Cousot, 1977). Abstract interpretation has taken a more significant role in the software verification and validation community, especially in the context of safety-critical software, such as embedded software for aircraft (Souyris J., 2007), cars (Yamaguchi T., 2019), spacecraft (O. Bouissou, 2009). It aims to tackle the issue of Software reliability in a broader sense (Cousot, 2000). It is a general framework for analysing large and complex deterministic (R. and Cousot, 1977) and stochastic (Cousot, 2012) systems in a scalable fashion. In the context of machine-learning models, it provides an incomplete, deterministic and white-box method that can verify the robustness of large machine-learning models. The verification process proceeds as follows:

- First, the provided test input and a robustness specification collectively define a region that contains all possible perturbed inputs that can be obtained by modifying the feedback based on the robustness specification. This region can be represented precisely or approximately using specific geometric shapes, such as boxes, zonotopes and polyhedra, or as custom abstract domains for neural networks (Singh et al., 2018) or SVM (Ranzato, 2019).
- This region is then propagated through the machine learning model, with every layer sequentially applied to the input region. The input region is transformed into an output region containing all outputs reachable from it. Depending on the layer, this can introduce approximations (unreachable outputs from the input region).
- Finally, an output region captures all possible network outputs for input perturbations formed according to the robustness specifications.

There is an inherent trade-off between precision and scalability. For example, simple abstract domains such as boxes can verify machine learning models within seconds but are often too imprecise to verify the desired robustness properties. On the other hand, semidefinite relaxations are more precise but do not scale to large machine-learning models. Balancing this trade-off is, therefore, key to achieving adequate verification.

The principles of abstract interpretation are as follows. First, the approach usually considers all the possible executions of a program through a semantic. This semantic is, for example, denotational (G., n.d.) or axiomatic (R., 1969). The set of all traces of execution expressed by a given semantic forms a lattice (a complete partial order defined over the set of all traces) or at least a partial order set. This lattice is called the concrete domain and is known to be intractable. Then, a second domain is defined, referred to as the abstract domain, since it is an abstraction of the concrete domain. The abstract domain is also a lattice or a partial order set. An abstraction is proven to be correct when a Galois connection is defined (and proven) between the two domains. The Galois connection is determined by two specific functions: the abstraction alpha (going from the concrete to the abstract domain) and the concretisation γ (going from the abstract to the concrete domain). These functions have specific properties to be proven beforehand, mainly the monotonicity of α and γ, $\gamma \circ \alpha$ is extensible and $\alpha \circ \gamma$ is tightening.
Once the Galois connection is proven between the two domains, the abstract domain becomes a sound over-approximation of all the concrete executions, which is also tractable (by construction). It is then possible to prove properties on the abstract domain and transfer them automatically to the corresponding concrete traces represented by the abstraction. The main difficulty with an abstract interpretation is defining a simple yet expressive enough abstract domain. The abstract domain is intended to be both tractable and representative of the concrete traces of the system. There is a vast literature on abstract domain definitions representing numerical computations. For example, it is possible to use intervals (Cousot P., 1976), pentagons (Logozzo F., 2008), octagons (Miné, 2006), templates (Mukherjee, 2017), polyhedrons (Cousot and Halbwachs, n.d.), zonotopes (Goubault et al., 2012), etc. Each one results from a different trade-off between the accuracy of the abstraction and the cost of its computation.
Recent work (Gehr, n.d.; Mirman M., 2018; Pulina and Tacchella, 2010; Singh et al., 2018) constructs new abstract domains specially tailored for the behaviour of neural networks. Indeed, the non-linear nature of machine learning tends to render some of the existing abstract domains ineffective, especially the ones that use the affine dynamics of a system to define the abstract domain. This is the case, for example, with a new zonotopic domain described in (Singh et al., 2018) that captures the specific dynamics of ReLU activation functions commonly used in image processing machine learning.

6.2.2.1.3 Reachability analysis in deterministic environments

Reachability-based machine learning verification techniques combine the outputs of the solvers described in Section 6.2.2.1.1 with techniques in reachability analysis to provide guarantees on the closed-loop performance of machine learning operating in a given environment. The first step in this analysis is to divide the input space into many smaller regions called cells. For each cell, the solvers from Section 6.2.2.1.1 can be used to determine the possible control outputs of the network in the area it defines. Using this information and the environment model, deciding on an over-approximation of the range of possible following states from any given cell is possible. Repeating this for all cells in the initial state region over multiple time steps can result in an over-approximation of the set of reachable states (K. Julian, 2019). Another approach to this problem is to encode an over-approximation of the environment dynamics as constraints in a mixed-integer program and use the mixed-integer verification technique from Section 6.2.2.1.1 to solve for an over-approximation of the output reachable set (C. Sidrane, 2019).

6.2.2.1.4 Reachability analysis in non-deterministic environments

When the environment is stochastic, the solvers in Section 6.2.2.1.1 can be combined with techniques in probabilistic model checking to determine the probability of reaching a set of states. Similar to the method described in Section 6.2.2.1.3, the input space is divided into a set of cells, and each cell is passed through a solver to determine the possible machine-learning outputs. Probabilistic model checking determines the probability of reaching a particular set of states from a given initial state using dynamic programming (M. Bouton, 2020). By adapting this framework to work with cells rather than single input states, we can obtain an overapproximated probability of reaching a set of states using a machine learning system (S. Katz, 2021).

6.2.2.1.5 Model checking

Model checking is a method to prove that a formal expression of a theory is valid under a particular interpretation. More detailed descriptions can be found in (ISO/IEC/IEEE, 2017) and (Ehrig H., 1985). A theory is expressed by a vocabulary of symbols comprising constants, functions, and predicates that build sentences that state assertions about the intended semantics of an idea. Sentences of predicate logic or data patterns can express a theory. Machine learning models are algorithms designed to discover and use data pattern models. The data pattern model is checked against the input.

For model checking to be valid, all models need to be checked. Model checking can be used in machine learning to prove relationships among sets that obey some relationship.

Example 1: The 'theory of family'(Manna Z., 1993) obeys the interpretation that implements the membership of persons belonging to a family. Thus, two arbitrary persons must be proven to be members of the family or not. Then, the sentence 'one person is the parent of the other person' has to be checked for all available pairs of persons.

Example 2: Model checking has been used (Sena L. H., 2019) to prove the existence of adversarial inputs for machine learning. The theory is the language constituted by the letters, weights, and biases described in machine learning. The interpretation is formed by the label attached to the image of the letter. It is possible to compute the distance between every possible pair of letters in the alphabet. Then, the model can be checked to ensure the predicate that every distance exceeds a specific threshold the user has fixed. User-predefined predicates are checked through machine learning against a theory.

6.2.2.2 **Applicability**

In this section, we study what methods can prove the properties presented in Section 6.1.5 using the technique introduced in (ISO/IEC, n.d.). For each, we review the literature associated with each method and the possible limitations of each technique.

6.2.2.2.1 Using uncertainty analysis to prove interpolation model stability

Uncertainty analysis is a general method used to measure the variability of any kind of mathematical function. For that, the methods tend to search for inputs that can cause significant drift in the behaviour. For the machine learning model, it is indeed a way to find the previously described issues in Section 5.4.2 of standard 24029-2 (ISO/IEC, 2022b). In particular, it is possible to compare the measured behaviour with the knowledge of the actual behaviour. The goal is to measure the capacity of the machine learning model to reproduce within an acceptable range of deviation the phenomenon it is intended to model. It especially helps measure the network's variation of response to check that no unstable behaviour has been introduced.

Several works have defined methods to quantify the capability of a machine learning model (especially for neural networks) to have a level of model stability property. In (Choi J. Y., 1992), a method is described to calculate the propagation of uncertainty in a network, thereby allowing it to detect the part of the ODD where the NN's response is abnormal and non-robust behaviour is to be expected. In (Montaño and Palmer, 2003), a method indicates the impact or importance of the input variables on the output, independently of the variables' nature (quantitative or discrete). In (Hess D. E., 2007), several sources of uncertainty are described, including uncertainty in the input, the sensitivity of the model itself, and the influence of random choices for training and testing data sets.
Uncertainty analysis also detects conditions under which a machine learning model can operate non-robustly. Another advantage is that it allows us to trace the cause of such behaviour by checking the source of the uncertainty change in the computation.

6.2.2.2.2 Using solvers to prove a maximum stable space property

Several examples of using solvers to compute a maximum stable space property have been published recently. For instance, in (Huang X., 2016), an SMT solver is used to prove the absence or the existence of an adversarial example, including the ability to construct it. In (R., 2017), a combination of satisfiability solving and linear programming on a linear approximation of the overall network behaviour is considered. These works illustrate the capacity to express the property needed as a logical formula and the capacity of a generic solver to prove this kind of property on a machine-learning model.

6.2.2.2.3 Using optimisation techniques to prove a maximum stable space property

A machine learning model can also verify standard optimisation techniques. In this process, any satisfiability problem is converted to an optimisation problem, which can then be solved using conventional optimisation techniques, such as the branch-and-bound algorithm (A. H. Land, 1960).
A maximum stable space property can be expressed as a Boolean formula on linear inequalities. For example, the property can express that the output of a neural network must not become more significant than a specific value. This requirement is then translated into a new (Boolean) layer in the network. Then, the model is analysed to find a solution to the optimisation problem. Consequently, the solution to the satisfiability problem is also solved by checking the sign of the solution.
In (Bunel, 2017), the construction of such an optimisation problem from a satisfiability problem on a neural network is described, combining classical gradient descent to find a local minimum, and a branch of bound optimiser to determine the global optimum.
In optimisation techniques, one can also use constraint programming to express the property to be checked. An example of such a technique is proposed in (Bastani O., 2016). The model is approximated by first modelling it as a linear program (using network composed of piece-wise linear functions), then approximating the feasible states using only convex sets, and finally applying iteratively a constraint solver to prove the robustness property.

6.2.2.2.4 Using abstract interpretation to prove a maximum stable space property

Abstract interpretation can easily be used to verify a maximum stable space property. First, the part of the input space must be expressed and checked using an abstract domain. Then, an abstract interpretation verifier can infer the over-approximation of the model using the abstract object over the considered input space. The abstract computation will contain all possible outputs of the system over the input space. For a classifier model, the abstract computation will produce the maximum and minimum values possible for each class. Then, two cases are possible: either one of the class outputs is more significant than any other (no overlap is detected), or no class always takes over the decision. Several works explore how to compute such properties for neural networks (A. Boopathy, 2018), (Singh et al., 2018) and (Gehr, n.d.), or for SVM in (Ranzato, 2019).

6.2.2.2.5 Using abstract interpretation to prove a relevance property

Although relevance has been quite extensively studied through approaches which are only valid on isolated inputs (e.g., deconvnet (M. D. Zeiler, n.d.), Grad-Cam (R. R. Selvaraju, 2016), LIME (M. T. Ribeiro, 2016) or SHAP (S. M. Lundberg, 2017)), it is a relatively new topic for formal methods. At their core, these approaches identify through a gradient analysis how the inputs have influenced the outputs of a machine learning algorithm. However, these approaches are not meant to be generalised to prove (local) relevance properties. For abstract interpretation to be applied, one needs first to generalise this approach to consider not only isolated data points but part of the input space. The notion of abstract gradient has been defined in that sense by Numalis (on an unpublished work yet) with the European Space Agency[186]. An abstract gradient allows the analysis of the impact of every input on every output of a machine learning system over a part of the ODD. It relies on classical abstract domains detailed in 6.2.2.1.2, but it applies the abstraction to the gradient space instead on just the input space.

6.2.2.2.6 Using optimisation to prove a reachability property

A forward reachability analysis method for the safety verification of nonlinear systems controlled by neural networks can be designed by leveraging the Linear-Parameter-Varying control (LPV) representation for the nonlinear systems. LPV frameworks have already received significant attention in tackling the control synthesis/analysis problem of nonlinear and time-varying systems via convex methods (C. Hoffmann, 2015). However, to the best of our knowledge, this idea has not yet been employed for the reachability analysis of nonlinear systems, such as the one used in ML. Finding an exhaustive method to address LPV embedding of the nonlinear system on a systematic basis is still an ongoing research topic; most of the available techniques are ad-hoc approaches with an inherent lack of generality and have shortcomings in addressing predominant issues regarding the constitution of the embedding (A. Kwiatkowsku, 2008; Sadeghzadeh, 2020).

6.2.2.2.7 Using solvers and abstract interpretation to prove a reachability property

It is possible to prove a reachability property by combining solvers and abstract interpretation. A detailed approach is described in Section 6.2.2.1.3 for deterministic environments and 6.2.2.1.4 for non-deterministic environments.

6.2.2.2.8 Overview

We grade the methods' ability to deliver results to assess each property by "+" or "++" and by N/A if the property cannot be assessed by this method.

[186] This work has been carried out from 2021 to 2022 through the ESA tender AO10403.

Method	Stability of the learning algorithm	Stability of the trained model	Robustness of the inference model	Bias	Variance	Relevance	Reachability
Solver	N/A	++	++	+	+	N/A	+
Abstract interpretation	N/A	++	++	+	+	++	++
Optimisation techniques	N/A	++	++	+	+	+	+

Table 58 – Applicability of the different techniques on the robustness and stability property

While formal techniques allow for the treatment of most properties, their scalability can vary widely. Currently, most of the most scalable solutions use abstract interpretation. One limitation of most of the tools available is that they do not easily process natural language models since the definition of local robustness has not been properly defined yet.

6.2.2.3 **Tools overview**

6.2.2.3.1 Saimple

Saimple is a proprietary tool based on abstract interpretation that verifies neural networks and supports vector machines. It uses several proprietary abstract domains developed by Numalis[187]. Saimple can verify both robustness and relevance properties and check reachability by building the necessary loop infrastructure around such use cases.
The product can currently handle convolutional, residual, and recurrent neural network architectures. In addition, detectors are currently in beta. It handles most of the ONNX specification of neural networks (with the notable exception of custom layers). Saimple can also process support vector machines with linear, quadratic, cubic, and RBF kernels.
The tool has SaaS capability and can be used either through a web interface or through its REST API.

6.2.2.3.2 PyRAT

PyRAT is a proprietary tool based on abstract interpretation that verifies neural networks[188]. It was developed at CEA after considering the lack of adaptability to neural networks of classical formal software verification tools developed at CEA. Through abstract interpretation techniques, PyRAT proposes multiple abstract domains to verify neural networks, such as boxes or zonotopes. PyRAT propagates the abstract domains through them to obtain the whole set of reachable values from an initial configuration to verify networks. In addition to this, an input partitioning approach allows PyRAT to improve its precision and prove more general properties on the network, such as the ACAS-Xu properties.

---

[187] https://www.saimple.com/
[188] https://list.cea.fr/en/page/development-environments/

6.2.2.3.3 ERAN

ERAN is an academic tool based on abstract interpretation that analyses neural networks[189]. ERAN stands for *ETH Robustness Analyzer for Neural Networks*. It refers to the work introduced by (Singh et al., 2018) (Gehr, n.d.). It operates on MNIST, CIFAR10, and ACAS Xu-based networks. Its design does not allow straightforwardly the handling of other neural networks than the ones already implemented as examples.
The goal of ERAN is to automatically verify safety properties of neural networks with feedforward, convolutional, and residual layers against input perturbations (e.g., L_∞-norm attacks, geometric transformations, vector field deformations, etc).
ERAN combines abstract domains with custom multi-neuron relaxations to support fully-connected, convolutional, and residual networks with ReLU, Sigmoid, Tanh, and MaxPool activations. Specifically, ERAN supports the following analysis:

- DeepZ: contains specialized abstract Zonotope transformers for handling ReLU, Sigmoid and Tanh activation functions.
- DeepPoly: based on an abstract domain that combines floating point Polyhedra with Intervals.
- GPUPoly: leverages an efficient GPU implementation to scale DeepPoly to much larger networks.
- RefineZono: combines DeepZ analysis with MILP and LP solvers for more precision.
- RefinePoly/RefineGPUPoly: combines DeepPoly/GPUPoly analysis with (MI)LP refinement and PRIMA framework to compute group-wise joint neuron abstractions for state-of- the-art precision and scalability.

ERAN supports only layers used with most classical activation layers used in MNIST, CIFAR10, and ACAS Xu based networks.

6.2.2.3.4 Marabou

Marabou is an academic tool based on an SMT solver to compute bounds on neural networks[190]. It is a framework that formerly verifies the mathematical properties of neural networks. It is a SMT (Satisfiability Modulo Theory) solver, which means it verify whether a mathematical formula is satisfiable. Its goal is to answer queries about a network's properties by transforming these queries into constraint satisfaction problems. This tool is built upon Reluplex (Katz G., 2017), a previous solver developed using SMT-based techniques to verify mathematical properties of DNNs (Deep Neural Networks). Marabou can be used with networks using different activation functions and topologies. It also supports parallel execution to enhance scalability further. Marabou accepts multiple input formats, including protocol buffer files generated by the popular TensorFlow framework for neural networks.
Marabou supports fully connected feed-forward and convolutional neural network with piece-wise linear activation functions, in the NN and TensorFlow formats. Properties can be specified using inequalities over input and output variables or via a Python interface.
There are several types of verification queries that Marabou can answer:

[189] https://github.com/eth-sri/eran
[190] https://neuralnetworkverification.github.io/

- Reachability queries: if inputs are in a given range is the output guaranteed to be in some, typically safe, range.
- Robustness queries: test whether there exist adversarial points around a given input point that change the output of the network.

6.2.2.3.5 MIPVerify

MIPVerify is an academic tool based on MILP solver to compute bounds on neural networks[191]. MIPVerify enables users to verify neural networks that are piecewise affine by finding the closest adversarial example to a selected input, or proving that no adversarial example exists for some bounded family of perturbations. It has been introduced in (V. Tjeng, n.d.).

6.2.2.3.6 DNNV

DNNV is a framework for verifying Deep Neural Networks. DNNV takes as input a neural network[192], and a property over the network, and checks whether the property is true, or false. One common DNN property is local robustness, which specifies that inputs near a given input will be classified similarly to that given input. It has been introduced in (D. Shriver, 2021).
DNNV standardizes the network and property input formats to enable multiple verification tools to run on a single network and property. This facilitates both verifier comparison, and artifact re-use.

6.2.2.3.7 ReluVal

ReluVal is one of the first systems proposed for formally verifying properties of feed-forward fully-connected neural networks with ReLU activations[193], and it is still considered among the state- of-the-art tools concerning the ACAS Xu benchmarks. Introduced in (S. Wang, 2018), ReluVal implements a search-based approach that combines interval abstractions and iterative refinement. Later, it was further improved with the integrations of the initial optimisations proposed in Neurify.

6.2.2.3.8 Nnenum

Nnenum is an exact verifier for sequential neural networks supporting fully-connected and convolutional network with ReLU activation functions[194]. It is developed by Stanley Bak (Bak, 2020) and is considered a state-of-the-art tool, especially on the ACAS-Xu benchmark. Nnenum combines an approach based on zonotopes and star-set over-approximations with an improved path enumeration verification method for case splitting on the ReLU function.

6.2.2.3.9 α, β-CROWN

α,β-CROWN is a formal neural network verifier based on the CROWN (Huan, 2018) verifier, GPU relaxation for ReLU and a branch and bound approach for case splitting on ReLU[195]. As mentioned, it

[191] https://github.com/vtjeng/MIPVerify.jl
[192] https://github.com/dlshriver/dnnv
[193] https://github.com/tcwangshiqi-columbia/ReluVal
[194] https://github.com/stanleybak/nnenum
[195] https://github.com/Verified-Intelligence/alpha-beta-CROWN

supports GPU computation to accelerate the analysis. Combined with its wide support of neural network analysis and thus possibility to run on various benchmarks, this led to very good performance in the VNN-COMP 2021.

6.2.2.3.10 Saver

Saver (**S**VM **A**bstract **Ver**ifier) is an abstract interpretation-based tool for proving properties of SVMs[196] (Ranzato, 2019), in particular it aims at proving robustness or vulnerability properties of classifiers.

6.2.2.4 **Tool applicability and maturity**

Most of the tools available are design to process robustness property on trained neural network, a significant part allows also to perform reachability analysis. Very few are able to perform analysis on other machine learning types than neural networks (Saimple and Saver). Only Saimple allows formal analysis for relevance property.
We grade the tool's ability to deliver results to assess each property by “+” or “++” and by N/A if the property cannot be assessed by this method. Assessment is based on the tool's ability to use the underlying method, the documentation provided, and our general knowledge of the tool.

[196] https://github.com/abstract-machine-learning/saver

Tool	Stability of the learning algorithm	Stability of the trained model	Stability of the inference model	Bias	Variance	Relevance	Reachability
Saimple	N/A	++	++	+	+	++	+
PyRAT	N/A	++	+	+	+	N/A	+
ERAN	N/A	++	+	+	+	N/A	+
Marabou	N/A	+	+	+	+	N/A	+
MIPVerify	N/A	+	+	+	+	N/A	
DNNV	N/A	+	+	+	+	N/A	+
ReluVal	N/A	++	+	+	+	N/A	+
Nnenum	N/A	N/A	N/A	N/A	N/A	N/A	++
α, β-CROWN	N/A	++	+	+	+	N/A	+
Saver	N/A	+	N/A	N/A	N/A	N/A	N/A

Table 59 – Tool applicability on the different robustness and stability properties

In terms of scalability for the robustness property, the tools Saimple and PyRAT have the best capabilities by processing large neural networks (over several dozen millions of parameters). It should be noted that no other use case is available to perform reachability property comparison.
In terms of ease of access Saimple is currently the only tool to be deployable as a service and fully usable in any CI/CD process through an API.

6.2.3 Empirical methods

6.2.3.1 **Methods**

This section presents methods described first in the ISO/IEC 24029-1 (ISO/IEC, n.d.).

6.2.3.1.1 Field trials

While there are several aspects that need to be studied for the establishment of trust in AI systems, the number of feasible approaches for analysing a system's behaviour and performance are limited. AI systems typically consist of software to a large extent. Therefore, standards for software testing are needed, such as ISO/IEC/IEEE 29119 for example.
The primary goals of software testing are stated in ISO/IEC/IEEE 29119-3:2013 (ISO/IEC/IEEE, 2013): *Provide information about the quality of the test item and any residual risk in relation to how much the test item has been tested; to find defects in the test item prior to its release for use; and to mitigate the risks to the stakeholders of poor product quality.*

The workflow to assess machine learning robustness described in 6.1.6.2, Figure 124 contains three steps that are crucial for every field trial:

1. Setting up the testing plan (plan testing)
2. Realization of the data acquisition (data sourcing)
3. Carrying out the trial in a real operation environment (conduct testing)

In contrast to other testing methods, in field testing the machine learning is integrated into a system that operates in a realistic environment for the respective application. Therefore, data acquisition and sourcing are integral to experimental design and execution. This is why the data sourcing step is to be taken particularly thoroughly for this approach to work.

Defects and poor product quality are concerns when testing machine learning systems too. However, the failure of an AI system in a functional test is not necessarily related to a "software bug" or an erroneous design. AI systems showing occasional malfunctions are sometimes accepted because they are still regarded as beneficial for their intended purpose, in particular where there are no feasible alternatives. AI systems reveal their degree of performance mainly during field trials or deployment, as is the case with systems like virtual assistants, for example. This applies to many machine learning systems that operate in interaction with or dependency of natural environments and humans. This imply that in order to be setup a machine learning model need to be integrated into a component or even a system that can be deployed. A field trial approach would not be adequate to test a machine learning model in itself.

In the medical sector, the efficacy and the balance between risk and benefit of a new product are subject to many regulations. For instance, in Europe medical devices, including those using AI components, need to comply with DIN/EN/ISO 14155(ISO, 2011). They need to undergo "clinical investigations", a procedure that resembles "clinical trials"(Läkemedelsverket, 2009) (Beede, 2020; Florek H.-J., 2015).

For some non-medical devices using AI components, field trials have for long been a recognized means of comparing and assessing the robustness of solutions. In these sectors it should be noted that regulations are also being made in order to control their safety before their entrance on the market.

Some prominent examples are:

- Facial recognition trials (BAA, 2009; BIS, 2004) (Vetter, 1997).
- Tests of decision support systems for agricultural applications (Burke J., 2008).
- Practice for testing driverless cars (UK-Government, 2015).
- Tests of speech and voice recognition systems (Isobe T, 1996; Lamel L, 1996).
- Networked robot at a train station (Shiomi M, 2011).

Field trials for machine learning systems greatly vary with respect to methodology, number of users or use samples involved, status of the responsible organization/persons, and documentation of the results.

6.2.3.1.2 A posteriori testing

In some cases, it is possible to formally validate the robustness of an intelligent system. When this is not possible, as is very often the case for machine learning (Tian, 2018), validation by testing empirically the robustness of the system is then implemented, and input-output evaluation are very popular in this context. In this kind of evaluation, there are priori testing and posteriori testing methods. While a priori testing knows in advance the expected output and statistical measures are therefore applicable, a posteriori testing does not know it in advance. In that case, it is sometimes

possible to design automated measures to still perform statistical measures by indirect means, but otherwise the only available method is empirical, relying on the judgment of human subjects.
In a posteriori testing, Step 4 and 5 of the process illustrated in Figure 124 in Section 6.1.6.2 are slightly altered. Step 4, Analyse outcome, is likely to be more complicated, because the correct answer is unknown. Interpreting the results in Step 5 is a matter of consensus and not based on an unambiguous truth.
In general, to validate the robustness of a system, data or test environment representing a wide variety of test scenarios, for normal operating conditions and critical cases, are identified (Step 2 of the process). These inputs are submitted to the system to be evaluated and the system outputs (called hypotheses) are compared to references, i.e., to a ground truth (Step 3). These inputs are designed to challenge the system to check its robustness, for example by using adversarial examples. These references are usually provided by human annotators performing the same task as that performed by the evaluated system, or by physical measurements.
In the case of a priori testing, references are provided by human annotators, and they usually all agree among themselves on the "right answer" to be provided (high inter-annotator agreement rate). The ground truth is therefore unambiguous. On the contrary, with a posteriori testing, the annotators' references vary from one to another. The field truth is therefore ambiguous. In general, this is the case when the task has several correct answers (G., 1979).
As it is not possible to determine a priori all the possible correct answers, thus posteriori evaluations are carried out. That is, it is by looking at the outputs of the systems that human annotators or automated measures will tell whether they are "acceptable" or "incorrect".
Machine translation is a classic example of a task for which a posteriori evaluation is a valuable complement to a priori testing. There are usually several ways to translate the same sentence from one language to another. While statistical methods are often applied in this case, by establishing an arbitrary set of correct or acceptable answers to compare the output with (Papineni, 2002), this is not an entirely reliable measure of performance and subjective a posteriori testing is often more precise.
Similarly, it is possible to accept several trajectories to move from one place to another during a navigation task. Depending on the ability to define an objective criterion for successful trajectories, a posteriori testing can be applied with either statistical or empirical means.
It is also possible to use a posteriori evaluation to validate a new robustness metric (a new method or formula to measure). Indeed, when the quality of task is subjective the metrics needs to assign quality scores that correlate with human judgment of quality. Human judgment is the benchmark for assessing automatic metrics (Graham, 2014).
However, the concepts of a posteriori evaluation and post-deployment evaluation are overlapping in some cases, particularly when testing with end-users. For example, in evaluating the quality of a human-machine interaction, the evaluation is done a posteriori because it is not possible to know how the interaction will generalise across the population before widespread interaction occurs. To carry such evaluation, it is possible to vary the user profile and having a user pool representative of the system's actual operating conditions and obtain an empirical analysis of the robustness of this interactive intelligent system.

6.2.3.1.3 Benchmarking of machine learning

Benchmarking a machine learning based system could help in establishing to some degree of the robustness of the system. Often the initial trust in an AI solution based on machine learning models is established by a benchmark test assembled by the system designer. For example, in pattern

recognition and similar applications of AI methods, benchmarking has for long been the gold standard for establishing trust in a certain method (Kohonen, 1988). However, benchmarking could introduce elements of subjectivity, such as in the tagging or annotation of test data sets by expert practitioners. Benchmarking measures the performance of a system on carefully designed data sets that are publicly available in most cases. Often, they are used for testing different systems competitively. Prominent examples of benchmarking are the Face recognition vendor tests conducted by the US Department of Commerce (Ngan, 2015). Other examples are the "Grand Challenges in Biomedical Image Analysis" (Van Ginneken et al., n.d.). Since they are public and are intended to be generic enough for a sector, they cover only a subset of the possible variation type of the inputs concerned by the machine learning model.
In contrast to field trials, benchmarking does not necessarily require an operational system in a real application scenario. For benchmarking, data sets are created to pose a serious but not impossible challenge for state-of-the-art classification or regression methods. The benchmark data sets need to be complemented by benchmarking rules that describe and standardise ways of setting up testing, documenting this setup, measuring and documenting these results (Prechelt, 1994). Benchmarking plays a strong role in pattern recognition research and has contributed decisively to the advancement of the field. However, benchmarking is usually not sufficient for a decision on a certain robustness goal. Benchmarking results are to be interpreted with care (Maier-Hein et al., 2018).

6.2.3.2 **Applicability and maturity**

We grade by "+" or "++" the ability of the methods to deliver results to assess each property, and by N/A if the property cannot be assessed by this method.

Method	Stability of the learning algorithm	Stability of the trained model	Robustness of the inference model	Bias	Variance	Relevance	Reachability
Field trials	N/A	+	+	+	+	N/A	~ (*)
A posteriori testing	N/A	+	+	+	+	++	N/A
Benchmarking	+	+	+	+	+	+	+

Table 60 – Applicability of the different empirical techniques on the robustness and stability property

(*) Reachability properties can be very challenging to be verified in field trials. However, if the ODD is rather small it might be possible to empirically test it.

Field trials and a posteriori testing present the same drawback that the system which include an ML constituent should be completed before being tested. It is therefore only suitable to assess the robustness of the trained model since this one has to be frozen for the evaluation to go forward. They both can be implemented by specific evaluator groups specialized in this kind of evaluation. There is no tool available capable of replacing the manual step of the evaluation protocol setup and realization. Scalability is completely dependent on the difficulty of setting up this protocol. But since it has to be done using human involvement the scalability of such methods is necessarily more costly and difficult.

Benchmarking however presents more scalability potential. A growing scientific literature is addressing this topic (J. Thiyagalingam, 2022; P. Malakar, n.d.; R. S. Olson, 2017; T. St. John, 2019). Benchmarks can either concerns data set that are shared in order to measure the performance of a new machine learning models, or it can also be a shared data set of already trained machine learning models to be benchmarked against. It should be noted that no large-scale benchmark data set is already internationally and formally recognised. Several benchmarks are still competing with each other to become the standard on each topic and many new ones are entering the fray each year. One important driver in the emergence of these benchmarks are competitions organized by academic conferences, which usually have one challenge in mind (image recognition, translation, etc). As these benchmarks are growing in number they also improve in maturity, allowing new evaluations to be done more and more easily. However, it is important to note that benchmarks are not always of the same quality, and outliers, incorrect labelling, or even incorrect data are often present in some. A recent study shown that even ImageNet, which is one of the most used data sets, contains a 6% error in average in its labelling (C. Northcutt, 2021). Some work has taken place at the ISO/IEC to try to standardise how benchmarks can be assembled, annotated, traced and validated to improve their quality and usability.

Data set	Type of input	Size
MNIST	Images	10 000
CIFAR-10	Images	10 000
CIFAR-100	Images	10 000
Caltech-256	Images	29 780
ImageNet	Images	50 000
QuickDraw	Images	50 426 266
20news	Text	7 532
IMDB	Text	25 000
Amazon Reviews	Text	9 996 437
AudioSet	Audio recording	20,371
MSTAR	X-band SAR	2 747
NORB	Images	29 160
COP_banknote	Images	20 800

Table 61 – Some examples of data sets used in machine learning competitions

Outside the benchmark used in some competitions we can also cite two benchmark that are specific to the aeronautic sector:

- The Air Operators database of incidents from the Federal Aviation Administration[197], with more than thousands of entries.
- The Opensky datasets[198] with more than dozens of thousands of commercial and non-commercial flights.

6.2.4 Selected tools and methods

Following the assessment done of each method and the maturity of the available tools, the section 6.2 of this MLEAP report will focus on experimenting some of them on the proposed use cases. From a practical point of view the use cases allow most of the statistical metrics to be tried, however not all formal methods can be evaluated. The maturity of the available tools can also impact their applicability. In practice, abstract interpretation will be the preferred technique to implement formal methods on the use cases. In particular the tools Saimple, ERAN and Nnenum can be available and compatible with the intended use cases (or at least some of them). Finally, empirical method requires a preparatory work alongside with the experts in charge of the evaluation with limited scalability. This type of evaluation is not covered by the MLEAP project in order to focus on scalable techniques.

6.3 Experimental evaluation

In this section, we illustrate the applicability of the requirements LM11, LM12, and LM13 to toy examples built on open-source data sets and avionic use cases. This section aims to provide some perspective on the applicability of the requirements expressed in the EASA CP.

6.3.1 Open-source examples

In this section are presented the open-source examples considered to analyse the applicability of the methods.

6.3.1.1 **Convolutional architecture**

Tentative ConOps definition: The system helps guide a space lander system by classifying images taken onboard that contain one or more craters, avoiding these areas during the landing process.

Tentative ODD definition: The system is operating on 8-bits grey scale image that are acquired by the on-board camera in high resolution. Prior to the machine learning model, the image is sliced in 64x64 images that are passed on to the model. The camera is turned on when approaching the objective, so

[197] https://av-info.faa.gov/dd_sublevel.asp?Folder=%5CAirOperators
[198] https://opensky-network.org/datasets/metadata/

nominal image could contain terrain or possible deep space portion (black background) depending on the lander orientation. The terrain is characterized by the presence or the absence of crater. Crater radius that should be detected can represent up to 25% of the image itself. A crater can also be arbitrary sliced during the slicing process (crater with less than 50% of their circumference represented on the image can be ignored). Multiple craters can be located on the same image, some can even overlap one another. The incidence angle from the lander can be from 90° (perpendicular to the surface) to 20° depending on which phase of the landing it is. Every image can bear two types of defects: gamma radiation and halo. The number of Gamma radiations is possibly unbounded and can appear anywhere on the picture. Each Gamma radiation can cause a defect on the image looking like a segment with a width inferior of 2 pixels, and an unbounded length. The number of halos is considered to be under 5 and would not represent more than 50% of the image surface.

<u>Architecture and data set</u>
In this example, we considered the following architecture, which performs an image classification task trained on a database from the Rosetta mission. Images are grey-scale images that can be affected by several optical defects and deep-space radiation perturbations. They are 32x32 in size and mono-channel, which can be considered a rather small-dimensionality use case. The training data set contains around 1000 images.
In this use case, the ERAN tool (presented in 6.2.2.3.3) has been used with significant modifications to define custom noises.

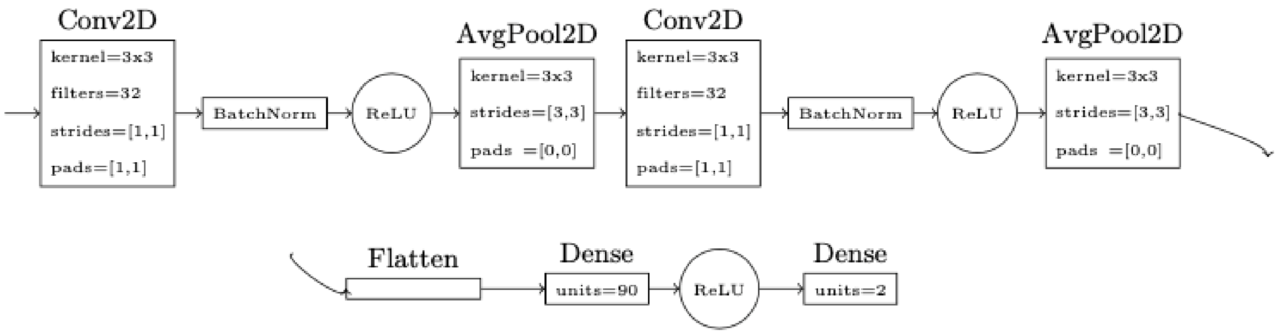


Figure 125 -Convolutional architecture considered

<u>LM11: Training algorithm stability</u>
To test the stability of the training algorithm, the deviation can be measured in terms of the trained model's behaviour or in terms of numerical values of its weights and biases.

As a first study, the model was tested against the gradual removal of the percentage of data points in its training set, and the accuracy of its training set was measured first. So here, the performance of the training is assessed, not the ability to generalise. Each removal is done symmetrically to keep classes balanced at all times. At each test, the previously removed data points are not re-injected in the training set, and the accuracy is assessed on the actual (reduced) training set used for the training (not the complete initial one). The model and the training algorithms used are from Keras version 2.10.0. The results of this experiment are presented in Figure 126, where the x-axis is the percentage of data points removed, and the y-axis is the accuracy (scaled between 0.65 and 1) of the model on

the training data set. Initially, the model had a good accuracy (around 97%). However, the accuracy starts to vary rapidly (up to plus or less 20% change). This behaviour is somewhat counter-intuitive since we would assume having fewer training data points would imply less accuracy, but in many cases, removing more data points helps increase the accuracy almost to the initial level. This is quite observable. Around 10% of the data set was removed, with a huge drop and an equivalent recovery.
As a first analysis, the training algorithm seems unable to guarantee the stability of its own output (i.e., of the trained model). The issue, however, can stem from the data point distribution inside the training data set, as suppressing some inner classes can impact the way the model is learning. It can also be due to some undetected overfitting.

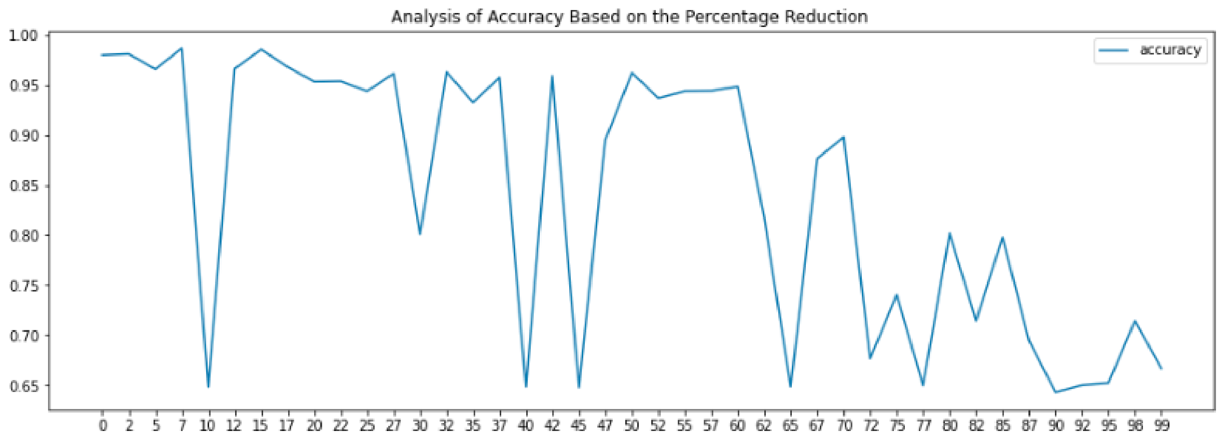


Figure 126 – Evolution of the accuracy of the model depending on how many percent of the training set is removed.

The choice of the metric to evaluate the model (here, the accuracy) may introduce some bias. So, a second verification would be to check if this behaviour is also attested using other metrics. This experiment uses the test data set to analyse the model's generalisation ability. The results of this experiment are presented in Figure 127; for more information about the statistical metrics used, the reader can refer to Section 2.2. The model's behaviour is still quite similar, regardless of the metric used, as the model also has counter-intuitive accuracy drop and recovery. This shows that some regularisation strategies can be helpful either by removing some data points or by adding ones to keep the performance more stable against suppression in the training set.

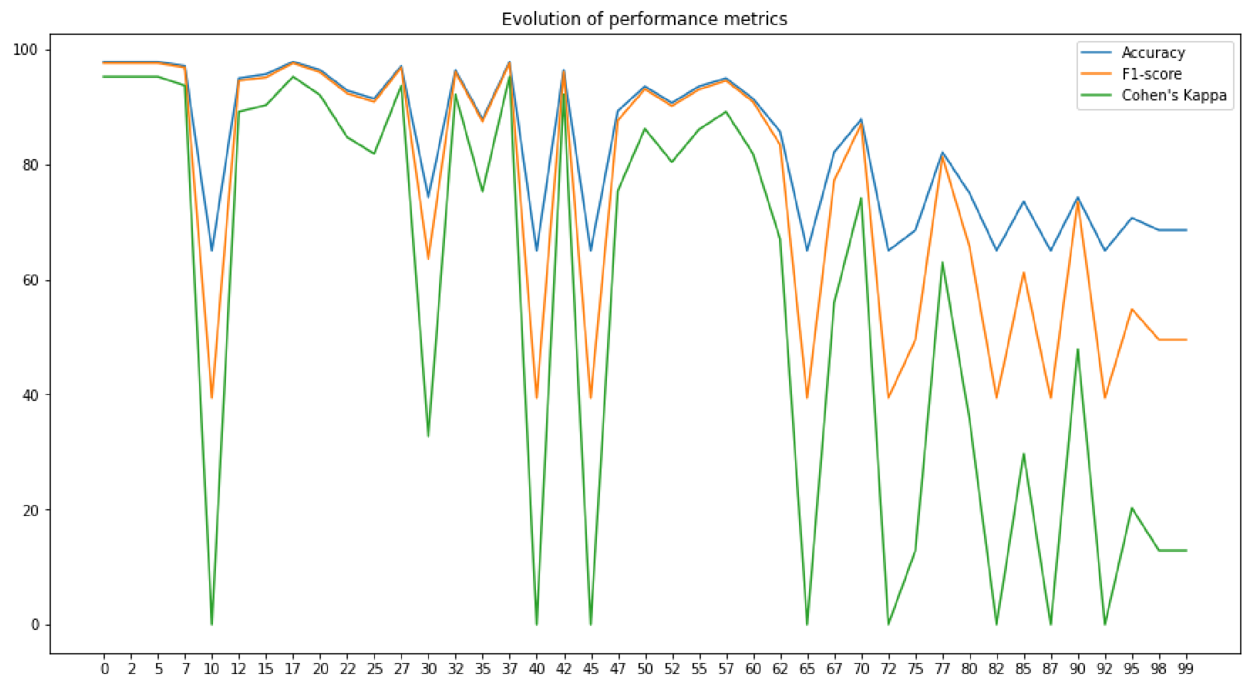
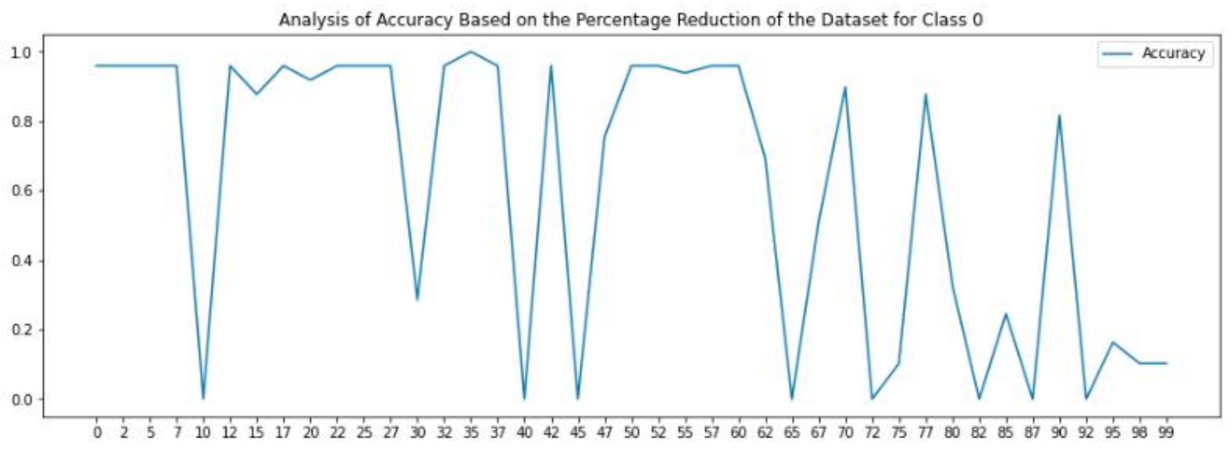


Figure 127 - Evolution using different performance metric of the model depending on how many percent of the training set is removed.

To better understand if the training algorithm or the data solely causes this behaviour, it is necessary to zoom in a little inside the dataset and consider what is removed. The behaviour observed by around 10% is a good candidate for understanding what is happening.

Figure 128 shows the evolution of accuracy by considering each class depending on how many data points are removed. It appears clearly that the accuracy of class 0 (“crater”) is widely more impacted by the suppression of data points, and in particular, the loss of accuracy observed around 10% is mostly caused by the poor performance on class 0 rather than class 1.



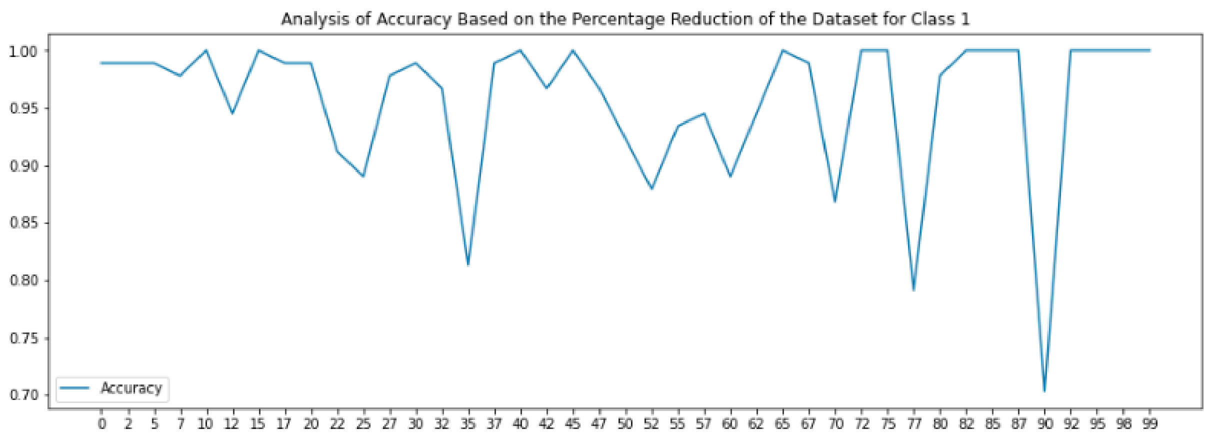


Figure 128 - Evolution of the accuracy on the test data set depending on the class consider (top is for class "crater", and bottom is for class "no crater")

If class 0 is more sensitive to the loss of some data points, it can be helpful to analyse the data points removed more precisely. Figure 129 and Figure 130 show the images of class 0 that have been removed to cause the drop at 10% and then the recovery at 12%. What can be seen in the red rectangle is that some images share similarities with others. Specifically, the images in Figure 129 are mostly craters seen at a 90° angle, whereas in Figure 130, they are seen at a more 30° angle. This shows that the model has constructed two subclasses inside the class crater, depending on the angle at which the crater is seen. Removing too many images from one subclass would cause an imbalanced data set that would cause a drop in accuracy, and then removing more of the other subclass would help restore the balance and thus improve the recovery of the accuracy.

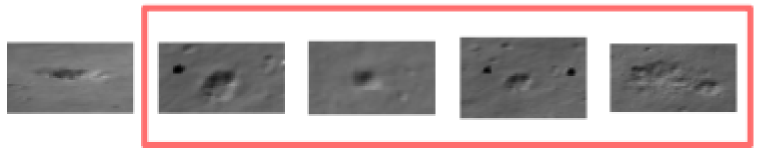


Figure 129 – Images removed of class 0 that cause the drop of the accuracy at 10%

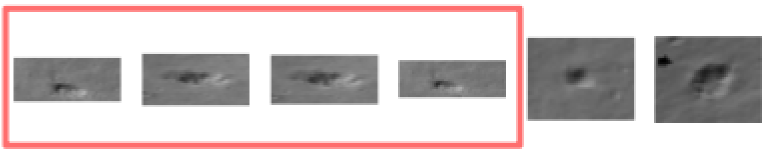


Figure 130 – Images removed of class 0 that cause the recovery of the accuracy at 12%

By analysing the stability of the training algorithm, the study uncovers new information regarding the model's accuracy and general behaviour. However, this is still insufficient to state whether the training algorithm can allow some stability property.

To do so, a second analysis can be attempted to measure the impact of the slightest perturbation of the training set, which removes only one picture of each class at a time. This study sequentially removes one picture of each class of the training set before re-injecting it. This allows us to measure the impact of each photograph on the training. The accuracy is measured here always on the test data set that is not impacted by the process. For each result shown, the x-axis is the identification number of the image, and the y-axis is the performance metric of the trained model.

Figure 131 shows that some images can cause vast accuracy drops, implying that the training algorithm is sensitive to small changes in the training data set. Similarly, as before, a study to check which class is more susceptible to causing instability of the training algorithm is shown in Figure 132. The training process is notably more sensitive to changes in class 0. The question of why this training procedure would be more susceptible remains still open as the time as of this report.

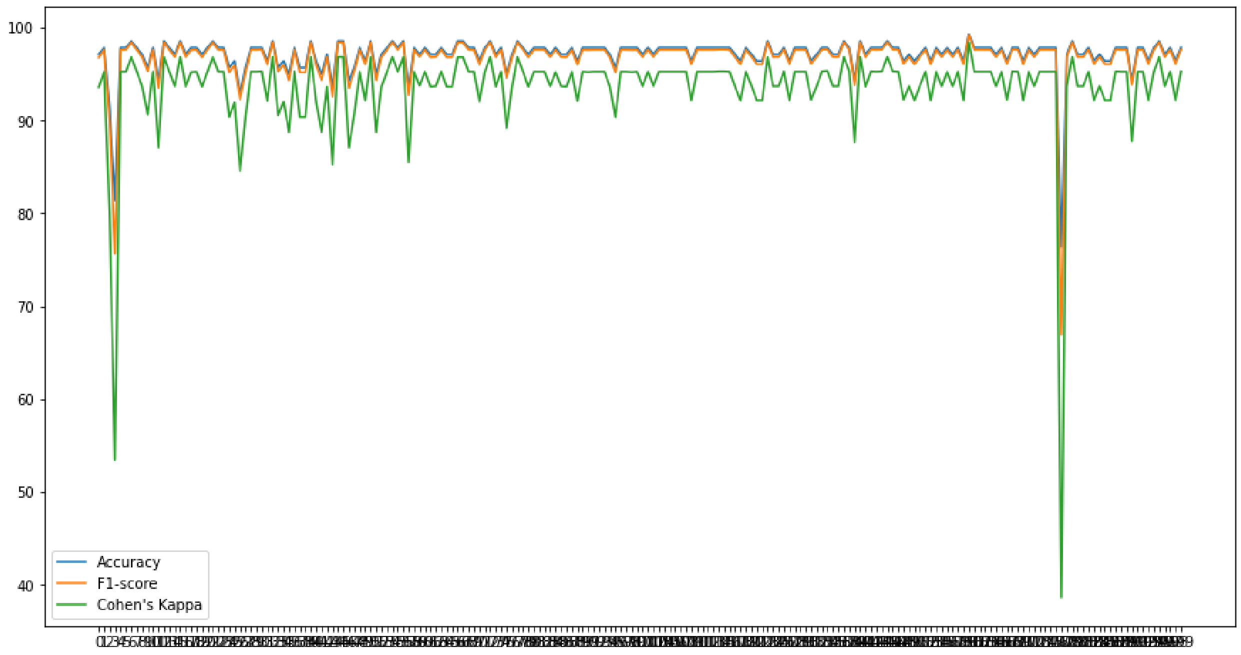
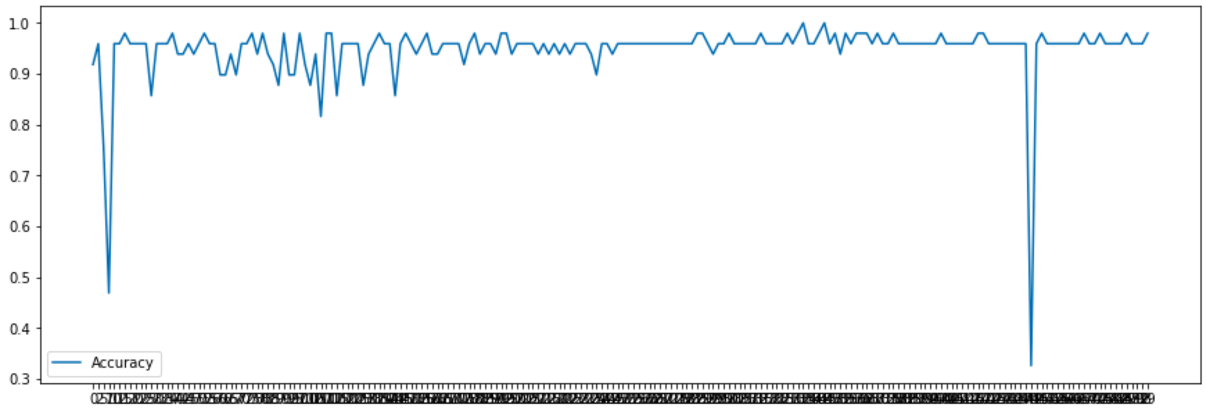


Figure 131 – Evolution of different performance metrics depending on the image removed from the training data set



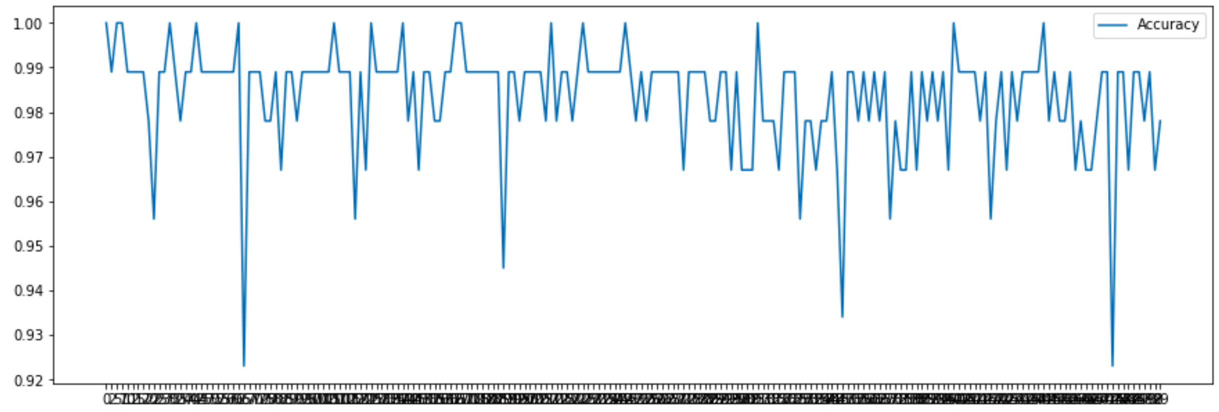


Figure 132 - Evolution of the accuracy on class 0 (top) or class 1 (bottom) depending on the image removed from the training data set

<u>LM12: Trained model stability</u>

Studying the stability of the decision allows the ML engineer to check how much the system can take in terms of perturbation before changing the classification. As a first approach one can decide to measure the stability over every possible perturbation (i.e., considering all dimension at once). This approach is the most general one but can several issues in terms of scalability since it will try to verify the stability against the largest number of cases.

In order to measure the ability of formal methods to tackle large subspace to verify we use two different abstract domains (zonotopes and polytopes). Each has different capability in terms of limiting the over-approximation done, but with a trade-off in terms of computational cost. This will allow to verify how much this can impact the stability formal verification. The study focuses on a set of 1312 validation images with two different (global) perturbation values.

	$\pm$1 pixel variation	**$\pm$2 pixels variation**
Zonotopes	1129/1312	72/1312
Polytopes	1212/1312	157/1312

Table 62 – Number of successful stability verification depending on the perturbation level and the abstract domain used.

As the table shows the stability property cannot be verified for most of the 1312 images considered when the perturbation intensity increases. This can be due to the fact that either the system can have adversarial in this area, or is poorly trained or that the abstract interpretation techniques tend to over-approximate too much. By using two different abstract domains (one more precise than the other) we can compare the impact of that over approximation.

As we have seen, global stability can be quickly difficult to validate, but local ones can still be achieved. In fact, validating stability should take into account the perturbations that can occur within the ODD. In this study, we considered two perturbations illustrated below.

- Gamma radiations are high energy beams that can cause on impact straight lines of (more or less) saturated pixels on a camera. This can be modelled by a maximal perturbation on a specific area

- Halo are optical defects caused by light reverberation on the lenses that can cause grey circles on the picture with a whiter perimeter. Two maximal perturbations, one in the shape of a disc and the other in the shape of a circle, with two different maximal intensities of noise, can model this.

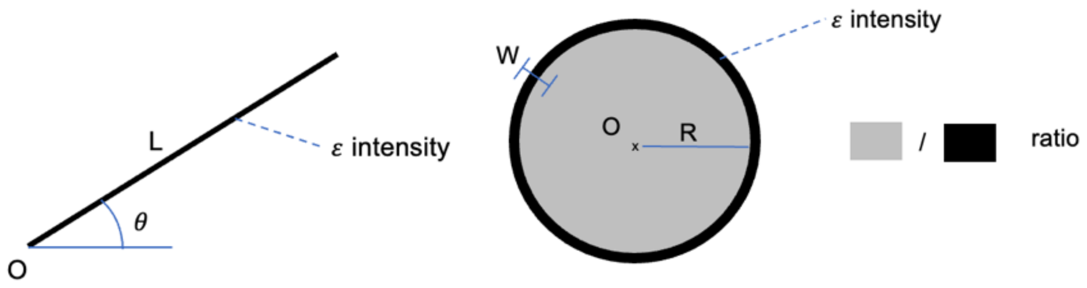


Figure 133 -illustration of perturbation on the ODD. On the left the radiation perturbation, on the right the halo perturbation

Using the radiation perturbation, it is possible to prove the stability of decision of a significant part of the evaluation data set even at noise intensity ranging from 0 to 255. Meaning that for some images it is possible to check the stability for pixels that can vary from 0 (black pixel, so no perturbation at all) to 255 (white pixel, so maximal perturbation). The difference with the previous study is that here we only consider the perturbed pixels that are on the straight line of the gamma radiation, not all the pixels at once. It is far less but under this assumption is possible to fully demonstrate the stability of the decision.
However, we can see a substantial imbalance between the two classes (crater or no-crater) which indicates that the system has not adequately trained on one of them.

Noise intensity :	15	35	55	75	95	115	135	155	175	195	215	235	255
Class 1	734 99.73%	732 99.46%	728 98.91%	721 97.96%	712 96.74%	703 95,52%	692 94.02%	665 90.35%	645 87.64%	619 84.10%	595 80.84%	563 76.49%	497 67.53%
Class 2	491 99.80%	481 97.76%	470 95.53%	438 89.84%	395 80.28%	338 68.70%	298 60.57%	266 54.07%	238 48.37%	214 43.50%	179 36.38%	149 30.28%	121 24.59%
All Images	1225 99.76%	1213 98.78%	1198 97.56%	1159 94.38%	1107 90.15%	1041 84.77%	990 80.62%	931 75.81%	883 71.91%	833 67.83%	774 63.03%	712 57.98%	618 50.33%

Table 63 – Stability results on the convolutional architecture using radiation perturbation. First value is the number of stable evaluation obtains; second value is the percentage over the total number of evaluation.

Using the halo perturbation, it is possible to prove the stability of the decision of a significant part of the evaluation data set even at noise intensity ranging from 0 to 10. We also see the same imbalance in training as before on this type of perturbation.

Noise intensity:	1	2	3	4	5	6	7	8	9	10
Class 1	717 97,42%	708 96,20%	697 94,70%	681 92,53%	640 86,96%	604 82,07%	577 78,40%	517 70,24%	413 56,11%	334 45,38%
Class 2	310 63,01%	272 55,28%	192 39,02%	139 28,25%	95 19,31%	88 17,89%	87 17,68%	81 16,46%	79 16,06%	76 15,45%
All Images	1027 83,63%	980 79,80%	889 72,39%	820 66,78%	735 59,85%	692 56,35%	664 54,07%	598 48,70%	492 40,07%	410 33,39%

Table 64 - Stability results on the convolutional architecture using halo perturbation.

LM13: Trained model robustness
This study has not been done. However, the stability results presented in the context of LM12 partially answer the extreme cases of radiation perturbation, for example, since stability was possible to prove on them.

Conclusion on the convolutional architecture
This study has allowed to demonstrate the ability of formal methods to demonstrate a high degree of stability against custom-made noise typical of an ODD. Also, scalability considerations were taken to measure the ability of abstract interpretation methods to be applied to large sets of data points.

6.3.1.2 **Detector architecture**

Tentative ConOps definition: the system is designed to detect cars and pedestrians on an image to help a navigation system.

Tentative ODD: The ODD for this use case is very large and very complex, it would cover all the possible parameters describing what can happen on a road. For the purpose of the short study conducted here it would be unnecessary to describe it in much detail, however the reader can refer to (Ito, 2021) for more.

Architecture and data set
The data set used for the training is from the coco data set[199]. The architecture used here is tiny version of Yolo-v3.

The study here is intended to verify the ability of formal methods to process detector architecture and exploit stability and relevance information. The Saimple tool has been evaluated (presented in 6.2.2.3.1).

LM11 : Training algorithm stability
This study has not been done.

LM12: Trained model stability
To measure the stability only local perturbation has been used since the behaviour of the system is to analyse only locally the patterns to make classification of objects. From the system's output on an image, several boxes with high probability have been selected, and a local noise located on and around these boxes has been set.

To determine precisely where to locate the noise around the box, the user can first rely on an analysis using perturbation everywhere on the picture, then use the relevance of one particular box to

[199] https://cocodataset.org/#overview

determine where the most relevant parts of the image used by this box are. This would indicate where to locate noise on part of the image that can impact the stability of the decision.

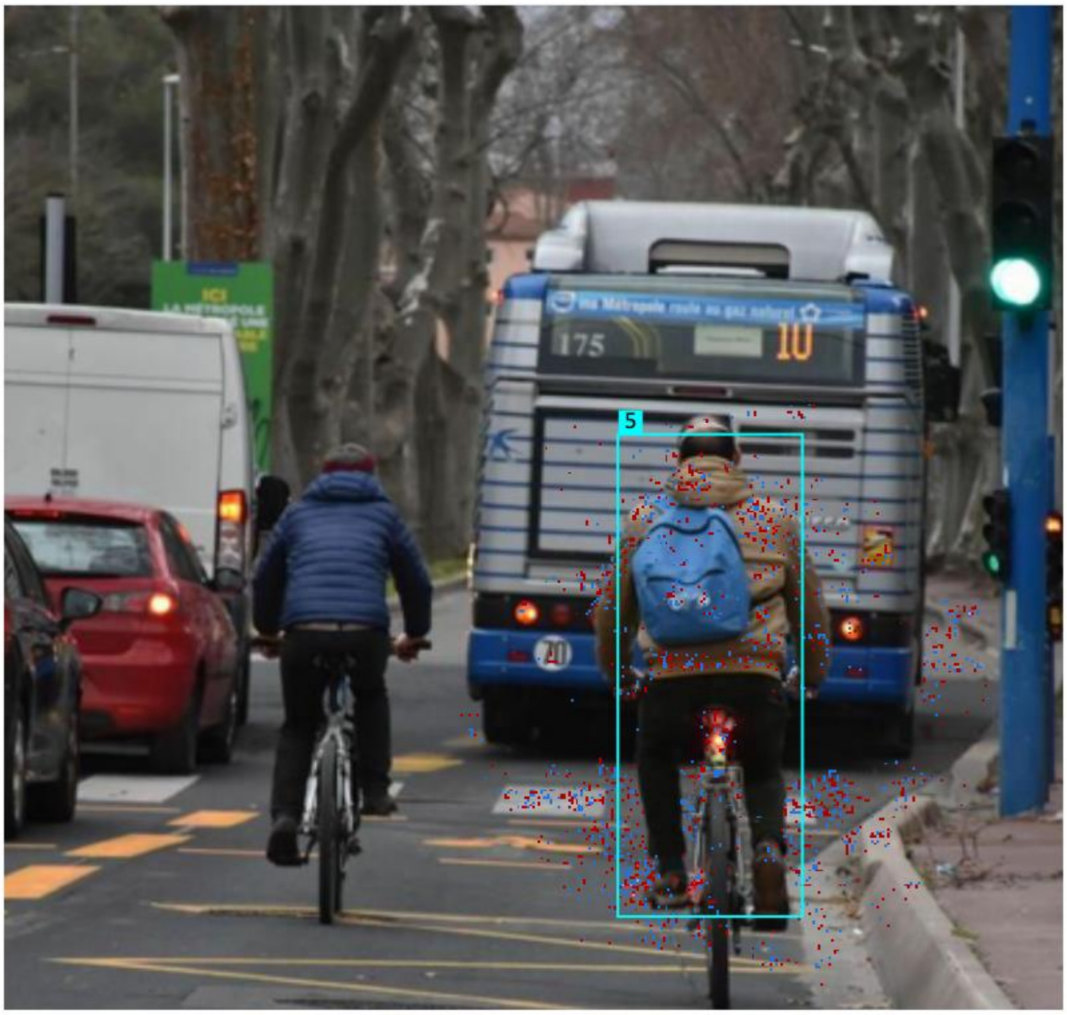


Figure 134 - Example of visualisation of the relevance computed for a particular box, here the perturbation was applied on the whole image

For this study, the image size considered is 640x640 (in RGB), and the local noise applied ranges from 1% to 15% of the image surface.
The study aims to analyse time to demonstrate a stability property and extract the image's associated relevance.

Warning: using abstract interpretation means computing an over-approximation of the output space for all the outputs generated by the ML system. In the case of a Yolo v3 detector, the "visible" output is the boxes with the property of "objectness" more significant than a predefined threshold, all the other boxes are not made visible. This does not mean these discard boxes were not computed and stored in the ML system; they were filtered in the display. This means that the outputs comprise millions of boxes on which an abstract interpreter would also compute the stability and relevance properties. This can lead to a huge output result file (more than several hundreds of gigabytes in this case). This also requires the abstract interpreter to implement some form of filtering based on the output space of the objectness. For example, by considering that the upper (resp. lower) bound of the objectness is more significant than a threshold

Computation time: The computations were run on several images of the same size but with different local noise surfaces. The computation run time takes 15 minutes to 75 minutes for one image with local noise.

<u>LM13: Trained model robustness</u>
Because the ODD of this use case is huge and complex to structure, a strategy to build a robustness assessment on (rare) edge and corner cases would imply using an empirical approach (e.g., field trials). An empirical methodology would imply constructing different scenarios for each class (e.g., vehicle, pedestrian). For each, it is possible to use Table 2 to build complex cases on different semantic levels. For example, a scenario to test the robustness of the pedestrian class could involve pedestrians occulted by décor or vehicle to verify the ability of the model to detect them even on minimal features available. More extreme scenarios could involve using images of disguised pedestrians (like for Halloween) where human features could be primarily distorted.

<u>Conclusion on the detector architecture</u>
Analysing significant detector architecture is feasible using formal methods. However, it requires two precautions:

- limit the localisation of the perturbation to verify one or more particular boxes to match the behaviour of the system,
- implement filtering on the outputs of the analyser to avoid massive result files.

During this study, formal methods were used to measure the stability of both decision (classification) and the objectness of a box. This allows the ML engineer to detect possible defects in its detection ML system.

6.3.1.3 **Time series processing architecture**

<u>Tentative ConOps definition</u>: The system helps detect abnormal heart rhythms for medical diagnosis.

<u>Tentative ODD definition</u>: A typical heart rate comprises several distinct waves, as shown in the figure below. These waves (P, QRS, and T) are essential for interpreting the ECG and assessing the heart's electrical activity by identifying potential anomalies or cardiac dysfunctions. According to the convention, the term "normal sinus rhythm" (NSR) or "normal sinus rhythm regular" (NSRR) indicates not only that the P waves (reflecting sinus node activity) have a normal morphology but also that all other ECG measurements are standard. The criteria include the following:

- A regular rhythm.
- A heart rate between 60 and 100 beats per minute.
- Straight P waves, present in a 1-to-1 ratio preceding each QRS complex.
- A PR interval not exceeding 0.12 to 0.20 seconds.
- A QRS complex not exceeding 0.08 seconds.

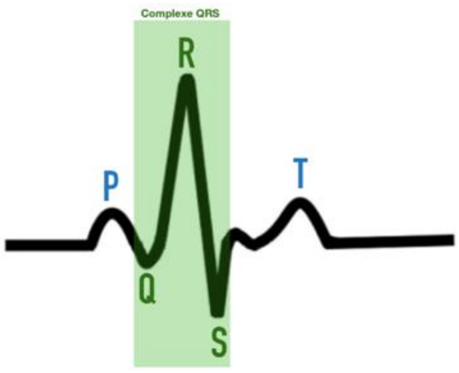


Figure 135 -Decomposition in several waves of a typical heart rate

Architecture and data set

The model has been trained on an open-source data set[200] and which is described in the figure below. Its output is a classification of the time series into 5 different classes (from a normal heart rate to 4 abnormal ones and one unknown one). The data set contains more than 800K time series; however, in order to keep each class balanced, it is restricted to 50,000 time series for each class.

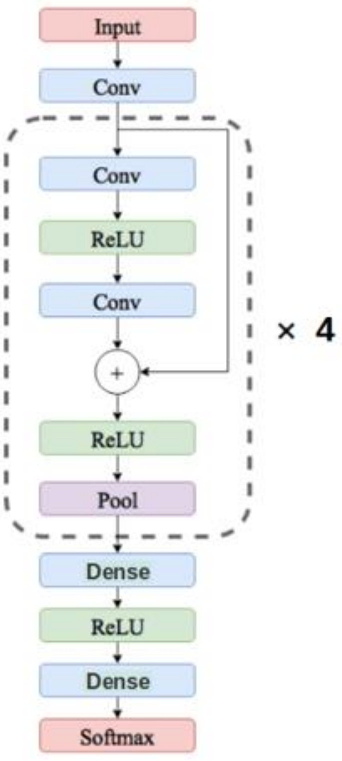


Figure 136 -Architecture of the model processing time series

[200] https://www.kaggle.com/datasets/shayanfazeli/heartbeat/data

In this use case, the Saimple tool (presented in 6.2.2.3.1) has been used for the formal evaluation, along with the scikit-learn Python package for the statistical evaluation.

<u>LM11 : Training algorithm stability</u>
Similarly to the study done on LM11 in Section 6.3.1.1, a stability analysis of this use case can be done using the same Keras 2.10.0. A similar approach will be conducted here in this section, and the reader can refer to Section 6.3.1.1 for more details on the protocol used.

First, an analysis is done to measure the impact of the reduction of the training set on the accuracy of this training set to assess if the training algorithm impacts the model's performance (but not its ability to generalise). Figure 137 shows that the accuracy is relatively stable until a significant portion of the training set (more than 80%) is removed, and the drop is about a few percentage points. This result would indicate that the training system is more stable in this use case. Additionally, as a side effect, this would suggest that the training set is overall too large for the complexity of the task to be performed by the model.

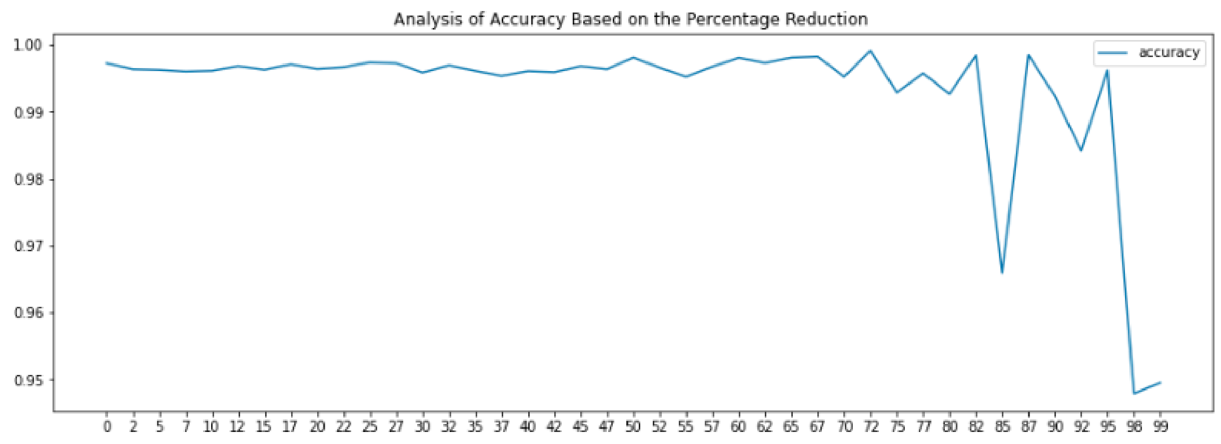


Figure 137 - Evolution of the accuracy of the model depending on how many percent of the training set is removed

When focusing on the performance evolution of the test data set, presented in Figure 138, the model tends to have much wider variation when removing data points for the training set. Again, some unexpected ups and downs in accuracy can be seen when continuously removing data points. This is attested using several performance statistical metrics. Since, in this use case, the number of data points is quite large (more than 500K images), it might be logical to assume that early variation of the accuracy could be less due to the data points removed rather than the training algorithm. On the contrary, removing data points from an already significantly reduced data set could cause wider behaviour fluctuations.

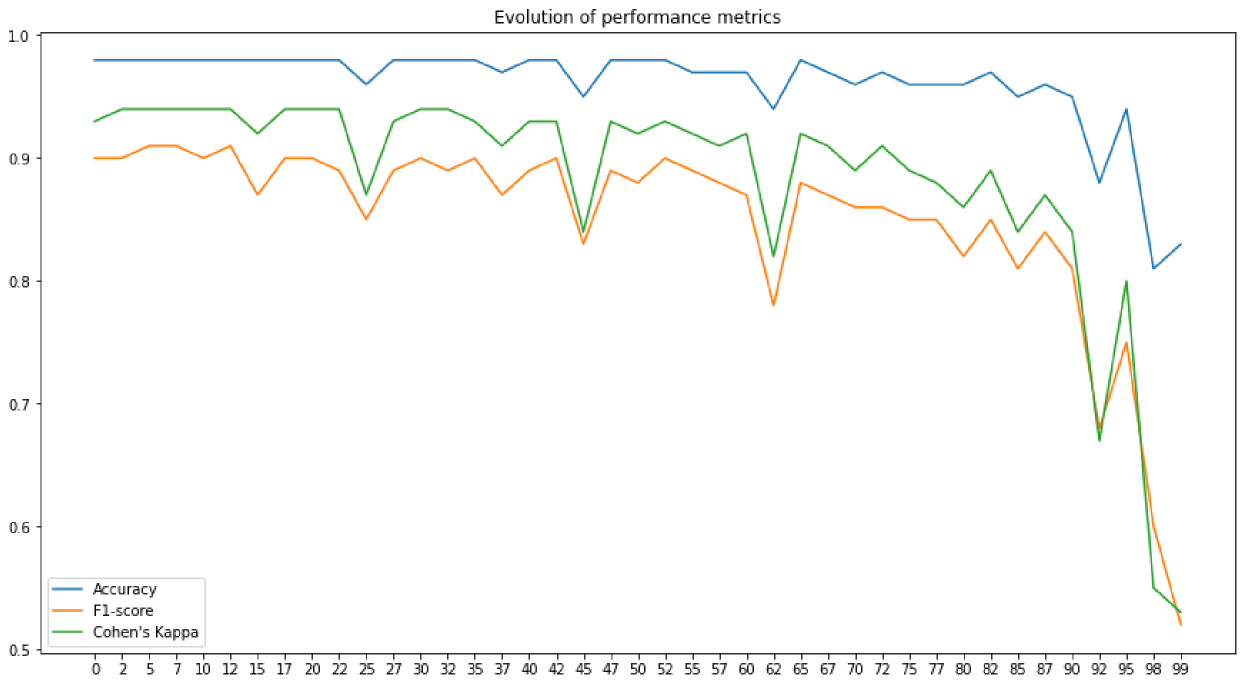
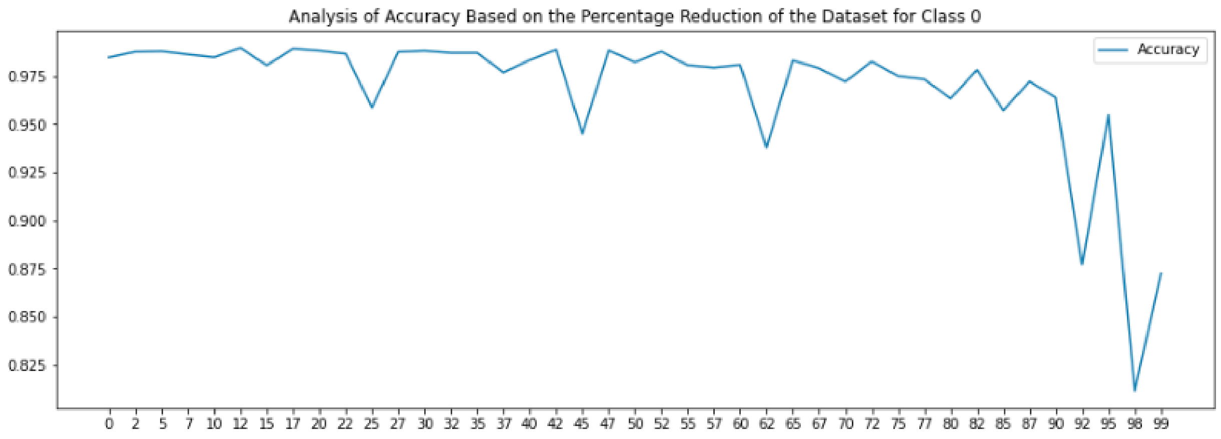
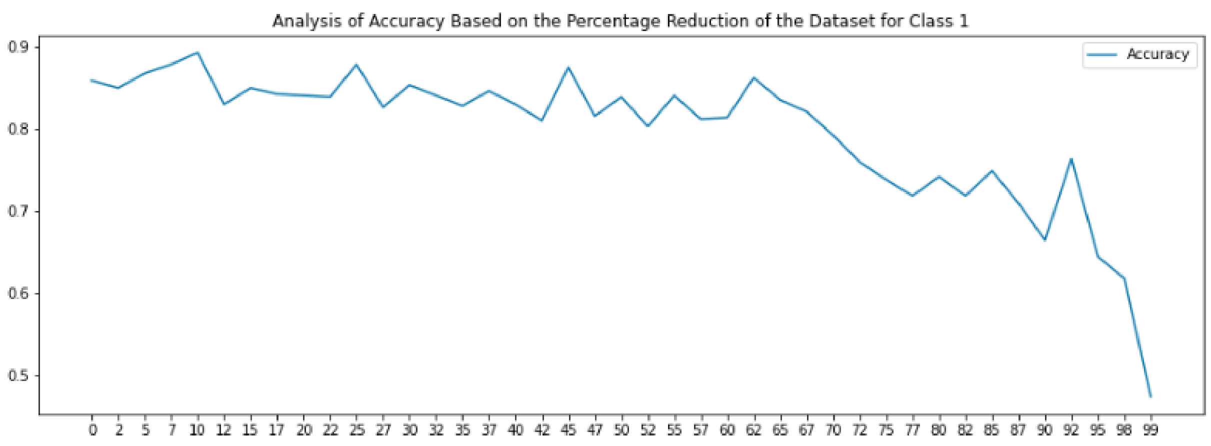
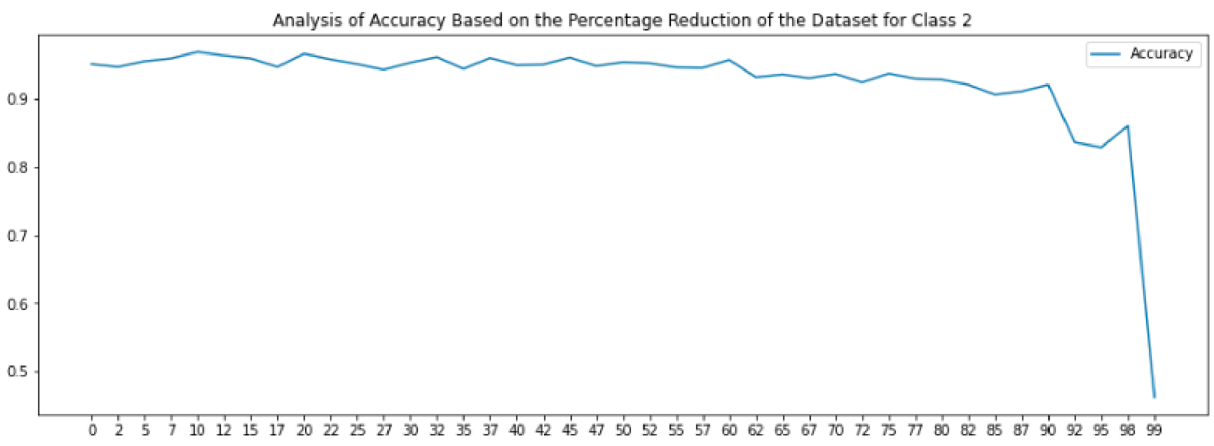


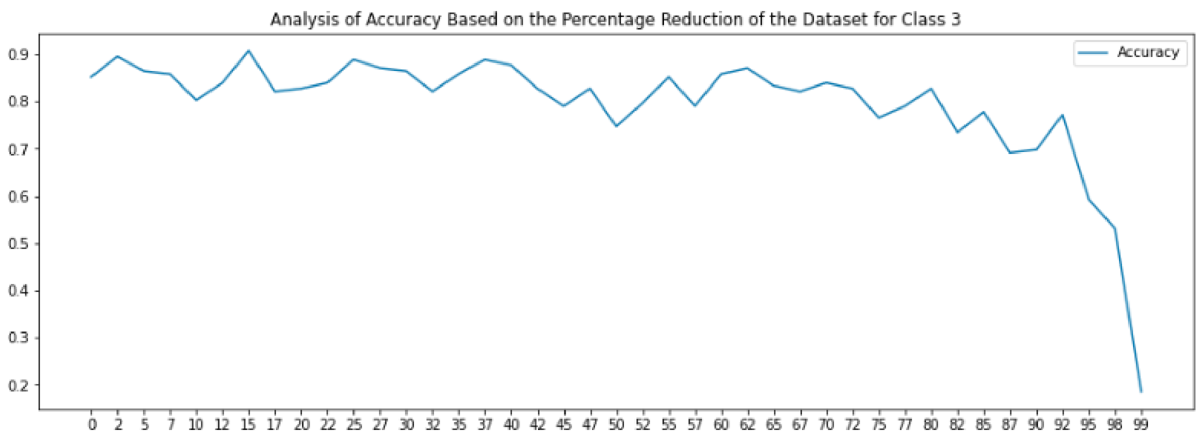
Figure 138 - Evolution using different performance metrics of the model depending on how many percent of the training set is removed.

To understand the source of these drops in accuracy, an analysis can be done for each class to measure if some are more prone to cause instability in the training process. Here again, Figure 139 shows that some classes do not impact the training algorithm much (classes 0 and 4), whereas others can have dramatic drops of accuracy (classes 1, 2 and 3). Again, it is still unclear if this is solely done due to the training algorithm or the quality of the training data set.




Analysis of Accuracy Based on the Percentage Reduction of the Dataset for Class 1
Accuracy
0.9
0.8
0.7
0.6
0.5
0 2 5 7 10 12 15 17 20 22 25 27 30 32 35 37 40 42 45 47 50 52 55 57 60 62 65 67 70 72 75 77 80 82 85 87 90 92 95 98 99


Analysis of Accuracy Based on the Percentage Reduction of the Dataset for Class 2
Accuracy
0.9
0.8
0.7
0.6
0.5
0 2 5 7 10 12 15 17 20 22 25 27 30 32 35 37 40 42 45 47 50 52 55 57 60 62 65 67 70 72 75 77 80 82 85 87 90 92 95 98 99


Analysis of Accuracy Based on the Percentage Reduction of the Dataset for Class 3
Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0 2 5 7 10 12 15 17 20 22 25 27 30 32 35 37 40 42 45 47 50 52 55 57 60 62 65 67 70 72 75 77 80 82 85 87 90 92 95 98 99

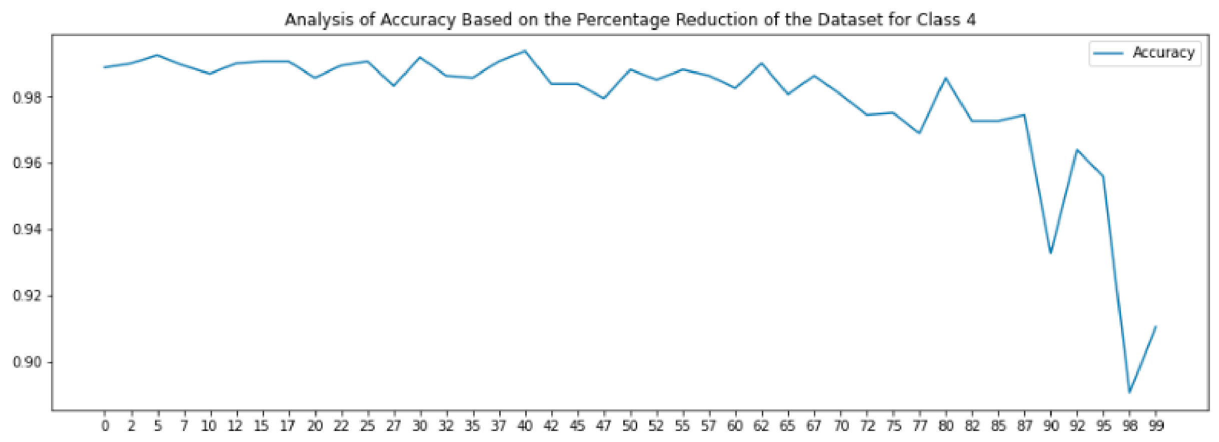


Figure 139 – Evolution of the accuracy on respectively (from top to bottom) 0, 1, 2, 3 and 4 depending of the image removed from the training data set

Finally, an analysis of the impact of removing only one data point for each class has been performed to focus more on the training algorithm itself and mitigate the effects of the training data set quality. Figure 139 shows that the accuracy on the training set is quite stable here since the deviation is roughly around $\pm 1\%$ which could be considered as acceptable. However, Figure 140 shows the variation of the accuracy on the test set, which is here more significant with variation around $\pm 10\%$. This suggests here again that the training process is sensitive to the training data set even in the case of a large one with only one data point removed.

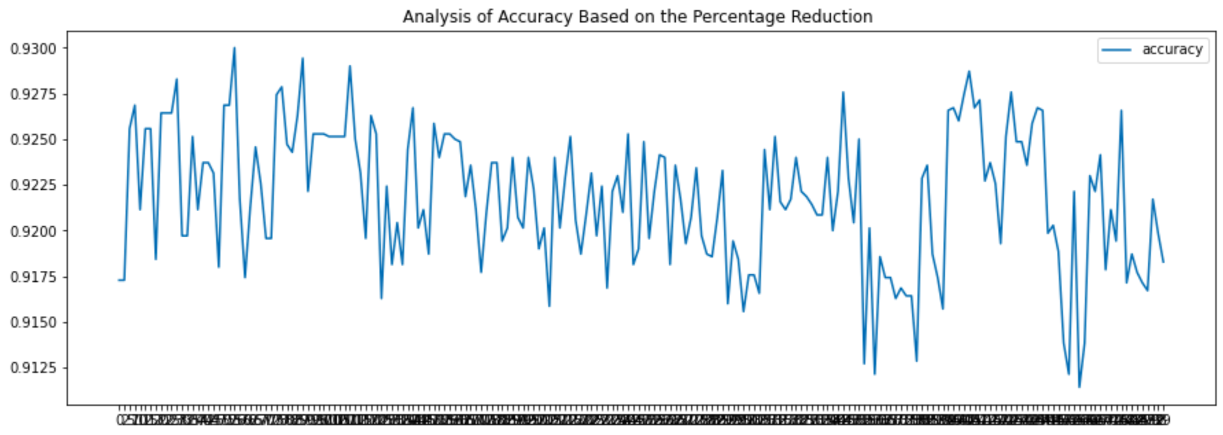


Figure 140 – Evolution of the accuracy on the training data set depending on the image removed from the training data set

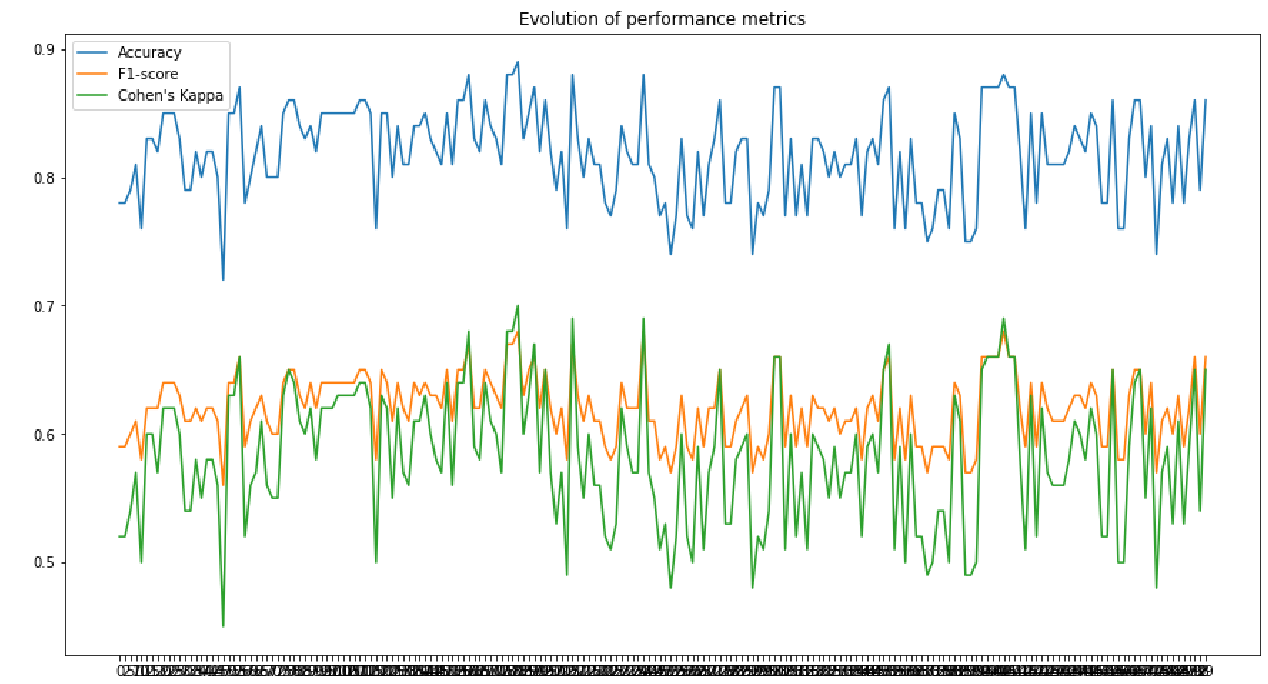


Figure 141 - Evolution of different performance metrics on the test data set depending on the image removed from the training data set

<u>LM12: Trained model stability</u>
Following a classical statistical performance assessment, the model is evaluated using the following confusion matrix (Figure 108), which indicates reasonable performance.

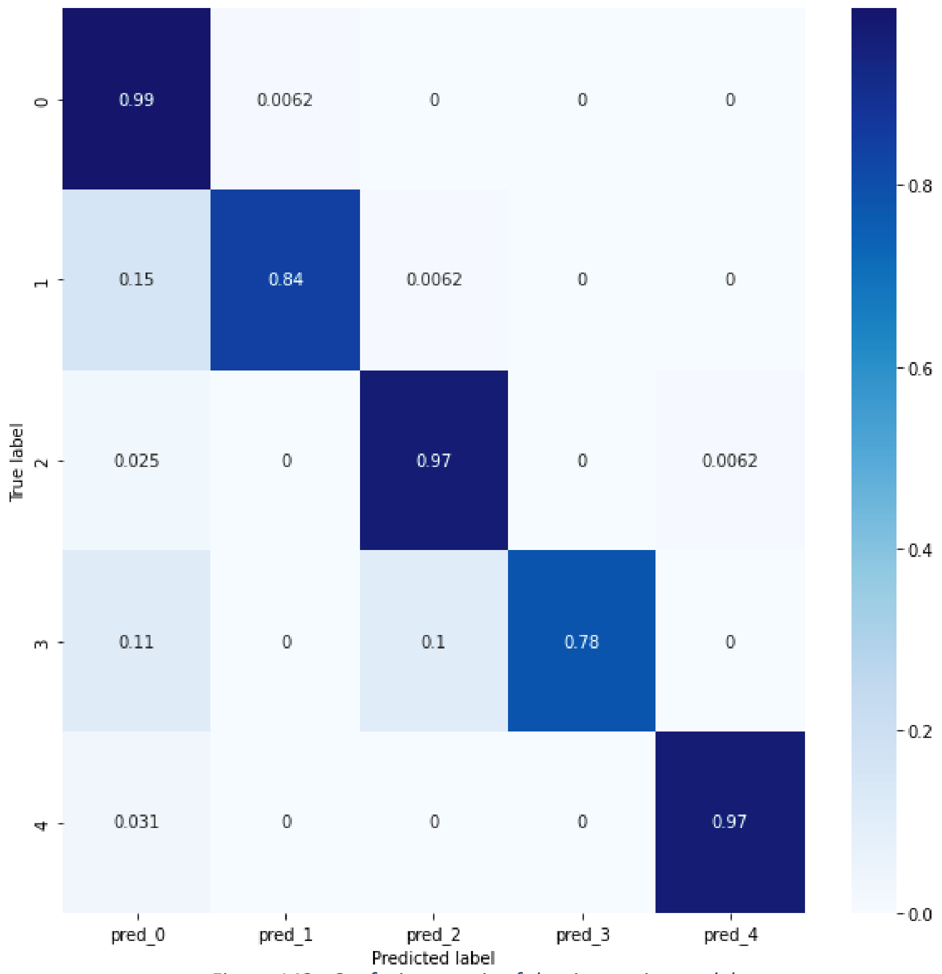


Figure 142 - Confusion matrix of the time series model

The confusion matrix for the model reveals high precision, with prediction scores of at least 97% for classes 0, 2, and 4, indicating overall accurate performance (accuracy). Classes 1 and 3 show at least 78% prediction scores but less than 80%, remaining acceptable. However, it is crucial to investigate why class 1 is predicted as class 0 at a rate of 15% and why class 3 is predicted as class 0 at a rate of 11%. These prediction errors highlight the need for in-depth analysis to identify underlying causes and improve the model's performance for these classes.

To assess the algorithm stability more thoroughly, the study focuses on the stability of the system by studying the regularity of stability between each output class selected. The underlying assumption is that if the system has been well-trained, then the stability should be comparable from one class to another. To ease the reading, we only display the results for two perturbation intensities here.

In the Figure 143 we can see that for class 3, the second perturbation intensity makes more than 2/3 of the inputs unstable. This result is obtained by checking the stability of the decision for two given levels of perturbation using the Saimple tool for each class over the whole data set (training and evaluation). This indicates a potential vulnerability for detecting this particular heart rate rhythm. It could stem from insufficient data set coverage, an intrinsically more complex characterisation of this specific heart rate rhythm (which would require more data to train properly on), or a poor data set quality regarding this class. In the ML development pipeline context, this would cause some feedback loop toward either the training or data qualification processes.

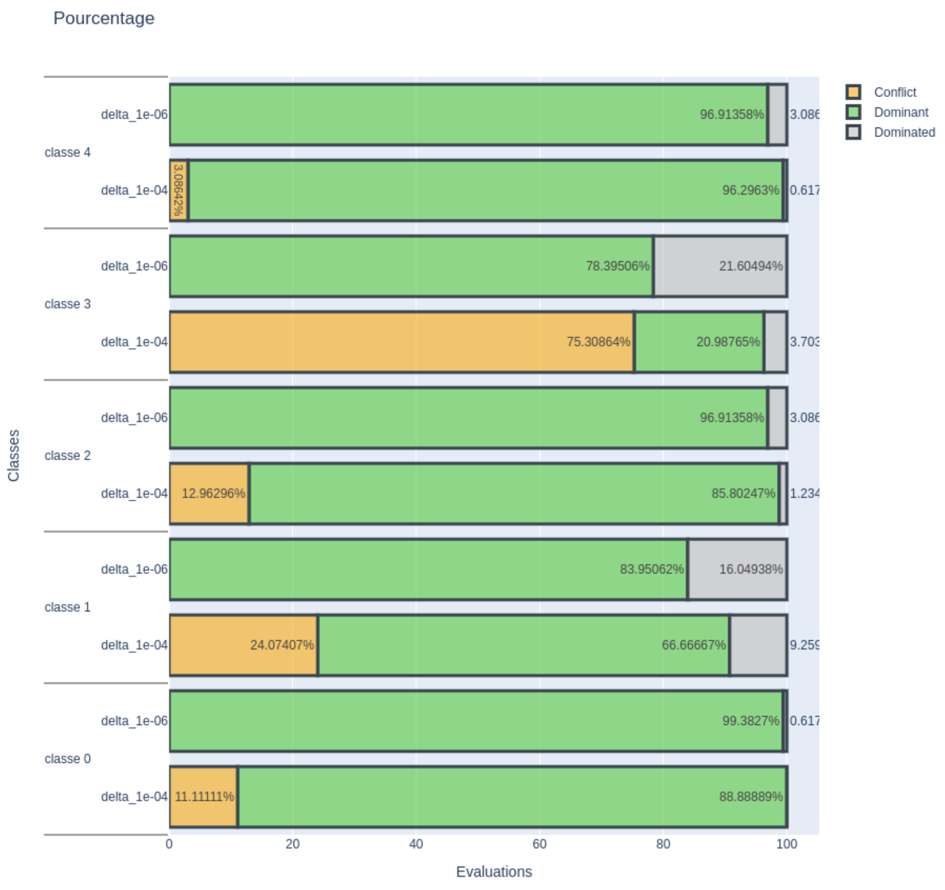


Figure 143 -Stability measurement depending regrouped by output class and for two different perturbations considered

<u>LM13: Trained model robustness</u>

This study has not been done.

However, this use case has allowed us to illustrate the ability of formal methods to detect wrongly annotated data by analysing their stability. Indeed, several of the data points that became unstable were misclassified heart rhythms in the dataset. Anomalies such as a lack of QRS complex or absence of visibility of P or T waves have been identified, indicating potential errors in annotation. This questioning of annotation quality is crucial as it can significantly impact the learning model's performance. Increasing the noise level revealed a high percentage of wrongly annotated heart rhythms, contributing to the loss of well-classified rhythms across all classes.

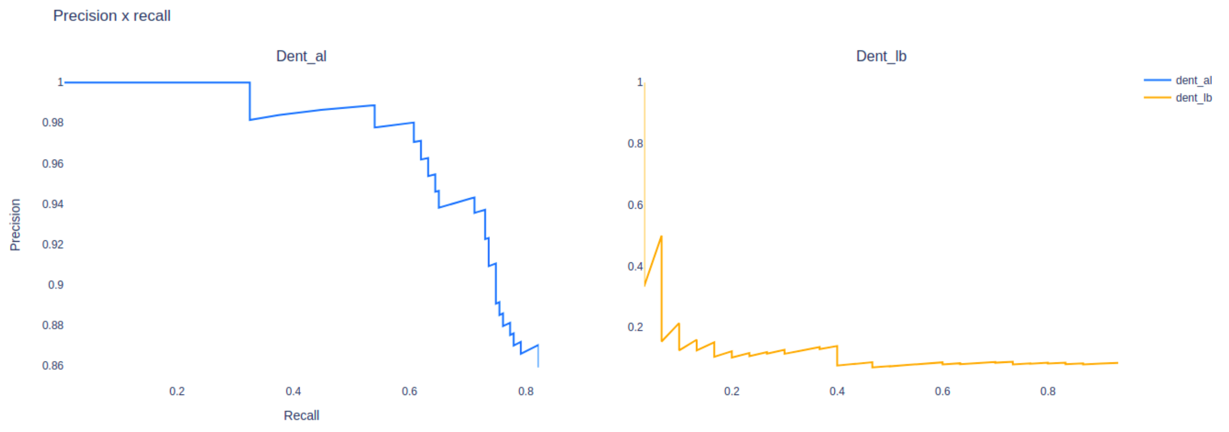
<u>Conclusion on the time series architecture</u>

This study has shown the potential feedback loop from the evaluation of robustness toward the qualification of the training set. Such design modification is part of the generic pipeline detailed in Chapter 7.

6.3.2 Aviation Use Cases Experimentations

6.3.2.1 **AVI use case**

A more complete description of the ConOps and the associate ODD of this use case is available in Section 3.4.This use case relies on an architecture commonly used for detection tasks called Yolo (also described in Section 3.4) in its version 5. This version is already quite an extensive neural network with more than 21 million parameters and attested performance in the state of the art (Jocher et al., 2022). The version initially set in the use case relied on a SiLU activation function (Elfwing et al., 2017) that presents some challenges in terms of computational power to perform its analysis. While still able to do so, it has been proposed that we revert to a LeakyReLU implementation to speed up the analysis. Using LeakyReLU was mainly motivated by hardware constraints to reduce the computing time and resources needed on the MLEAP server. SiLU is a more costly activation function to approximate, but it often has better performance metrics. This change has simplified the analysis without altering the performances too much compared with the original SiLU model on dent_al, but it has impacted the performance on dent_lb.
The training of the LeakyReLU architecture was done over 300 epochs and over 64 batches of data over four days using the CoCo data set[201]. It has been performed over the 80 classes of the data set and 330K training images. The fine-tuning step[202] done on the Airbus data takes a few minutes for 15 epochs and is done using 3695 images over 3 classes. This step is done using the Ultralytics feature to reduce from 80 to 3 classes automatically. Figure 144 presents the performance of the fine running over the classes dent_al and dent_lb. The model is globally good at detecting the dent_al defect (AP 80.13%), but not at all on dent_lb (AP 14,64%); therefore, it should be used chiefly for dent_al detection. It should be noted that the original SiLU model's performance was 52,10% on dent_al and 66,91% on dent_lb. The results mentioned here were obtained on a smaller test set. With the entirety of the data set, we obtain 54,23% for dent_al and 69.28% on dent_lb (consistent with the result obtained in 5.8.3.1.2.4).



[201] https://cocodataset.org/
[202] Following the definition of the EASA CP this step of fine tuning can be viewed as a transfer learning step.

Figure 144 - Fine-tuning performance of the model on the Airbus data.

LM11: Training algorithm stability

To implement the LM11 requirement in this use case in a scalable way, one could either analyse the behaviour of the training algorithm on the initial Coco training set (Lin et al., 2014) or on the fine-tuning done using the Airbus data. The initial training usually takes several days, whereas the fine-tuning can be done in minutes. Analysing the stability of the fine-tuning algorithm is still relevant even if it is only part of the process since the pre-trained model can be considered a fixed model. In contrast, most of the stability required regarding the ODD depends on fine-tuning.

The methodology used here consisted of gradually removing more and more data points from the fine-tuning data set and measuring the model's performance. Figure 145 presents the results obtained by considering the evolution of the accuracy of the trained model under the progressive deletion of its fine-tuning dataset. Results are shown for two classes the model should detect (dent-al and dent-lb). The dent_al' performance (on the left) holds around 55%, even with more than 60% of the dataset being removed. The dent_lb model performance (on the right) is more unstable under the same deletion process. A significant increase in its accuracy can be seen when removing around 70% of the dataset. However, it should be noted that in the case of dent_lb, the initial accuracy is very low (around 10%). Therefore, it is not unrealistic for different fine-tuning conditions to lead to significant performance changes.

Demonstrating the LM11 on this particular use case has been challenging in terms of resources needed, with results that echoed those obtained on the open-source use case presented in Section 6.3.1.

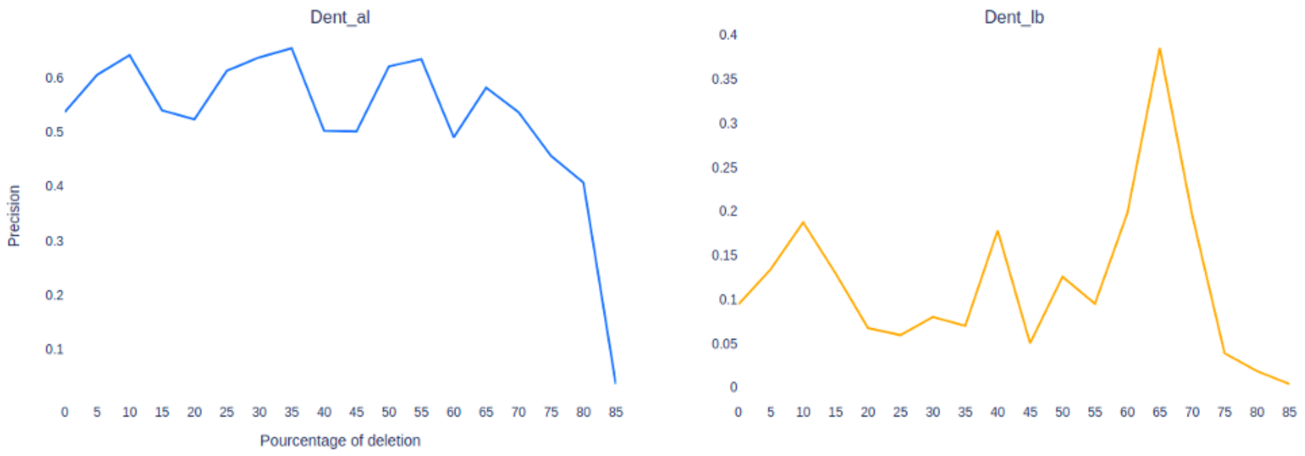


Figure 145 -Evolution of the precision performance of Yolo v5 while gradually removing data points in the fine-tuning data set.

LM12: Trained model stability

Stability can first be assessed using statistical methods; for this, a confusion matrix can be created. Figure 146 presents the results obtained using this method on the original SiLU model. This shows that the model is highly accurate on the class dent_lb. However, there can be some confusion between the background elements and this class (since 13% can be misclassified as such and create false positives). Although the precision for class 0 is high, the inclusion of the background can affect the overall performance of the model, notably by decreasing overall precision or affecting other metrics such as recall or specificity. For the dent_al class, the accuracy is 0.89, but 0.67 of the background is classified as dent_al, which lowers the accuracy.

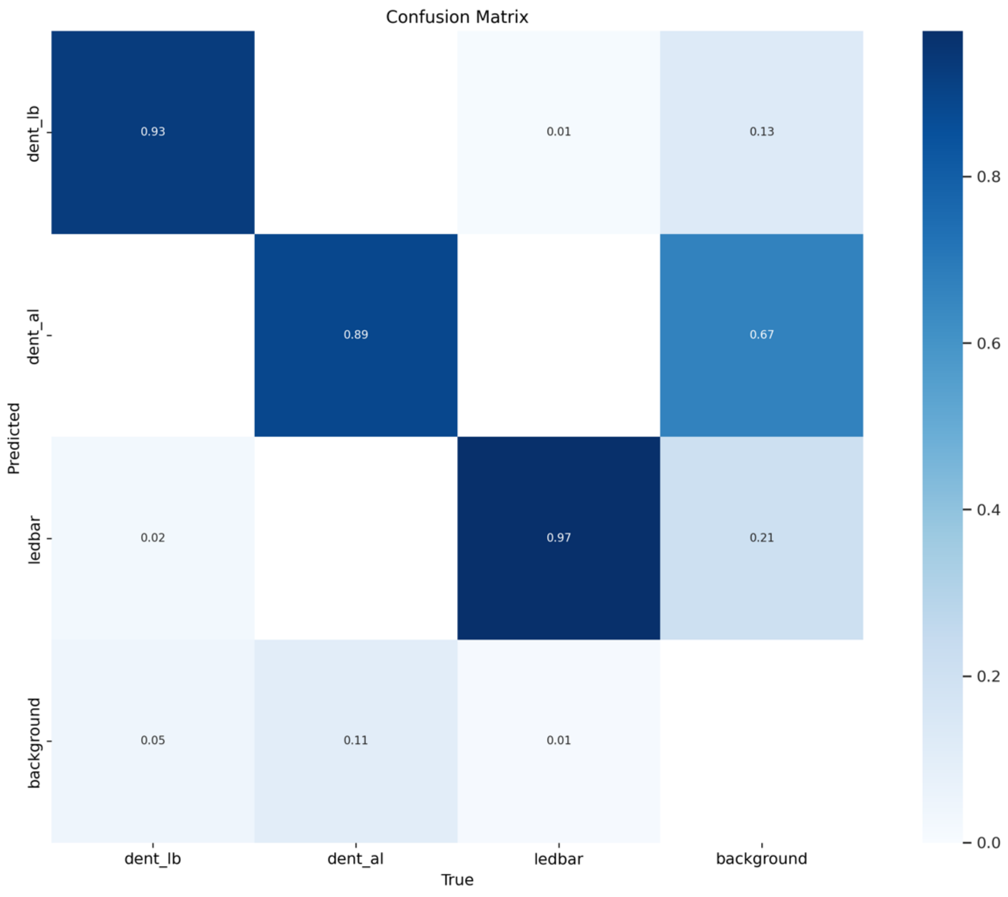


Figure 146 – Confusion matrix of the YoLo-v5 architecture

A formal validation approach has also been performed on this model to verify its stability and relevance using Saimple (see Section 6.2.2.3.1). The results show that analysing a Yolo v5 is feasible using formal methods (echoing the results presented in Section 6.3.1.2) but remains a challenge regarding computational resources. Formal verification allows us to check whether or not the detection would remain stable if the images were different. It also allows us to verify what features are used to make the classification (using explainability techniques). The analysis is designed to explore the system's behaviour against local perturbation located around the boxes detected on the initial image. The goal is to measure if the system stays stable in detecting a defect, considering some perturbations. The usual box for dent_al or dent_lb is around 40x40 to 50x50 in size. Therefore, the perturbation considered was 10 to

20% larger than these in size, considering perturbations both on the defect and around it. The study was done using 48 images randomly selected in the test data set[203], which contains 105 boxes; only 61 of those are correctly classified by the model (without any perturbation). On the 61 correct boxes, 31 of 49 were of class dent_al, 6 of 14 were of class dent_lb, 24 of 42 were of class Led Bar. Each validation tool takes around 30 minutes to be performed on the MLEAP server.

For small perturbations (around ± 1 in pixel variation) the model stays stable for all its pre-existing boxes (whether correctly or wrongly classified). Also, the boxes with an initially low objectness value are more prone to become unstable. This is measured by comparing the width of the possible outcome for objectness and confidence on each box. Boxes with an initial objectness inferior to 0.5 had a two-order-of-magnitude larger width than the box with an objectness superior to 0.9 (further study would be necessary to confirm this result). Figure 147 and Table 65 present a more detailed view of an outcome using a formal method. Both the relevance and the outcome of objectness and classification are presented.

We can observe the model's behaviour from different angles by combining multiple results, such as stability and relevance. For example, during analysis of two relevant with very different shapes between box 1 and box 3, where the latter seems to use pixels that are not particularly tied to the defect. Interestingly, this very box's objectness is inferior mainly to the one from box 1 (see Table 65). This could either warn the AI system designer of a possible misclassification of this box (see Section 7.2.3.4) or it could be a warning of a subclass of dent_al defect that is not properly treated (like in the use case described in Section 6.3.1.1).

No meaningful results were obtained for larger perturbations since the over-approximation caused by the formal method was too large. This suggests that for large neural networks such as Yolo v5, a better compromise of formal verification should be defined to limit the over-approximation while still ensuring the method's scalability.

Box number	Class	Confidence	Objectness
1	dent_al	[0.99727,0.99728]	[0.9296,0.9297]
2	dent_lb	[0.99739,0.99739]	[0.7836,0.7837]
3	dent_al	[0.99462,0.99468]	[0.4477,0.4616]

Table 65 – Detail of the abstract results for each box for the picture show in Figure 114.Objectness and confidence are two key indicators used by Yolo v5 architecture to determine whether a box is accurate.

[203] As a reminder a local perturbation on a 50x50 area with ± 1 pixel variation corresponds to a 3^{2500} potential images.

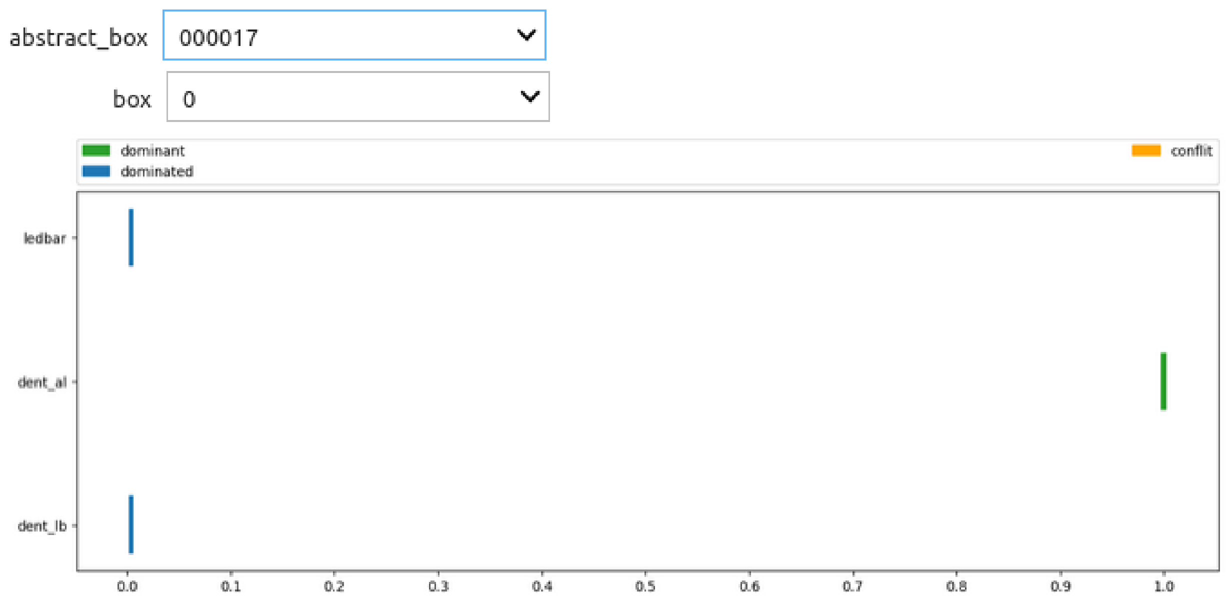


Figure 147 – Detailed view of the stability of the classification (confidence parameter) of a box using formal method. In this case the box is stable on its classification dent_al

<u>LM13: Trained model robustness</u>

Verifying the model's robustness corresponds to exploring more extreme perturbations defined within the ODD to explore the behaviour around edge and corner cases. This study focused on corner cases constituted by blur variations (global, horizontal, or vertical) and luminosity variations. These types of perturbation represent the conditions to which a camera can be subjected. For LM13, the original SiLU model was used. For each noise considered, the results were obtained by analysing the model's performance on the test set composed of 522 images.

Blur noise is defined as an intensity scale from 0 to 9. Noise intensity corresponds to a Gaussian noise (with mean set to zero and standard deviation set to noise intensity parameter/10). Figure 148 gives an example of the blurring obtained at each intensity. Figure 149 shows that the model's performance (mean Average Precision) does not hold at most of the intensity testing of this type of perturbation for the class dent_al. The class dent_lb is resisting more but also suffers from a drastic performance drop before stabilising itself around 30%.

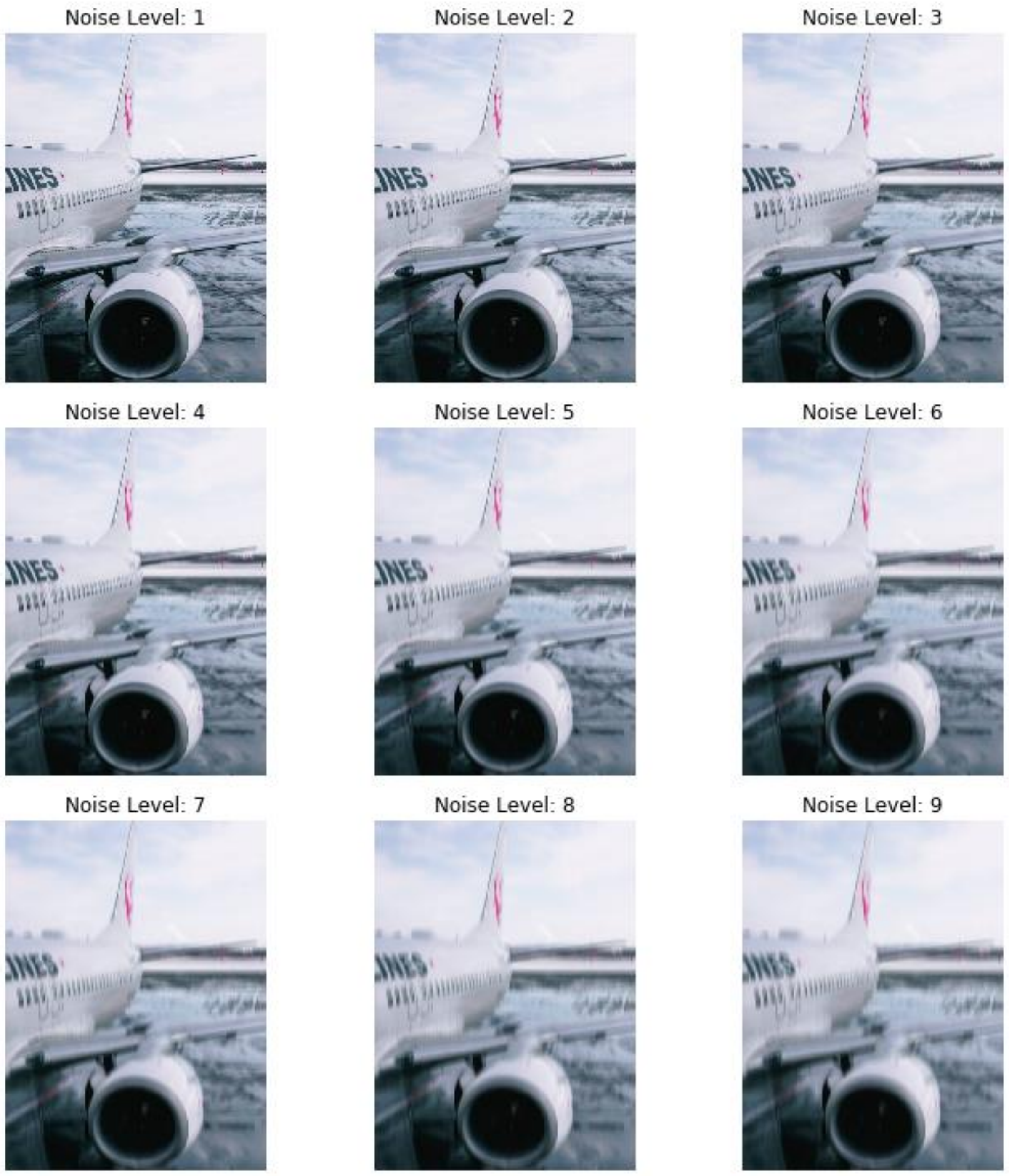


Figure 148 – Example of variation of an image using different blur intensity (without any blur due to movement)

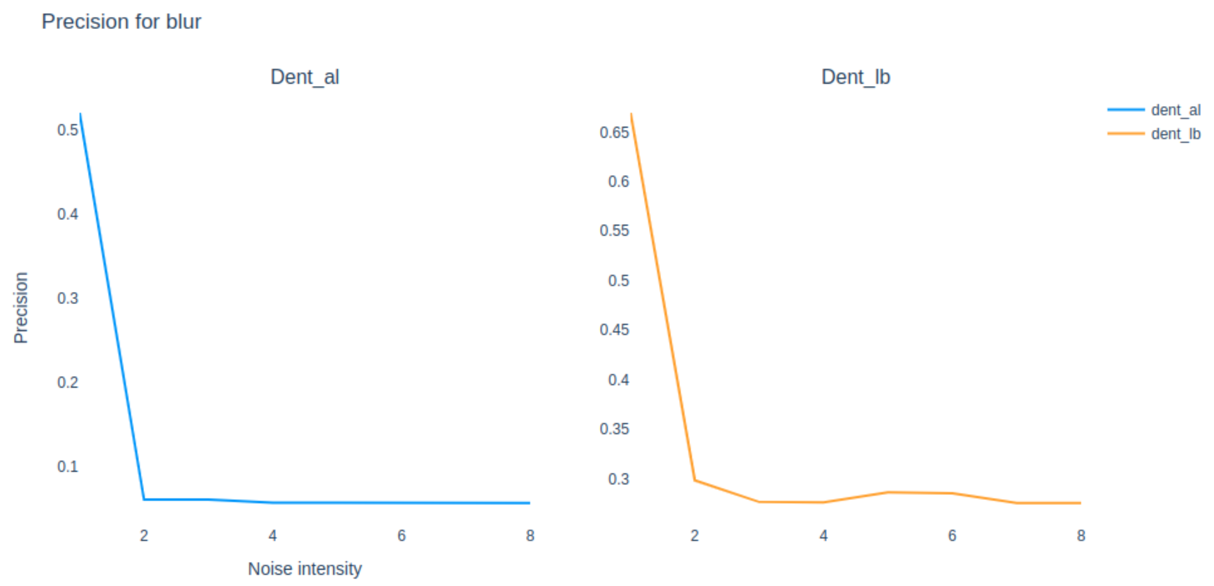


Figure 149 – Performance of the model on edge cases defined by the gaussian blur perturbation.

Horizontal and vertical blur is defined using the length to which pixel we move around given a specific direction. Here, the length corresponds to the distance over which the image pixels will be moved in the direction specified by an angle. A higher length value results in more pronounced motion blur, with pixels moving over a greater distance. The length is defined as 10 times the noise intensity. Figure 150 and Figure 151 show that the model's performance (mean Average Precision) does not hold under this kind of perturbation for any of the two classes considered. The only nuance is that the class dent_lb seems to resist a bit longer before having the same poor performance as class dent_al.

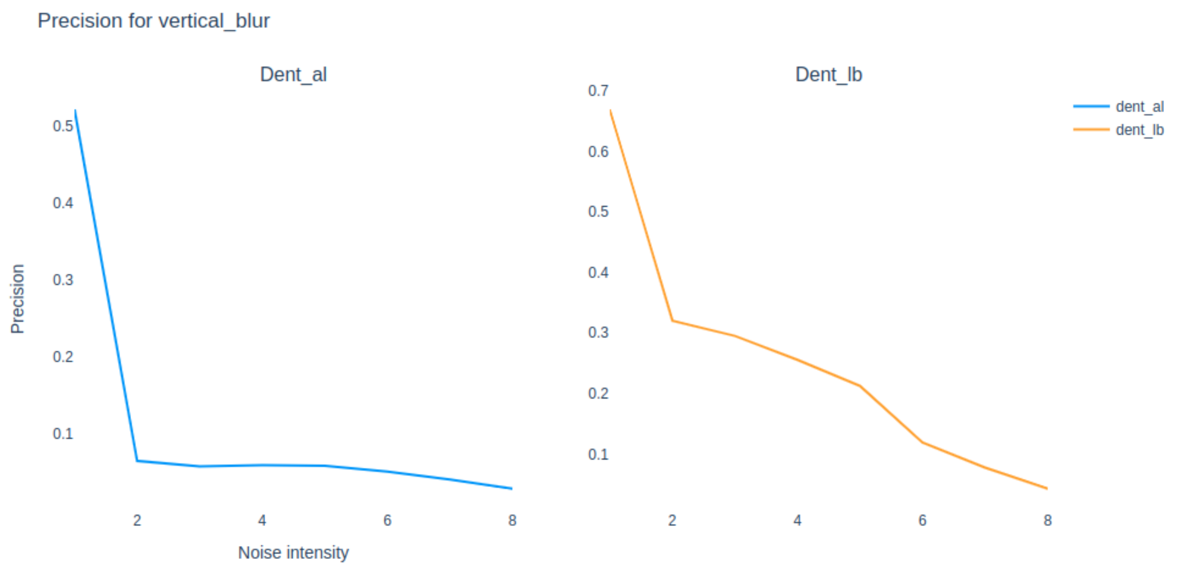


Figure 150 – Performance of the model on edge cases defined by the vertical blur perturbation.

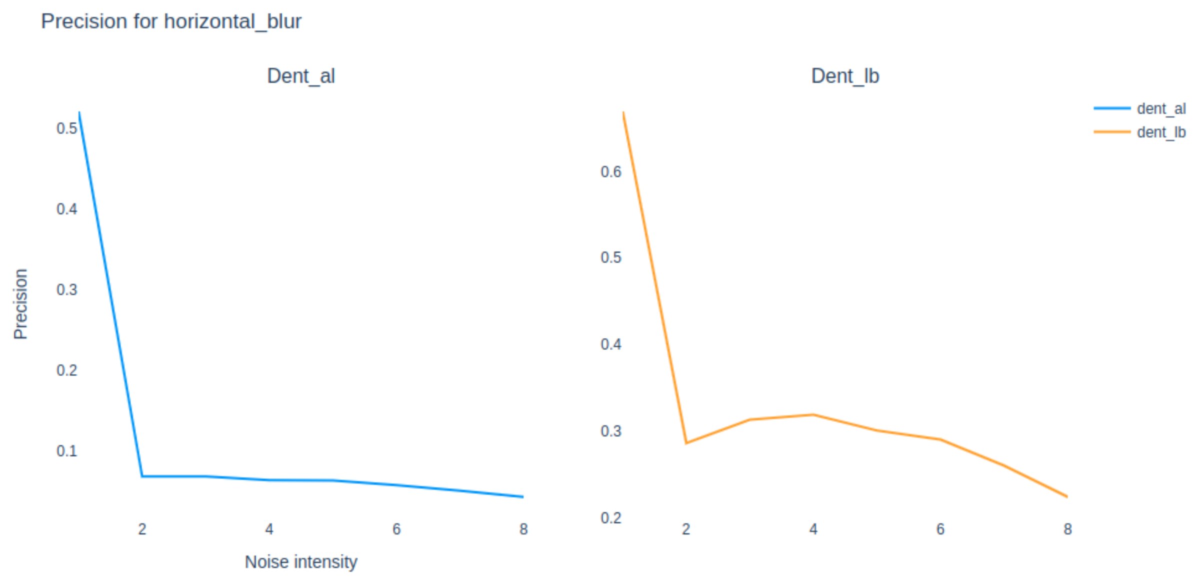


Figure 151 – Performance of the model on edge cases defined by the horizontal blur perturbation.

Gaussian noise corresponds to the standard deviation, which controls the extent to which the added pixel values will be dispersed around the mean (generally zero for Gaussian noise). A higher standard deviation value means that the noise values will be more dispersed, resulting in more intense noise in the image. Here, the standard deviation is set by dividing the noise intensity the user sets by ten. Figure 152 shows that the model's performance (mean Average Precision) does not hold under this kind of perturbation for the class dent_al. For the class dent_lb defect, the AI system also resists a little before performing poorly.

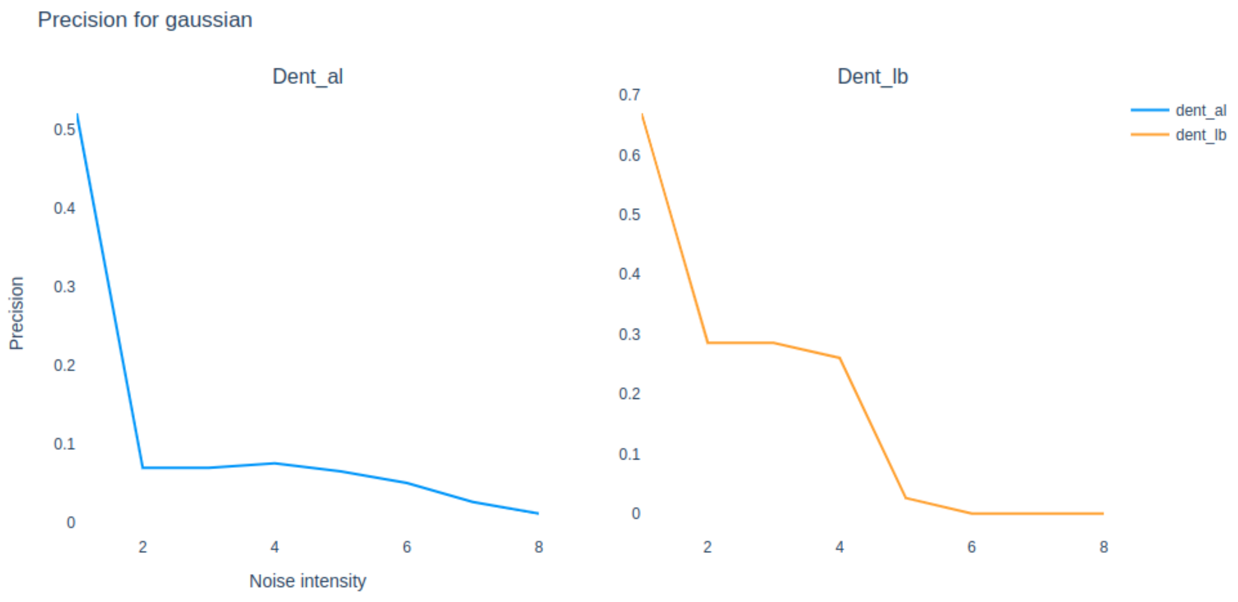


Figure 152 – Performance of the model on edge cases defined by the Gaussian noise perturbation.

Brightness intensity is defined by a positive shift of every channel of every pixel of the image, which causes the image to get whiter and whiter. The shift is defined as $1 + (1/4) \times \mathrm{noise\ intensity}$. Figure 153 shows that the model's performance does not hold under this kind of perturbation for any of the two classes considered.

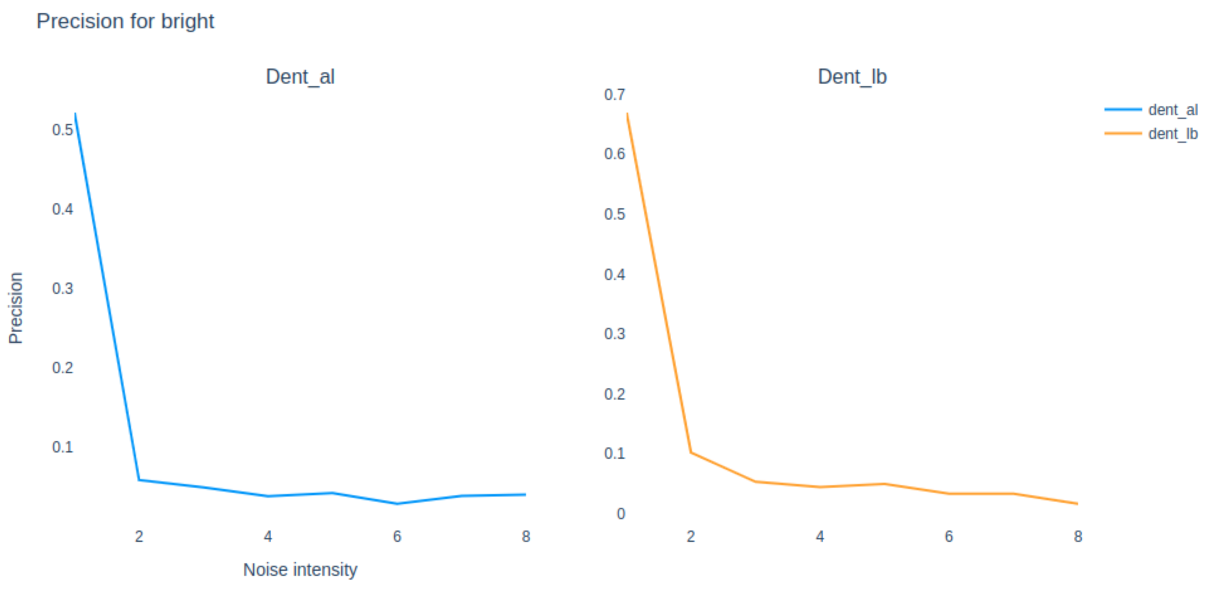


Figure 153 – Performance of the model on edge cases defined by the brightness noise perturbation.

The different experimentations done for LM13 show that the original SiLU model seems to better handle more perturbation for class dent_lb than dent_al. This shows a possible imbalance during the training process in terms of the number of defects of each class present in the training set or a difference in training difficulty between the two types of defects (dent_al being maybe more difficult to train on than dent_lb). It is possible to consider data augmentation to improve the results of this model (see Sections 5.5.4.2 and 6.2.1.2).

<u>Conclusion on the AVI model</u>

The analysis performed on the AVI (using LeakyReLU) showed that:

- LM11 can be very hard to perform on large neural networks that require training for many days. This could be a real bottleneck for the applicability of this requirement as it will impose either to rely on an extensive HPC infrastructure (which is not easy for every company, especially SMEs), or rely on a smaller neural network (but frugality is still an open-topic for the state of the art). In any case, the fine-tuning AVI model seems to handle even considerable variation of the training data set, which might imply that in this use case, the process does not necessarily require as much data as it was used;
- LM12 can benefit from both statistical and formal methods to assess the system's vulnerability. In particular, combining several metrics (e.g., stability and relevance) can help identify how to improve the system's overall performance. However, formal

methods still suffer some scalability challenges to be deployed on large-scale neural networks such as YoloV5;

- The LM13 application on the AVI use case has shown an imbalance in terms of robustness between two classes. This would suggest revising the training by either enhancing the training database or using data augmentation techniques.

6.3.2.2 STT use case

Section 3.3 provides a complete description of this use case's ConOps and associated ODD. This use case relies on several architectures described in Pytorch or the KALDI project. Depending on the architecture, the workflow is not always the same. The study here compares the Wav2vec and KALDI architectures.

LM11: Training algorithm stability

LM11 requires retraining the model by changing the training data set and measuring the change on the trained neural network. It should be noted that both architectures (KALDI and Wav2Vec large) are massive and require a lot of time to train on the available datasets, which are also very large. For example, the training on the GPU of the MLEAP server was impossible due to the maximum amount of memory (16 Go) not being sufficient; resorting to CPU training was possible but required an estimate of 2000 days of computing time to perform one training. This limitation reminds us of the strong dependence on the hardware capability (mainly GPU) that this kind of architecture imposes and the potential problem that this dependency implies (accessibility, sovereignty…). Resorting to a smaller version of the neural network (possibly for the Wave2Vec architecture) can make the training feasible but does not necessarily end with a neural network with good performance.
The Wav2Vec Base is a variant of the Wav2Vec Large, which is smaller (95 million parameters instead of 317 million). This model was used to test the server's ability to train repeatedly using always the same parameters.

- learning_rate: 0.0001
- warmup_steps: 1
- num_train_epochs: 15
- logging_steps: 250
- weight_decay: 0.005
- number of data samples for training: 10k

Table 66 presents the time required to train the Wav2Vec Base architecture on the MLEAP server. Only the last test allows GPU training to be feasible when performing only 2 batches (to speed up the process), in only 10h. However, regardless of the hardware used (CPU or GPU), the performance of the model was still very insufficient. All networks were having a WER performance metric of around 1.0, whereas the initial model had a WER of 0.35. The LM11 cannot be applied without the proper hardware infrastructure for all these reasons. This is a crucial point regarding the reproducibility of the methodology employed for certification purposes. LM11 would maybe impose to the auditor to access the whole computational infrastructure the applicant uses or have an equivalent (costly) one.

Table 66 – Training time cost for the Wav2Vec Base architecture on the MLEAP server

Number of batches	GPU training time	CPU training time
8	CUDA out of memory.	75h
4	CUDA out of memory.	53h
2	10h	41h

<u>LM12: Trained model stability</u>

Section 5.9.3.1.4 presents a study and its conclusion on the performance of the KALDI model using statistical analysis. In this context, the WER metric was selected to measure the model's performance.

This section focuses on the feasibility of analysing the STT use case using formal methods. To the author's knowledge, no work has been done in the state of the art regarding using formal methods in such cases. This experiment is thus a world premiere that has been performed with the tool Saimple (see Section 6.2.2.3.1). The study uses one recording selected for its length and difficulty in the test data set. The study consists of adding a small perturbation on the whole input recording to consider this specific recording and the recording that would be a little bit different at each instant.

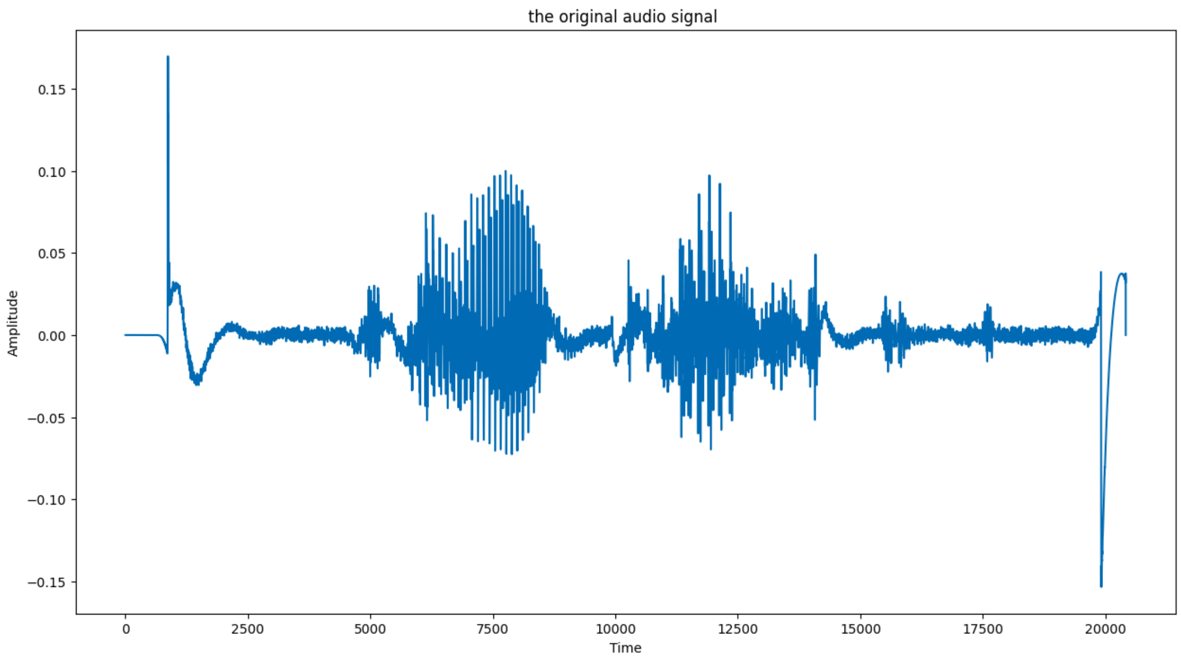
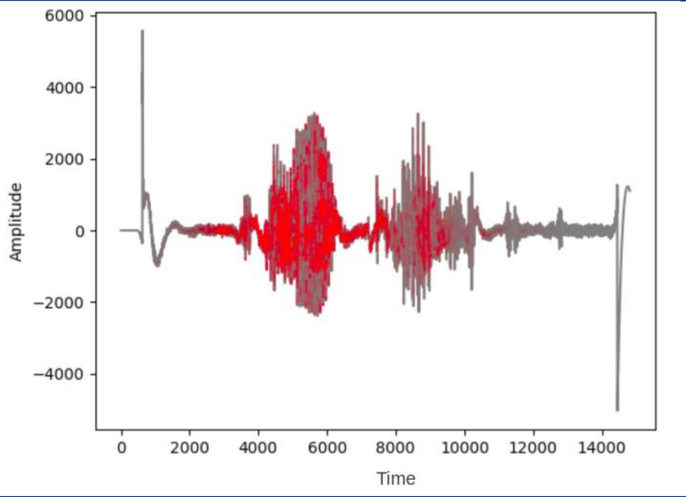
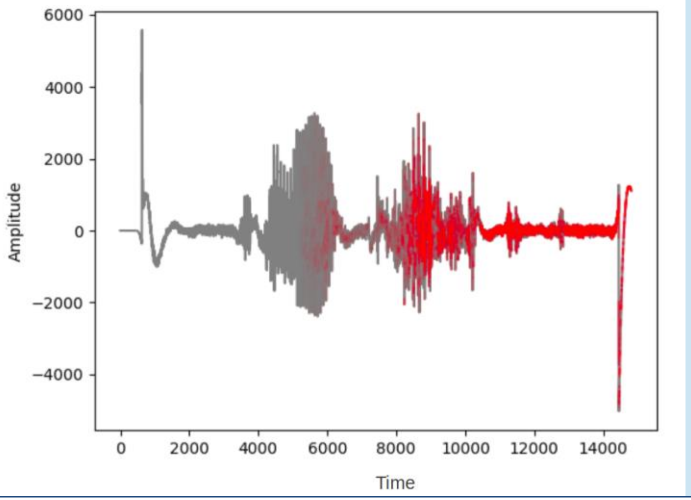


Figure 154 – Recording used for the testing of the Wav2Vec model, whose transcription is "and jersey eight four six uniform on landing do you want us to backtrack to vacate mike four or vacate at the end mike eight"

An output's relevance corresponds to the input's dimension to its calculation. In the case of the speech-to-text application, the output is the transcript. However, each word in the transcript can be viewed as its output. Therefore, it is possible for each word produced by the system to compute using formal methods what dimension in input contributed to its appearance. The goal here is to confirm if the word output matches the right part of the input sound and measure which part of the signal is susceptible to influencing the change if it is subjected to noise. Here, the input is a recording, therefore, the dimensions in the input are the volume amplitude at each instant of the recording. Figure 155 illustrates the results obtained for three different sentence words in the transcript. The relevance is shown by drawing in red the most critical instant of the recording for this word. It can be seen that the model uses parts of the recording that are not entirely associated with the given word, which is in part expected in the architecture of the Wav2Vec model, which uses context to make the transcript. These kinds of results are new and do not allow for proper interpretation and positive feedback for the system's design yet. However, producing this kind of result can have a significant impact in the future since it would be possible to reconstruct the contributing part of the input from the output even in the presence of a perturbation in the recording.

Re-transcription	Relevance
MINIMUM CLIMB SPEED	
MINIMUM CLIMB SPEED	

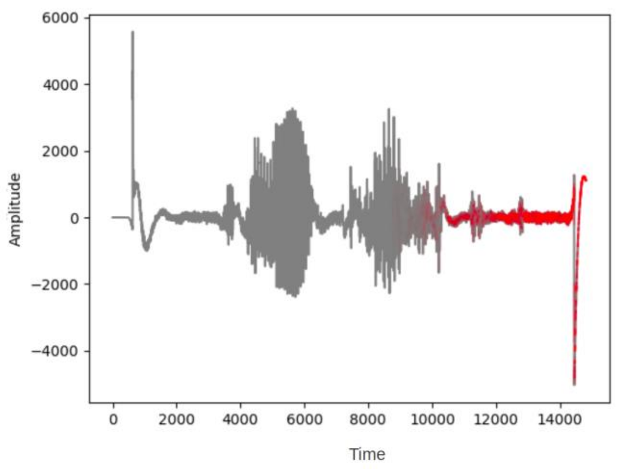
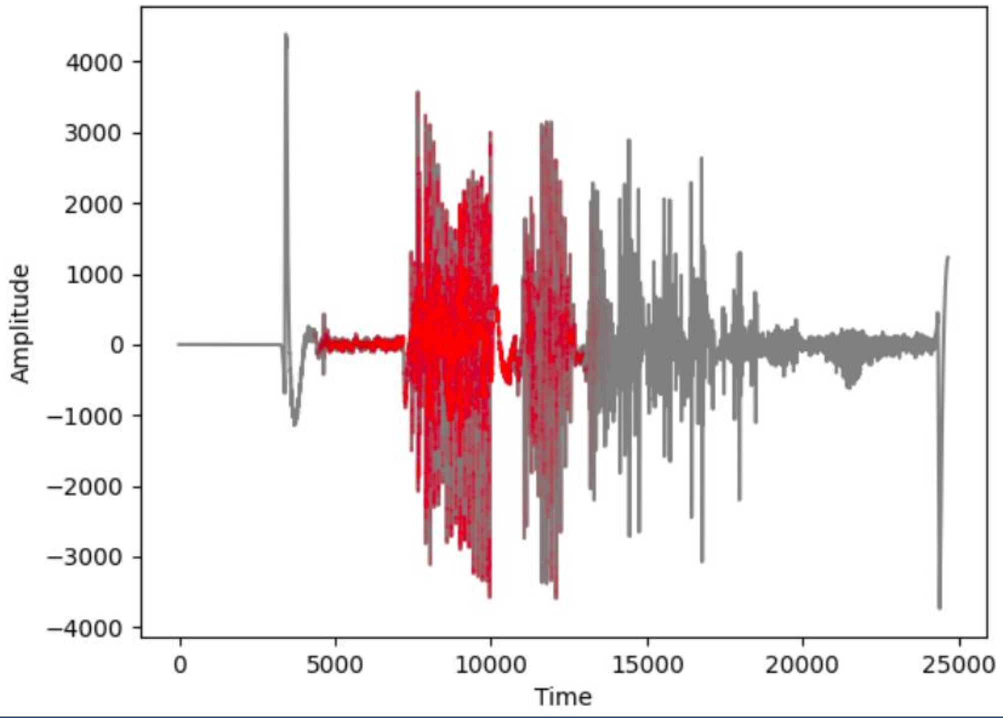
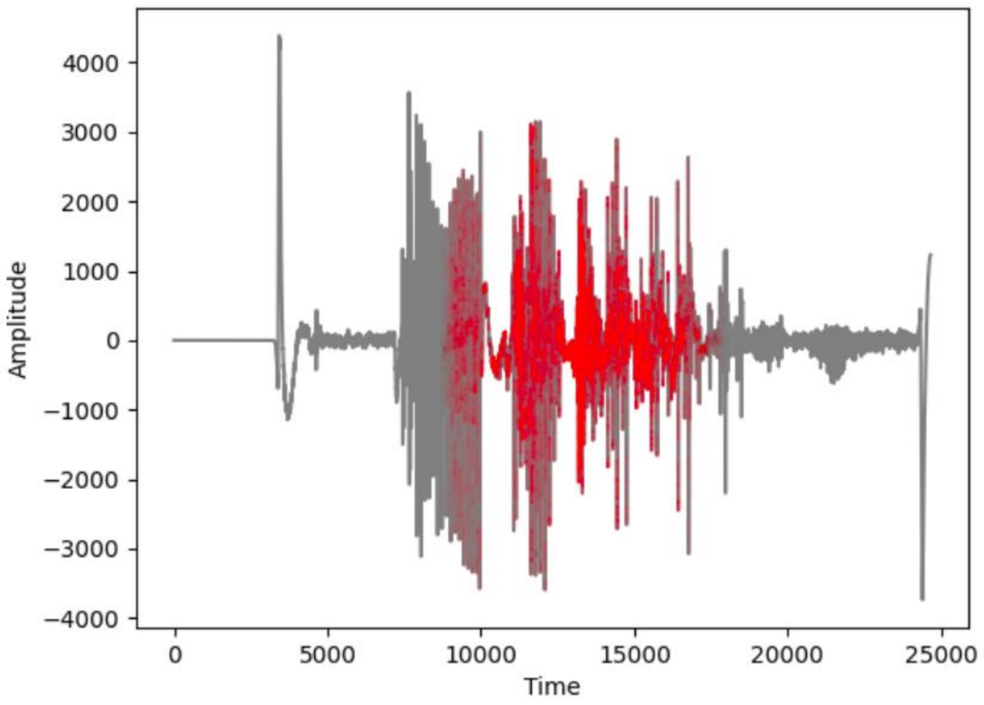
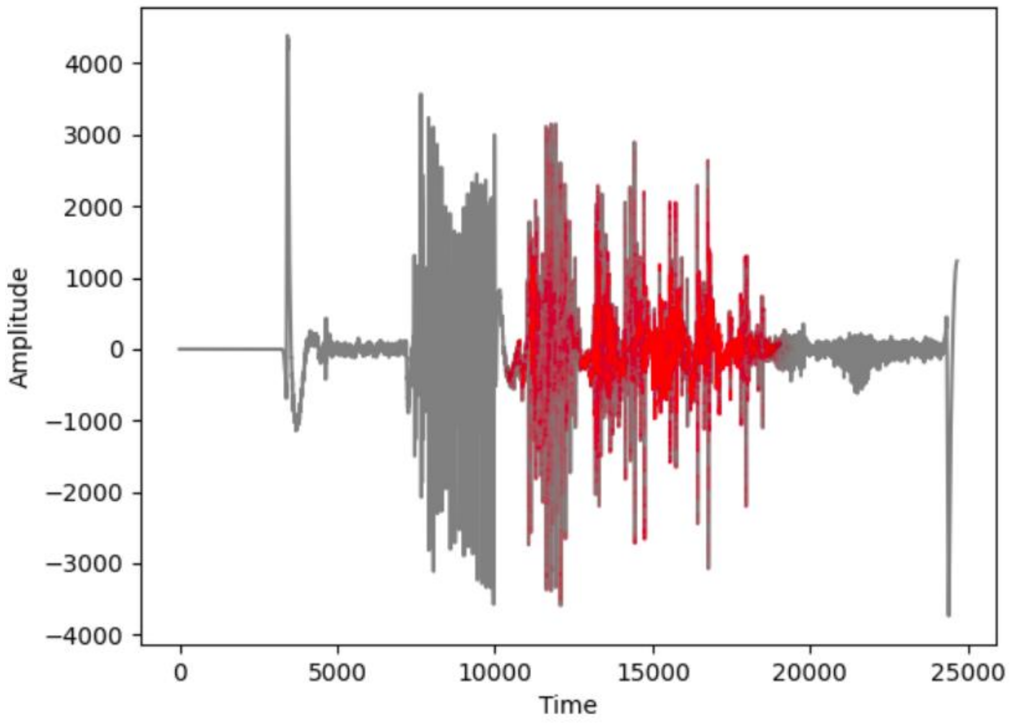
MINIMUM CLIMB **SPEED**	

Figure 155 - Calculated relevance through the STT algorithm of the recording described in Figure 154

The previous analysis was performed on a recording that the system was processed correctly. Another interesting approach is to consider recordings that are partially or completely misunderstood in order to verify whether or not their relevance presents different characteristics. Figure 156 presents the results of such a recording.

sound	**Sentence is "*i report when ready*" but is translate "*and report when ready*"**
I report when ready	

I report when ready	
I report when ready	

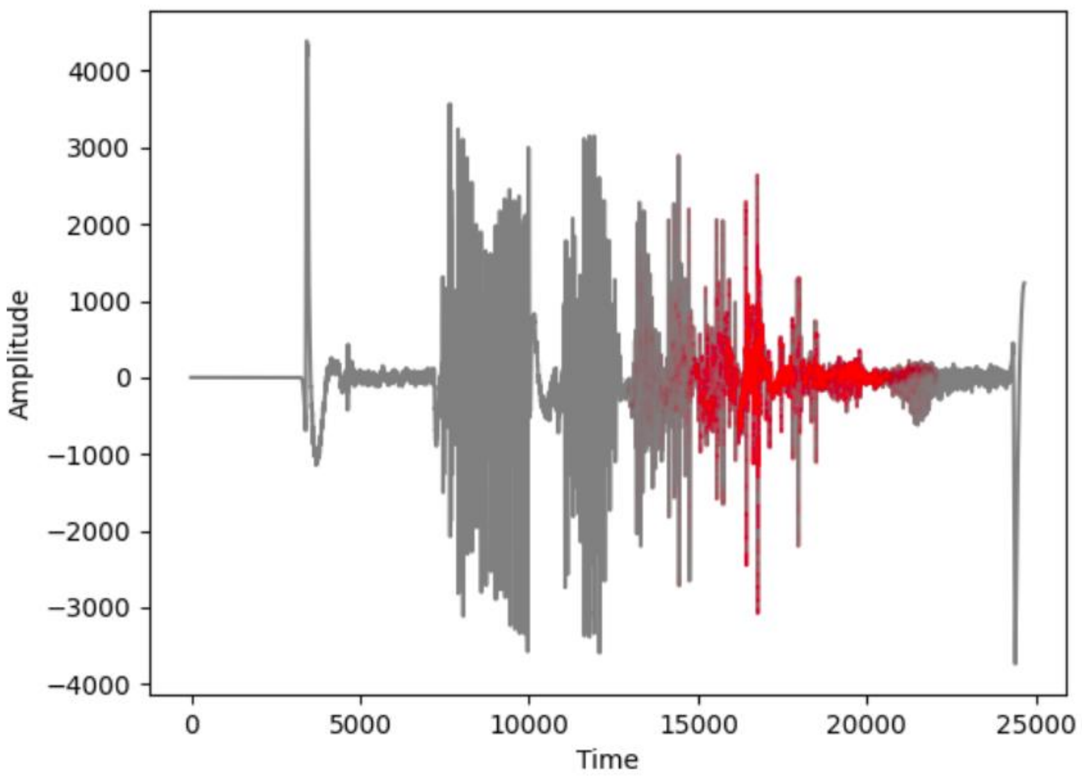
I report when ready	

Figure 156 – Evaluation on a different recording whose transcript is incorrect. In red is the relevance of each word. The word "I" translated into "And" is using a larger portion of the input space than anticipated.

Another approach to the LM12 regarding this use case is to consider how much noise a given word continues to appear in the transcript. This would correspond to some measure of the maximum stability of the transcript given a perturbation in the input. Figure 157 illustrates the analysis of the initial recording considered. The first column shows how much noise intensity is considered on the whole signal; the second one is regarding the relevance of the first word ("Thank"), and the last column is about the second word ("you"). Relevance indicates whether or not the output uses its inputs differently than before. The goal was to measure how much the system handles perturbation before changing its output. However, this point was not reached due to hardware constraints, so only stable behaviour was measured in the study. Here, the perturbation was introduced on the whole audio, as was done for the first result in the LM12' study of STT. The goal is here to see how much it is possible to prove its appearance for every considered input included in the level of noise introduced.

Noise intensity	THANK YOU	THANK YOU

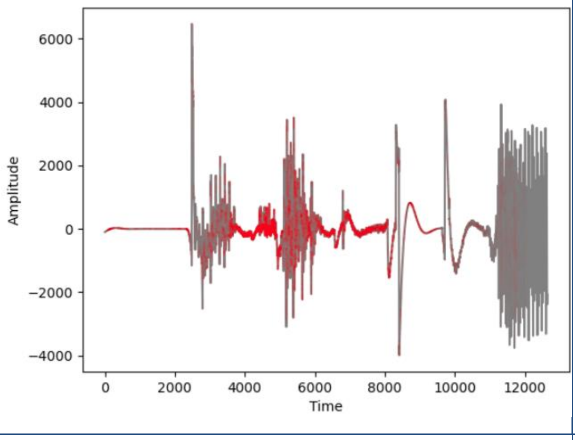
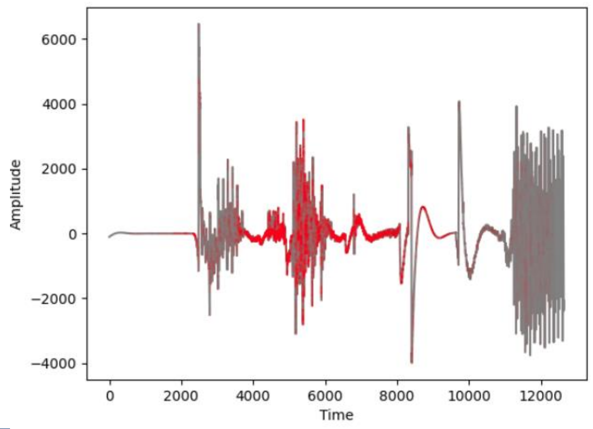
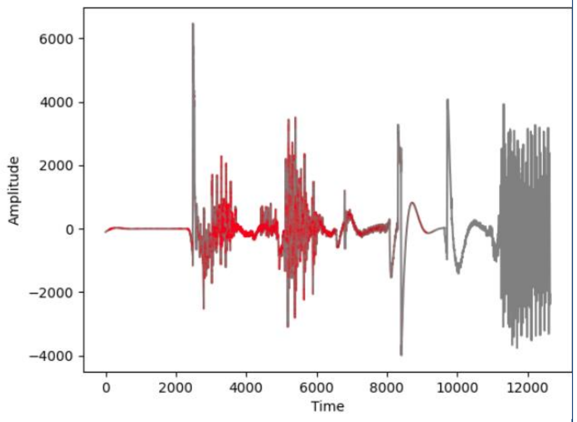
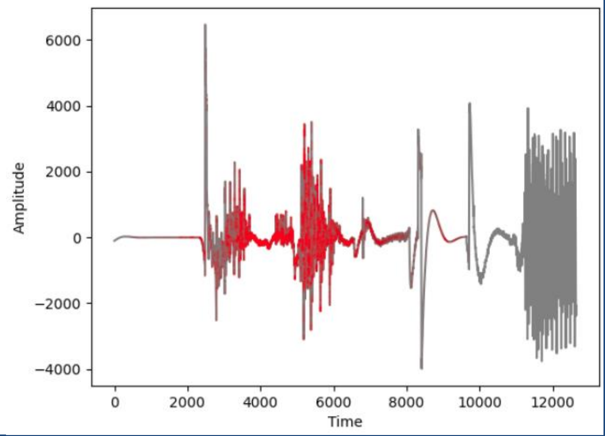
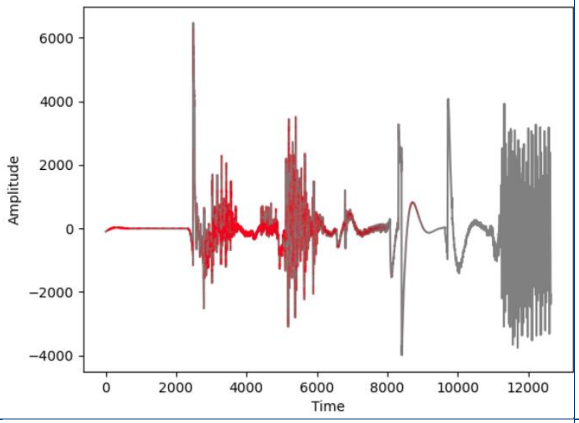
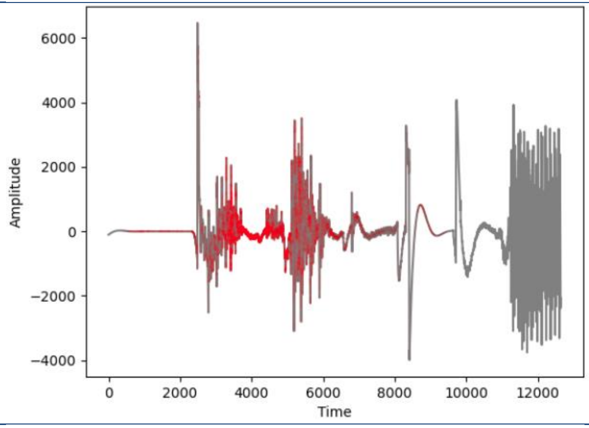
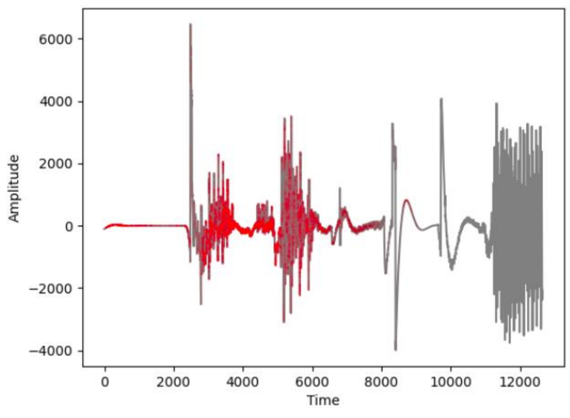
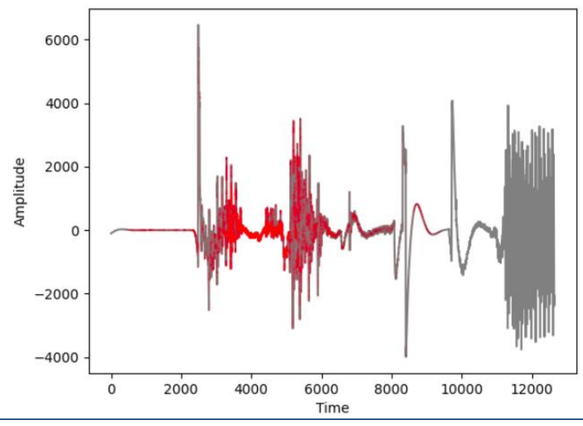
10^{-5}		
10^{-4}		
5×10^{-3}		
8×10^{-3}		

Figure 157 – Evaluation of the stability of a transcript. The relevance metrics present in this case good stability. In red is the relevance of each word at various noise intensity

<u>LM13: Trained model robustness</u>

To evaluate the robustness of the KALDI and Wav2Vec model, the study relies on a perturbation that can occur on the ODD of the AI system. Specifically, a perturbation regarding the speed rate has been introduced. The intensity of this perturbation is again defined on a scale from 0 to 9, with a reduction of the length output signal defined as $1 - 0.05 \times noise\ intensity$. Figure 158 illustrates what a perturbed signal (orange) can look like compared to the original signal (blue).

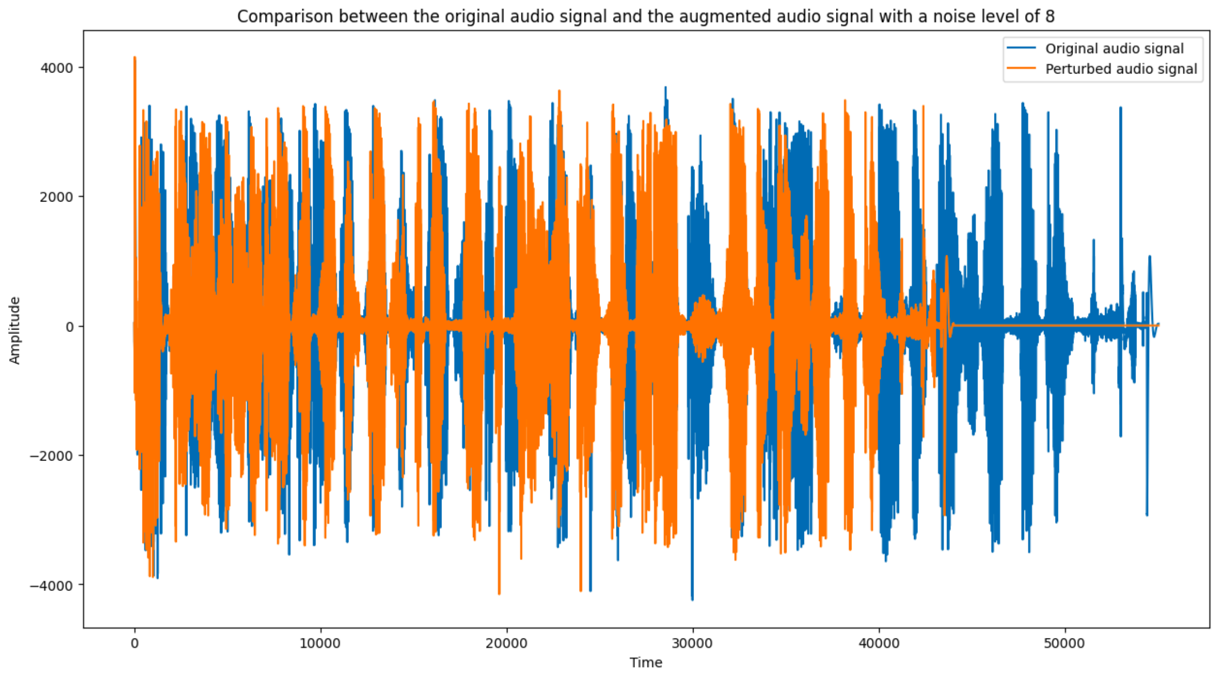


Figure 158 – Example of a perturbated recording under the speed perturbation (orange) from the original recording (blue).

Given this perturbation model, both models (KALDI and Wav2Vec) have been tested to verify their ability to maintain their WER metric against it. Figure 159 illustrates the average WER score for various noise intensities on the 3595 recordings extracted from the test data set. Both models tend to have relatively good performance until a level of noise intensity of 5 after which the KALDI model has faster degrading performance than the Wav2Vec. But both are having trouble coping with this growing intensity of noise.

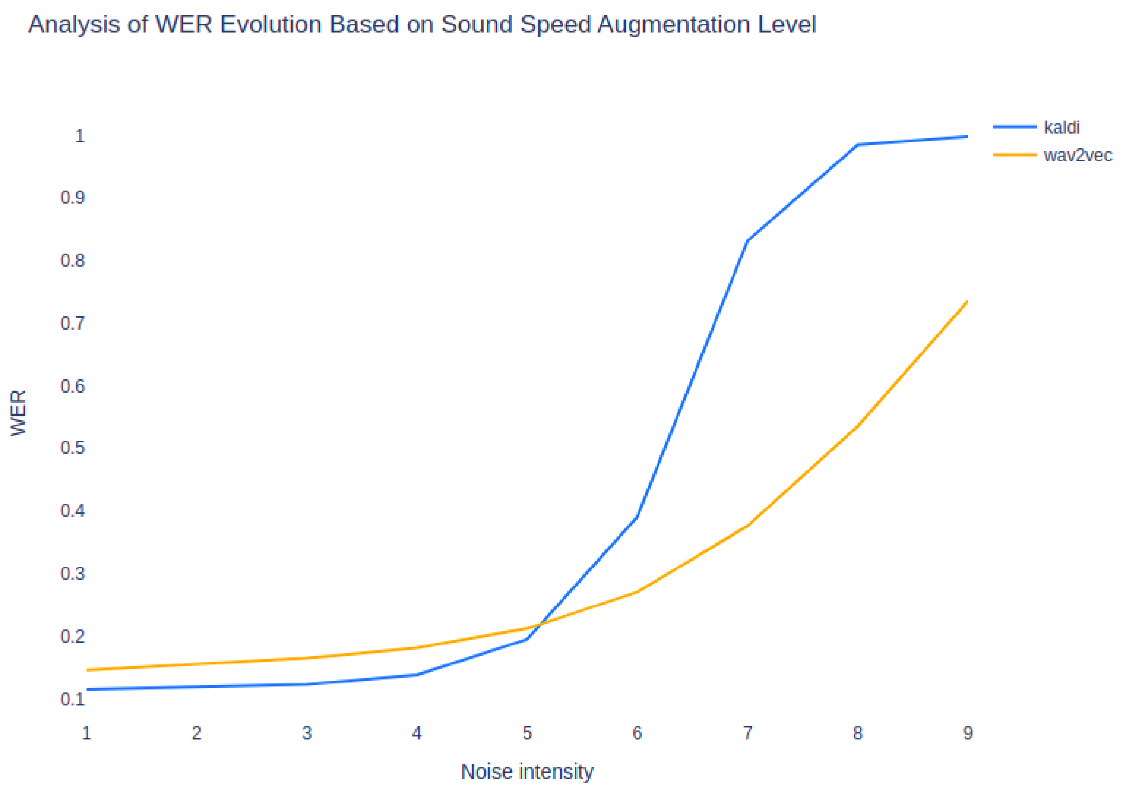


Figure 159 – Evolution of the WER metric for the KALDI and Wav2Vec model under the speed perturbation.

This evaluation against a specific noise, such as the speed rate, is somehow insufficient to evaluate the model's robustness. Given the nature of the use case, more particular perturbations should be considered to explore the ODD more thoroughly, especially under more “subjective” perturbations, such as an accentuation of the pronunciation. This type of noise poses a particular challenge since new samples cannot just be generated from existing ones. They must acquire more data points from external databases that can also be biased in their own way. Therefore, a more empirical approach would be required to evaluate against it. This type of validation, however, is subjected to their limitation, such as the fact that they will be subjective and would not necessarily bear any strong generalisation properties over the ODD.

The evaluation against accent variation has also been performed to verify the robustness of the model. Three different accents were considered: German, Swiss German, and Swiss French. First, an analysis of the robustness of the model (through the WER metric) was done using a sampling of 200 audio files from the 2000 available. For each data set, we measure the WER metric of the KALDI model. Figure 160 shows the WER results on the model. The distribution of WER values for each accent shows that the most represented value is 1, suggesting that the model tends to produce erroneous transcriptions in many cases. This indicates that the model is not performing well on the ATCOSIM data, as a WER of 1 means a utterly erroneous transcription. Looking at the average WER for each accent, we observe the following values:

- German accent: 0.9089
- Swiss-German accent: 0.8779
- For Swiss-French: 0.9019

These relatively high averages indicate poor model performance for all accents examined, with an average WER close to 1, confirming the initial observation that the model fails to transcribe ATCOSIM data correctly.

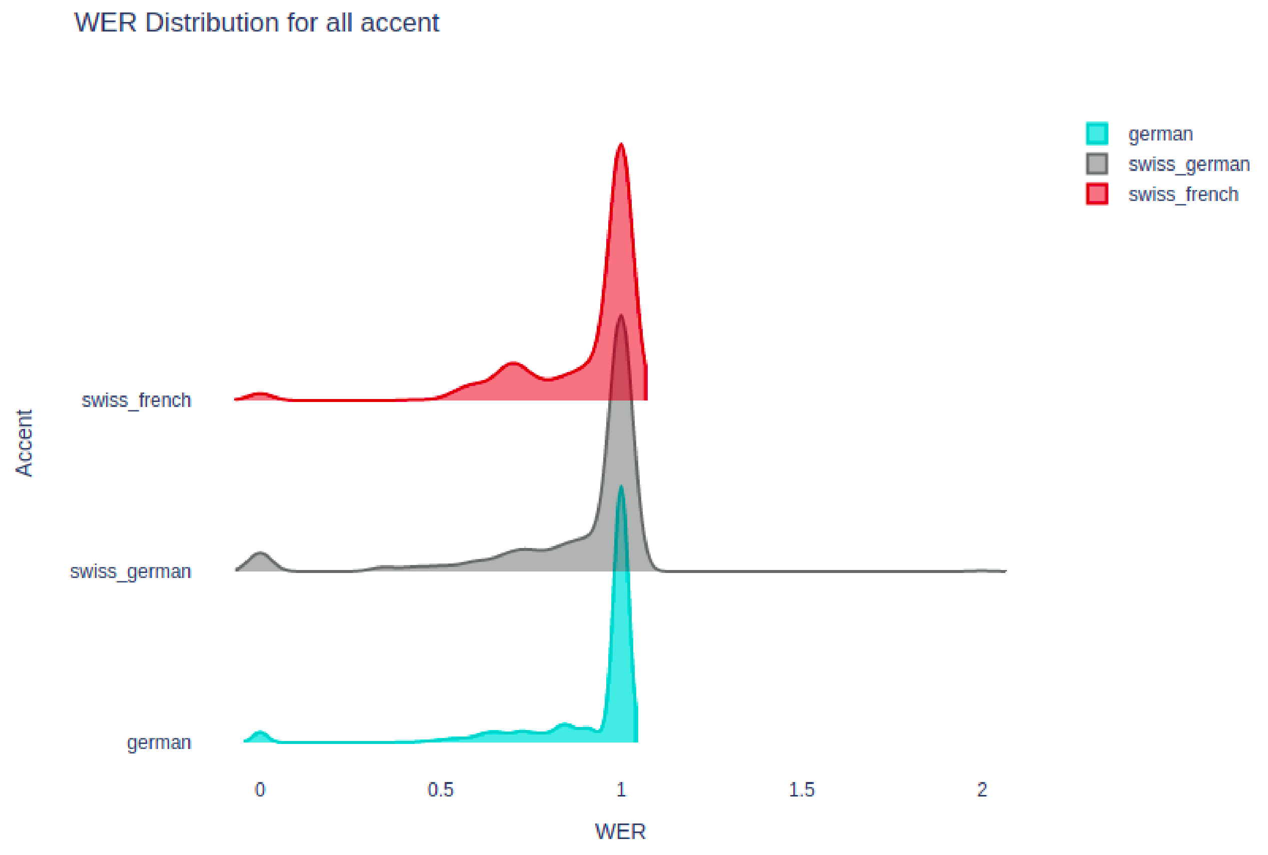


Figure 160 - Evolution of the WER metric for the KALDI model under the speed perturbation.

The three corresponding data sets were evaluated with speed rate perturbation to consider the corner case between accent and speed rate. As in the previous protocol, the three data sets were tested by increasing speed rate perturbation from 1 to 9. At first glance, the Swiss-German accent seems less robust to noise, as the WER increases rapidly, especially compared to the other two curves. However, an important observation is that at noise level 6, the curve remains constant at 0.96, indicating a specific stabilisation despite the increase in noise. On the other hand, the curve for the German accent, although initially having the worst score, seems less sensitive to noise up to level 4. Beyond this level, an apparent increase in WER to 0.97 indicates a more gradual deterioration. The result for the Swiss-French accent shows an increase from the first noise level, reaching a WER value of 1. This shows a high sensitivity to noise, with very degraded performance even at relatively low noise levels.
It's also important to note that the initial WER values for each accent were already quite high, reaching 0.8 at most, suggesting that the models' original performance was already not optimal before the addition of noise.

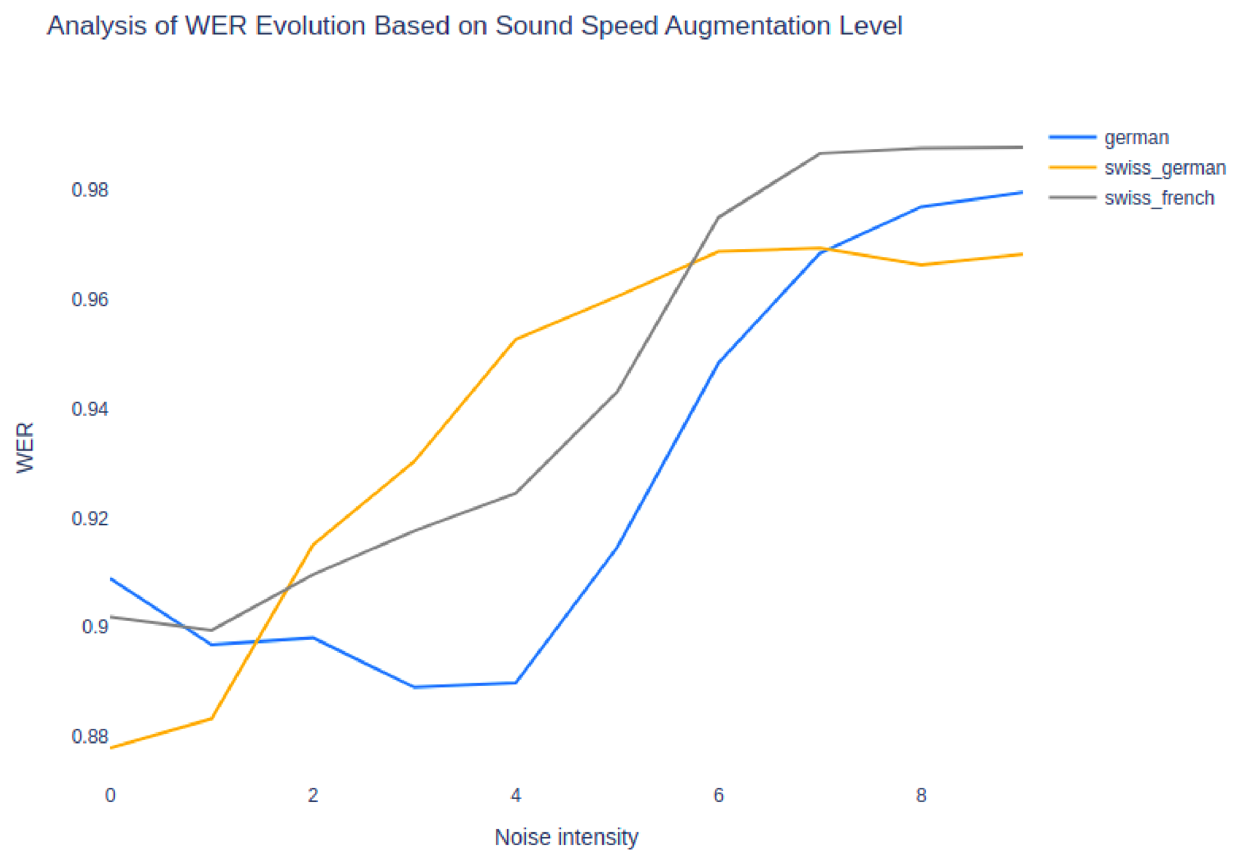


Figure 161 - Evolution of the WER metric for the KALDI model under the speed perturbation for three different accents.

<u>Conclusion on the STT model</u>

The analysis performed on the STT use case (using KALDI and Wav2vec architectures) showed that:

- LM11 can be almost impossible without a powerful infrastructure to verify this requirement.;
- LM12 can be performed successfully using a statistical approach, but ground-breaking results also allow in the future use of formal methods to point out defects and vulnerabilities that can be addressed during design;
- LM13 has allowed some comparison between the models considered, one more robust to a specific noise. However, due to the nature of the use case, a more empirical approach would be required to explore the ODD on attributes that are not easily quantifiable nor easy to sample from a validation database.

6.3.2.3 **ACAS-Xu use case**

<u>LM11: Training algorithm stability</u>

The ACAS-Xu use case is a set of pre-trained models made available to verify their reachability property (see Section 6.2.2.1.3), tied to the LM12 requirement. Their initial training was not available to test the LM11 requirement on it.

LM12: Trained model stability

In the case of ACAS-Xu, the primary safety requirement is related to the safety of the system's reachability analysis. What is expected of the ACAS-Xu is that given initial conditions between the two aircraft, the safety property of separation of both aircraft is always validated for a given number of steps of applying the ACAS-Xu AI components. A piece-wise linear algorithm is applied at each step to decide between 5 possible actions. The linearity of the neural networks used (Dense layer with ReLU activation function) makes the LM12 requirement almost directly satisfied since the response of each neural network is linear. However, the linearity alone is not enough to enforce the reachability analysis. Here, formal verification techniques can be used with significant advantages to validate the ODD piece by piece. In this study, we rely on the Nnenum tool (Bak, 2021), which is currently at the state of the art. The methodology relies on a geometric path enumeration method, a formal verification technique for neural networks using ReLU activation functions. The process generates paths through the network using point sets rather than individual data points. Data point sets are propagated through each layer using geometric operations such as intersection and union. Paths are then checked to satisfy the safety property. If all paths satisfy these properties, the model is considered safe; otherwise, it is unsafe. One limitation of the tool is that when it determines that the input space is unsafe (by finding one or more counter-examples), it does not provide the segmentation of the input space to distinguish between safe and hazardous areas. This would have been useful to identify how much input space is not proven safe to set in place at the system level, such as some failsafe or fallback mechanism complementing the ACAS-Xu algorithm (e.g., other ML components, other algorithms). In most case tests using the high-level property, the ACAS-Xu is proven safe.
This exact method provides a mathematical guarantee of the model's safety or safety. This method requires, however, that the dimensionality of the space to be explored is low and that the layers are linear. Thus, it is fully applicable in this use case but hardly in others that would not satisfy these requirements. The method automatically segments the given input space chosen by the user. If one piece of the input space is not proven safe, then a counter-example will be computed. The input space to be tested is sliced into several pieces, usually called properties (see Figure 162), which correspond to a specific range of the attributes of the ODD. For each property, the tool can either prove the safety reachability result or produce a counter-example (see Figure 163 and Table 67).

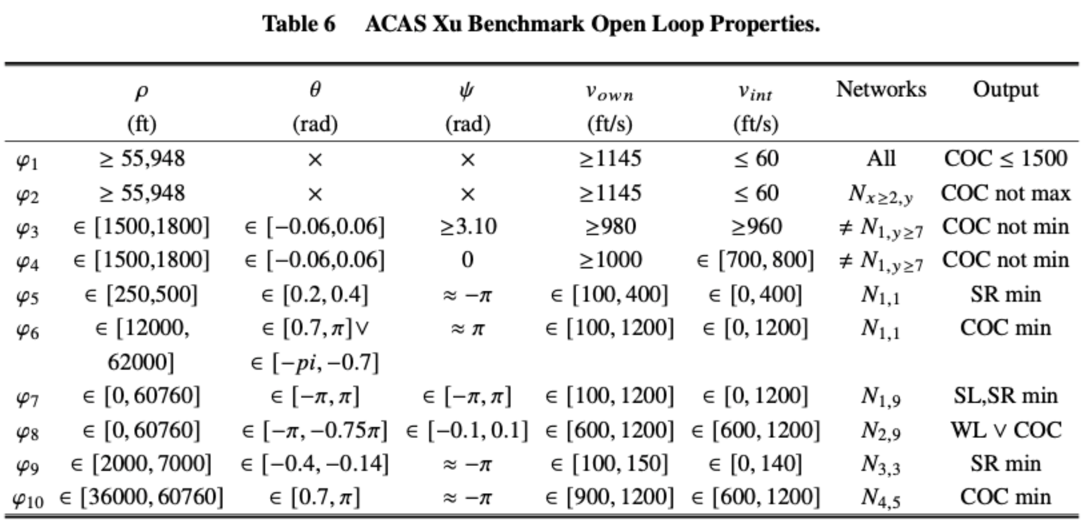
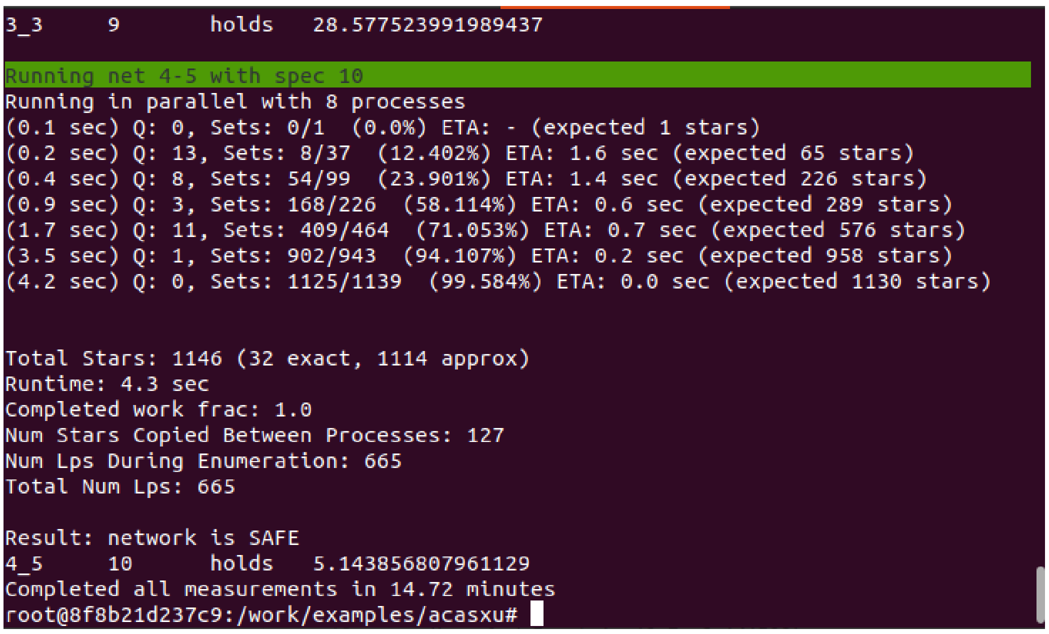


Table 6 ACAS Xu Benchmark Open Loop Properties.

	ρ (ft)	θ (rad)	ψ (rad)	v_{own} (ft/s)	v_{int} (ft/s)	Networks	Output
φ_1	$\geq 55{,}948$	$\times$	$\times$	≥ 1145	≤ 60	All	COC ≤ 1500
φ_2	$\geq 55{,}948$	$\times$	$\times$	≥ 1145	≤ 60	$N_{x\geq 2,y}$	COC not max
φ_3	$\in [1500{,}1800]$	$\in [-0.06{,}0.06]$	≥ 3.10	≥ 980	≥ 960	$\neq N_{1,y\geq 7}$	COC not min
φ_4	$\in [1500{,}1800]$	$\in [-0.06{,}0.06]$	0	≥ 1000	$\in [700, 800]$	$\neq N_{1,y\geq 7}$	COC not min
φ_5	$\in [250{,}500]$	$\in [0.2, 0.4]$	$\approx -\pi$	$\in [100, 400]$	$\in [0, 400]$	$N_{1,1}$	SR min
φ_6	$\in [12000, 62000]$	$\in [0.7, \pi] \vee \in [-pi, -0.7]$	$\approx \pi$	$\in [100, 1200]$	$\in [0, 1200]$	$N_{1,1}$	COC min
φ_7	$\in [0, 60760]$	$\in [-\pi, \pi]$	$\in [-\pi, \pi]$	$\in [100, 1200]$	$\in [0, 1200]$	$N_{1,9}$	SL,SR min
φ_8	$\in [0, 60760]$	$\in [-\pi, -0.75\pi]$	$\in [-0.1, 0.1]$	$\in [600, 1200]$	$\in [600, 1200]$	$N_{2,9}$	WL $\vee$ COC
φ_9	$\in [2000, 7000]$	$\in [-0.4, -0.14]$	$\approx -\pi$	$\in [100, 150]$	$\in [0, 140]$	$N_{3,3}$	SR min
φ_{10}	$\in [36000, 60760]$	$\in [0.7, \pi]$	$\approx -\pi$	$\in [900, 1200]$	$\in [600, 1200]$	$N_{4,5}$	COC min

Figure 162 – Properties to be demonstrate on the ACAS-Xu use case, extracted from (Manzanas Lopez et al., 2023)



```
3_3     9       holds   28.577523991989437

Running net 4-5 with spec 10
Running in parallel with 8 processes
(0.1 sec) Q: 0, Sets: 0/1  (0.0%) ETA: - (expected 1 stars)
(0.2 sec) Q: 13, Sets: 8/37  (12.402%) ETA: 1.6 sec (expected 65 stars)
(0.4 sec) Q: 8, Sets: 54/99  (23.901%) ETA: 1.4 sec (expected 226 stars)
(0.9 sec) Q: 3, Sets: 168/226  (58.114%) ETA: 0.6 sec (expected 289 stars)
(1.7 sec) Q: 11, Sets: 409/464  (71.053%) ETA: 0.7 sec (expected 576 stars)
(3.5 sec) Q: 1, Sets: 902/943  (94.107%) ETA: 0.2 sec (expected 958 stars)
(4.2 sec) Q: 0, Sets: 1125/1139  (99.584%) ETA: 0.0 sec (expected 1130 stars)


Total Stars: 1146 (32 exact, 1114 approx)
Runtime: 4.3 sec
Completed work frac: 1.0
Num Stars Copied Between Processes: 127
Num Lps During Enumeration: 665
Total Num Lps: 665

Result: network is SAFE
4_5     10      holds   5.143856807961129
Completed all measurements in 14.72 minutes
root@8f8b21d237c9:/work/examples/acasxu#
```

Figure 163 – Example of the use of Nnenum on part of the input space of ACAS-Xu

Property	Verification
1	All related models are proven safe
2	35 counter-examples found. One is for example: {"ACASXU_run2a_2_1_batch_2000.onnx": {"input":[0.6493751406669617, 0.01211466919630766, -0.09158006310462952, 0.44999998807907104, -0.47905993461608887], "output":[0.026150472462177277, -0.022936437278985977, 0.019640766084194183, -0.01705879345536232, 0.020409993827342987]},}
3	All related models are proven safe
4	All related models are proven safe

5	Related model is safe
6	Related model is safe
7	Related model is unsafe, counter-example found: [-0.328422874212265, -0.49999991059303284, -0.4887717664241791, 0.30708637833595276, -0.4506964385509491]
8	Related model is unsafe, counter-example found: [-0.2729834318161011, -0.38378289341926575, -0.008519993163645267, 0.06708698719739914, 0.4784213602542877]
9	Related model is safe
10	Related model is safe

Table 67 – Example of an output of Nnenum on the different properties defined

LM13: Trained model robustness

The ODD of the ACAS-Xu presented in Section 3.5 somewhat differs from any of the use cases in Section 6.2. Contrary to other use cases, the ODD of this use case is fully defined and perfectly bounded. Its dimensionality is sufficiently small to be covered in its entirety. In this ODD the edge and corner cases are as any other cases that can be tested. Implementing any particular specific perturbation to explore rare instances is unnecessary. If the ODD is already proven safe by the LM12 requirement, it is also true for the edge and corner cases.

Conclusion on the ACAS-Xu model

The ACAS-Xu constitutes a specific use case that presents little applicability of the different requirements described in LM11, LM12 and LM13 of the EASA CP. Its main contribution to the discussion for the MLEAP project is that the reachability property this use case proposes to verify can be successfully handled using formal methods.

6.4 Conclusion

Assessing the robustness and stability of machine learning is a complex task that deals with:

- the definition of the operational domain on which the system is going to operate;
- the definition of the requirements that are needed from the system;
- the choice and the setup of a methodology to assess these properties over the ODD;
- the practical realisation of such a methodology in a tractable way.

As several methods are available, the document regroups them into 3 categories: statistical, formal, or empirical. Several metrics and techniques are available in each with varying maturity in the existing tools. These categories can fit within both ISO/IEC and EASA documents on the topic. They can also be used to implement some of the requirements described in those standardisation reference documents.

Statistical methods might be the most straightforward way to analyse these properties. However, they require lots of preparation work to set up the right data sets. Also, any attempt to sample exhaustively is immediately limited by the high dimensionality of the input space. In terms of tools, many libraries provide the necessary functions to evaluate statistical metrics. However, few tackle the data issues associated with such methodologies. Statistics help assess the LM11 requirement and explore edge or corner cases. They offer a first look at stability and robustness that formal or empirical methods can then complement. Considering availability, the tools necessary to implement statistical methods are widely available in the open-source community.
Formal methods, while promising in overcoming this limitation, suffer partly from some scalability issues that few tools can overcome. The available tools can vary in terms of maturity. Most are still academic tools, but a few industrial solutions are starting to appear. In some cases, with limited dimensionality, highly specialised tools tailored especially for the task demonstrate high performance (such was the case for nanmu). In general, having a unique tool for every use case requires an industry-grade solution (such as the tool Saimple). These methods offer more substantial properties in robustness and stability; however, they are often limited in what they can prove over the input space. The evaluation showed that it is possible to obtain meaningful stability and robustness properties if those can be specified precisely and do not affect every input dimension simultaneously. Several benefits can be obtained by formally validating a machine learning system (more detailed in Section 7.2.3) to improve stability, robustness and the quality of the training data.
Finally, empirical methods might be considered the most practical because they need to have the system up and running to be evaluated. However, these approaches can only provide a black-box understanding of the system properties. Contrary to statistical or formal approaches, they will not allow assessment to the same level of confidence as the properties required on the system. Empirical methods rely on a concrete assessment evaluated by some human experts, which tends to reach a consensus regarding methodology and its evaluation. However, it can also propose benchmarking using standard and widely spread data sets as reference points. Their use might be considered for applications of low-level criticality depending on the objective the system has to meet.

Overall, combining techniques would benefit any meaningful evaluation of robustness and stability. This way, the process would cover as much input space as possible while maintaining tractability. Given the objective of the EASA CP and the results obtained, it is possible to recommend that:

- LM11 (which concerns the stability of the training algorithm) could be handled using statistical methods to slice the training set in several ways. However, it should be noted that LM11 can suffer from severe limitations in practice due to the cost of each training session and the number of training sessions required.
- LM12 (which concerns the stability of the trained algorithm) could be handled using formal methods since they allow measuring stability over the nominal part of the ODD. Formal methods have proven useful in cases where dimensionality is small (to prove the complete ODD) and large (to point out specific vulnerabilities).
- LM13 (which concerns the robustness of the trained algorithm to more adverse inputs like edge or corner cases) could be handled by statistical methods to explore the border of the ODD or even empirical methods to construct a particularly dangerous scenario that can only be thought of by an expert.

7. Towards application agnostic development pipeline implementing the W-shaped learning assurance

7.1 Introduction

This chapter aims to provide an end-to-end development pipeline that implements the different phases of the W-shaped process (cf. section 1.4). While giving a set of recommendations about the primary verifications and analysis that need to be completed at each stage of the learning assurance, the intention is to draw on all the experimental and theoretical studies carried out so far, to point out the steps and checklist that should not be missed when developing AI components for critical aviation systems, to meet the target objectives. Focusing on specific characteristics of the data, ML/DL models and performances at each stage of the ML development pipeline, different methods have been discussed in the previous chapters that can be used to ensure:
(1) the data sets quality in terms of completeness and representativeness, w.r.t the defined ODD of the task being addressed;
(2) the generalisation assessment, evaluation, and improvement;
(3) the robustness and stability of performances of the trained models.

These features are part of a high-quality and secure development workflow through different steps in the AI-learning assurance (cf. section 1.4) process developed by EASA. In the previous chapters, we reviewed a set of methods enabling us to meet the three main objectives (1), (2) and (3) through analyses of the methods selected for each of the tasks. Hence, one of the most important observations is that the quality and volume of the training data significantly impact the models developed, the choice of algorithms, the architecture and configuration, and the selection of target performance evaluation metrics. A range of data and model performance evaluation methods have been selected, and the first analytical results for the selected use cases have been summarised in the previous chapters. So far, the initial conclusions and investigations into model behaviour have enabled the completion of a generic ML/DL development pipeline (see section 7.3), helping to meet the various requirements and ConOps defined in the EASA documents. In addition, to meet the verification objectives for the development assurance process, it is essential to synchronise the steps carried out in the pipeline and the backtracking (if necessary for verification).
In this chapter, each task in the MLEAP project is mapped to its corresponding stage(s) in the W-shaped development process. The main issues related to the development of ML/DL components are then summed up, and possible strategies for tackling them are discussed. Next, the generic AI development pipeline is presented. Independently of the use case, it provides a detailed approach to the various steps to carry out the different tasks in the W-process. The objective is to revisit the AI development pipeline and implement the learning assurance process, along with a set of requirements verifications proposed to meet the goals of the target application.

7.2 Protocol/Checklist to Avoid Common Issues

In the previous chapters, various common practices have been discussed regarding data preparation, model architecture selection, choice of evaluation metrics, setting the learning objectives, and the performance verifications to be made. This ensures that the ML/DL delivered model meets the application requirements. However, at some point, some of these practices are still not conducive to achieving the objective and may even be detrimental to it. This section highlights the main issues in data analysis, model development and training, and performance evaluation to be addressed to meet the targeted objectives. In the following, the experimental results and subsequent analyses and discussions of the previous chapters will be synthesised to extract empirical good practices when defining an end-to-end AI development pipeline. These elements will then be leveraged to propose a generic development pipeline in the next section to complete the different tasks of the whole development pipeline. In Table 68, a list of identified issues, along with the corresponding W-shaped learning assurance stage, is provided with the identified impacts of these pitfalls and how one can remedy them. It represents a synthesis of limitations identified and verified during the experimental analysis of the project.

Phase W-Process	Pipeline Phase	Issue/Limitation	Impact on	To be checked / recommendation
Data management	(1): Data management & Qualification	Inappropriate data in training/testing (lack of coverage, noise, domain shift ...)	Learned elements by the model: The trained model will not have enough coverage, hence impacting generalisation. This latter will not be considered when the training data does not cover the targeted ODD for use.	Specifications of Data Quality Requirement in the ODD. Use the assessment method to document how these specifications could be enforced during data collection and preparation.
Data management	(1): a priori Evaluation	Inadequate data processing to the target application requirements in terms of input.	Model training and learning management. Even if the data contains all the required classes and information to represent the targeted domain, the processing of the data (representations construction, noise management, balancing ...) has a big impact on the information to be fed to the models.	Simplify data processing to ensure it does not introduce additional complexity.
Data management	(1): a priori Evaluation	Ignoring outliers	Generalisation abilities and dealing with OOD data samples. Indeed, specific cases are important in ODD definition. When outliers are not well defined and managed, the model will not be able to derive patterns from the data correctly, hence it will underfit. Besides, OOD data samples could also impact the performances.	Outliers' handling: w.r.t ODD must be identified upstream and appropriate dispositions regarding handling must be demonstrated (i.e., justifications regarding the selected types and number of associated samples)

Phase W-Process	Pipeline Phase	Issue/Limitation	Impact on	To be checked / recommendation
Data management	(1): Data management & Qualification	Inappropriate pre-processing policy	Generalisation and model convergence	Pre-processing should be part of a data preparation plan that aligns with DQR and model specifications.
Data Management	(1): Data management & Qualification (1): a priori Evaluation	violation of data related hypothesis: training-validation independence assumption and the IID	Performance generalisation to the initially defined ODD: due to weak data preparation (Inaccurate data oversampling, data augmentation before splitting into training, validation, and test sets, or performing feature selection before splitting data, are all procedural errors that may lead to the violation of the train-test data sets independence assumption...)	In addition to verifying the data distributions (IID in training and testing sets), use resampling (repeatedly selecting samples on which the model is trained to ensure that it will perform correctly in different samples) and introduce the validation dataset to tune the model's hyperparameters.
Learning Process Management & Model Training	(3): Development (training – a posteriori evaluation)	Overfitting	Model Performances: (big model, small data, high model variance) model highly dependent on the raining data and context: memorization of data points, noise included, instead of signals for mapping	Train with more data (add real data when it's possible, or use augmentation), introduce intentional noise in the training to make the model able to learn over noisy data (which is more frequent in the real life),
Learning Process Management & Model Training	(3): Development (training – a posteriori evaluation)	Underfitting	Model Performances: The model is too simple compared to the provided data for training and has a high bias due to the inability to capture the relationship between the input examples and the target values.	This can be managed with the regularization, which implies learning a more complex model. Reduce the risk of overfitting by applying a penalty to some parameters (e.g., dropout, l1-regulariser) which introduces more complexity. We can also manage the early stopping to let the model learn more by increasing its training duration. Finally, make sure that the noise is removed from data.
Learning process management	(2): a priori evaluation (2): Model Design	Misuse/understanding of generalisation bounds	Model development and training configuration. The generalisation bounds represent an estimating tool, but they are not always usable.	Uncertainty quantification can be used. Some bounds are not applicable to all use cases, or their results must be interpreted specifically.
Learning Process Management & Model Training	(3): Model Design & Adaptation	Inappropriate training objective	performance of the model (generalisation & robustness) in addition to its trainability	Adaptation and customisation of the loss function and the training objective is crucial to converge to the accurate performance level

Phase W-Process	Pipeline Phase	Issue/Limitation	Impact on	To be checked / recommendation
			(convergence to the accurate loss minimum)	and meet the target objective of the application. To do so: start by identifying the problem (i.e. classification or regression, how many classes, what do we need to maximise/minimise...), then depending on the model type (neural networks, decision trees ...) and data distribution (balance) choose a performance metric evaluation to be optimised w.r.t the industrial KPI (i.e. accuracy, precision over recall, ...) the loss function should be defined accordingly to penalise the wrong predictions, maximise/minimise a distance between classes, balance the weights between features ...
Learning process management	(2): a priori evaluation (3): Model Design & Adaptation	Inappropriate model capacity vs task complexity	model training and learning management: this can result in an underfitting model (if the model capacity is too essential for the task w.r.t the number of classes, data volume ...), or even in an underfitting model if its capacity is not enough to address the task complexity (take into account all possible cases)	The existing tools for model capacity computation can help driving the choice and setting of the correct model capacity to bear the task complexity (e.g., in terms of number of classes to be learned at once, or the precision of the probability/score to be computed by the model). However, a balance between the model expressive capacity (VC-dim, Rademacher complexity...) and the practical complexity (depth, parameters...) need to be balanced to avoid the High-Capacity Low-Reality Phenomenon, which refers to the enormous gap between the practical effective complexity of a deep learning model and its expressive capacity, hence, the shallow (less complex models) can achieve comparable performance but cannot be trained on the same data set with the same complexity (they need a simplified problem)
Model training	(3): Model Design & Adaptation (3): Development (training – a posteriori evaluation)	Bad global minima	Performances: bad model generalisation. The convergence will not lead to good global minima.	Regularisation methods and optimisers help the training algorithm converge to better values.

Phase W-Process	Pipeline Phase	Issue/Limitation	Impact on	To be checked / recommendation
Learning process management	(3): Model Design & Adaptation (3): Development (Model training)	Misuse of the model-driven regularisers	Model training and learning management: Regularisation is needed when the model is complex enough to start modelling the noise in the training data. However, when this is misused later, it can lead to performance deterioration rather than improvement.	One good practice is to go from more straightforward to complex in terms of functions for the training optimization and regularization. The regulariser aims to drive the model learning and avoid overfitting. Several regularisers can be used in the model development: . Kernel regulariser: Regulariser to apply a penalty on the layer's kernel . Bias regulariser: Regulariser to apply a penalty on the layer's bias . Activity regulariser: Regulariser to apply a penalty on the layer's output These regularisers help the balancing of bias and variance of the model. To better choose the regulariser at the model architecture level, we will need to study the impact of each of them on the learning development and also understand the exercised changes on the model function.
Learning Process Management & Model Training	(3): Model Design & Adaptation	Non adapted optimization algorithm to the model complexity	Learning management. Since the optimizer aims to update model parameters to reduce the error. When we plot the loss function, w.r.t the model parameters, there exist values of the parameters that correspond to the lowest error value. The objective is to help the model getting there using an adequate optimizer (SGD, ADAM, ADAGRAD, ...)	The learning rate will be different depending on the model complexity, so we rely on optimisers that adapt the learning rate (AdaGrad and Adam). However, the major issue is that the adaptive learning rate tends to get really small over time. Hence, SGD remains the best since it converges quickly but may not converge to the optimum minima.
Learning Management & Model Training	(3): Model Design & Adaptation	Adapting a large hypothesis space	model training & learning management	bound the learning space
Learning process verification	(3): Model Design & Adaptation (3): A posteriori Evaluation	Inappropriate evaluation metrics	performance evaluation and model adaptation	Not all evaluation metrics are appropriate for the industrial objective. Sometimes we need to customize the accuracy definition, and evaluation metrics to better represent the industrial targeted performances. For example, suppose the accuracy acceptance

Phase W-Process	Pipeline Phase	Issue/Limitation	Impact on	To be checked / recommendation
				is conditioned on some criterion presence (impact of a given class over the others, since the accuracy is the ratio of correct predictions to the total number of input samples.). Hence, an adapted definition of this one should be considered instead of relying simply on existing ones.
Learning process verification & Independent data and learning verification	(3): Development (a posteriori evaluation) (5): Implementation	Gap between the experimentation and industrial expectation (reality gap)	Performances & Acceptability of the model in the target application and system	Review the evaluation metrics and the training objectives. There is no typical rule to do it, however, analysing the representativeness of the training objective and the selected evaluation measures is more than important. Relying on common practices and state-of-the-art recommendations to choose the accurate measures is helpful, but the applicability of these functions w.r.t the KPIs and the correlation between the objectives is necessary (not all real objectives can simply be targeted by existing metrics). Also, a more accurate description of the ODD alongside the expected perturbation and the evaluation of the system's robustness against them can improve reducing the risk associated with the reality gap.
Independent data and learning verification	(3): a posterior evaluation (5): implementation	Relying only on testing of ML/DL components using testing scenarios	Coverage of the testing, generalisation guarantee, robustness and stability verification	Due to limitations of the testing scenarios, even if a massive testing is adopted to validate the model performance, this is not enough to guarantee the model performances. Hence, in case where the ML/DL model needs to be tested in the target application, a combination of the massive testing with model performances verification and errors analysis, after its validation and integration in the target system, can be recommended (a posterior evaluation of the model along with testing scenarios execution). Formal validation can also help avoid some limitations of massive testing since it can validate properties on a high-dimensional

Phase W-Process	Pipeline Phase	Issue/Limitation	Impact on	To be checked / recommendation
				space at once, whereas statistical testing would struggle to explore each dimension sufficiently.
Learning process verification	(3): a posterior evaluation	Insufficient stability against specific noise	Performance, robustness, stability	Check the maximum stability with different intensity of each custom perturbation listed as constitutive of the ODD.
Learning process verification	(3): a posterior evaluation	Learned bias	Performance, robustness, stability	Using relevance properties check the features learned by the ML component
Learning process verification	(3): a posterior evaluation	Insufficient stability	Performance, robustness, stability	Check the possible outcome using class separability and then adapt data set to improve separability (and stability as a result)
Learning process verification	(3): a posterior evaluation	Unstable training algorithm	Performance, robustness, stability	Check different strategy of slicing the training set.
Learning process verification	(3): a posterior evaluation	Some data can have incorrect annotations	Performance, robustness, stability	Check the stability on the training and validation set and detect outlier measurement of maximum stable space.

Table 68 Synthesis of the identified weak common issues in the different stages of the W-shaped learning assurance, along with their impact on the performances and/or design of the target application.

Table 68 synthesises the main elements to be checked to avoid the main design errors along the development pipeline, following the W-shaped learning assurance. This table is filled based on the ongoing experimental work, where the different findings complete the recommendations for all the stages of the W-shaped process covered by MLEAP and for which the generic pipeline (cf. section 7.3) aims to provide accurate answers.

7.2.1 Drive the data management

7.2.1.1 The ODD as a centrepiece of data qualification

As discussed in Chapter 4, data characterisation is a difficult task. In most cases, completeness or representativeness cannot be assessed regarding some real-life distribution of phenomena. Relying on the ODD as a reference point seems to be a more reasonable approach. As a consequence, the ODD becomes a central element of data quality.
The ODD identifies influence factors defining its limits, i.e., edge cases, as well as their interactions (i.e., possible corner cases). Such information can drive data requirements specification before the collection process. In addition, these requirements can be used as meta-data for better data

assessment, including for completeness and representativeness. It also allows a first assessment of available resources (e.g., public data sets) in order to estimate the collection efforts. Indeed, assembling a data set from scratch is significantly superior in terms of time, cost, and complexity. Starting from an existing dataset and applying enrichment strategies is far less costly, although not always possible or desirable.

Besides, the ODD strongly interacts with other specification documents, including the statement of intended levels of performance, especially for edge and corner cases. This information may complement data requirements by helping the estimation of the volumetry needed. Qualitative elements describing the use case may provide preliminary insight on the relevance of data enrichment strategies, i.e., data augmentation and/or synthesis.

7.2.1.2 **The model as a necessary feedback source**

Assembling a data set using a priori assessment methods for data quality may not suffice to reach the performance expectations covering the full ODD. Indeed, the model's learning behaviour is not entirely foreseeable, and designers should anticipate using it as feedback to evolve the data set.
Learning patterns may emerge at training time and provide empirical insight on the best course of action to solve dilemmas such as the trade-off between completeness and representativeness, i.e., strict representativeness may not enable correct learning of certain classes or edge and corner cases. Unanticipated influence factors may also arise, requiring specific adaptations, either through modifying the existing data set, or initiating a complementary collection effort.

7.2.1.3 **Data completeness and representativeness assessment - a visual synthesis**

Several assessment methods have been tested during the MLEAP project. Some of them are fit only for certain profiles of data. Most of them should not be used in isolation but rather combined, even though the potential synergies could not be explored during the course of the project. Nonetheless, it seemed worthwhile to provide a recapitulative tool that would allow the reader to get a reminder of when to use which method in a glance. Table 69 offers such outlook.

It is important to keep in mind that this tool is for convenience only, the list of methods is neither exhaustive nor prescriptive. The list could be arbitrarily extended. More importantly, on the long term, an indicator of the maturation of the data quality assessment domain would be to be able to build such table with properties (i.e., completeness, representativeness, and other specific data quality requirements) as well as influence factors as structuring elements of the rows and columns, rather than just the elements used hereafter (i.e., data profile and other broad element about the applicative context).

Method name	Method type	Assessment step		Data type					Input type		Dimensionality (> 50 ?)	
		A priori	A posteriori	Tabular	Time series	Text	Image Video	Speech Audio	Raw data	Embed ding	Low-dimensional	High-dimensional
Catania et al., Caiafa et al.	PCA	X		X		X		#	X	#		X
Asudeh et al.	MUP (Graph-based)	X		X					X		X	
Dourado et al.,	Entropy	X				X	X	?	X	?		X
Cleanlab	Integrated suite (confidence learning paradigm)	*(some features)	X	X		X	X (images)	X	X		X	X
BSA	Risk-based analysis	X		X	X	X	X	X	X		X	X
Liu et al.	Completeness ratios	X		X	X		*(meta-data)	*(meta-data)	X		X	
Mani et al.	Learned feature space analysis		X		?	X	X	X		X		X
Pei et al.	Neuron coverage		X	X	X	X	X	X		X	X(dimensionality depends on the size of the network)	X(dimensionality depends on the size of the network)
Cabitza et al., Mountrakis et al.	Sample similarity	X		X	?	#	#	#	X	#	X	#

Table 69 - Synthesis of the methods' applicability

Where:

- X indicates unconditional applicability
- * means partial applicability. For example, Cleanlab can be used as an a priori method on tabular data, as its outlier and near-duplicate detection features would then rest on an external KNN algorithm exploiting the raw data. For other data types, it exploits embeddings and is, therefore, an a posteriori method.
- # means conditional applicability. For instance, sample similarity could only be applied to text, speech, or image embeddings, not the raw data. Such embeddings would be high-dimensional in nature.
- ? means the method could apply, but feasibility could not be confirmed during the project.

7.2.2 Drive the model development and training

7.2.2.1 **Rely on data analysis outcomes**

In the complete ML/DL development pipeline, starting from the data analysis and task specification, we need to leverage the data management step outcomes to tackle better the ML/DM model's design and development. Knowing the rationales in terms of data type (e.g., images, textual data, audio…) and nature (e.g., time-based data, evolving data, static data...) is essential for a relevant model design. Based on the state-of-the-art analysis and the foreseen comparisons of the identified model architectures, the accurate one will be carefully selected in line with the data requirements.
The task complexity, drawn from the data analysis step, including the data volume and availability, is a key element in the model design and development. The point is *"go from simple to complex"*. Based on the state-of-the-art, some configurations and ML architectures are highly recommended, depending on the task's objective. As seen in the previous analysis (cf. chapter 5), several factors come together: the number of classes to be learned for a classifier helps to identify the volume of data, but also the adequate capacity of the model to develop, depending on the nature of the data and the information we want to retain, choose a representation method for the inputs that will emphasise the value of the data; knowing the maximum volume of data is also a key element for limiting the search space for a training algorithm to converge more easily on the best model.
Based on data analysis, highlighting required quality criterion and volume availability, in addition to the possible influencing elements to be taken into account (e.g. target environment features), the model architecture will be set and the data input representations will be configured accordingly (e.g. identify the main features to be emphasised on each data sample, by the input representations layer of the target model), including the data representation scheme, the outliers/corner cases handling if needed (e.g. use of appropriate data weighting layers to guarantee a balanced impact of the features).

7.2.2.2 **Focus on target performance objectives from the industrial perspective**

As discussed before, one of the reasons why several AI solutions cannot be deployed is the wide gap between evaluation objectives and industrial objectives. While focusing on the ML/DL model performance evaluation, several tools and means exist to estimate and assess the model performances. The generalisation bounds are statistical tools that provide some information about the ability of a model to generalise the learned information to unseen data during training (cf. section 5). The generalisation bounds combined with a complete evaluation pipeline, including several metrics adapted to the current application (cf. section 2.2) along with their interpretability and the known thresholds (e.g., the required minimum accuracy value), is unfortunately not enough to guarantee that the produced model will be deployed on the target system. In fact, from the industrial perspective, the required performance of the ML component is more strictly defined when it comes to critical systems (such as those dealt with in the aeronautical industry). Because the AI integration must comply with the safety requirements of the target system, the AI module must, therefore, be analysed beyond its simple recall and precision values. The same policy can be applied here, where the AI module must be optimised to meet the KPIs defining what a *«good model»* is. To do so, the industrial KPIs need to be translated into quantitative and measurable performance objectives using specific functions:

- The training objective definition, where the loss functions must be defined in accordance with the behaviour we want the model to produce the least. Several loss functions can be found in the state-of-the-art related to solving the task. However, those functions must always be adapted to the current application needs.
- The evaluation measures selection and the definition of acceptance criteria. This is a key when reflecting on the industrial performance objective from an empirical perspective. Among sets of measures for classification and regression applications found in the literature, there will be very often several options that can be combined to reflect the *"hidden"* metric on the system-level objective. Besides, benchmarking these metrics is essential through defining accurate thresholds and "*acceptance*" conditions and criteria. For this latter, system-level requirements can give more information about the possibility of acceptance with an accuracy of 95%, for instance.

A rigorous error analysis and performance understanding are required based on these two elements. Know the distribution of the 95% accuracy on the whole test dataset. This will help identify the areas of weakness of the model and understand their criticality and impact on the target system. In some cases, this stage requires interaction with the expert to see whether the errors that the model may have in certain particular cases (outliers) can be accepted (if the system can handle them) or not (if their impact could be catastrophic).
Other indications of the relevant model design and configuration can be drawn from the system-level requirements regarding model complexity and capacity vs volume, response time, embeddability, and possible implementation items (e.g., CPU or GPU). Finally, the expectations toward the certainty of the model outputs and the tolerable confidence margins, as well as the already known operating conditions (e.g., light intensity for video/image applications) that can influence the model behaviour.

7.2.2.3 **Anticipate ways to enhance the performances**

Here, the objective is two folds:

- Avoid discarding the developed model along a costly process and leverage the effort invested in the model;
- Enable a safe integration into the target system, with some guarantees on the model adaptability to make it more performant and compliant with the expectations.

One possible way to ensure the objectives above is to anticipate integrating high-influencing elements in the model setting to limit their impact on performances. Another essential element is identifying how to make the model learn better; regularisation, boosting, and optimisation practices can enhance

model learning. Besides, during the model learning and evaluation-adaptation steps, the training objective definition and adjustment should include those identified elements with a high impact on performances. Every task, whatever its complexity, can be simplified through iterative steps: make the model learn a simple version of the task, and then add the complexity criterion step-by-step while leveraging what is already learned. For example, while training a model to correctly classify images into ten (10) different classes, the problem can first be simplified, where the target labels can be grouped into two main classes, making the problem more straightforward for the model, then iteratively, perform other separations and make a re-training-based adaptation of the current model making it learn all the classes gradually. This process will, on the one hand, make the task more uncomplicated to solve and, on the other hand, enable identifying the more complex classes to be learned so data-related issues can be identified. A data augmentation, for example, can then be foreseen to ensure some balance among classes, or more training examples can be added to make it easier to learn the most complicated classes with more examples.
Finally, including the anticipated AI-component requirements (from the target application perspective) in the learning objective will help converge to the target performances better.

7.2.3 Reinforce the model's robustness and stability

7.2.3.1 **Using class separation to improve stability**

An ML system learns classification by building decision boundaries through manifold separation. However, each decision boundary remains implicit for the AI system designer. It is embedded in the function of the model, which can be set in (usually) high dimensionality. A better knowledge of the boundaries may help understand the system’s errors (i.e., class confusion). A possible way to do so is to analyse each class's maximum stability space (using formal methods) by checking the closest boundaries and corresponding distance of each data point. This will allow us to check each one's closest frontier and the distance at which this frontier takes place. Proper training should ensure that the distance between classes is sufficient and that each frontier distance is comparable in the same class. For example, the adequate distance may be expressed in terms of the minimal perturbation required to change the decision. Studying the boundary may help the AI system designer to monitor the risk of class confusion.
As an example, two classes close to a common decision boundary (compared to other classes) would tend to indicate that both classes have not been well-separated. Such information can then be leveraged for devising mitigations strategies (e.g., modifying the training set or the model’s hyperparameters).

7.2.3.2 **Using ODD perturbations to reinforce robustness**

When designing an ML system, the ODD plays a fundamental role as it will describe both the nominal operating conditions and edge cases. These cases can represent rare occurrences of the nominal conditions or borderline instances in which some perturbations occurred. Special care should be taken to ensure the robustness of the trained model during the design and the training phase. Data augmentation techniques can be used, particularly edge and corner case generation methods. To validate the effect of such techniques, stability measures can be performed using the test set on specific perturbations modelled formally (when possible) or statistically. The point is to measure the impact of these perturbations on the system behaviour and determine at which point the system's performance starts degrading. Available methods include maximum stable space measurement (using formal methods) or the statistical study of the performance evolution against increasing levels of perturbations.

7.2.3.3 Using relevance properties to avoid some bias

Relevance properties allow an ML system designer to set requirements to justify its system's behaviour. During the early phases of the system design and training, they can help identify the potential learning bias of the system. However, this usually requires human assessment of the relevance of the results produced by some independent tool (be it formal or statistical). Some degree of automation can be setup if some form of semantic segmentation is available to compare automatically with the results of the relevance analysis. Detecting abnormal relevance can help characterise the quality of the training process. For example, fuzzy relevance results (scattered across the data set) usually indicate underfitting and incomplete training processes. Highly localised relevance on some detail unrelated to the semantic segmentation may indicate a bias in the training data set or an over-fitted model.

7.2.3.4 Using stability to crosscheck data sets

Measuring the stability around an input can bear different results depending on whether this input is correctly annotated or not. Indeed, even if the ML system treats the input correctly, it tends to have less stability around it since it has learnt wrongly annotated data. Poor stability on particular data points may indicate that the system did not learn a correct representation or that the data point was not correctly annotated. This would call for two opposite correction strategies:

- Studying the maximum stable space (using formal methods) in the training, validation, and test data sets may help identify data points that might be incorrectly annotated and increase the data set's quality.
- Measuring the training algorithm's stability can help identify that the training is not stable enough to capture the data behaviour.

7.3 Understanding of dependencies in system-level and ML-level requirements

In order to understand the expected AI-level performance and behaviour, the correlation between system-level requirements and AI-level targeted behaviour needs to be made.

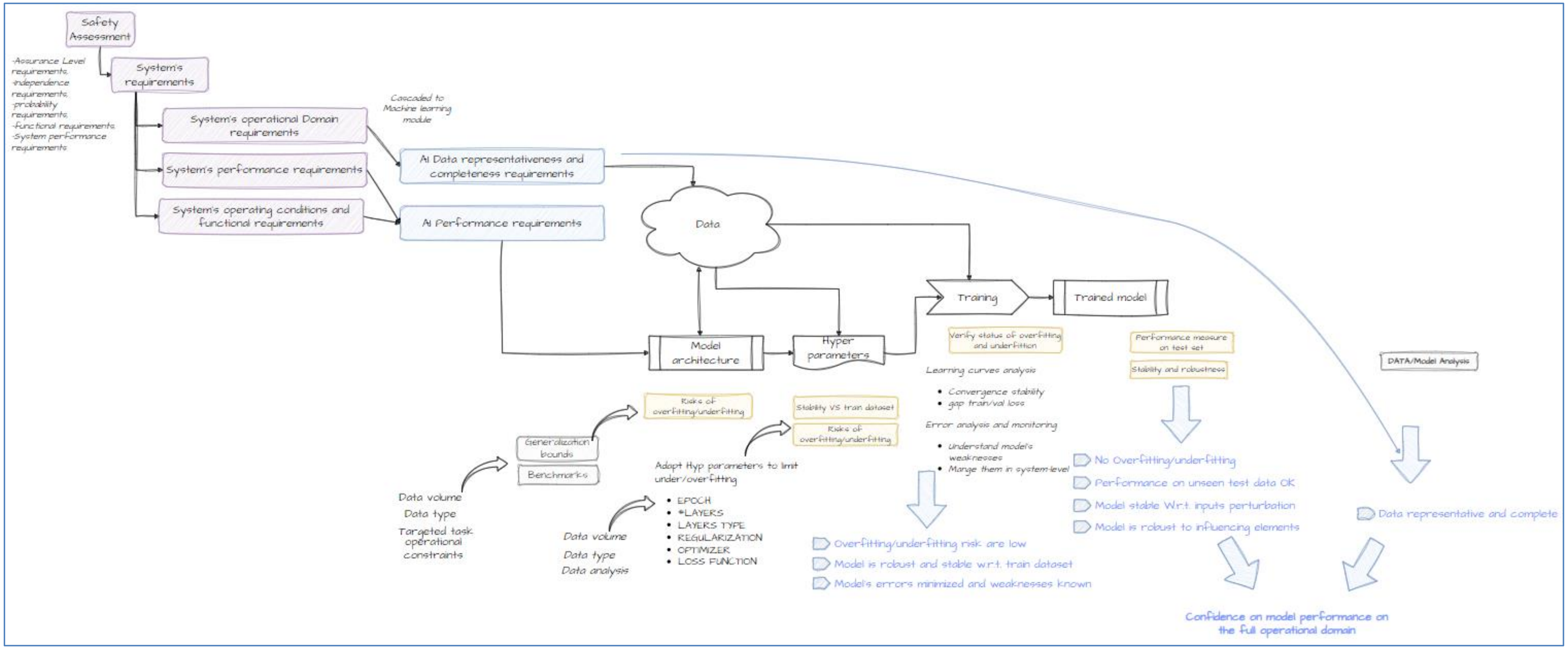

Safety Assessment
-Assurance Level requirements, -independence requirements, -probability requirements, -functional requirements -System performance requirements
System's requirements
System's operational Domain requirements
System's performance requirements
System's operating conditions and functional requirements
Cascaded to Machine learning module
AI Data representativeness and completeness requirements
AI Performance requirements
Data
Training
Trained model
Model architecture
Hyper parameters
Verify status of overfitting and underfittion
Performance measure on test set
Stability and robustness
DATA/Model Analysis
Learning curves analysis
Convergence stability
gap train/val loss
Error analysis and monitoring
Understand model's weaknesses
Mange them in system-level
Risks of overfitting/underfitting
Generalization bounds
Benchmarks
Stability VS train dataset
Risks of overfitting/underfitting
Adapt Hyp parameters to limit under/overfitting
EPOCH
#LAYERS
LAYERS TYPE
REGULARIZATION
OPTIMIZER
LOSS FUNCTION
Data volume
Data type
Targeted task operational constraints
Data volume
Data type
Data analysis
Overfitting/underfitting risk are low
Model is robust and stable w.r.t. train dataset
Model's errors minimized and weaknesses known
No Overfitting/underfitting
Performance on unseen test data OK
Model stable W.r.t. inputs perturbation
Model is robust to influencing elements
Data representative and complete
Confidence on model performance on the full operational domain

Figure 164. Understanding ML/DL performances definition and verification w.r.t system-level requirements and the operating and environment conditions

As shown in Figure 164, the objective of this step is to ensure, at the AI-level, compliance with the system-level requirements. In this latter, safety[204]-related analysis and requirements verification include the definition and the verification of requirements of the AI-level. These can be derived from the system-level requirements, design and considerations. The following criteria are worth being considered while addressing expectations of the ML/DL development when safety is a matter of the target system:

- *System's Operational Domain requirements* – cascaded requirements to data-driven model development based on the system's architecture. As explained in (Van Royen et al., 2023), the functional requirements of the system define the function of the product, its purpose, what it does, and the intended use. Having understood and defined these prerequisites, a better understanding of the operational requirements will be possible. These latter explain what human action is needed to keep the product operational, such as maintenance, observing and fixing issues. Finally, the technical requirements and product build decisions will drive the design accordingly. At this point, among the cascaded requirements for the AI-level:
    - *Data Completeness and Representativeness Requirements*. As discussed in Chapter 4, data completeness and representativeness are crucial criteria for validating the data quality assessment regarding the operational conditions in which the model will operate. Based on the definition of system-level operating conditions, the operating conditions for the AI-level components will be defined, and the training and testing datasets will be built upon those conditions.

- *System's Operating Conditions and Functional Requirements*: The definition of operating conditions refers to the specific circumstances under which a system is designed to function safely and effectively. This needs to be defined at the system-level in addition to the functional requirements and conditions. Hence, at the AI-level, the followings need to be considered:
    - *Performances Requirements – cascaded to ML modules*. Performance requirements at the system level refer to the system's desired outcomes, capabilities, and efficiency, independently of its included components. These requirements ensure that the system meets its intended objectives and functions. Note that, based on the definition of system-level requirements and operating conditions, the performance requirements for the AI level will be defined accordingly. These should be measurable and specified based on an alignment of the objectives concerning the efficiency of the ML/DL models that will be implemented in the target system (e.g., accuracy, precision, maximum error rate …), as shown in Figure 165. The model should show trusted behaviour by being robust and stable and meeting the industrial objectives for an acceptable ML module in the target system. In addition to having verified performances in the ODD. Hence, it is necessary to translate the system-level requirements into specific performance targets for the AI components. For example, if the system aims to achieve high accuracy in decision-making, the AI component may need to meet a minimum accuracy threshold (e.g., 95%). Besides, when speed and responsiveness are part of the system-

[204] Note that safety analysis is not in the scope of MLEAP, deriving AI-level requirements from system design and system considerations were not addressed.

level requirements (e.g., real-time applications), the criteria for response time, throughput, or latency will be considered. This affects the AI component's requirements regarding computational efficiency, inference speed, and optimisation techniques. In addition,

- *Performances impacting environment elements*. This includes environmental factors, which encompass noise and light intensity and can significantly impact data quality and, hence, the ML/DL model's output. The operating conditions should specify the possible and acceptable ranges for these environmental factors to ensure the safe operation of the system. In the AI-level, these factors should be handled to ensure an acceptable level of performance. Knowing these elements and the impact that they could have at the AI-level, it is required to define performance requirements with allowances for uncertainty and variability that could be caused by unexpected changes in input data samples for the ML/DL model and establish mechanisms for monitoring and adapting to changing conditions over time.

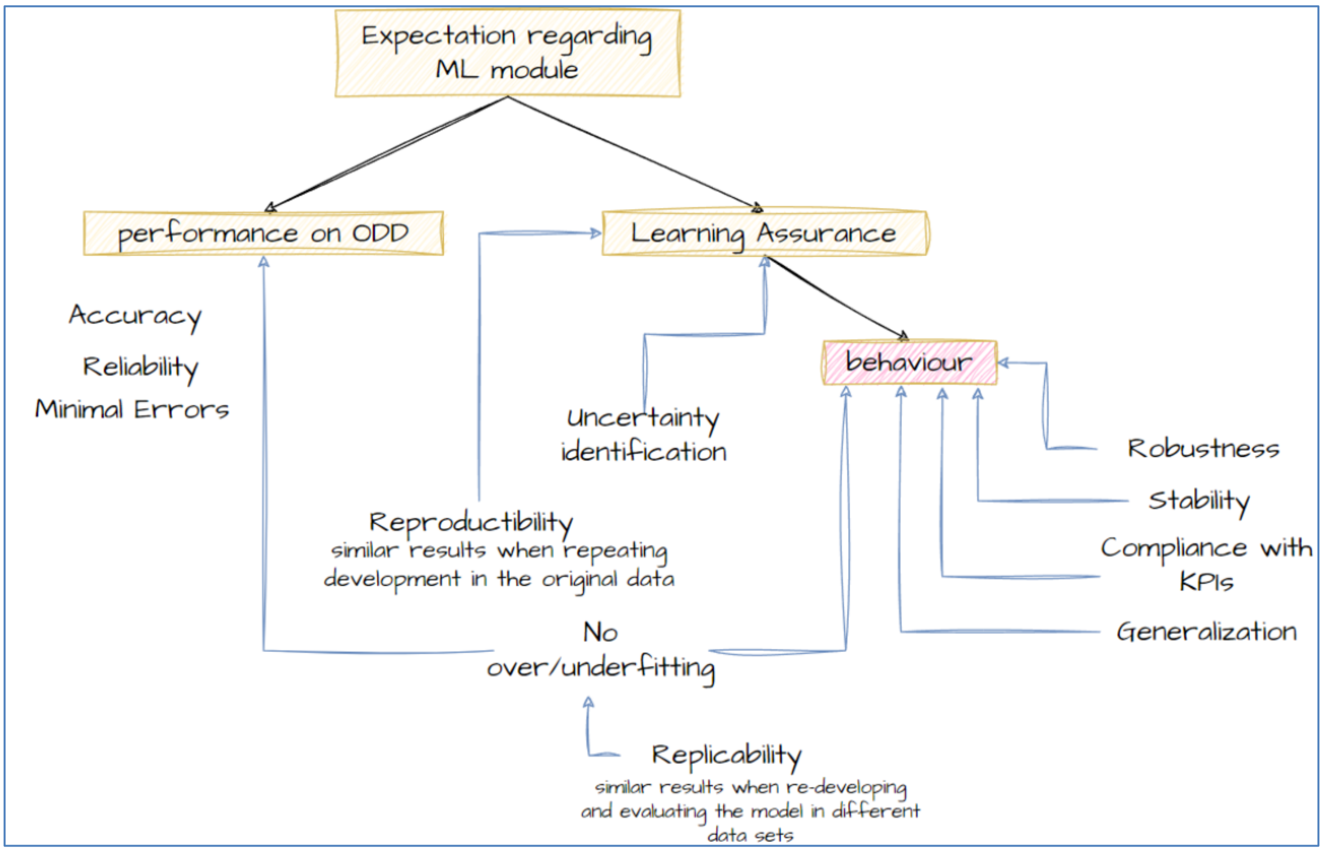

Expectation regarding ML module
performance on ODD
Learning Assurance
Accuracy
Reliability
Minimal Errors
behaviour
Uncertainty identification
Robustness
Stability
Compliance with KPIs
Generalization
Reproductibility
similar results when repeating development in the original data
No over/underfitting
Replicability
similar results when re-developing and evaluating the model in different data sets

Figure 165. Expectations regarding the ML module of a system and the related features[205]

[205] More details about the features mentioned in this figure can be found in Chapter 5.
Reproducibility and Replicability here refer, to the ability to reproduce the same results by the AI's evaluation using the same benchmarks.

In addition to the aforementioned criteria, to assure a compliance with safety at the system-level, several considerations could impact the design and the deployment of the AI models. These later must adhere to safety-critical standards, undergo rigorous testing and evaluation, and incorporate mechanisms for error analysis and detection, as well as a relevant data management and representation.

To ensure that target performances are met with the least possible impact on system-level objectives and safety, we need to verify that the model is not underfitting/overfitting by knowing the leading causes of both of them, such as not having enough data for training or a model that is too simple/complex[206] to handle the task's complexity or to be trainable in the available amount of data[207], or even the lack of representativeness of this one. Besides, as shown in Figure 166, during the model training, several information can be drawn from the loss behaviour, such as the correlation between the model learning and the model complexity, the training epochs and the training dataset size. The more complex the model is, the more data it needs for training and the longer the training should be. Moreover, to ensure performance stability, the model's training should be meant to reduce the model's sensibility to noise and, hence, anticipate the drop in performances after the model deployment.

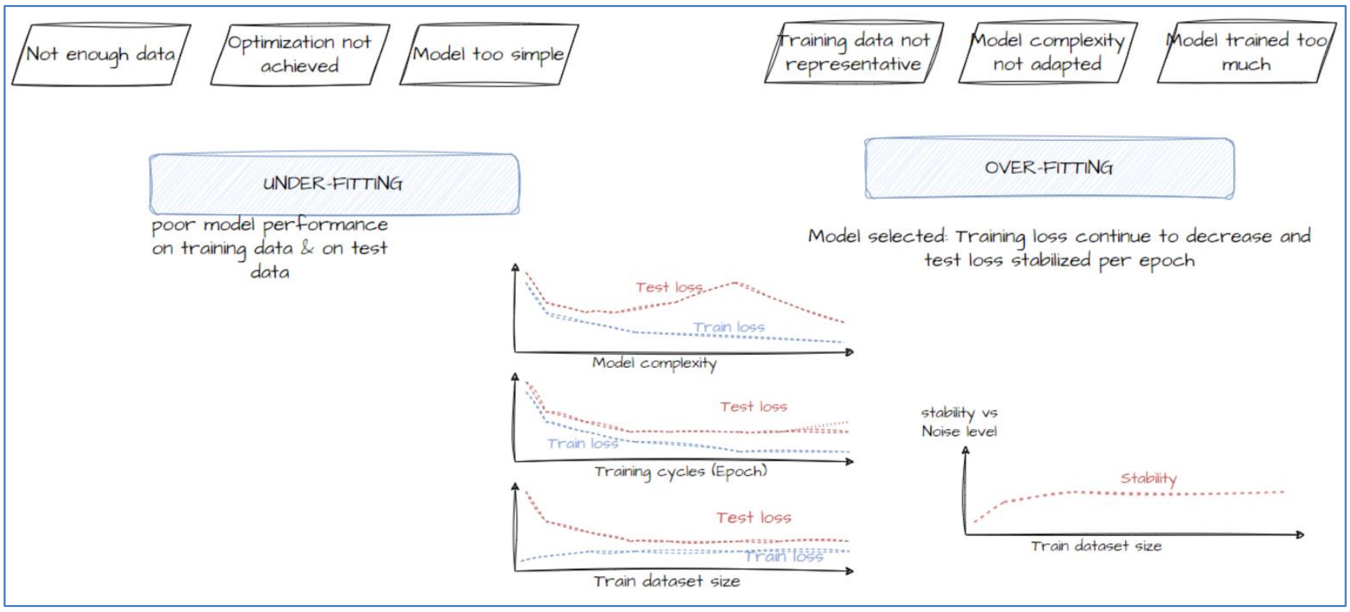


Figure 166. ML/DL models generalisability conditions

[206] For a too simple model, we will not be able to properly correlate input and outputs. For too complex model, we have a risk to overfit. However, in deep learning, it is possible to achieve good generalization, despite increasing the uncertainty in some inputs.
[207] With a validation dataset included (Riley et al., 2021).

7.4 Generic development approach

To verify the data quality and the ML/DL model performances mentioned in the previous section, the W-shaped learning assurance provides a development process that shows the importance of correlating the system-level and the AI-level requirements. In the earlier chapters, a set of methods and tools was identified to meet the development objectives of each stage of the W-shaped process, i.e., data qualification, model training and evaluation, performance validation and requirements verification. Nevertheless, several weak practices have been identified, and a list of crucial verification elements and a potential protocol to avoid common mistakes and issues were discussed.

Based on the previous section's discussions regarding the allocation of performance requirements and verification to the AI-level, w.r.t the system-level, and conclusions drawn from different tasks of the MLEAP project, this section proposes a generic protocol for developing AI solutions compatible with industrial requirements.

In this section, the target application is first defined, providing the main elements that need to be addressed in the ODD, w.r.t system-level requirements and known operating conditions, the input-to-output spaces mapping, the target performance objectives w.r.t industrial objectives, and the model's performances assessment, w.r.t the requirements verification that should be made. Next, an implementation of the W-shaped learning assurance process is provided, in line with the guidelines of the anticipated means of compliance mentioned before (cf. section 1.5), to meet each of the requirements and specificities related to the application domain, as well as to achieve the target industrial objectives. Furthermore, an experimental exploration shows how the ML/DL models' learning behaviour and evaluation results can be explored to enhance data quality and the importance of the design phase of the models.

7.4.1 Target application definition

The targeted application must be defined in terms of expected input data, the ML/DL model function, and expected outputs at the AI-component level. First, for a data set D, we need to define a function T_f representing a series of transformations applied to every sample in D, to build the corresponding data representation, in the input space X, and that will be fed into the model to produce the corresponding element from the output space Y:

$$D \xrightarrow{T_f} (X, Y),\ with\ x \in X\ \ and\ \ y \in Y$$

While building the ODD definition, the required data samples and corresponding characteristics are specified in line with the anticipated means of compliance for data management (cf. section 1.5.1.1). Hence, the transformation function can include a series of data analyses needed to identify the technical requirements (cf. section 4.3) and perform a set of processes (cf. section 4.4), such as synthesis, labelling, or sampling. The goal is to produce data collections suitable for the target domain representation in terms of quality and necessary volume while also being sufficiently representative of the domain of application and learning objectives. This is while maintaining several required data qualities, such as a trade-off between balance and completeness-representativeness (cf. section 4.5).

Further, a need for volume amplification and quality improvement will be identified. Finally, suitable data representation approaches will prove how the data will be processed to fit into the model's architecture. The data representation can be performed upstream, based on a simple transformation method (e.g., statistics and weights computation), as part of the data processing pipeline, or be based on a trainable method (i.e., automatic feature extraction), as part of the trainable model. In both cases, it aims to transform the dataset into mathematical representations to be fed into the model.

In the following, we consider the data representation computation as a part of the function T_f. Next, we define the essential elements of a target application A.

- Given a model $f \in F$, the input space X, and the output space Y, we define the target application A as a system of 8 elements:

$$A = \begin{cases} f \in F, & (1) \\ X : x_i = [x_i^0, \dots, x_i^n], & (2) \\ Y : y_j = [y_j^0, \dots, y_j^m], & (3) \\ f: X \xrightarrow{f(x)} Y, & (4) \\ M = \{m_1, \dots, m_k\}, & (5) \\ B = \{b_1, \dots, b_l\}, & (6) \\ b_t(m_t \circ f(x_t)) \in \{0, 1\} & (7) \\ E = \{e_1, \dots, e_z; (e_i \odot x_i) = x'_i\} & (8) \end{cases}$$

  In the system A:
  1. Corresponds to a mapping strategy (trained model), selected from the hypothesis space F. This can be defined with respect to a set of observations in the environment, events, and rules. All these elements should be formalised in the definition[208] of f to ensure a coherent mapping ;
  2. Is the definition of the input space X, and every element $x_i \in X$ is defined as a vector of dimension n, where values correspond to weights of the features associated with every sample in X;
  3. Is the definition of the output space Y, it provides a set of results and consequences emerging from the inputs in X, when fed to the model f. Every element $y_j \in Y$ can be viewed as a vector of resulting observations (output values, classes…). Depending on the target application, the output y_i can be a single value or a vector of output values in space of dimension m;
  4. Corresponds to the mapping from the input to the output space, performed by the trained model f, viewed as a function;
  5. It represents a set of indices related to a SMART[209] objective definition, it provides information about the effectiveness of f while performing the mappings in (4). Hence, this is defined as M, a set of metrics that formulate the objectives of the application. Where, in order to f be effective, it should conform to all the elements $m_t \in M$ corresponding to

[208] For the aim of simplicity, we consider *f* as the trained model, where hyperparameters have been optimized to perform the task *A*.
[209] SMART (Haughey, 2014) objectives are: specific, measurable, achievable, relevant and time-bounded.

evaluation measures (cf. Section 2.2 for different possibilities), and KPIs expression as well. This should provide some rules to have an accurate mapping in A (check the correctness and relevance of the mappings), and that should be valid whatever the elements of both spaces X and Y. This verification corresponds to the generalisation and robustness that need to be evaluated;

(6) Represents a *benchmarking* scheme of the industrial KPIs. Every element b_t corresponds to a validity/acceptance criterion and/or a target performance index or threshold value, as intended in the ground truth. Hence, for each metric $m_t \in M$, there exist an element $b_t \in B$ which is a validity criterion (e.g., a minimum threshold value, or indicator p ϵ [0,1] that could even represent a probability tolerance of the uncertainty or the error of the model). This enables the interpretation of the computed value by the measure correctly;

(7) Represents the benchmark exploration (all the defined elements b_t for the target application), based on the defined measures (m_t) for evaluation of the mappings performed by f. The objective here is to validate or not the industrial assessments of the performance. Hence, the application $b_t\left(m_t \circ f(x_t)\right)$ provides a binary value (i.e., is the performance accepted or not?). This validation scheme should provide a-priori hypothesis on the performance requirements of the component including the ML/DL model under development;

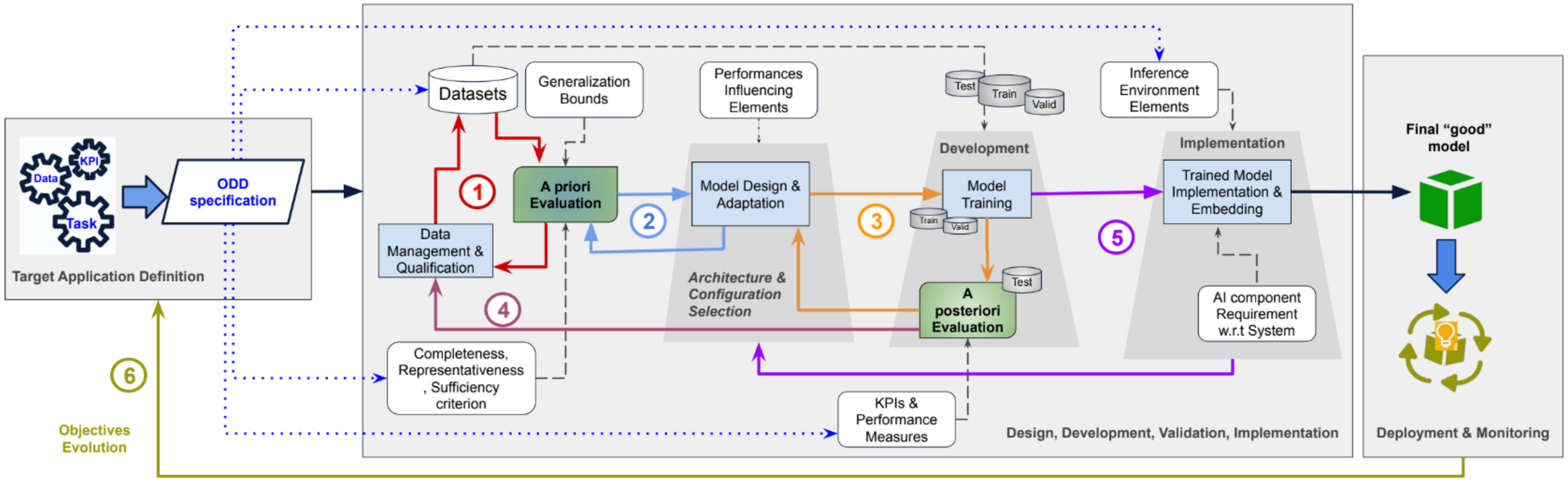
(8) It provides a set of elements, operating conditions, and circumstances that directly impact the model inputs and, hence, the predictions after implementation of the trained model. This could represent the set of environment elements (e.g., blurring, noise, signal errors, interference… several implementations can be found in ISO 24029-2[210]) that could impact the inputs of the model (the input space X) during inference. The function $\odot$ applies the set of elements E to every sample in X, resulting in a modified (or not[211]) data set.
Note that this function can be considered in two different ways (regarding the development stage):
   a. *Training time:*
      i. Used as an "*identity function*": if the inputs are perfectly clean (no noise, no perturbation …),
      ii. Considered as a noise introducer function or corruption function (e.g., image rotator, cutting words, background noise function …) during the model validation phase or to boost its performances during the training by including corrupted data samples,
   b. *Implementation time:*
      i. After embedding in the target system, the function $\odot$ represents the introduction of real-life factors, where the elements impacting the data inputs, represented by E, could harm the model performance. Hence, these elements are included in the model evaluation and training process to anticipate their impact on model performances.

[210] https://www.iso.org/obp/ui/#iso:std:iso-iec:24029:-2:dis:ed-1:v1:en
[211] When these specific conditions do not appear, the function $\odot$ will have no impact on the data input.

7.4.2 Pipeline definition

Following the previous workflows for the development and release of ML/DL models, they are defined in Section 5.2.1 and Figure 94 of Section 5.7.2.3, as well as the requirements discussed previously (for data preparation, generalisation guarantee and verification, and the performance evaluation pipelines), considering the training, evaluation, and implementation steps, an alignment with the learning assurance W-shaped process is made, including the targeted objectives for MLEAP (cf. section 1.5). As a result, an ML/DL development pipeline promoting the performances of implemented models is provided. Putting ahead the best practices and potential means to avoid common mistakes (cf. Section 5.6.1), an application-agnostic development pipeline for ML/DL applications is defined.

Figure 167 shows the recommended development steps and feedforward loops, enabling the development of ML/DL components for industry. This includes appropriate data qualification, model architecture and configuration selection, development through training and evaluation, and finally, implementation and potential objective evolution anticipation.


Data
KPI
Task
ODD specification
Target Application Definition
Datasets
Generalization Bounds
Performances Influencing Elements
Test
Train
Valid
Inference Environment Elements
A priori Evaluation
Data Management & Qualification
Model Design & Adaptation
Architecture & Configuration Selection
Development
Model Training
A posteriori Evaluation
Implementation
Trained Model Implementation & Embedding
AI component Requirement w.r.t System
Completeness, Representativeness, Sufficiency criterion
KPIs & Performance Measures
Design, Development, Validation, Implementation
Final "good" model
Deployment & Monitoring
Objectives Evolution
1
2
3
4
5
6

.... ODD elements to be considered

– – – – Specific inputs/requirements per task

——— Forward/backward actions of the pipeline

Figure 167. A general framework for ML/DL models development and evaluation, implementing the W-shaped learning assurance.

The pipeline of Figure 167 consists of three main steps, each presented separately in the following.

7.4.2.1 **Target application definition**

This corresponds to the first step, which was implemented as soon as possible before starting the ML/DL design and development. It includes the analysis of the target application needs and the specification of the expected outcomes of the ML component. In this step, based on an already known task, the ML/DL component's intended function will be defined. To do so, the target application can be defined as explained in the previous section 7.4.1, where the equations of system *A* are specified. By analysing the objective of the target application and in correlation with the KPIs, the data requirements will be specified, and the operational scope of the module expected at the end will be determined. This is the definition of the ODD, including all the conditions for correct functioning, and the causes of error known beforehand can be defined upstream. As a result, these essential elements, among others, will be derived from the ODD:

- **Datasets:** these are the first collected data that are made of the input and output spaces. Based on the completeness, representativeness, and sufficiency criteria, the datasets will be built in a data management and qualification process (cf. section 7.2.1).
- **Performances Influencing Elements:** These are the known characteristics of the target environment that are more likely to influence the model's behaviour and impact its performance. These elements are included in the model design and adaptation while performing architecture and configuration selection actions.
- **KPIs & Performance Measures:** This is the set of expectations from the industrial perspective toward the ML component under development. In addition to the target performances, the ML-level requirements should be derived from the system-level requirements, including safety and certification requirements, where system safety analysis and compliance demonstration may directly impact ML design activities[212]. Hence, system-level requirements and compliance with applicable regulations (for safety and performance verification) must be considered while defining the expected input/outputs of the ML/DL model as an intended function. Finally, the acceptability criteria and conditions (e.g., tolerable error margins and unacceptable errors) will also be derived from the system-level expectations and the AI-component level, considering the KPIs specification and evaluation measures selection (or definition).
- **Inference Environment Elements:** to anticipate the impact of the deployment environment on the ML/DL model, the criteria of the target implementation environment can be specified when possible. These elements can be related to the system-level requirements and operating conditions that, when they are not verified, can have an impact on the ML-component too, or elements related to changing conditions that cannot be controlled at the system or ML-component level (e.g., weather conditions and light intensity, which have an impact on video applications).

[212] Taking into account system-level safety and performance requirements do not aim solely to integrate AI recommendations and practices in existing system engineering practices. The aim is also to ensure that the AI/ML, when introduced in aerospace products, meets the applicable regulation and that the necessary compliance demonstration activities are performed and documented

7.4.2.2 **Design, development, validation, and implementation**

This step is built around two primary evaluation stages: *a priori evaluation* of data quality and generalisation estimation and *a posteriori evaluation* after training, including several development loops that will be run until the delivery of the final *good model*. Note that this process is designed for an offline model development setting. It applies to supervised model development; the data quality management and model performance evaluation must be specified accordingly. In the pipeline of Figure 167, the design, development, validation and implementation steps are made of a two folds evaluation scheme:

- **A priori evaluation.** This first evaluation concerns the performance objectives assessment before developing the ML/DL model and data management. Here, the listed development and design pitfalls (cf. sections 5.6.1 and 5.6.2) and mishandling of the task (cf. section 5.6.3) should be investigated in order to prevent a weak learning and development process of the model. The main inputs are the data quality and volume criteria and the generalisation bounds that could be assessed at this level, enabling verification of the conformity of the model's complexity with the used architecture. In this first evaluation, data requirements, in terms of completeness, representativeness and necessary volume for model training, should be specified (cf. section 4.3), as well as the generalisation bounds selection and computation;
- ***A posteriori evaluation***. This second evaluation comes after the model training. It concerns the performance evaluation and verification of model generalisability, robustness and performance stability. In this evaluation, the main inputs are the KPIs and selected performance measures (that are supposed to express well the expected level of performance and intended function conformity to the objective), in addition to the test dataset, which is assumed to be selected w.r.t several data management criteria. Note that the set of performance evaluation metrics (cf. section 2.2) should be used, w.r.t the target task, along with a set of the computed measures' domain-specific (business) acceptance criteria. This is to verify how well the model can meet the expectations of the final application. Besides, the a-posteriori evaluation of the generalisation bounds provides, in some cases (e.g., this is the case when there is a massive volume of data and a shallow neural network, the bounds are not vacuous), statistical guarantees regarding the average performance of the trained model in the full ODD. Finally, the hypothesis on the performance requirements of the ML/DL model will be either verified, hence validating the resulting model (loop 5 on Figure 167), or trigger the feedforward loop (4) to check the data and refine the objectives.

In addition to the two-fold evaluation objective, the different forward and backward actions on the pipeline are defined as follows:

(1) After the ODD specification, the datasets come with the definition of the objectives and the data quality criteria. Hence, to ensure that the collected data complies with the quality and volume criteria required to build an ML model, it must be evaluated and qualified using several tools and methods (cf. chapter 4) w.r.t the target application. This evaluation includes three main steps:
    a. Identify influencing elements of the data quality in terms of representativeness and completeness. This includes the data samples distribution, the corner cases and outliers' identification, as well as their impact on the training of an ML/DL model;
    b. Based on the ODD specification, identify the requirements for a minimal size of data needed. This should be performed early enough to ensure that data collection activities

are launched if required. Besides, if there are any specific cases handling the data (specific measures for some outliers), this should be specified;

c. If data has not been collected yet, based on (a) and (b), perform data collection, preparation, and pre-processing actions as needed. At this stage, the data quality evaluation will be conducted, where completeness and representativeness will be verified to build an accurate dataset for the remaining steps of the pipeline.

After these data qualification and evaluation steps, a set of operations, like data augmentation, specific processing (e.g., balance), and cleaning, can be performed when needed. Finally, the data is divided into at least three sets for training, validation, and testing regarding the IID hypothesis.

(2) Once the target application is specified and the data requirements are defined, the model architecture is chosen. In the *Model Design & Adaptation* step, the model will be determined using an approach that could help better meet the objectives while remaining compliant with the constraints at the system-level and the target application (e.g., real-time application, be embedded in a resource-limited system …) in addition to data-related constraints (e.g., available data volume, inputs size and type). The mappings between the inputs and outputs, as well as the generalisation bounds (cf. section 5.4.2) that provide signals on the ability of the designed model to generalise to unseen data after training, are elements included in this loop to enhance the model architecture. Furthermore, some known elements that could influence the model's results/performance should be anticipated, namely in *Performances Influencing Elements*. For instance, several regularisation and model adaptation methods (cf. section 5.5) can be used in the model architecture design phase. Finally, factors that could change the inputs' structure or included information (e.g., background noise on speech data, fog of dust on camera for image data) must be included in the input features description and representations learning.

(3) When loop (2) is completed, the resulting model architecture and configuration will be used for running the development process, including the *Model Training* and the *A posteriori evaluation,* using the data sets that are qualified for running. To do so, a benchmark including a set of industrial KPIs, evaluation measures, and acceptability criteria, set during the ODD definition (cf. sections 7.2.3.2 and 7.4.2.1), will be included in the training-evaluation iterations to perform the target task effectively. *A posteriori* evaluation of the trained model is then performed to ensure it meets the industrial objectives, including generalisation and robustness evaluation. In this crucial step, measures defined in section 2.2 can be combined and adapted to match the target application objectives. Besides, adapted loss functions and training objective functions should be used as recommended in the state of the art (Q. Wang et al., 2022; Zhu et al., 2022). Based on the evaluation using the test dataset, a backward action can be considered to re-work the model design and configuration.

(4) It is important to remember that the basic rule for this loop is to adopt an iterative process concerning the improvement and adaptation of both the training and test data and the construction of the model. During this loop, the aim is to make each stage as secure as possible, with the necessary verifications to avoid backtracking, which is costly, once the model has been implemented. At the end of this last stage, the verification in the target environment is a decisive step in the success of the approach and the model built. To do so, in-depth analysis and

improvement actions are required after training if the model does not meet specified performance requirements. That said, identifying the leading cause of the lack of performance, which could either rely on poor training (e.g., inadequate loss function or bad hyperparameter optimisation), inadequate architecture choice (i.e., poor or excessive capacity), insufficient data (either in quality or quantity) or poor specifications. Each of these sources would be increasingly costly to modify. It is then essential to isolate which one(s) are involved in the case under study. This is where combining assessment methods working directly on data (e.g., PCA) with methods using the model as feedback (e.g., Cleanlab) would be helpful, as it allows one to observe the interaction between the data and the model. This enables a more complete and holistic understanding of the influence of data on the model's performance and could help to identify a weak point more quickly.

Note that one of the main issues that could impact the model performance acceptability, concerning both robustness and generalisation, is the reproducibility of the results (Tatman et al., 2018). For the same inputs processed in equivalent circumstances/technical contexts, running at different times, the model is supposed to be deterministic and expected to produce the same outputs. To ensure this, the randomness[213] of some ML/DL architectures (e.g., due to some layer's weights initialisation) needs to be considered, and model settings that help detect and avoid random behaviours need to be used. Hence, in the *A Posteriori Evaluation*, one of the main objectives is to ensure that the model would not produce unexpected behaviours while running with the same data several times in the same conditions. This aspect can trigger the loop (4) for further analysis if necessary. In the literature, little work has been done on this aspect (McDermott et al., 2021; Olorisade et al., 2017) so far. Mainly, in deep learning models, reproducibility is a significant concern, as several factors can prevent the reproducibility of the results of the same model, including random initialisations and hardware features. Although recent studies (Chen et al., 2022) have investigated some solutions to promote results reproducibility in DL, no specific approach defines how to design and develop DL models to guarantee reproducible results.

(5) After the model evaluation and validation on the test dataset, the implementation phase would reveal if the expected behaviour is met by the model's behaviour while interacting with the target domain in the phase *Trained Model Implementation & Embedding*. On this later, the *Inference Environment Elements,* initially included/represented during the design phase as *Performances Influencing Elements*, will be injected, and the model performance will be verified. Besides, the *AI component Requirement w.r.t System* will be verified to assess the compliance of the implemented model with the expectations. At this stage, two possibilities remain: the model can be validated and go to the *Deployment & Monitoring* phase (cf. section 7.4.2.3), or a backward action is needed

[213] In ML/DL models, mainly when NNs are used, the randomness can be due to: Random initialization of weights and biases of the model, randomness in regularization techniques like Dropouts and randomness in optimization techniques like Stochastic Gradient Descent (SGD). These technics result in non-deterministic model if not taken into account. Hence, to get predictable and repeatable results every time the same model is run, *seed* parameter is used. Setting the *seed* parameter to some value, (in general 0, 1, 2, or 3) will generate the same random numbers during multiple executions of the code on the same machine or different machines.

if validation fails. In the latter case, the implemented model does not meet the requirements, hence:

a. If performances considerably deteriorate, the influencing factors should be considered in the loop (5) backward to the design and adaptation. Hence, a new model should be developed and then evaluated. In this case, there is no need to re-build the datasets unless the problem remains on the data;
b. Otherwise, the *Performances Influencing Elements* can be included before training (in step 3) to ensure good performance after implementation and less impact on the released model.

At this stage, there are two main points to bear in mind:

- Backtracking is an action that calls into question the various choices made in the "*model training and evaluation* " stages. Indeed, the generalisation assessment functions and evaluation metrics will be revised if they do not reflect the target performance or even the designed model's configuration and architecture.
- In this case, the return should not use the same family (type and architecture) of already trained models. The objective of the design stage is to select a new family of models to train from, taking into account the particularities of the implementation environment, which led to the model developed in the previous iteration being rejected due to a considerable drop in performance. To illustrate this problem, if we were to select samples from the test dataset with a bias (where the samples come from a particular source) that would give an advantage to certain types of model after implementation, through the training-test-validation iterations, we might at some point discover that a specific type of model performs better with this verification dataset. On the other hand, after its implementation in the target environment, the performance of the selected model, when confronted with unbiased data, may not be as good as observed during the iterations of the loop (3).

7.4.2.3 **Deployment and monitoring**

After the model is deployed in the target environment, the objectives of the target system could evolve. This evolution can lead to changes in the objectives, which may mean that the model is inadequate to meet the new requirements and that the definition of the objectives needs to be reconsidered. This iteration, represented by loop (6), will trigger two principal activities:

a) The definition of new objectives and, therefore, the re-execution of the entire development pipeline, including the construction of new datasets or the evolution of those already used for the first pass, ensures data quality requirements for the model development.
b) Reusing (retraining or fine-tuning) the initially validated good model and evolving it by adapting the training objectives. In some cases, this action may not be possible (e.g., when the evolution of the objective at the system level impacts the task to be performed by the ML level by changing the data type and number of inputs), so a new model will be developed from scratch using an architecture that is more adapted to the new objectives.

In the case of model retraining, attention must be paid in step (a) to avoid reusing the identical training data distributions and, therefore, to avoid model overfitting. Besides, the model's generalisation and robustness will be evaluated through the execution of the complete pipeline. During this new pass, the already selected generalisation bounds and selected evaluation measures may also evolve. Hence, the loop (6) in Figure 167can be defined as follows:

(6) **Objectives' evolution**. When we would like to use the same model trained on the previous task A, providing a new task A^*. In this latter, the different elements of the equation A defined in section 7.3.1 should be adapted and redefined. Indeed, in this loop, the extension of the objectives, implies important changes on the target ML/DL function, where the dataset D^*(including input and output spaces) and the mapping function T_f^* will evolve. The measures, benchmarks, and acceptability criteria of the performance values (respectively, M, B, and b_t could remain unchanged (depending on the new objectives set during the system evolution). However, the mapping function (the ML/DL model) $f(x)$ and its impacting elements E will systematically be replaced by new definitions. Several scenarios are possible, here are some examples:

   a. Same targeted performances for the new system after evolution. No changes on M, B, and b_t, use of the same model validated through the precedent loops (1 – to 5). In this case, a new data qualification is required, including the verification of completeness and representativeness toward the new information to be learned, in addition to constructing new distributions *train, test,* and *valid* sets to ensure that the new model will not overfit.
   b. The targeted performances differ regarding the measures used and acceptance criteria. This means that M, B, and b_t are not the same, due to the larger ODD and datasets (e.g. learn to detect new classes in addition to previously learned ones). The model re-training/fine-tuning will rely on defining a new objectives, including the metrics selection w.r.t KPIs and a verification of the generalisation bounds will be needed.
   c. The targeted performances may not be the same, with different learning objectives and more complex tasks (e.g. running in a new environment). The Performance evaluation scheme may differ in terms of used measures and acceptance criteria. This means that M, B, b_t, and E may not be the same. We may rely on the previous model, performing a transfer learning to solve the new task. However, the whole pipeline must be run carefully (including clarification of the metrics and objectives). This represents a simple initialisation of the new model to be qualified during the entire process. Bearing in mind that the architecture may also evolve, depending on changes in the data and, therefore, expectations at the system level.

7.4.3 Experimental Exploration

To evaluate the complete pipeline described in section 7.4.2, in this section, we show some experimental results of the backward actions, where the most important analysis goes to the information that the model's performance can provide to the earlier steps, including data management, the model design and development, and the a priori evaluation of the choices already made, such as the generalisation bounds, the evaluation metrics and the data quality assessment.

7.4.3.1 **Impact of data augmentation**

7.4.3.1.1 Analysis of Augmented Data

On the AVI/lightning strike case, contrasting PCA results on the non-augmented and augmented data sets revealed a notable increase in space coverage, suggesting improved dataset completeness. On the contrary, the augmentation protocol significantly altered the trends observed in the original data set for dents classes, which could indicate a degradation of the representativeness of the sample population.

The entropy analysis shows that data augmentation boosts the entropy values of classes Smoky (i.e., lightning strikes) and more marginally dent_al. The difference in amplitude can be related to the augmentation protocol, with the lightning strikes being augmented with 20 supplementary samples per original occurrence. In contrast, each dent class is augmented with only 5 generated samples for each original occurrence. On the other hand, class dent_lb exhibits uneven results, from a slight increase in absolute value to a decrease if a ponderation scheme is applied to the inputs where samples with more bounding boxes have more weight. Even considering the lower sample count of dent_lb samples compared to the dent_al population, the lower average number of bounding boxes per sample for dent_al points toward confirming that in the AVI data set (and possibly in any multiple object detection task), samples containing more bounding boxes should be expected to contribute more to the learning process.

Neuron coverage provided an interesting insight into the model's activation behaviour. If the non-fine-tuned model (i.e., on augmented date) could have been contrasted with the fine-tuned one, it would have been interesting to run the original data set on the fine-tuned model first to look and interpret the discrepancies in activation patterns, then run on the complete data set (including the augmented samples) to investigate the consistency of the activation patterns. This could have provided information related to the model's learning process (in which case not much additional insight on the data quality could have been inferred), or if no information could be related to the model, this would have hinted at conclusions on the data augmentation protocol.

7.4.3.1.2 Model's Generalisation

As shown in section 5.8.2.4, the data augmentation has an essential impact on the model's performance. For FMNIST trained with augmented data, when the volume of augmented data is < 40,000 (compared with the 60,000 of the initial training dataset), we gain in performances compared with the test dataset (unchanged during the experiment), in stability and robustness for perturbations of low rotation or small translations. Then, when the volume of augmented data becomes more significant than the initial images, the performances deteriorate, and the training and test datasets are no longer derived from the same distribution.
Note that this only applies to the augmentations used in the experiment. The conclusions could be different with augmented data generated differently, and for which it could be challenging to say whether they are "similar" to the ODD. For instance, several augmentation functions are used in the AVI dataset to generate new images. However, the behaviour of the model's training stays the same. The performances seem to have deteriorated (cf. section 5.8.3.1.2.6), since the amount of synthetic data is higher than the real ones. To this end, an analysis of the impact of the different augmentation

functions is necessary to understand their influence on the representativeness and, therefore, on the model's performance.

7.4.3.1.3 Model's robustness

In addition to the data analysis and the generalisation bounds, a study regarding the model stability and robustness has been performed following the same evaluation protocol as in Section 6.2.

<u>LM11: Training algorithm stability</u>

Both models were evaluated by gradually reducing the number of images within their corresponding training dataset. The original training set is composed of 6,000 images for each class (so 60,000 images in total), and the augmented one is composed of 1 019 945 data points in total. The images are removed proportionally in each class at each step to keep the overall data set balanced. Figure 168 presents the evolution of the accuracy of the trained model depending on the percentage of data points removed in the corresponding training set. Both models seem to have comparable behaviour regarding data removal until around 60%, where the model trained on the original data set has a steep accuracy drop. At the same time, the accuracy of the model trained using data augmentation resists up to 90% of its training data being removed. This indicates that the training algorithm is more stable when using data augmentation (and consequently, the model seems to have better accuracy even with fewer data points to train on).

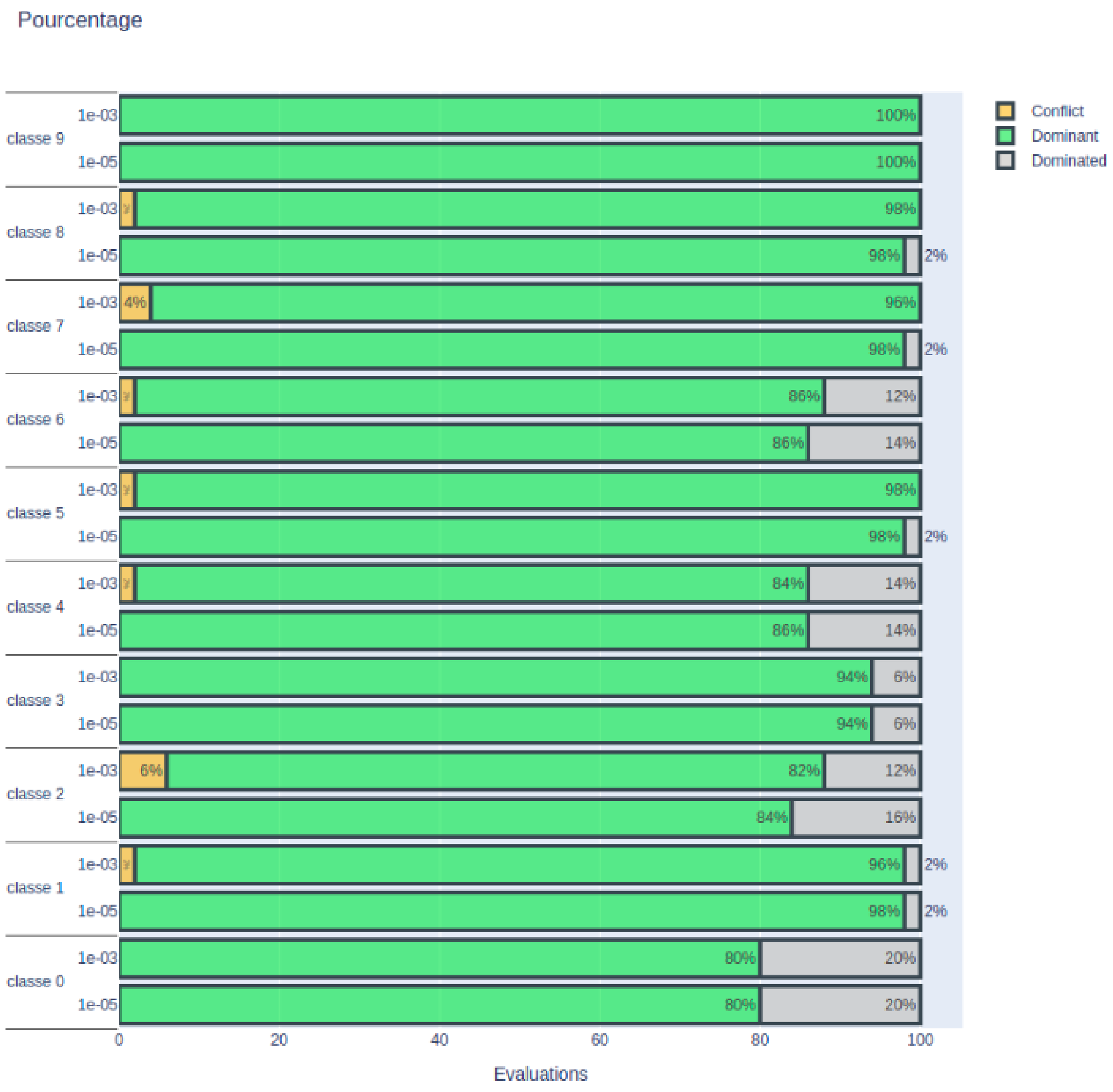


Figure 168. Stability of the original model

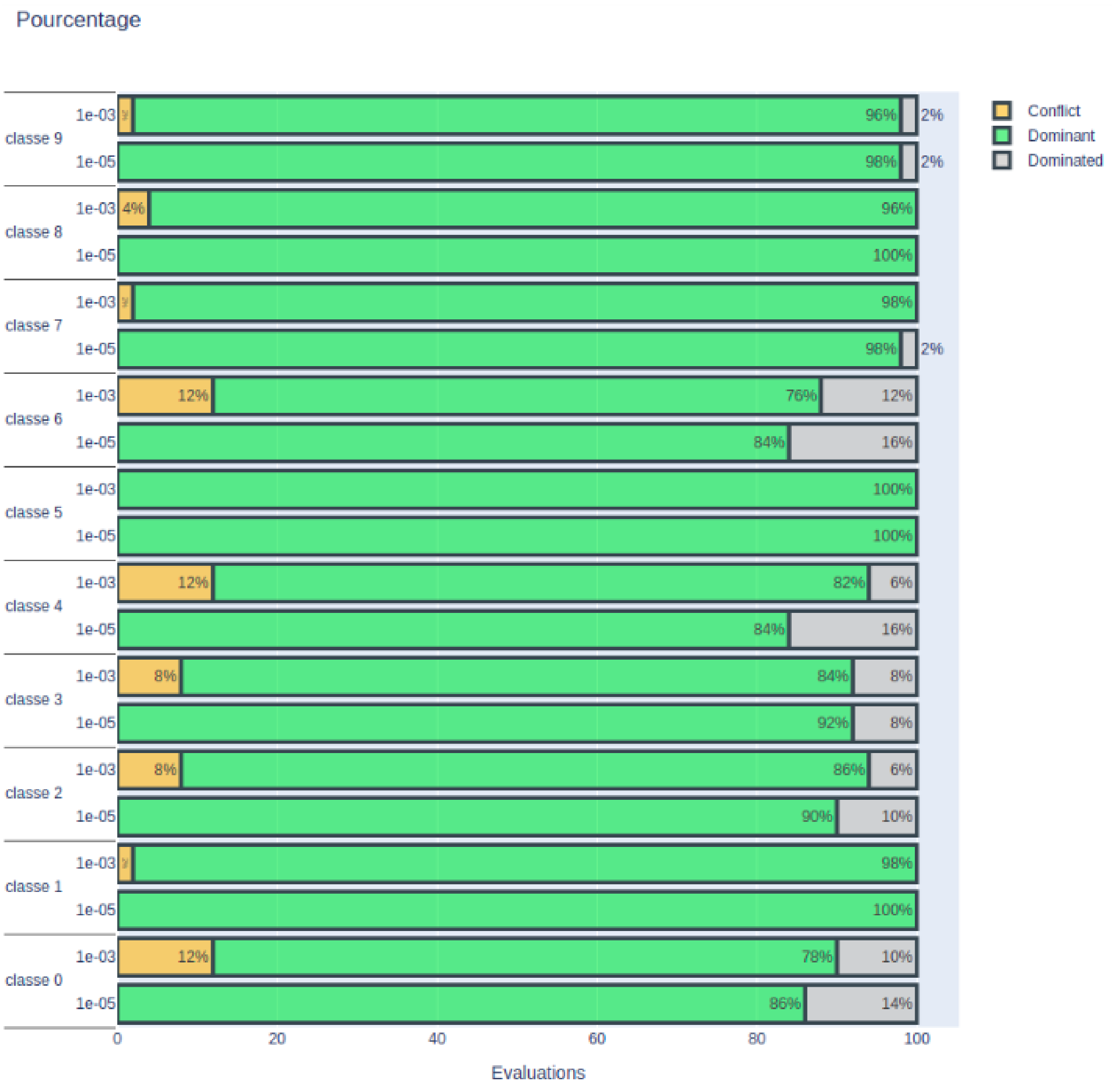


Figure 169. Stability of the data-augmented model

<u>LM13: Trained model robustness</u>

This study evaluated both models against a gradually stronger version of a Gaussian perturbation of over 500 images each time. This perturbation setup is the same as the one described in Section 6.3.2.1, using 9 intensity levels.

Figure 170 presents the evolution of the accuracy of both models following an increase in the perturbation. Both models seem to have the same dynamic in terms of robustness against this perturbation, with a bit of advantage to the model trained on the original data set. This indicates that the data augmentation had no or even a slight negative impact on the robustness against this type of perturbation. If the data augmentation had considered this type of perturbation, the results would have been different.

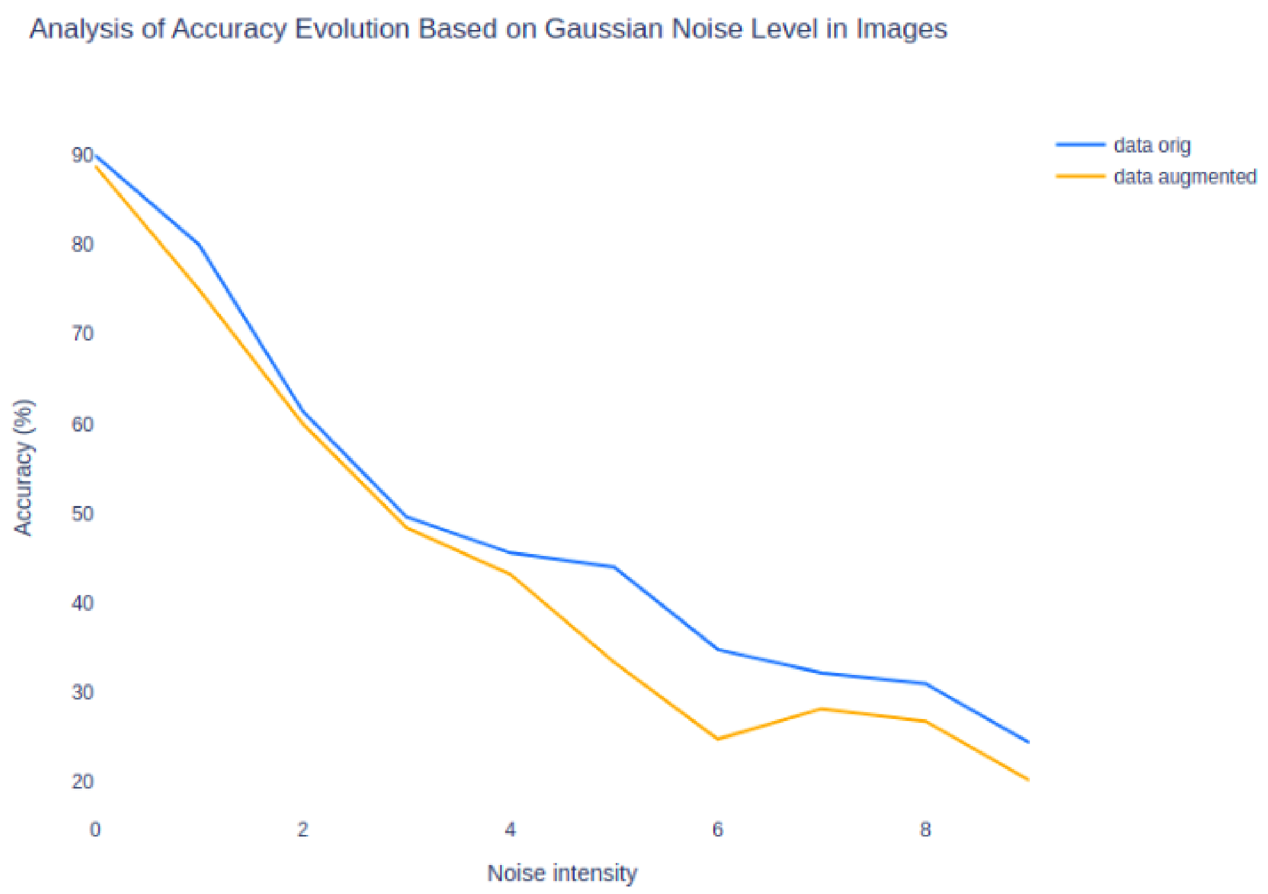


Figure 170. Evaluation of the accuracy of both models against a gradually stronger Gaussian perturbation

7.4.3.2 **Impact of model design**

The design and architecture of AI models play a crucial role in determining their performance and efficiency in the industry. Model architecture, configuration, and hyperparameters must align with the target application and deployment environment to ensure optimal performance, stability, and scalability. A well-suited model architecture considers factors such as the task's complexity, the input data's nature, computational resources available for training and inference (e.g., CPU, GPU), and constraints related to deployment, such as latency and memory requirements. In critical use cases where reliability, interpretability, and safety are paramount, model design choices must prioritise robustness, interpretability, and uncertainty quantification. Furthermore, continuous monitoring, error analysis, and data analysis are integral to model development and maintenance, enabling iterative improvements and ensuring that AI systems perform effectively in real-world applications. Ultimately, thoughtful consideration of model design and architecture is fundamental for promoting trust in AI-based systems, especially in aviation. In this later, where safety-critical systems are employed, the design of AI models carries even greater significance. To drive the model design and ensure less impact on safety, several verifications need to be made and adapt the model design accordingly:

- Robustness to adverse conditions: models should be resilient to variations in environmental conditions, such as noisy data collected in different situations (e.g., extreme weather);
- Performance optimisation and design adaptation: Careful selection and tuning of hyperparameters, including learning rates, batch sizes, and regularisation techniques, are essential for achieving a better minimum closer to the global minimum. The balanced model's effective complexity and capacity also impact generalisation capabilities.

Finally, the continuous monitoring and maintenance of the model after deployment are important. Indeed, the post-deployment/implementation in the target system needs to handle the model's predictions, especially when errors are produced. This will feed a continuous monitoring of the deployed model's performance. Performance verification during the model's use is essential to detect anomalies or degradation in performance. Regular maintenance and retraining cycles can be implemented to adapt the model to evolving conditions and ensure ongoing reliability (cf. section 7.4.2.3).

7.4.4 Discussion

In the previous sections, the importance of revisiting experimentation, particularly in qualitative analysis of models for industrial applications, is highlighted. To do so, some conclusions need to be leveraged:

- *Understanding the impact of data augmentation as one of the data management practices that enhance data quality*. The data augmentation makes more data available for development. However, the analysis reveals nuanced impacts of data augmentation on model performance. While it enhances the space coverage and boosts entropy values for certain classes like lightning strikes, it also alters trends observed in the original data set, potentially degrading representativeness for specific classes such as dents. Understanding these impacts is crucial for ensuring the reliability and generalisation of the model. Revisiting experimentation allows a deeper understanding of how data augmentation affects different aspects of model behaviour, enabling informed decisions on protocol adjustments to improve model performance and data representativeness.
- *Model generalisation and performances stability are the main indicators used to validate the model.* The findings suggest that the volume and nature of augmented data significantly influence the model's performance and generalisation capabilities. Revisiting experimentation is essential to analyse how different volumes and types of augmented data impact model stability, robustness, and alignment with the test dataset distribution. This analysis can inform decisions on the optimal volume and types of augmented data to be used, ensuring that the model maintains stable performance across various conditions and remains aligned with real-world data distributions.
- *Don't take the Model Design for granted*. Model architecture and design choices significantly influence performance, reliability, and interpretability, particularly in safety-critical industries like aviation. Revisiting experimentation allows for evaluating different model architectures, configurations, and hyperparameters to identify the most suitable design for the target application. By continuously iterating on model design and architecture based on experimental insights (cf. section 7.4.2.2, where loops 2 and 3 of the pipeline in Figure 167 highlighted the

importance of the design phase of the model), developers can optimise performance, enhance robustness to adverse conditions, and ensure compliance with safety and reliability requirements.

- *Continuous Monitoring and Maintenance, providing feedback on previous steps*. The post-implementation[214] (cf. Figure 167) monitoring and maintenance are essential for detecting anomalies in the model's behaviour, performance degradation, and evolving conditions that may impact model reliability. Hence, revisiting experimentation during the monitoring and maintenance phase allows for identifying performance anomalies and adapting the model to changing conditions. Regular retraining cycles based on experimental insights ensure ongoing reliability and alignment with evolving data distributions, contributing to the long-term success of AI-based systems with less impact on safety.

Finally, understanding the system-level requirements and conditions allocating the performance requirements for the AI-level, is necessary to take actions during the AI development to ensure acceptable performances by the produced models, such as data augmentation and other data management practices, model design setting and choices, performance indicators (generalisation, robustness and stability) assessment, and some anticipated recommendations for the post-implementation maintenance of model performance, to have constant reliability and alignment with real-world conditions.

7.5 Conclusion

In this chapter, an application-agnostic development workflow is provided. This later is designed in line with the W-shaped learning assurance and the MLEAP objectives (cf. sections 1.4 and 1.5). It includes synthesising best practices that can be adopted to avoid common issues during several stages of the W-shaped process. To do so, we provided a set of instructions and anticipated recommendations for evaluation and verification actions that need to be made. These focus on the alignment between AI-level requirements and the required verifications to meet the expected behaviour and performances, thanks to the understanding of the dependencies between system-level and AI-level requirements (cf. section 7.3). Furthermore, a generic development pipeline (cf. Figure 167) is provided, putting all together the different findings of the experimental work done in this project, in data management (cf. Chapter 4), model development and training promoting generalisation properties (cf. Chapter 5), and the robustness and performances stability and assessment (cf. Chapter 6). In section 7.4.3, we show some experimental results of the backward actions in the pipeline, where the most critical analysis goes to the information that the model's performance can provide to the earlier steps, including data management, the model design and development, and the a priori evaluation of the choices already made, such as the generalisation bounds, the evaluation metrics and the data quality assessment.

[214] Note that in MLEAP project, the implementation in target system of the different use-cases is not in the scope of the experimental analysis, we do, nevertheless, provide a set of guidelines to follow that are evident based on the results of established analysis, and also based on the state of the art. Please note that the conclusions concerning the post-implementation stage must be verified on the application case being processed, and adaptations may be necessary.

Hence, the impact of data augmentation on the model's performance and the impact of the model design (architecture and configuration have been explored. The main conclusion is that the data is the centrepiece of the whole process, making it possible to influence the model's performance. Besides, the trust of any AI brick is built around several quantitative criteria about the trained model's performance. These latter should be aligned with requirements verification processes to ensure they comply with the system-level requirements regarding expected behaviour and function accomplishment as well as safety-related criteria.

8. Conclusions

This document represents the 4th and final deliverable of the MLEAP project. It includes the synthesis of the work completed since the beginning of the project. The main objective was to provide experimental-based recommendations on implementing several steps of the EASA's W-shaped learning assurance (EASA, 2024). The achievements of the MLEAP project, the tools and methodologies needed to instantiate several steps of the EASA's W-shaped development approach, and the verifications of AI-level requirements and how these can be evaluated are investigated. The activities are divided into three technical tasks:

- **Task 1:** Data completeness and representativity, with the handling of the corner cases (cf. Chapter 4)
- **Task 2:** Model design and development through the handling of the generalisation properties (cf. Chapter 5)
- **Task 3:** Model evaluation in terms of robustness and stability (cf. Chapter 6)

These tasks follow an equivalent process to meet the specified objectives (cf. Section 1.5) and for which they are meant. First is the requirements and needs analysis, where the expected outcomes for each task have been specified, and then a state-of-the-art analysis of existing methods and tools is provided. In this report, aviation use cases (cf. Chapter 3) have served as examples to support the study and investigation of existing methods and tools and made recommendations. In the second step, specific development and investigations of the different tasks are addressed, and a set of methods and tools have been selected for experimental work. The main objective is to provide empirical analysis concerning the applicability of the methods. The chosen methods have been subjects for exploration using the selected use cases of MLEAP and several toy-use cases. The objective is to assess the methods' applicability on low-dimensional datasets while comparing their behaviour and then apply them to the aviation use cases to support the different hypotheses and findings.

During the data management, assessing data quality is a complex topic. Completeness and representativeness are usually not handled per se, and almost no dedicated tools exist. Thus, indicators from several metrics, such as diversity and relevance, must be leveraged. Besides, different tools (like sample similarity) can be used. Essentially, the ODD definition remains the key, where an objective estimation of completeness or representativeness requires knowing the exact extent and distributions of the phenomena to observe. In addition, there is a necessary trade-off between representativity and diversity since rare cases must be amplified to be modelled correctly. Hence, Chapter 4 analyses the requirements for the ODD to set the expectations for the representativity-diversity trade-off. Several tools and methods described in the selection grid should provide the means to achieve the data qualification objectives. Several experimental analyses were carried out using some of these methods and tools. The lesson to be learned from this analysis is that even off-the-shelf software suites are not sufficient solutions for estimating the completeness and representativeness of the data to be managed to ensure that the level of quality and volume corresponds to what is expected. Indeed, these tools do not exempt data analysts from having a good knowledge of the characteristics of the data and the problem to be addressed. However, the methods these tools offer can be combined very well with other approaches, which must be adapted to the

needs of the current application. Finally, we would stress that the assessment of completeness and representativeness can only be carried out based on the ODD since it has already been said that it is impossible to do so based on real data distributions and must, therefore, be supplemented by a characterisation of the data in terms of what the model does with it.
To take these analyses further, the methods selected from the grid developed in Chapter 4 are evaluated, enabling this hypothesis to be verified with more data from the more complex use cases selected for the MLEAP project, where data dimensionality is not the same.

Based on the data volume and quality, the generalisability of trained models, assessment and evaluation can be performed (cf. Chapter 5) using different methods. However, when insufficient data is available, the model can over-fit or under-fit the training data compared to the model and task complexity. The state–of–the–art analysis, made in Chapter 5, presented methodologies to right-scale the complexity and capacity of the models depending on the scope of the task under development and the volume and nature of input data while measuring the level of generalisation reached by a training session. Besides, the generalisation bounds are statistical tools that could help assess upstream the ability of a model to generalise performances. However, their efficiency remains an open discussion since several interpretations can be found in the literature. Hence, during the complete ML/DL development, the influencing elements have been analysed. The main limitations of the common practices have been pointed out, and a set of challenges are raised. The objective is to provide means to enhance the generalisability of developed models. Hence, several key elements are identified, among which regularisation and optimisation techniques can be combined with a better definition of the target objectives of the application, in addition to other means, such as error analysis and uncertainty evaluation that could help investigate the limitations of the trained models and assess better their acceptance by the target application.

Indeed, there are several issues related to the design and development pipelines of ML and DL solutions, such as overfitting and underfitting and the design pitfalls that result in a weak model in production. Most previous research has focused on empirical methods for generalising from many training examples using no domain-specific knowledge. In the past few years, new methods have been developed to apply domain-specific knowledge and formulate valid generalisations from single domain training examples. Other techniques were designed to solve problems related to areas with little qualified data and train algorithms to accelerate and improve model learning. Regarding generalisation evaluation, it is highlighted that the classical approach that uses a set of technical metrics to assess the model's performance is limited in capturing performance-related aspects, where the industrial objectives (KPIs) and reproducibility of results matter. Hence, the generalisation evaluation is ultimately more crucial than initially thought. Therefore, the experimental analysis of the generalisation assessment is focused on the applicability and interpretation of the results of selected generalisation bounds as part of a complete experimental evaluation set up of existing ML/DL development pipelines (i.e., corresponding to selected use-cases). The existing pipelines, providing the different use cases, have been discussed, where several inconsistencies have been identified regarding the models' design and training regarding the target objective. The experimental endeavours focused on use-cases of different dimensionalities and complexity to address these aspects. The findings unearth substantial disparities between target objectives and actual model performances. For instance, the ATC-STT analysis underscores the necessity of diverse training data to mitigate biases. In contrast, the AVI analysis emphasises the importance of segmentation models to fulfil the target performance objectives. Furthermore, the ACAS Xu analysis underscores the significance of models accurately reproducing input data and addressing data class imbalances. These insights illuminate the critical nuances in model development and highlight the importance of aligning AI initiatives with industrial objectives and ensuring generalisation in real-world applications.

Regarding the stability and robustness evaluation, the work put into perspective the framework defined by the EASA CP and the one proposed by the ISO/IEC sub-committee on AI. Both share similarities in terms of definitions, properties introduced and requirements defined. These shared commonalities allow us to bridge methods from both sources to reinforce the strategy detailed by the EASA in the W-shape approach. In particular, the different methodologies described in the ISO/IEC 24029 series to measure the stability and robustness of neural networks can be leveraged to build a validation scheme adapted to the W-shape approach. Various techniques are available to perform such validation by relying, for example, on formal, statistical or empirical methods. While formal can tackle machine learning models with high dimensionality, they offer strong local stability and robustness properties. Statistical methods allow for a more widespread verification but with little guarantee of the results. Finally, empirical methods allow specific scenarios and benchmarks to be applied to follow an expert-centric approach to the validation, which would be done by a human rather than an automated one. The main findings and recommendations regarding the experiments show that combining these approaches is necessary to cover as many of the requirements expressed in the EASA CP as possible.

Measuring the quality of the training step takes part in the larger question of evaluating the model. We present multiple approaches in Chapter 6, from pure performance measures with empirical, data-based approaches to validating explicit properties, particularly stability, through an array of analytic or formal methods. Those methods, while sometimes challenging to put into practice, allow for compelling analysis of the behaviour of the models, including at runtime, allowing monitoring of the whole system in a live setup. The stability and robustness of an ML system remain key requirements that are both a necessity and a challenge to assess when dealing with ML components. In practice, this validation implies combining several methods to cover as much of the definition of the ODD as possible by taking into account the various perturbations that can be anticipated. Following the definition of the ODD, the validation process can be applied using formal, statistical, and empirical methods depending on where we validate the ODD and under which risk assumption we are.

Finally, based on the different findings concerning the MLEAP's tasks, an operational proposal (cf. Chapter 7) on how to project the different recommendations into the W-shaped approach, which extends to the whole set of tools, methods, evaluations, and verification presented in the document.

For further information on the MLEAP achievements and to provide feedback, do not hesitate to contact the project team at ai@easa.europa.eu.

9. Annexes

Note that some details that are presented in this section are confidential and cannot be made public. This content will be removed before the public release of the deliverable, due to confidential statements (cf. Disclaimer of the report).

A. Computational assumption for a priori evaluation of generalization bounds.

To compute bounds evaluation a priori, we need to make some assumptions concerning parameters used in theorems formula. We assume, for the a priori evaluation that:

- The spectral norms of the corresponding fully connected weight matrix are lower than 10
- The maximum value of the weight matrix is lower than 10
- KL divergence approximation: $N_{param}. ln\left(\frac{2}{\varepsilon}\right)$; where N_{param} is the number of trainable parameters and ε is the generalization gap tolerance.
- Cover complexity is less than 1
- The loss function is bounded by 1 (10 for AVI use case)
- The gradient of the training risk is lower than 0.8 at each step of the training phase
- The gradient of parameters norms of each layer is lower than 50 during the training phase

As mentioned in section 5.9.2.3.1, those assumptions are conservative as it is applicable for all functions contained in the hypothesis space.

B. *This content has been removed in accordance with the confidentiality statements mentioned in the document's disclaimer*

10. References

A. Boopathy, L.D., T.W. Weng, P.Y. Chen, S. Liu, 2018. CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks.

A. H. Land, A.G.D., 1960. An automatic method of solving discrete programming problems 28.

A. Kwiatkowsku, H.W., 2008. PCA-Based Parameter Set Mappings for LPV Models With Fewer Parameters and Less Overbounding 16.

Abidin, N.Z., Ismail, A.R., Emran, N.A., 2018. Performance analysis of machine learning algorithms for missing value imputation. International Journal of Advanced Computer Science and Applications 9.

Adhikari, N., Behera, N., Tripathi, V., 2022. Modeling of Optimal Deep Learning Enabled Object Detection and Classification on Drone Imagery. https://doi.org/10.1109/ICAISS55157.2022.10010957

Agarap, A.F., 2018. Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

Aghababaeyan, Z., Abdellatif, M., Briand, L., S, R., Bagherzadeh, M., 2023. Black-Box Testing of Deep Neural Networks Through Test Case Diversity. IEEE Transactions on Software Engineering 1–26. https://doi.org/10.1109/TSE.2023.3243522

Ahamed, S.S.R., 2010. Studying the Feasibility and Importance of Software Testing: An Analysis. https://doi.org/10.48550/ARXIV.1001.4193

Alecu, L., Bonnin, H., Fel, T., Gardes, L., Gerchinovitz, S., Ponsolle, L., Mamalet, F., Jenn, É., Mussot, V., Cappi, C., others, 2022. Can we reconcile safety objectives with machine learning performances?, in: ERTS 2022.

Ali, Peshawa Jamal Muhammad, Faraj, Rezhna Hassan, Koya, E., Ali, Peshawa J Muhammad, Faraj, Rezhna H, 2014. Data normalization and standardization: a technical report. Mach Learn Tech Rep 1, 1–6.

Allen, D.M., 1971. Mean square error of prediction as a criterion for selecting variables. Technometrics 13, 469–475.

Almaimouni, A., Ademola-Idowu, A., Kutz, J.N., Negash, A., Kirschen, D., 2018. Selecting and evaluating representative days for generation expansion planning, in: 2018 Power Systems Computation Conference (PSCC). IEEE, pp. 1–7.

Almeida, R., Vieira, M., 2011. Benchmarking the resilience of self-adaptive software systems: perspectives and challenges, in: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 190–195.

Alquier, P., 2021. User-friendly introduction to PAC-Bayes bounds. arXiv preprint arXiv:2110.11216.

Altman, N., Krzywinski, M., 2018. The curse(s) of dimensionality. Nature Methods 15, 399–400. https://doi.org/10.1038/s41592-018-0019-x

Alvarez-Coello, D., Klotz, B., Wilms, D., Fejji, S., Gómez, J.M., Troncy, R., 2019. Modeling dangerous driving events based on in-vehicle data using Random Forest and Recurrent Neural Network. 2019 IEEE Intelligent Vehicles Symposium (IV) 165–170.

An, J., Lee, W., Kim, J., 2007. Adaptive Detection and Concealment Algorithm of Defective Pixel. 2007 IEEE Workshop on Signal Processing Systems 651–656.

Anaby-Tavor, A., Carmeli, B., Goldbraich, E., Kantor, A., Kour, G., Shlomov, S., Tepper, N., Zwerdling, N., 2020. Do not have enough data? Deep learning to the rescue!, in: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 7383–7390.

Anthony, M., 2004. Generalization error bounds for threshold decision lists. Journal of Machine Learning Research 5, 189–217.
Anthony, M., Bartlett, P.L., Bartlett, P.L., others, 1999. Neural network learning: Theoretical foundations. cambridge university press Cambridge.
Anttila, S., Ketola, M., Vakkilainen, K., Kairesalo, T., 2012. Assessing temporal representativeness of water quality monitoring data. Journal of Environmental Monitoring 14, 589–595.
Ao, J., Wang, R., Zhou, L., Wang, C., Ren, S., Wu, Y., Liu, S., Ko, T., Li, Q., Zhang, Y., Wei, Z., Qian, Y., Li, J., Wei, F., 2021. SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing. https://doi.org/10.48550/ARXIV.2110.07205
Arnold, A., Ernez, F., Kobus, C., Martin, M.-C., 2022. Knowledge extraction from aeronautical messages (NOTAMs) with self-supervised language models for aircraft pilots, in: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track. Association for Computational Linguistics, Hybrid: Seattle, Washington + Online, pp. 188–196. https://doi.org/10.18653/v1/2022.naacl-industry.22
Arora, S., Ge, R., Neyshabur, B., Zhang, Y., 2018. Stronger generalization bounds for deep nets via a compression approach, in: International Conference on Machine Learning. PMLR, pp. 254–263.
Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., others, 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information fusion 58, 82–115.
Ashok Kumar, L., Karthika Renuka, D., Shunmuga Priya, M.C., 2021. Analysis of Audio Visual Feature Extraction Techniques for AVSR System. EAI. https://doi.org/10.4108/eai.7-12-2021.2314528
Asudeh, A., Jin, Z., Jagadish, H., 2019. Assessing and remedying coverage for a given dataset, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE, pp. 554–565.
Aust, J., Pons, D., 2022. Comparative Analysis of Human Operators and Advanced Technologies in the Visual Inspection of Aero Engine Blades. Applied Sciences 12. https://doi.org/10.3390/app12042250
B. Barz, E. Rodner, Y.G. Garcia, Denzler, J., 2019. Detecting Regions of Maximal Divergence for Spatio-Temporal Anomaly Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 1088–1101.
Ba, J., Caruana, R., 2014. Do deep nets really need to be deep? Advances in neural information processing systems 27.
BAA, 2009. Automated Border Control (ABC) Trial Stansted Airport. BAA.
Baevski, A., Zhou, H., Mohamed, A., Auli, M., 2020. Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20. Curran Associates Inc., Red Hook, NY, USA.
Baggenstoss, P.M., 2004. Class-specific classifier: avoiding the curse of dimensionality. IEEE Aerospace and Electronic Systems Magazine 19, 37–52.
Bai, E.-W., Cheng, C., Zhao, W.-X., 2019. Variable selection of high-dimensional non-parametric nonlinear systems by derivative averaging to avoid the curse of dimensionality. Automatica 101, 138–149.

Bai, X., Wang, X., Liu, X., Liu, Q., Song, J., Sebe, N., Kim, B., 2021. Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments. Pattern Recognition 120, 108102.

Bai, Y., Li, Y., Shen, Y., Yang, M., Zhang, W., Cui, B., 2022. AutoDC: an Automatic Machine Learning Framework for Disease Classification. Bioinformatics.

Bajgar, M., Berlingieri, G., Calligaris, S., Criscuolo, C., Timmis, J., 2020. Coverage and representativeness of Orbis data.

Bak, S., 2021. nnenum: Verification of ReLU Neural Networks with Optimized Abstraction Refinement, in: NASA Formal Methods Symposium. Springer, pp. 19–36.

Bak, S., Tran, H.-D., 2022. Neural Network Compression of ACAS Xu Early Prototype Is Unsafe: Closed-Loop Verification Through Quantized State Backreachability, in: Lecture Notes in Computer Science. Springer International Publishing, pp. 280–298. https://doi.org/10.1007/978-3-031-06773-0_15

Bak, T., S.,. Tran, H.D.,. Hobbs, K.,. Johnson, 2020. Improved geometric path enumeration for verifying relu neural networks 32.

Balaji, Y., Sankaranarayanan, S., Chellappa, R., 2018. Metareg: Towards domain generalization using meta-regularization. Advances in neural information processing systems 31.

Balaraman, V., Razniewski, S., Nutt, W., 2018. Recoin: relative completeness in Wikidata, in: Companion Proceedings of the The Web Conference 2018. pp. 1787–1792.

Balasubramanian, V., 1997. Statistical inference, Occam's razor, and statistical mechanics on the space of probability distributions. Neural computation 9, 349–368.

Balduzzi, G., Ferrari Bravo, M., Chernova, A., Cruceru, C., van Dijk, L., de Lange, P., Jerez, J., Koehler, N., Koerner, M., Perret-Gentil, C., others, 2021. Neural Network Based Runway Landing Guidance for General Aviation Autoland. United States. Department of Transportation. Federal Aviation Administration ….

Bansal, A., Sikka, K., Sharma, G., Chellappa, R., Divakaran, A., 2018. Zero-shot object detection, in: Proceedings of the European Conference on Computer Vision (ECCV). pp. 384–400.

Bansal, M.A., Sharma, D.R., Kathuria, D.M., 2021. A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications. ACM Computing Surveys (CSUR).

Barbiero, P., Squillero, G., Tonda, A., 2020. Modeling generalization in machine learning: A methodological and computational study. arXiv preprint arXiv:2006.15680.

Barr, D.J., Levy, R., Scheepers, C., Tily, H.J., 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. Journal of memory and language 68, 255–278.

Bartlett, P., Freund, Y., Lee, W.S., Schapire, R.E., 1998. Boosting the margin: A new explanation for the effectiveness of voting methods. The annals of statistics 26, 1651–1686.

Bartlett, P.L., Foster, D.J., Telgarsky, M.J., 2017. Spectrally-normalized margin bounds for neural networks. Advances in neural information processing systems 30.

Bartlett, P.L., Harvey, N., Liaw, C., Mehrabian, A., 2019. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. The Journal of Machine Learning Research 20, 2285–2301.

Bartlett, P.L., Mendelson, S., 2002. Rademacher and Gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research 3, 463–482.

Bashir, D., Montañez, G.D., Sehra, S., Segura, P.S., Lauw, J., 2020. An Information-Theoretic Perspective on Overfitting and Underfitting, in: Gallagher, M., Moustafa, N., Lakshika, E.

(Eds.), AI 2020: Advances in Artificial Intelligence. Springer International Publishing, Cham, pp. 347–358.

Bastani O., C.A., Ioannou Y.,. Lampropoulos L.,. Vytiniotis D.,. Nori A., 2016. Measuring Neural Net Robustness with Constraints. Proceedings of the 30th International Conference on Neural Information Processing Systems.

Batini, C., Rula, A., Scannapieco, M., Viscusi, G., 2015. From data quality to big data quality. Journal of Database Management (JDM) 26, 60–82.

Bayasi, N., Hamarneh, G., Garbi, R., 2022. BoosterNet: Improving Domain Generalization of Deep Neural Nets Using Culpability-Ranked Features, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 538–548.

Beede, L., E.,. Elliott, E.,. Hersch, F.,. Iurchenko, A.,. Wilcox, L.,. Ruamviboonsuk, P.,.M. Vardoulakis, 2020. A Human-Centered Evaluation of a Deep Learning System Deployed in Clinics for the Detection of Diabetic Retinopathy.

Belkin, M., Hsu, D., Ma, S., Mandal, S., 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proceedings of the National Academy of Sciences 116, 15849–15854.

Bellman, R., 1966. Dynamic programming. Science 153, 34–37.

Bendale, A., Boult, T., 2015. Towards open world recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1893–1902.

Ben-Gal, I., 2005. Outlier detection, Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers.

Bengio, Y., Courville, A., Vincent, P., 2013. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence 35, 1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2006. Greedy layer-wise training of deep networks. Advances in neural information processing systems 19.

Bengio, Y., Louradour, J., Collobert, R., Weston, J., 2009. Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 41–48.

Bianchini, M., Scarselli, F., 2014. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. IEEE transactions on neural networks and learning systems 25, 1553–1565.

Biere, A., Heule, M., Van Maaren, H., 2009. Handbook of Satisfiability: Handbook of Satisfiability. IOS Press, Amsterdam.

Biessmann, F., Golebiowski, J., Rukat, T., Lange, D., Schmidt, P., 2021. Automated Data Validation in Machine Learning Systems. IEEE Data Eng. Bull. 44, 51–65.

Bilgic, E., Gorgy, A., Yang, A., Cwintal, M., Ranjbar, H., Kahla, K., Reddy, D., Li, K., Ozturk, H., Zimmermann, E., others, 2021. Exploring the roles of artificial intelligence in surgical education: A scoping review. The American Journal of Surgery.

BIS, 2004. An investigation into the performance of facial recognition systems relative to their planned use in photo identification documents – BioP. BIS.

Bishop, C.M., others, 1995. Neural networks for pattern recognition. Oxford university press.

Bittner, L., 1962. R. Bellman, adaptive control processes. A guided tour. XVI+ 255 S. Princeton, NJ, 1961. Princeton University Press. Preis geb. $6.50. Zeitschrift Angewandte Mathematik und Mechanik 42, 364–365.

Blalock, D., Gonzalez Ortiz, J.J., Frankle, J., Guttag, J., 2020. What is the state of neural network pruning? Proceedings of machine learning and systems 2, 129–146.

Blanchard, G., Deshmukh, A.A., Dogan, Ü., Lee, G., Scott, C., 2021. Domain generalization by marginal transfer learning. The Journal of Machine Learning Research 22, 46–100.

Blanchard, G., Lee, G., Scott, C., 2011. Generalizing from several related classification tasks to a new unlabeled sample. Advances in neural information processing systems 24.

Blatchford, M.L., Mannaerts, C.M., Zeng, Y., 2021. Determining representative sample size for validation of continuous, large continental remote sensing data. International Journal of Applied Earth Observation and Geoinformation 94, 102235.

Bohanec, M., Bratko, I., 1994. Trading accuracy for simplicity in decision trees. Machine Learning 15, 223–250.

Bolte, J.-A., Kamp, M., Breuer, A., Homoceanu, S., Schlicht, P., Huger, F., Lipinski, D., Fingscheidt, T., 2019. Unsupervised Domain Adaptation to Improve Image Segmentation Quality Both in the Source and Target Domain, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

Bolya, D., Foley, S., Hays, J., Hoffman, J., 2020. Tide: A general toolbox for identifying object detection errors, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. Springer, pp. 558–573.

Bottou, L., others, 1991. Stochastic gradient learning in neural networks. Proceedings of Neuro-Nımes 91, 12.

Bouthillier, X., Konda, K., Vincent, P., Memisevic, R., 2015. Dropout as data augmentation. arXiv preprint arXiv:1506.08700.

Breitenstein, J., Termöhlen, J.-A., Lipinski, D., Fingscheidt, T., 2021. Corner Cases for Visual Perception in Automated Driving: Some Guidance on Detection Approaches. https://doi.org/10.48550/ARXIV.2102.05897

Brennan, J.R., Dyer, C., Kuncoro, A., Hale, J.T., 2020. Localizing syntactic predictions using recurrent neural network grammars. Neuropsychologia 146, 107479.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R., 1993. Signature verification using a" siamese" time delay neural network. Advances in neural information processing systems 6.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., others, 2020. Language models are few-shot learners. Advances in neural information processing systems 33, 1877–1901.

Brubaker, J., Kilic, T., Wollburg, P., 2021. Representativeness of individual-level data in COVID-19 phone surveys: Findings from Sub-Saharan Africa. PloS one 16, e0258877.

Brunton, S.L., Nathan Kutz, J., Manohar, K., Aravkin, A.Y., Morgansen, K., Klemisch, J., Goebel, N., Buttrick, J., Poskin, J., Blom-Schieber, A.W., others, 2021. Data-driven aerospace engineering: reframing the industry with machine learning. AIAA Journal 59, 2820–2847.

Brutzkus, A., Globerson, A., 2019. Why do larger models generalize better? A theoretical perspective via the XOR problem, in: International Conference on Machine Learning. PMLR, pp. 822–830.

BSA, 2021. AI Bias Risk Management Framework. BSA - The Software Alliance.

Bucci, S., D'Innocente, A., Liao, Y., Carlucci, F.M., Caputo, B., Tommasi, T., 2021. Self-supervised learning across domains. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Buczko, M., Willert, V., 2017. Monocular Outlier Detection for Visual Odometry. pp. 739–745.

Bulso, N., Marsili, M., Roudi, Y., 2019. On the complexity of logistic regression models. Neural computation 31, 1592–1623.

Bunel, K.M.P., R, Turkaslan I.,. Torr P. HS.,. Kohli P., 2017. Piecewise linear neural network verification. CoRR.

Burke J., D.B., 2008. Field testing of six decision support systems for scheduling fungicide applications to control Mycosphaerella graminicola on winter wheat crops in Ireland. The Journal of Agricultural Science 146.

C. Hoffmann, H.W., 2015. LFT-LPV modeling and control of a control moment gyroscope. 54.

C. Northcutt, J.M., A.n. Athalye, 2021. Benchmarks, Pervasive Label Errors in Test Sets Destabilize Machine Learning.

C, R.T., 2001. Data Quality. The Field Guide, Boston Digital Press.

C. Sidrane, M.K., 2019. OVERT: Verification of nonlinear dynamical systems with neural network controllers via overapproximation. Workshop on Safe Machine Learning.

Cabitza, F., Campagner, A., Soares, F., de Guadiana-Romualdo, L.G., Challa, F., Sulejmani, A., Seghezzi, M., Carobene, A., 2021. The importance of being external. methodological insights for the external validation of machine learning models in medicine. Computer Methods and Programs in Biomedicine 208, 106288.

Cachi, P.G., Ventura, S., Cios, K.J., 2020. Fast convergence of competitive spiking neural networks with sample-based weight initialization, in: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. Springer, pp. 773–786.

Caiafa, C.F., Solé-Casals, J., Marti-Puig, P., Zhe, S., Tanaka, T., 2020. Decomposition methods for machine learning with small, incomplete or noisy datasets. Applied Sciences 10, 8481.

Carlini, N., Erlingsson, U., Papernot, N., 2019. Distribution density, tails, and outliers in machine learning: Metrics and applications. arXiv preprint arXiv:1910.13427.

Caruana, R., Lawrence, S., Giles, C., 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. Advances in neural information processing systems 13.

Caruana, R., Niculescu-Mizil, A., 2006. An empirical comparison of supervised learning algorithms, in: Proceedings of the 23rd International Conference on Machine Learning. pp. 161–168.

Caruana, R., Niculescu-Mizil, A., 2004. Data mining in metric space: an empirical analysis of supervised learning performance criteria, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 69–78.

Catania, C., Guerra, J., Romero, J.M., Caffaratti, G., Marchetta, M., 2022. Beyond Random Split for Assessing Statistical Model Performance. arXiv preprint arXiv:2209.03346.

Celebi, M.E., Aydin, K., 2016. Unsupervised learning algorithms. Springer.

Celis, L.E., Deshpande, A., Kathuria, T., Vishnoi, N.K., 2016. How to be fair and diverse? arXiv preprint arXiv:1610.07183.

Cha, J., Cho, H., Lee, K., Park, Seunghyun, Lee, Y., Park, Sungrae, 2021. Domain generalization needs stochastic weight averaging for robustness on domain shifts. arXiv preprint arXiv:2102.08604 3.

Chakarov, A., Nori, A., Rajamani, S., Sen, S., Vijaykeerthy, D., 2016. Debugging machine learning tasks. arXiv preprint arXiv:1603.07292.

Chalapathy, R., Toth, E., Chawla, S., 2018. Group Anomaly Detection using Deep Generative Models, in: ECML/PKDD.

Challa, H., Niu, N., Johnson, R., 2020. Faulty requirements made valuable: On the role of data quality in deep learning, in: 2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE). IEEE, pp. 61–69.

Changyu, C., Yong, Y., Gang, Z., 2014. DMU-based project monitoring technology of aircraft development. Journal of Beijing University of Aeronautics and Astronautics 40, 198–203.

Chaudhari, P., Soatto, S., 2015. On the energy landscape of deep networks. arXiv preprint arXiv:1511.06485.

Chehreghan, A., Ali Abbaspour, R., 2018. An evaluation of data completeness of VGI through geometric similarity assessment. International Journal of Image and Data Fusion 9, 319–337.

Chen, B., Wen, M., Shi, Y., Lin, D., Rajbahadur, G.K., Jiang, Z.M., 2022. Towards Training Reproducible Deep Learning Models, in: 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE). pp. 2202–2214. https://doi.org/10.1145/3510003.3510163

Chen, H., Chen, J., Ding, J., 2021. Data evaluation and enhancement for quality improvement of machine learning. IEEE Transactions on Reliability 70, 831–847.

Chen, L.-C., Papandreou, G., Schroff, F., Adam, H., n.d. Rethinking Atrous Convolution for Semantic Image Segmentation. https://doi.org/10.48550/ARXIV.1706.05587

Chen, S.F., Goodman, J., 1999. An empirical study of smoothing techniques for language modeling. Computer Speech & Language 13, 359–394. https://doi.org/10.1006/csla.1999.0128

Chen, T.Y., Cheung, S.C., Yiu, S.M., 2020. Metamorphic Testing: A New Approach for Generating Next Test Cases. https://doi.org/10.48550/ARXIV.2002.12543

Chen, W., Yu, Z., De Mello, S., Liu, S., Alvarez, J.M., Wang, Z., Anandkumar, A., 2021. Contrastive syn-to-real generalization. arXiv preprint arXiv:2104.02290.

Chen, Y., Li, W., Sakaridis, C., Dai, D., Van Gool, L., 2018. Domain Adaptive Faster R-CNN for Object Detection in the Wild. https://doi.org/10.48550/ARXIV.1803.03243

Cheng, L., 2013. Unsupervised topic discovery by anomaly detection.

Cheng, M.-Y., Firdausi, P.M., Prayogo, D., 2014. High-performance concrete compressive strength prediction using Genetic Weighted Pyramid Operation Tree (GWPOT). Engineering Applications of Artificial Intelligence 29, 104–113.

Chernoff, H., 1952. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. The Annals of Mathematical Statistics 493–507.

Cho, J., Lee, K., Shin, E., Choy, G., Do, S., 2015. How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? arXiv preprint arXiv:1511.06348.

Choi J. Y., C.C.-H., 1992. Sensitivity analysis of multilayer perceptron with differentiable activation functions. Transactions on Neural Networks 3.

Chong, Y.S., Tay, Y.H., 2017. Abnormal Event Detection in Videos using Spatiotemporal Autoencoder, in: International Symposium on Neural Networks.

Choudhary, T., Mishra, V., Goswami, A., Sarangapani, J., 2020. A comprehensive survey on model compression and acceleration. Artificial Intelligence Review 53, 5113–5155.

Chrominski, K., Tkacz, M., 2010. Comparison of outlier detection methods in biomedical data. Journal of Medical Informatics & Technologies 16/2010.

Chu, B., Qureshi, S., 2022. Comparing Out-of-Sample Performance of Machine Learning Methods to Forecast US GDP Growth. Computational Economics 1–43.

Chu, P., Bian, X., Liu, S., Ling, H., 2020. Feature space augmentation for long-tailed data, in: European Conference on Computer Vision. Springer, pp. 694–710.

Chung, Y., Haas, P.J., Upfal, E., Kraska, T., 2018. Unknown examples & machine learning model generalization. arXiv preprint arXiv:1808.08294.

Cluzeau, J.M., Henriquel, X., Rebender, G., Soudain, G., van Dijk, L., Gronskiy, A., Haber, D., Perret-Gentil, C., Polak, R., 2020. Concepts of design assurance for neural networks (CoDANN). Public Report Extract Version 1, 1–104.

Cohan, A., Feldman, S., Beltagy, I., Downey, D., Weld, D.S., 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. https://doi.org/10.48550/ARXIV.2004.07180

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. https://doi.org/10.48550/ARXIV.1604.01685

Cousot, P., 2012. Probabilistic Abstract Interpretation. Programming Languages and Systems 7211.

Cousot, P., 2000. Abstract Interpretation Based Formal Methods and Future Challenges.

Cousot P., C.R., 1976. Static determination of dynamic properties of programs. Proceedings of the Second International Symposium on Programming.

Cousot, P., Halbwachs, N., n.d. Automatic Discovery of Linear Restraints Among Variables of a Program.

Creusot, C., Munawa, A., 2015. Real-Time Small Obstacle Detection on Highways Using Compressive RBM Road Reconstruction 162–167.

Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V., 2019. Autoaugment: Learning augmentation strategies from data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 113–123.

Cui, P., Athey, S., 2022. Stable learning establishes some common ground between causal inference and machine learning. Nature Machine Intelligence 4, 110–115.

D. Shriver, M.B.D., S. Elbaum, 2021. DNNV: A Framework for Deep Neural Network Verification 33.

Dai, D., Van Gool, L., 2018. Dark Model Adaptation: Semantic Image Segmentation from Daytime to Nighttime. https://doi.org/10.48550/ARXIV.1810.02575

Dar, Y., Muthukumar, V., Baraniuk, R.G., 2021. A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning. arXiv preprint arXiv:2109.02355.

Dauphin, Y., Cubuk, E.D., 2020. Deconstructing the regularization of BatchNorm, in: International Conference on Learning Representations.

de la Fuente Garcia, S., Ritchie, C.W., Luz, S., 2020. Artificial intelligence, speech, and language processing approaches to monitoring Alzheimer's disease: a systematic review. Journal of Alzheimer's Disease 78, 1547–1574.

de Mello, R.F., Ponti, M.A., 2018. Statistical Learning Theory. Rodrigo Fernandes de Mello 75.

De Prado, M.L., 2018. The 10 reasons most machine learning funds fail. The Journal of Portfolio Management 44, 120–133.

Dehak, N., Kenny, P.J., Dehak, R., Dumouchel, P., Ouellet, P., 2011. Front-End Factor Analysis for Speaker Verification. IEEE Transactions on Audio, Speech, and Language Processing 19, 788–798. https://doi.org/10.1109/TASL.2010.2064307

Delpech, E., Laignelet, M., Pimm, C., Raynal, C., Trzos, M., Arnold, A., Pronto, D., 2018. A Real-life, French-accented Corpus of Air Traffic Control Communications, in: Language Resources and Evaluation Conference (LREC). Miyazaki, Japan.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition. Ieee, pp. 248–255.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

DeVries, T., Taylor, G.W., 2017. Dataset augmentation in feature space. arXiv preprint arXiv:1702.05538.

Dhurandhar, A., Sankaranarayanan, K., 2015. Improving classification performance through selective instance completion. Machine Learning 100, 425–447.

Dickinson, E.R., Adelson, J.L., Owen, J., 2012. Gender balance, representativeness, and statistical power in sexuality research using undergraduate student samples. Archives of Sexual Behavior 41, 325–327.

Ding, K., Xu, Z., Tong, H., Liu, H., 2022. Data augmentation for deep graph learning: A survey. arXiv preprint arXiv:2202.08235.

DING, M., WU, B., XU, J., KASULE, A.N., ZUO, H., 2022. Visual inspection of aircraft skin: Automated pixel-level defect detection by instance segmentation. Chinese Journal of Aeronautics 35, 254–264. https://doi.org/10.1016/j.cja.2022.05.002

Ding, Z., Fu, Y., 2017. Deep domain generalization with structured low-rank constraint. IEEE Transactions on Image Processing 27, 304–313.

D'Innocente, A., Caputo, B., 2018. Domain generalization with domain-specific aggregation modules, in: German Conference on Pattern Recognition. Springer, pp. 187–198.

Djolonga, J., Yung, J., Tschannen, M., Romijnders, R., Beyer, L., Kolesnikov, A., Puigcerver, J., Minderer, M., D'Amour, A., Moldovan, D., others, 2021. On robustness and transferability of convolutional neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16458–16468.

Doku, R., Rawat, D.B., Liu, C., 2019. Towards federated learning approach to determine data relevance in big data, in: 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI). IEEE, pp. 184–192.

Dong, X., Taylor, C., Cootes, T., 2019. Small Defect Detection Using Convolutional Neural Network Features and Random Forests: Munich, Germany, September 8-14, 2018, Proceedings, Part IV. pp. 398–412. https://doi.org/10.1007/978-3-030-11018-5_35

Doshi-Velez, F., Kim, B., 2018. Considerations for Evaluation and Generalization in Interpretable Machine Learning, in: Escalante, H.J., Escalera, S., Guyon, I., Baró, X., Güçlütürk, Y., Güçlü, U., van Gerven, M. (Eds.), Explainable and Interpretable Models in Computer Vision and Machine Learning. Springer International Publishing, Cham, pp. 3–17. https://doi.org/10.1007/978-3-319-98131-4_1

Dou, Q., Coelho de Castro, D., Kamnitsas, K., Glocker, B., 2019. Domain Generalization via Model-Agnostic Learning of Semantic Features, in: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d', Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.

Dourado Filho, L.A., Calumby, R.T., 2022. Data Augmentation policies and heuristics effects over dataset imbalance for developing plant identification systems based on Deep Learning: A case study. Revista Brasileira de Computação Aplicada 14, 85–94.

Du, J., 2016. The "weight" of models and complexity. Complexity 21, 21–35.

Duvar, R., Böyük, M., Urhan, O., 2021. A Review on Visual Inspection Methods for Aircraft Maintenance. Journal of Aeronautics and Space Technologies 14, 185–192.

Dziugaite, G.K., Drouin, A., Neal, B., Rajkumar, N., Caballero, E., Wang, L., Mitliagkas, I., Roy, D.M., 2020. In search of robust measures of generalization. Advances in Neural Information Processing Systems 33, 11723–11733.

Dziugaite, G.K., Roy, D.M., 2017. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. arXiv preprint arXiv:1703.11008.

EASA, 2024. EASA Concept Paper: Guidance for Level 1&2 machine learning applications.
EASA, Collins Aerospace, 2023, Formal Methods use for Learning Assurance (ForMuLA), April 2023
EASA, Daedalean, 2024. Concepts of Design Assurance for Neural Networks (CoDANN) II, with Appendix B, January 2024.
ED-79A, 2010. Guidelines for Development of Civil Aircraft and Systems. EUROCAE.
Efron, B., 1992. Bootstrap methods: another look at the jackknife, in: Breakthroughs in Statistics. Springer, pp. 569–593.
Ehrig H., M.B., 1985. Fundamentals of Algebraic Specification 1: Equations and Initial Semantics. Springer.
Elfwing, S., Uchibe, E., Doya, K., 2017. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning. https://doi.org/10.48550/ARXIV.1702.03118
Elliott, R.J., Moore, J.B., Aggoun, Lakhdar., 1995. Hidden Markov models : estimation and control.
Emran, N.A., 2015. Data completeness measures, in: Pattern Analysis, Intelligent Security and the Internet of Things. Springer, pp. 117–130.
Erdogan, A., Ugranli, B., Adalı, E., Sentas, A., Mungan, E., Kaplan, E., Leitner, A., 2019. Real- World Maneuver Extraction for Autonomous Vehicle Validation: A Comparative Study. 2019 IEEE Intelligent Vehicles Symposium (IV) 267–272.
Errattahi, R., El Hannani, A., Ouahmane, H., 2018. Automatic speech recognition errors detection and correction: A review. Procedia Computer Science 128, 32–37.
Eskin, E., 2000. Detecting Errors within a Corpus using Anomaly Detection, in: 1st Meeting of the North American Chapter of the Association for Computational Linguistics.
Even, A., Shankaranarayanan, G., 2007. Utility-driven assessment of data quality. ACM SIGMIS Database: the DATABASE for Advances in Information Systems 38, 75–93.
Evgeniou, T., Pontil, M., Poggio, T., 2000. Regularization networks and support vector machines. Advances in computational mathematics 13, 1–50.
Fallon A., S.Ch., n.d. Detection and Accommodation of Outliers in Normally Distributed Data Sets.
Fang, Y., Guo, X., Chen, K., Zhou, Z., Ye, Q., 2021. Accurate and automated detection of surface knots on sawn timbers using YOLO-V5 model. BioResources 16, 5390.
Fei, G., Wang, S., Liu, B., 2016. Learning cumulatively to become more knowledgeable, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1565–1574.
Feldman, V., Vondrak, J., 2018. Generalization bounds for uniformly stable algorithms. Advances in Neural Information Processing Systems 31.
Feng, S.Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., Hovy, E., 2021. A survey of data augmentation approaches for NLP. arXiv preprint arXiv:2105.03075.
Fernandes, S., Abreu, J., Almeida, P., Santos, R., 2019. A Review of Voice User Interfaces for Interactive TV, in: Abásolo, M.J., Silva, T., González, N.D. (Eds.), Applications and Usability of Interactive TV. Springer International Publishing, Cham, pp. 115–128.
Filzmoser, P., 2004. A MULTIVARIATE OUTLIER DETECTION METHOD.
Fingscheidt, T., Lipinski, D., Bär, A., Bolte, J.-A., 2019. Towards Corner Case Detection for Autonomous Driving IV, 438–445.
Fingscheidt, T., Lipinski, D., Termohlen, J.-A., Breitenstein, J., 2020. Systematization of Corner Cases for Visual Perception in Automated Driving, in Proc. of IV, Las Vegas, NV, USA 986–993.
Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning. PMLR, pp. 1126–1135.

Florek H.-J., D.R., Brunkwall J.,. Orend K.H.,. Handley I.,. Pribble J., 2015. Results from a First-in-Human Trial of a Novel Vascular Sealant. Frontiers in Surgery 2.

Foote, K.D., 2022. The history of machine learning and its convergent trajectory towards AI. Machine Learning and the City: Applications in Architecture and Urban Design 129–142.

Frank, I.E., Todeschini, R., 1994. The data analysis handbook. Elsevier.

Frankle, J., Dziugaite, G.K., Roy, D.M., Carbin, M., 2019. Stabilizing the lottery ticket hypothesis. arXiv preprint arXiv:1903.01611.

Frye, M., Schmitt, R.H., 2020. Structured Data Preparation Pipeline for Machine Learning-Applications in Pro-duction. 17th IMEKO TC 10, 241–246.

G. H, H., 1918. Sir George Stokes and the concept of uniform convergence. Proceedings of the Cambridge Philosophical Society 148–156.

G., V.S., 1979. Critical study of methods for evaluating the quality of machine translation.

G., W., n.d. The formal semantics of programming languages: an introduction. 1993.

Gabreau, C., Gauffriau, A., Grancey, F.D., Ginestet, J.-B., Pagetti, C., 2022. Toward the certification of safety-related systems using ML techniques: the ACAS-Xu experience, in: 11th European Congress on Embedded Real Time Software and Systems (ERTS 2022). Toulouse, France.

Gal, Y., Ghahramani, Z., 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: International Conference on Machine Learning. PMLR, pp. 1050–1059.

Gandomi, A.H., Alavi, A.H., Sahab, M.G., 2010. New formulation for compressive strength of CFRP confined concrete cylinders using linear genetic programming. Materials and Structures 43, 963–983.

Ge, R., Kakade, S.M., Kidambi, R., Netrapalli, P., 2018. Rethinking learning rate schedules for stochastic optimization.

Gehr, M.T., T.,. Mirman, M.,. Drachsler-Cohen, D.,. Tsankov, P.,. Chaudhuri, S.,. Vechev, n.d. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. IEEE Symposium on Security and Privacy 2018.

Geifman, Y., El-Yaniv, R., 2019. Selectivenet: A deep neural network with an integrated reject option, in: International Conference on Machine Learning. PMLR, pp. 2151–2159.

Geifman, Y., El-Yaniv, R., 2017. Selective classification for deep neural networks. Advances in neural information processing systems 30.

Ghifary, M., Balduzzi, D., Kleijn, W.B., Zhang, M., 2016. Scatter component analysis: A unified framework for domain adaptation and domain generalization. IEEE transactions on pattern analysis and machine intelligence 39, 1414–1430.

Glasgow, M., Wei, C., Wootters, M., Ma, T., 2022. Max-Margin Works while Large Margin Fails: Generalization without Uniform Convergence. arXiv preprint arXiv:2206.07892.

Golafshani, E.M., Behnood, A., 2018. Automatic regression methods for formulation of elastic modulus of recycled aggregate concrete. Applied Soft Computing 64, 377–400.

Golbraikh, A., Shen, M., Xiao, Z., Xiao, Y.-D., Lee, K.-H., Tropsha, A., 2003. Rational selection of training and test sets for the development of validated QSAR models. Journal of computer-aided molecular design 17, 241–253.

Gonen, A., Shalev-Shwartz, S., 2017. Fast rates for empirical risk minimization of strict saddle problems, in: Conference on Learning Theory. PMLR, pp. 1043–1063.

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S., Hengel, A. van den, 2019. Memorizing Normality to Detect Anomaly: Memory-Augmented Deep Autoencoder for

Unsupervised Anomaly Detection. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) 1705–1714.

Gong, Z., Zhong, P., Hu, W., 2019. Diversity in machine learning. IEEE Access 7, 64323–64350.

Goodfellow, A., Papernot, N., 2017. The challenge of verification and testing of machine learning.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Goodman, J., 2022. Statistical and Geometric Data Augmentation for Robust Machine Learning-Enabled Decision Support Systems (PhD Thesis). The George Washington University.

Goodman, J., Sarkani, S., Mazzuchi, T., 2022. Distance-based probabilistic data augmentation for synthetic minority oversampling. ACM/IMS Transactions on Data Science (TDS) 2, 1–18.

Google, 2016. Google Auto Waymo Disengagement Report for Autonomous Driving.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., Turner, R.E., 2020. Meta-Learning Probabilistic Inference For Prediction.

Gordon, M.L., Zhou, K., Patel, K., Hashimoto, T., Bernstein, M.S., 2021. The Disagreement Deconvolution: Bringing Machine Learning Performance Metrics In Line With Reality, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, CHI ’21. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445423

Goubault, E., Le Gall, T., Putot, S., 2012. An Accurate Join for Zonotopes, Preserving Affine Input/Output Relations. Electronic Notes in Theoretical Computer Science 287.

Graham, T., Y.,.&. Baldwin, 2014. Testing for significance of increased correlation with human judgment. Conference on Empirical Methods in Natural Language Processing.

Green, J.R., MacDonald, R.L., Jiang, P.-P., Cattiau, J., Heywood, R., Cave, R., Seaver, K., Ladewig, M.A., Tobin, J., Brenner, M.P., others, 2021. Automatic Speech Recognition of Disordered Speech: Personalized Models Outperforming Human Listeners on Short Phrases., in: Interspeech. pp. 4778–4782.

Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A., 2012. A kernel two-sample test. The Journal of Machine Learning Research 13, 723–773.

Gruhl, C., Sick, B., Tomforde, S., 2021. Novelty Detection in Continuously Changing Environments, Future Generation Computer Systems 114, 138–154.

Gülçehre, Ç., Bengio, Y., 2016. Knowledge matters: Importance of prior information for optimization. The Journal of Machine Learning Research 17, 226–257.

Gulcehre, C., Moczulski, M., Visin, F., Bengio, Y., 2016. Mollifying networks. arXiv preprint arXiv:1608.04980.

Guo, W., Lou, Y., Qin, J., Yan, M., 2021. A novel regularization based on the error function for sparse recovery. Journal of Scientific Computing 87, 1–22.

Gupta, N., Mujumdar, S., Patel, H., Masuda, S., Panwar, N., Bandyopadhyay, S., Mehta, S., Guttula, S., Afzal, S., Sharma Mittal, R., others, 2021. Data quality for machine learning tasks, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 4040–4041.

Gupta, V., Rebout, L., Boulianne, G., Ménard, P.-A., Alam, J., 2019. CRIM’s Speech Transcription and Call Sign Detection System for the ATC Airbus Challenge Task, in: Proc. Interspeech 2019. pp. 3018–3022. https://doi.org/10.21437/Interspeech.2019-1131

Hagendorff, T., 2021. Linking Human And Machine Behavior: A New Approach to Evaluate Training Data Quality for Beneficial Machine Learning. Minds and Machines 31, 563–593.

Han, Y., Lam, J.C., Li, V.O., Zhang, Q., 2020. A domain-specific Bayesian deep-learning approach for air pollution forecast. IEEE Transactions on Big Data.

Hanin, B., Rolnick, D., 2019. Complexity of linear regions in deep networks, in: International Conference on Machine Learning. PMLR, pp. 2596–2604.

Hardt, M., Recht, B., Singer, Y., 2016. Train faster, generalize better: Stability of stochastic gradient descent, in: International Conference on Machine Learning. PMLR, pp. 1225–1234.

Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A.K., Davis, L.S., 2016. Learning Temporal Regularity in Video Sequences. https://doi.org/10.48550/ARXIV.1604.04574

Hassaballah, M., Hosny, K.M., 2019. Recent advances in computer vision. Studies in computational intelligence 804, 1–84.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The elements of statistical learning: data mining, inference and prediction, 2nd ed. Springer.

Haughey, D., 2014. A brief history of SMART goals. Project Smart Website. https://www. projectsmart. co. uk/brief-history-of-smart-goals. php.

Hawkins, D.M., 1980. Identification of Outliers, ser. Monographs on applied probability and statistics. Dordrecht: Springer.

He, H., Garcia, E.A., 2009. Learning from imbalanced data. IEEE Transactions on knowledge and data engineering 21, 1263–1284.

Heidecker, F., Breitenstein, J., Rösch, K., Löhdefink, J., Bieshaar, M., Stiller, C., Fingscheidt, T., Sick, B., 2021. An Application-Driven Conceptualization of Corner Cases for Perception in Highly Automated Driving IV, 644–651.

Hein, M., Audibert, J.-Y., 2005. Intrinsic dimensionality estimation of submanifolds in Rd, in: Proceedings of the 22nd International Conference on Machine Learning. pp. 289–296.

Heinrich, B., Hristova, D., Klier, M., Schiller, A., Szubartowicz, M., 2018. Requirements for data quality metrics. Journal of Data and Information Quality (JDIQ) 9, 1–32.

Helmke, H., Kleinert, M., Ohneiser, O., Ehr, H., Shetty, S., 2020. Machine Learning of Air Traffic Controller Command Extraction Models for Speech Recognition Applications, in: 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC). pp. 1–9. https://doi.org/10.1109/DASC50938.2020.9256484

Hendrycks, D., Dietterich, T., 2019. Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261.

Hertzum, M., Jacobsen, N.E., 2001. The evaluator effect: A chilling fact about usability evaluation methods. International journal of human-computer interaction 13, 421–443.

Hess D. E., R.R.F.F.W.E., 2007. Uncertainty Analysis Applied to Feedforward Neural Networks. Applied Simulation Technologies 54.

Heyn, H.-M., Subbiash, P., Linder, J., Knauss, E., Eriksson, O., 2022. Setting AI in context: A case study on defining the context and operational design domain for automated driving.

Hinton, G., Vinyals, O., Dean, J., others, 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 2.

HLEG, 2020. High-Level Expert Group on Artificial Intelligence - Assessment List for Trustworthy Artificial Intelligence (ALTAI). European Commission.

Hodge, V., Austin, J., 2004. A Survey of Outlier Detection Methodologies. Artificial Intelligence Review 22, 85–126. https://doi.org/10.1023/B:AIRE.0000045502.10941.a9

Hofbauer, K., Petrik, S., Hering, H., 2008. The ATCOSIM Corpus of Non-Prompted Clean Air Traffic Control Speech., in: LREC. Citeseer.

Hoffer, E., Hubara, I., Ailon, N., 2016. Deep unsupervised learning through spatial contrasting. arXiv preprint arXiv:1610.00243.

Hoffer, E., Hubara, I., Soudry, D., 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. Advances in neural information processing systems 30.

Ho-Phuoc, T., 2018. CIFAR10 to compare visual recognition performance between deep neural networks and humans. arXiv preprint arXiv:1811.07270.

Howard, J., Ruder, S., 2018. Universal Language Model Fine-tuning for Text Classification. https://doi.org/10.48550/ARXIV.1801.06146

Hsu, D., Ji, Z., Telgarsky, M., Wang, L., 2021. Generalization bounds via distillation. arXiv preprint arXiv:2104.05641.

Hsu, W.-N., Bolte, B., Tsai, Y.-H.H., Lakhotia, K., Salakhutdinov, R., Mohamed, A., 2021. HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. https://doi.org/10.48550/ARXIV.2106.07447

Hu, S., Zhang, K., Chen, Z., Chan, L., 2020. Domain generalization via multidomain discriminant analysis, in: Uncertainty in Artificial Intelligence. PMLR, pp. 292–302.

Hu, X., Chu, L., Pei, J., Liu, W., Bian, J., 2021. Model complexity of deep learning: A survey. Knowledge and Information Systems 63, 2585–2619.

Hu, X., Liu, W., Bian, J., Pei, J., 2020. Measuring model complexity of neural networks with curve activation functions, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1521–1531.

Hu, Y., Jiang, M., Underwood, T., Downie, J.S., 2020. Improving digital libraries' provision of digital humanities datasets: A case study of htrc literature dataset, in: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020. pp. 405–408.

Huan, D., Z.,. Tsui-Wei W.,. Pin-Yu, C.,. Cho-Jui, H.,. Luca, 2018. Efficient neural network robustness certification with general activation functions 31.

Huang X., W.M., Kwiatkowska M.,. Wang S., 2016. Safety Verification of Deep Neural Networks. Computer Aider Vision.

Hubens, N., Mancas, M., Gosselin, B., Preda, M., Zaharia, T., 2021. One-cycle pruning: Pruning convnets under a tight training budget. arXiv preprint arXiv:2107.02086.

Hyontai, S., 2018. Performance of machine learning algorithms and diversity in data, in: MATEC Web of Conferences. EDP Sciences, p. 04019.

IEC 31010, 2019. Risk management — Risk assessment techniques. ISO/TC 262 Risk management.

Ilse, M., Tomczak, J.M., Louizos, C., Welling, M., 2020. Diva: Domain invariant variational autoencoders, in: Medical Imaging with Deep Learning. PMLR, pp. 322–348.

Inoue, T., Choudhury, S., De Magistris, G., Dasgupta, S., 2018. Transfer learning from synthetic to real images using variational autoencoders for precise position detection, in: 2018 25th IEEE International Conference on Image Processing (ICIP). IEEE, pp. 2725–2729.

INSEE, 2024. Institut national de la statistique et des études économiques, INSEE. 2024. Population par sexe, données annuelles de 1990 à 2024. [WWW Document]. URL https://www.insee.fr/fr/statistiques/2381466

INSEE, 2023. Institut national de la statistique et des études économiques, INSEE . 2023. L'essentiel sur... les immigrés et les étrangers. [WWW Document]. URL https://www.insee.fr/fr/statistiques/3633212

INSEE, 2021. Institut national de la statistique et des études économiques, INSEE. 2021. Emploi, chômage, revenus du travail Édition 2021. [WWW Document]. URL https://www.insee.fr/fr/statistiques/5391964?sommaire=5392045

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning. PMLR, pp. 448–456.

Iphar, C., Napoli, A., Ray, C., 2015. Detection of false AIS messages for the improvement of maritime situational awareness, in: Oceans 2015-Mts/Ieee Washington. IEEE, pp. 1–7.

Irie, K., Tüske, Z., Alkhouli, T., Schlüter, R., Ney, H., others, 2016. LSTM, GRU, highway and a bit of attention: an empirical overview for language modeling in speech recognition, in: Interspeech. pp. 3519–3523.

ISO, 2011. ISO 14155: Clinical investigation of medical devices for human subjects - Good clinical practice. ISO.

Isobe T, M.K., Morishima M, Yoshitani F, Koizumi N., 1996. Voice-activated home banking system and its field trial. Proceedings Fourth International Conference on Spoken Language 3.

ISO/DIS, n.d. ISO/DIS 5259-2 Artificial intelligence — Data quality for analytics and machine learning (ML) — Part 2: Data quality measures. ISO/DIS.

ISO/IEC, 2022a. ISO/IEC 22989, Information technology — Artificial Intelligence (AI) — Artificial intelligence concepts and terminology. ISO/IEC.

ISO/IEC, 2022b. ISO/IEC DIS 24029-2, Information technology — Artificial Intelligence (AI) — Assessment of the robustness of neural networks — Part 2: Methodology for the use of formal methods. ISO/IEC.

ISO/IEC, n.d. ISO/IEC TR 24029-1, Information technology — Artificial Intelligence (AI) — Assessment of the robustness of neural networks — Part 1: Overview. ISO/IEC.

ISO/IEC 14496, 2019. Information technology — Coding of audio-visual objects — Part 3: Audio.

ISO/IEC 22989, 2022. Information technology — Artificial intelligence — Artificial intelligence concepts and terminology. ISO/IEC JTC 1/SC 42 Artificial intelligence.

ISO/IEC 25012, 2008. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Data quality model. ISO/IEC JTC 1/SC 7 Software and systems engineering.

ISO/IEC 27000, 2018. Information technology — Security techniques — Information security management systems — Overview and vocabulary. ISO/IEC JTC 1/SC 27 Information security, cybersecurity and privacy protection.

ISO/IEC CD 5259-2, 202X. Artificial intelligence — Data quality for analytics and machine learning (ML) — Part 2: Data quality measures (under development). ISO/IEC JTC 1/SC 42 Artificial intelligence.

ISO/IEC DIS 25059, 202X. Software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality model for AI systems (under development). ISO/IEC JTC 1/SC 42 Artificial intelligence.

ISO/IEC DIS 42001, 202X. Information technology — Artificial intelligence — Management system. ISO/IEC JTC 1/SC 42 Artificial intelligence.

ISO/IEC TR 24027, 2021. Information technology — Artificial intelligence (AI) — Bias in AI systems and AI aided decision making. ISO/IEC JTC 1/SC 42 Artificial intelligence.

ISO/IEC/IEEE, 2017. ISO/IEC/IEEE 24765: Systems and software engineering — Vocabulary.

ISO/IEC/IEEE, 2013. ISO/IEC/IEEE 29119-3 Software and systems engineering - Software testing - Part 3: Test documentation. ISO/IEC/IEEE.

Issa, S., Adekunle, O., Hamdi, F., Cherfi, S.S.-S., Dumontier, M., Zaveri, A., 2021. Knowledge graph completeness: A systematic literature review. IEEE Access 9, 31322–31339.

Ito, M., 2021. ODD description methods for automated driving vehicle and verifiability for safety. jucs 27, 796–810. https://doi.org/10.3897/jucs.72333

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G., 2018. Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407.

J. Thiyagalingam, T.H., M. Shankar, G. Fox, 2022. Scientific machine learning benchmarks 4.

Jabbar, H., Khan, R.Z., 2015. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). Computer Science, Communication and Instrumentation Devices 70.

Jain, A.K., Chandrasekaran, B., 1982. 39 Dimensionality and sample size considerations in pattern recognition practice, in: Classification Pattern Recognition and Reduction of Dimensionality, Handbook of Statistics. Elsevier, pp. 835–855. https://doi.org/10.1016/S0169-7161(82)02042-2

Jatzkowski, I., Wilke, D., Maurer, M., 2018. A Deep-Learning Approach for the Detection of Overexposure in Automotive Camera Images. 2018 21st International Conference on Intelligent Transportation Systems (ITSC) 2030–2035.

Javed, R., Rahim, M.S.M., Saba, T., Rehman, A., 2020. A comparative study of features selection for skin lesion detection from dermoscopic images. Network Modeling Analysis in Health Informatics and Bioinformatics 9, 1–13.

Jesmeen, M., Hossen, J., Sayeed, S., Ho, C., Tawsif, K., Rahman, A., Arif, E., 2018. A survey on cleaning dirty data using machine learning paradigm for big data analytics. Indonesian Journal of Electrical Engineering and Computer Science 10, 1234–1243.

Ji, Y., Li, H., Edwards, A.V., Papaioannou, J., Ma, W., Liu, P., Giger, M.L., 2019. Independent validation of machine learning in diagnosing breast Cancer on magnetic resonance imaging within a single institution. Cancer Imaging 19, 1–11.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., Bengio, S., 2020. Fantastic Generalization Measures and Where to Find Them.

Jin, D., Jin, Z., Zhou, J.T., Szolovits, P., 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment, in: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 8018–8025.

Jin, P., Lu, L., Tang, Y., Karniadakis, G.E., 2020. Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness. Neural Networks 130, 85–99.

Jin, X., Lan, C., Zeng, W., Chen, Z., 2020. Feature alignment and restoration for domain generalization and adaptation. arXiv preprint arXiv:2006.12009.

Jing, L., Tian, Y., 2020. Self-supervised visual feature learning with deep neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence 43, 4037–4058.

Jocher, G., Ayush Chaurasia, Stoken, A., Borovec, J., NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, Imyhxy, Lorna, 曾逸夫(Zeng Yifu), Wong, C., Abhiram V, Montes, D., Zhiqiang Wang, Fati, C., Jebastin Nadar, Laughing, UnglvKitDe, Sonck, V., Tkianai, YxNONG, Skalski, P., Hogan, A., Dhruv Nair, Strobel, M., Jain, M., 2022. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. https://doi.org/10.5281/ZENODO.3908559

Jocher, G., Chaurasia, A., Qiu, J., 2023. YOLO by Ultralytics. GitHub. GitHub, January.

Jocher, G., Stoken, A., Borovec, J., Changyu, L., Hogan, A., Diaconu, L., Poznanski, J., Yu, L., Rai, P., Ferriday, R., others, 2020. ultralytics/yolov5: v3. 0. Zenodo.

Jolliffe, I.T., Cadima, J., 2016. Principal component analysis: a review and recent developments. Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences 374, 20150202.

J.R, T., 1999. Introduction to the analysis of measurement error.

Juba, B., Le, H.S., 2019. Precision-recall versus accuracy and the role of large data sets, in: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 4039–4048.

Juddoo, S., 2015. Overview of data quality challenges in the context of Big Data, in: 2015 International Conference on Computing, Communication and Security (ICCCS). IEEE, pp. 1–9.

Juddoo, S., George, C., 2020. A qualitative assessment of machine learning support for detecting data completeness and accuracy issues to improve data analytics in big data for the healthcare industry, in: 2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM). IEEE, pp. 58–66.

Julian, K.D., Kochenderfer, M.J., Owen, M.P., 2019. Deep neural network compression for aircraft collision avoidance systems. Journal of Guidance, Control, and Dynamics 42, 598–608.

K. Julian, M.K., 2019. Guaranteeing safety for neural network-based aircraft collision avoidance systems. IEEE/AIAA 38th Digital Avionics Systems Conference (DASC).

K. Leino, M.F., 2021. Relaxing Local Robustness.

Kähler, F., Schmedemann, O., Schüppstuhl, T., 2022. Anomaly detection for industrial surface inspection: application in maintenance of aircraft components. Procedia CIRP 107, 246–251. https://doi.org/10.1016/j.procir.2022.05.197

Kakade, S.M., Sridharan, K., Tewari, A., 2008. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. Advances in neural information processing systems 21.

Kalimeris, D., Kaplun, G., Nakkiran, P., Edelman, B., Yang, T., Barak, B., Zhang, H., 2019. SGD on Neural Networks Learns Functions of Increasing Complexity, in: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d', Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.

Kamikubo, R., Wang, L., Marte, C., Mahmood, A., Kacorri, H., 2022. Data Representativeness in Accessibility Datasets: A Meta-Analysis. arXiv preprint arXiv:2207.08037.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1725–1732.

Karras, T., Laine, S., Aila, T., 2019. A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4401–4410.

Katsaggelos, A.K., Tsaftaris, S.A., Yuan, J., Jiang, F., 2010. Video Anomaly Detection in Spatiotemporal Context, in Proc. of ICIP, Hong Kong 705–708.

Katz, G., Barrett, C., Dill, D., Julian, K., Kochenderfer, M., 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. https://doi.org/10.48550/ARXIV.1702.01135

Katz G., K.M., Barrett C.,. Dill D.,. Julian K., 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. Computer Aided Verification.

Kaur, H., Pannu, H.S., Malhi, A.K., 2019. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. ACM Computing Surveys (CSUR) 52, 1–36.

Kawaguchi, K., Bengio, Y., Kaelbling, L., 2023. Generalization in Deep Learning, in: Mathematical Aspects of Deep Learning. Cambridge University Press, pp. 112–148. https://doi.org/10.1017/9781009025096.003

Kawaguchi, K., Bengio, Y., Kaelbling, L., 2022. Generalization in Deep Learning, in: Mathematical Aspects of Deep Learning. Cambridge University Press, pp. 112–148. https://doi.org/10.1017/9781009025096.003

Kawaguchi, K., Kaelbling, L.P., Bengio, Y., 2017. Generalization in deep learning. arXiv preprint arXiv:1710.05468.

Kay, M., Patel, S.N., Kientz, J.A., 2015. How Good is 85%? A Survey Tool to Connect Classifier Evaluation to Acceptability of Accuracy, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI '15. Association for Computing Machinery, New York, NY, USA, pp. 347–356. https://doi.org/10.1145/2702123.2702603

Kayed, M., Anter, A., Mohamed, H., 2020. Classification of garments from fashion MNIST dataset using CNN LeNet-5 architecture, in: 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE). IEEE, pp. 238–243.

Kendall, A., Badrinarayanan, V., Cipolla, R., 2015. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding.

Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P., 2016. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836.

Keskes, N., Fakhfakh, S., Kanoun, O., Derbel, N., 2022. Representativeness consideration in the selection of classification algorithms for the ECG signal quality assessment. Biomedical Signal Processing and Control 76, 103686.

Khalid, S., Khalil, T., Nasreen, S., 2014. A survey of feature selection and feature extraction techniques in machine learning, in: 2014 Science and Information Conference. IEEE, pp. 372–378.

Khosla, C., Saini, B.S., 2020. Enhancing performance of deep learning models with different data augmentation techniques: A survey, in: 2020 International Conference on Intelligent Engineering and Management (ICIEM). IEEE, pp. 79–85.

Kiela, D., Bartolo, M., Nie, Y., Kaushik, D., Geiger, A., Wu, Z., Vidgen, B., Prasad, G., Singh, A., Ringshia, P., others, 2021. Dynabench: Rethinking benchmarking in NLP. arXiv preprint arXiv:2104.14337.

Kim, J., Feldt, R., Yoo, S., 2019. Guiding Deep Learning System Testing Using Surprise Adequacy, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). pp. 1039–1049. https://doi.org/10.1109/ICSE.2019.00108

Kim, J., Ju, J., Feldt, R., Yoo, S., 2020. Reducing DNN labelling cost using surprise adequacy: an industrial case study for autonomous driving, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM. https://doi.org/10.1145/3368089.3417065

Kim, S., Yoo, S., 2020. Evaluating Surprise Adequacy for Question Answering, in: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW'20. Association for Computing Machinery, New York, NY, USA, pp. 197–202. https://doi.org/10.1145/3387940.3391465

Kim, T.S., Jones, J., Peven, M., Xiao, Z., Bai, J., Zhang, Y., Qiu, W., Yuille, A., Hager, G.D., 2021. Daszl: Dynamic action signatures for zero-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 1817–1826.

Knight, E.C., Poo Hernandez, S., Bayne, E.M., Bulitko, V., Tucker, B.V., 2020. Pre-processing spectrogram parameters improve the accuracy of bioacoustic classification using convolutional neural networks. Bioacoustics 29, 337–355.

Kocour, M., Veselý, K., Blatt, A., Gomez, J.Z., Igor Szöke, Černocký, J., Klakow, D., Motlicek, P., 2021. Boosting of Contextual Information in ASR for Air-Traffic Call-Sign Recognition, in: Proc. Interspeech 2021. pp. 3301–3305. https://doi.org/10.21437/Interspeech.2021-1619

Koehrsen, W., 2018. Overfitting vs. underfitting: A complete example. Towards Data Science.

Koenecke, A., Choi, A.S.G., Mei, K., Schellmann, H., Sloane, M., 2024. Careless Whisper: Speech-to-Text Hallucination Harms. arXiv preprint arXiv:2402.08021.

Kohonen, R.L., T.,. Barna, G.,. Chrisley, 1988. Statistical pattern recognition with neural networks: benchmarking studies Proceedings of International Conference on Neural Networks (ICNN'88).

Kohut, A., Keeter, S., Doherty, C., Dimock, M., Christian, L., 2012. Assessing the representativeness of public opinion surveys. Washington, DC: Pew Research Center.

Konieczka P., N.J., 2007. Evaluation and quality control of analytical measurements.

Koopman, P., Kane, A., Black, J., 2019. Credible Autonomy Safety Argumentation in Proc. of the 27th Safety-Critical Systems Symposium, Bristol, UK: Safety-Critical Systems Club.

Kopf, L.M., Huh-Yoo, J., 2023. A User-Centered Design Approach to Developing a Voice Monitoring System for Disorder Prevention. Journal of Voice 37, 48–59. https://doi.org/10.1016/j.jvoice.2020.10.015

Kostakos, V., Musolesi, M., 2017. Avoiding pitfalls when using machine learning in HCI studies. interactions 24, 34–37.

Krueger, D., Ballas, N., Jastrzebski, S., Arpit, D., Kanwal, M.S., Maharaj, T., Bengio, E., Fischer, A., Courville, A., 2017. Deep nets don't learn via memorization.

Kukačka, J., Golkov, V., Cremers, D., 2017. Regularization for deep learning: A taxonomy. arXiv preprint arXiv:1710.10686.

Kumar, P., Bhatnagar, R., Gaur, K., Bhatnagar, A., 2021. Classification of imbalanced data: review of methods and applications, in: IOP Conference Series: Materials Science and Engineering. IOP Publishing, p. 012077.

Kumar, V., Banerjee, A., Chandola, V., 2009. Anomaly Detection: A Survey, ACM Computing Surveys 41, 1–58.

Kuzborskij, I., Lampert, C., 2018. Data-dependent stability of stochastic gradient descent, in: International Conference on Machine Learning. PMLR, pp. 2815–2824.

Läkemedelsverket, 2009. Proposal for guidelines regarding classification of software based information systems used in health care.

Lakshminarayanan, B., Pritzel, A., Blundell, C., 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.

Lamel L, R.S., Gauvain JL, Bennacef SK, Devillers L, Foukia S, Gangolf J.J., 1996. Field trials of a telephone service for rail travel information. Third IEEE Workshop on Interactive Voice Technology for Telecommunications Applications. IEEE.

Laranjeiro, N., Soydemir, S.N., Bernardino, J., 2015. A survey on data quality: classifying poor data, in: 2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC). IEEE, pp. 179–188.

Laskar, M.T.R., Hoque, E., Huang, J.X., 2022. Domain Adaptation with Pre-trained Transformers for Query-Focused Abstractive Text Summarization. Computational Linguistics 48, 279–320.

Le, Q.V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Ng, A.Y., 2011. On optimization methods for deep learning, in: ICML.

Leavy, S., 2018. Gender bias in artificial intelligence: The need for diversity and gender theory in machine learning, in: Proceedings of the 1st International Workshop on Gender Equality in Software Engineering. pp. 14–16.

LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R., 2012. Efficient BackProp, in: Montavon, G., Orr, Geneviève B., Müller, K.-R. (Eds.), Neural Networks: Tricks of the Trade: Second Edition. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 9–48. https://doi.org/10.1007/978-3-642-35289-8_3

Ledo, D., Houben, S., Vermeulen, J., Marquardt, N., Oehlberg, L., Greenberg, S., 2018. Evaluation Strategies for HCI Toolkit Research, in: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI ’18. Association for Computing Machinery, New York, NY, USA, pp. 1–17. https://doi.org/10.1145/3173574.3173610

Lee, D.H., Yoon, Y.J., Kang, S.J., Ko, S.J., 2014. Correction of the overexposed region in digital color image. IEEE Transactions on Consumer Electronics 60, 173–178. https://doi.org/10.1109/TCE.2014.6851990

Lee, E.H., Cherkassky, V., 2022. VC Theoretical Explanation of Double Descent. arXiv preprint arXiv:2205.15549.

Lee, J., Yoon, Y., Kwon, J., 2020. Generative Adversarial Network for Class-Conditional Data Augmentation. Applied Sciences 10, 8415.

Lee, L.C., Jemain, A.A., 2021. On overview of PCA application strategy in processing high dimensionality forensic data. Microchemical Journal 169, 106608.

Lee, S., Cha, S., Lee, D., Oh, H., 2020. Effective white-box testing of deep neural networks with adaptive neuron-selection strategy, in: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. Presented at the ISSTA ’20: 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, ACM, Virtual Event USA, pp. 165–176. https://doi.org/10.1145/3395363.3397346

Lei, M., Felix, J.-X., Jiyuan, S., Chunyang, C., Ting, S., Fuyuan, Z., Minhui, X., Bo, L., Li, L., Yang, L., others, 2018. Deepgauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems. arXiv preprint arXiv:1803.07519.

Lei, S., He, F., Yuan, Y., Tao, D., 2022. Understanding deep learning via decision boundary. arXiv preprint arXiv:2206.01515.

Lei, Y., Bao, S., Perez, M.A., Wang, J., 2017. Enhancing generalization of visuomotor adaptation by inducing use-dependent learning. Neuroscience 366, 184–195.

Leino K., F.M., Wang Z., 2021. Globally-Robust Neural Networks 139.

Lemley, J., Bazrafkan, S., Corcoran, P., 2017. Smart augmentation learning an optimal data augmentation strategy. Ieee Access 5, 5858–5869.

Li, B., Jin, J., Zhong, H., Hopcroft, J.E., Wang, L., 2022. Why robust generalization in deep learning is difficult: Perspective of expressive power. arXiv preprint arXiv:2205.13863.

Li, B., Wu, F., Lim, S.-N., Belongie, S., Weinberger, K.Q., 2021. On feature normalization and data augmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12383–12392.

Li, C.H., Lee, C., 1993. Minimum cross entropy thresholding. Pattern recognition 26, 617–625.

Li, D., Gouk, H., Hospedales, T., 2022. Finding lost DG: Explaining domain generalization via model complexity. arXiv preprint arXiv:2202.00563.

Li, D., Yang, Y., Song, Y.-Z., Hospedales, T., 2020. Sequential learning for domain generalization, in: European Conference on Computer Vision. Springer, pp. 603–619.

Li, D., Yang, Y., Song, Y.-Z., Hospedales, T., 2018. Learning to generalize: Meta-learning for domain generalization, in: Proceedings of the AAAI Conference on Artificial Intelligence.

Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T., 2018. Visualizing the loss landscape of neural nets. Advances in neural information processing systems 31.

Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X., Zhang, C., 2021. CleanML: a study for evaluating the impact of data cleaning on ML classification tasks, in: 2021 IEEE 37th International Conference on Data Engineering (ICDE). IEEE, pp. 13–24.

Li, Q., Nourbakhsh, A., Shah, S., Liu, X., 2017. Real-Time Novel Event Detection from Social Media. 2017 IEEE 33rd International Conference on Data Engineering (ICDE) 1129–1139.

Li, X., Fang, M., Li, H., Wu, J., 2020. Zero shot learning based on class visual prototypes and semantic consistency. Pattern Recognition Letters 135, 368–374.

Li, X., Lu, J., Wang, Z., Haupt, J., Zhao, T., 2018. On tighter generalization bound for deep neural networks: Cnns, resnets, and beyond. arXiv preprint arXiv:1806.05159.

Li, Y., Chen, C.P., Zhang, T., 2022. A survey on siamese network: Methodologies, applications, and opportunities. IEEE Transactions on Artificial Intelligence 3, 994–1014.

Li, Y., Hao, Z., Lei, H., 2016. Survey of convolutional neural network. Journal of Computer Applications 36, 2508.

Li, Z., Liu, L., Dong, C., Shang, J., 2020. Overfitting or Underfitting? Understand Robustness Drop in Adversarial Training. arXiv preprint arXiv:2010.08034.

Liang, S., Li, Y., Srikant, R., 2017. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. https://doi.org/10.48550/ARXIV.1706.02690

Liang, T., Poggio, T., Rakhlin, A., Stokes, J., 2019. Fisher-rao metric, geometry, and complexity of neural networks, in: The 22nd International Conference on Artificial Intelligence and Statistics. PMLR, pp. 888–896.

Lin, S., Zhang, J., 2019. Generalization bounds for convolutional neural networks. arXiv preprint arXiv:1910.01487.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P., 2014. Microsoft COCO: Common Objects in Context. https://doi.org/10.48550/ARXIV.1405.0312

Lin, Y., 2021. Spoken Instruction Understanding in Air Traffic Control: Challenge, Technique, and Application. Aerospace 8. https://doi.org/10.3390/aerospace8030065

Lin, Y., Li, Q., Yang, B., Yan, Z., Tan, H., Chen, Z., 2021a. Improving speech recognition models with small samples for air traffic control systems. Neurocomputing 445, 287–297. https://doi.org/10.1016/j.neucom.2020.08.092

Lin, Y., Tan, X., Yang, B., Yang, K., Zhang, J., Yu, J., 2019. Real-time controlling dynamics sensing in air traffic system. Sensors 19, 679.

Lin, Y., Yang, B., Li, L., Guo, D., Zhang, J., Chen, H., Zhang, Y., 2021b. ATCSpeechNet: A multilingual end-to-end speech recognition framework for air traffic control systems. Applied Soft Computing 112, 107847. https://doi.org/10.1016/j.asoc.2021.107847

Lis, K., Nakka, K., Fua, P., Salzmann, M., 2019. Detecting the Unexpected via Image Resynthesis. https://doi.org/10.48550/ARXIV.1904.07595

Liu, C., Talaei-Khoei, A., Zowghi, D., Daniel, J., 2017. Data completeness in healthcare: a literature survey. Pacific Asia Journal of the Association for Information Systems 9, 5.
Liu, N., Ma, X., Xu, Z., Wang, Y., Tang, J., Ye, J., 2020. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates, in: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 4876–4883.
Liu, Q., Dou, Q., Heng, P.-A., 2020. Shape-aware meta-learning for generalizing prostate MRI segmentation to unseen domains, in: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp. 475–485.
Liu, R., Gillies, D.F., 2016. Overfitting in linear feature extraction for classification of high-dimensional image data. Pattern Recognition 53, 73–86.
Liu, S., 2021. Learning sparse neural networks for better generalization, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. pp. 5190–5191.
Liu, S., Papailiopoulos, D., Achlioptas, D., 2020. Bad global minima exist and sgd can reach them. Advances in Neural Information Processing Systems 33, 8543–8552.
Liu, T., Lugosi, G., Neu, G., Tao, D., 2017. Algorithmic Stability and Hypothesis Complexity, in: Precup, D., Teh, Y.W. (Eds.), Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, pp. 2159–2167.
Liu, W., Luo, W., Lian, D., Gao, S., 2017. Future Frame Prediction for Anomaly Detection – A New Baseline. https://doi.org/10.48550/ARXIV.1712.09867
Liu, X., Nourbakhsh, A., Li, Q., Fang, R., Shah, S., 2015. Real-time Rumor Debunking on Twitter. Proceedings of the 24th ACM International on Conference on Information and Knowledge Management.
Liu, Z., Lin, Y., Sun, M., 2020. Document Representation, in: Representation Learning for Natural Language Processing. Springer Singapore, Singapore, pp. 91–123. https://doi.org/10.1007/978-981-15-5573-2_5
Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T., 2018. Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270.
LNE, 2021. REFERENTIEL DE CERTIFICATION DE PROCESSUS POUR L'IA - Conception, développement, évaluation et maintien en conditions opérationnelles. LNE - Laboratoire national de métrologie et d'essais.
Logozzo F., F.M., 2008. Pentagons: a weakly relational abstract domain for the efficient validation of array accesses. Proceedings of the 2008 ACM symposium on Applied computing.
Loosli, G., Canu, S., Bottou, L., 2007. Training invariant support vector machines using selective sampling. Large scale kernel machines 2.
Lopes, R.G., Yin, D., Poole, B., Gilmer, J., Cubuk, E.D., 2019. Improving robustness without sacrificing accuracy with patch gaussian augmentation. arXiv preprint arXiv:1906.02611.
Lopez, D.M., Johnson, T.T., Bak, S., Tran, H.-D., Hobbs, K.L., 2021. Neural Network Verification Methods for Closed-Loop ACAS Xu Properties.
Lotter, W., Kreiman, G., Cox, D., 2016. Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. https://doi.org/10.48550/ARXIV.1605.08104
Lowerre, B.T., 1976. The Harpy speech recognition system (PhD Thesis). Carnegie Mellon University, Pennsylvania.
Lu, J.H., Callahan, A., Patel, B.S., Morse, K.E., Dash, D., Shah, N.H., 2021. Low adherence to existing model reporting guidelines by commonly used clinical prediction models. MedRXiv.

Lugosch, L., Ravanelli, M., Ignoto, P., Tomar, V.S., Bengio, Y., 2019. Speech Model Pre-Training for End-to-End Spoken Language Understanding, in: Proc. Interspeech 2019. pp. 814–818. https://doi.org/10.21437/Interspeech.2019-2396

Lust, J., Condurache, A.P., 2020. A survey on assessing the generalization envelope of deep neural networks: predictive uncertainty, out-of-distribution and adversarial samples. arXiv preprint arXiv:2008.09381.

Lyu, S.-H., Wang, L., Zhou, Z.-H., 2022. Improving generalization of deep neural networks by leveraging margin distribution. Neural Networks 151, 48–60.

M. Bouton, M.K., J. Tumova, 2020. Point-based methods for model checking in partially observable Markov decision processes. AAAI Conference on Artificial Intelligence 24.

M. D. Zeiler, R.F., n.d. Visualizing and Understanding Convolutional Networks 2014.

M. T. Ribeiro, C.G., S. Singh, 2016. “Why Should I Trust You?”: Explaining the Predictions of Any Classifier.

Ma, H., King, I., Lyu, M.R., 2007. Effective missing data prediction for collaborative filtering, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 39–46.

Maass, W., 1995. Vapnik-Chervonenkis dimension of neural nets. The handbook of brain theory and neural networks 1000–1003.

Macgregor, C.J., Evans, D.M., Pocock, M.J., 2017. Estimating sampling completeness of interactions in quantitative bipartite ecological networks: incorporating variation in species’ specialisation. Biorxiv 195917.

Madrigal, A., 2017. Inside Waymo’s Secret World for Training Self-Driving Cars. The Atlantic.

Mahanti, R., 2019. Data quality: dimensions, measurement, strategy, management, and governance. ASQ Quality Press.

Maier-Hein, A.&, L.,. Eisenmann, M.,. Reinke, A.,. Onogur, S.,. Stankovic, M.,. Scholz, P.,. Arbel, T.,. Bogunovic, H.,. Bradley, A.P.,. Carass et al., 2018. Why rankings of biomedical image analysis competitions should be interpreted with care 9.

Maji, S., Berg, A.C., 2009. Max-margin additive classifiers for detection, in: 2009 IEEE 12th International Conference on Computer Vision. IEEE, pp. 40–47.

Maleki, F., Ovens, K., Gupta, R., Reinhold, C., Spatz, A., Forghani, R., 2022. Generalizability of Machine Learning Models: Quantitative Evaluation of Three Methodological Pitfalls. arXiv preprint arXiv:2202.01337.

Malik, M., Khanam, R., 2022. The state of the art on ASR systems and feature extraction technique, in: 7th International Conference on Computing in Engineering & Technology (ICCET 2022). pp. 67–72. https://doi.org/10.1049/icp.2022.0594

Malisiewicz, T., Gupta, A., Efros, A.A., 2011. Ensemble of exemplar-svms for object detection and beyond, in: 2011 International Conference on Computer Vision. IEEE, pp. 89–96.

Mani, S., Sankaran, A., Tamilselvam, S., Sethi, A., 2019. Coverage testing of deep learning models using dataset characterization. arXiv preprint arXiv:1911.07309.

Maniyar, U., Deshmukh, A.A., Dogan, U., Balasubramanian, V.N., others, 2020. Zero shot domain generalization. arXiv preprint arXiv:2008.07443.

Manna Z., W.R., 1993. The deductive foundations of computer programming - The logical basis for computer programming.

Manzanas Lopez, D., Johnson, T.T., Bak, S., Tran, H.-D., Hobbs, K.L., 2023. Evaluation of Neural Network Verification Methods for Air-to-Air Collision Avoidance. Journal of Air Transportation 31, 1–17. https://doi.org/10.2514/1.D0255

Mas' ud, M.Z., Sahib, S., Abdollah, M.F., Selamat, S.R., Yusof, R., 2014. Analysis of features selection and machine learning classifier in android malware detection, in: 2014 International Conference on Information Science & Applications (ICISA). IEEE, pp. 1–5.

Masters, D., Luschi, C., 2018. Revisiting Small Batch Training for Deep Neural Networks.

Mathet, Y., Widlöcher, A., Metivier, J.-P., 2015. The Unified and Holistic Method Gamma for Inter-Annotator Agreement Measure and Alignment 41.

Mathieu, M., Cobib_task3_2.textuprie, C., LeCun, Y., 2015. Deep multi-scale video prediction beyond mean square error. https://doi.org/10.48550/ARXIV.1511.05440

Matousek, M., El-Zohairy, M., Al-Momani, A., Kargl, F., Bösch, C., 2019. Detecting Anomalous Driving Behavior using Neural Networks. 2019 IEEE Intelligent Vehicles Symposium (IV) 2229–2235.

McAllester, D., Akinbiyi, T., 2013. Pac-bayesian theory. Empirical inference 95–103.

McAllester, D.A., 2003. PAC-Bayesian stochastic model selection. Machine Learning 51, 5–21.

McAllester, D.A., 1998. Some pac-bayesian theorems, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory. pp. 230–234.

McCloskey, M., Cohen, N.J., 1989. Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of Learning and Motivation. Elsevier, pp. 109–165.

McDermott, M.B., Wang, S., Marinsek, N., Ranganath, R., Foschini, L., Ghassemi, M., 2021. Reproducibility in machine learning for health research: Still a ways to go. Science Translational Medicine 13, eabb1655.

Mei, G., Guo, Z., Liu, S., Li, P., 2019. SGNN: A Graph Neural Network Based Federated Learning Approach by Hiding Structure. pp. 2560–2568. https://doi.org/10.1109/BigData47090.2019.9005983

Meir, R., Fontanari, J.F., 1993. Data compression and prediction in neural networks. Physica A: Statistical Mechanics and its Applications 200, 644–654.

Mentaschi, L., Besio, G., Cassola, F., Mazzino, A., 2013. Problems in RMSE-based wave model validations. Ocean Modelling 72, 53–58.

Miller, R.G., 1974. The jackknife-a review. Biometrika 61, 1–15.

Milletari, F., Navab, N., Ahmadi, S.-A., 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation, in: 2016 Fourth International Conference on 3D Vision (3DV). IEEE, pp. 565–571.

Miné, A., 2006. The octagon abstract domain. Higher-Order and Symbolic Computation 19.

Mirman M., V.M., Gehr T., 2018. Differentiable Abstract Interpretation for Provably Robust Neural Networks. Proceedings of the 35th International Conference on Machine Learning.

Mittal, Puneet, Sharma, S., 2023. Automatic Speech Recognition Models, Tools, and Techniques: A Systematic Review, in: Kumar, L.A., Renuka, D.K., Geetha, S. (Eds.), Deep Learning Research Applications for Natural Language Processing. IGI Global, pp. 18–40.

Mohri, M., Pereira, F., Riley, M., 2002. Weighted finite-state transducers in speech recognition. Computer Speech & Language 16, 69–88. https://doi.org/10.1006/csla.2001.0184

Mohseni, S., Pitale, M., Singh, V., Wang, Z., 2019. Practical Solutions for Machine Learning Safety in Autonomous Vehicles. CoRR abs/1912.09630.

Mohseni, S., Wang, H., Yu, Z., Xiao, C., Wang, Z., Yadawa, J., 2021. Taxonomy of Machine Learning Safety: A Survey and Primer. arXiv e-prints arXiv-2106.

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J., 2019. Importance estimation for neural network pruning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11264–11272.

Montaño, J.J., Palmer, A., 2003. Numeric sensitivity analysis applied to feedforward neural networks. Neural Computing & Applications 12.

Morerio, P., Cavazza, J., Volpi, R., Vidal, R., Murino, V., 2017. Curriculum dropout, in: Proceedings of the IEEE International Conference on Computer Vision. pp. 3544–3552.

Motiian, S., Piccirilli, M., Adjeroh, D.A., Doretto, G., 2017. Unified deep supervised domain adaptation and generalization, in: Proceedings of the IEEE International Conference on Computer Vision. pp. 5715–5725.

Mountrakis, G., Xi, B., 2013. Assessing reference dataset representativeness through confidence metrics based on information density. ISPRS journal of photogrammetry and remote sensing 78, 129–147.

Moussa, G.S., Owais, M., 2021. Modeling Hot-Mix asphalt dynamic modulus using deep residual neural Networks: Parametric and sensitivity analysis study. Construction and Building Materials 294, 123589.

Mukherjee, T., R.,. Schrammel, P.,. Haller, L.,. Kroening, D.,. Melham, 2017. Lifting CDCL to Template-based Abstract Domains for Program Verification. Automated Technology for Verification and Analysis.

Müller, K.-R., 2012. Regularization techniques to improve generalization, in: Neural Networks: Tricks of the Trade. Springer, pp. 49–51.

Muller, M., Wolf, C.T., Andres, J., Desmond, M., Joshi, N.N., Ashktorab, Z., Sharma, A., Brimijoin, K., Pan, Q., Duesterwald, E., others, 2021. Designing ground truth and the social life of labels, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. pp. 1–16.

Murphy, C., Kaiser, G.E., Hu, L., Wu, L.L., 2008. Properties of Machine Learning Applications for Use in Metamorphic Testing, in: International Conference on Software Engineering and Knowledge Engineering.

Mustafa, M.B., Yusoof, M.A., Khalaf, H.K., Rahman Mahmoud Abushariah, A.A., Kiah, M.L.M., Ting, H.N., Muthaiyah, S., 2022. Code-Switching in Automatic Speech Recognition: The Issues and Future Directions. Applied Sciences 12. https://doi.org/10.3390/app12199541

Nagarajan, V., Kolter, J.Z., 2019. Deterministic PAC-bayesian generalization bounds for deep networks via generalizing noise-resilience. arXiv preprint arXiv:1905.13344.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., Sutskever, I., 2021. Deep double descent: Where bigger models and more data hurt. Journal of Statistical Mechanics: Theory and Experiment 2021, 124003.

Nallaperuma, D., Nawaratne, R., Bandaragoda, T., Adikari, A., Nguyen, S., Kempitiya, T., De Silva, D., Alahakoon, D., Pothuhera, D., 2019. Online incremental machine learning platform for big data-driven smart traffic management. IEEE Transactions on Intelligent Transportation Systems 20, 4679–4690.

Narkhede, M.V., Bartakke, P.P., Sutaone, M.S., 2022. A review on weight initialization strategies for neural networks. Artificial intelligence review 55, 291–322.

Naseem, U., Razzak, I., Khan, S.K., Prasad, M., 2021. A Comprehensive Survey on Word Representation Models: From Classical to State-of-the-Art Word Representation Language

Models. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 20, 1–35. https://doi.org/10.1145/3434237

Naser, M., Alavi, A., 2020. Insights into performance fitness and error metrics for machine learning. arXiv preprint arXiv:2006.00887.

Naser, M., Alavi, A.H., 2021. Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences. Architecture, Structures and Construction 1–19.

Nassif, A.B., Shahin, I., Attili, I., Azzeh, M., Shaalan, K., 2019. Speech Recognition Using Deep Neural Networks: A Systematic Review. IEEE Access 7, 19143–19165. https://doi.org/10.1109/ACCESS.2019.2896880

Nataraja, P., Raju, G., 2013. Quantitative influence of HCI characteristics in a blended learning system. Education and Information Technologies 18, 687–699.

Neelakantan, A., Vilnis, L., Le, Q.V., Sutskever, I., Kaiser, L., Kurach, K., Martens, J., 2015. Adding gradient noise improves learning for very deep networks. arXiv preprint arXiv:1511.06807.

Netrapalli, P., 2019. Stochastic gradient descent and its variants in machine learning. Journal of the Indian Institute of Science 99, 201–213.

Neu, G., Dziugaite, G.K., Haghifam, M., Roy, D.M., 2021. Information-Theoretic Generalization Bounds for Stochastic Gradient Descent, in: Belkin, M., Kpotufe, S. (Eds.), Proceedings of Thirty Fourth Conference on Learning Theory, Proceedings of Machine Learning Research. PMLR, pp. 3526–3545.

Neu, G., Lugosi, G., 2022. Generalization Bounds via Convex Analysis. arXiv preprint arXiv:2202.04985.

Nevzorov, A.A., Perchenko, S.V., Stankevich, D.A., 2022. Truncation: A New Approach to Neural Network Reduction. Neural Processing Letters 54, 423–435.

Neyshabur, B., 2017. Implicit regularization in deep learning. arXiv preprint arXiv:1709.01953.

Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N., 2017. Exploring Generalization in Deep Learning. CoRR abs/1706.08947.

Neyshabur, B., Salakhutdinov, R.R., Srebro, N., 2015a. Path-sgd: Path-normalized optimization in deep neural networks. Advances in neural information processing systems 28.

Neyshabur, B., Tomioka, R., Srebro, N., 2015b. In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning., in: ICLR (Workshop).

Neyshabur, B., Tomioka, R., Srebro, N., 2015c. Norm-based capacity control in neural networks, in: Conference on Learning Theory. PMLR, pp. 1376–1401.

Neyshabur, B., Tomioka, R., Srebro, N., 2014. In search of the real inductive bias: On the role of implicit regularization in deep learning. arXiv preprint arXiv:1412.6614.

Ngan, P.J., M.,. Grother, 2015. Face recognition vendor test (FRVT) performance of automated gender classification algorithms. US Department of Commerce, National Institute of Standards and Technology.

Nguyen, V.N., Holone, H., 2015. Possibilities, Challenges and the State of the Art of Automatic Speech Recognition in Air Traffic Control. International Journal of Computer and Information Engineering 9, 1933–1942.

Niaz, M.S., Saake, G., 2015. Merkle hash tree based techniques for data integrity of outsourced data. GvD 1366, 66–71.

Nitesh Varma Rudraraju, N., Varun Boyanapally, V., 2019. Data Quality Model for Machine learning.

Nobles, A.L., Vilankar, K., Wu, H., Barnes, L.E., 2015. Evaluation of data quality of multisite electronic health record data for secondary analysis, in: 2015 IEEE International Conference on Big Data (Big Data). IEEE, pp. 2612–2620.

Northcutt, C., Jiang, L., Chuang, I., 2021. Confident learning: Estimating uncertainty in dataset labels. Journal of Artificial Intelligence Research 70, 1373–1411.

Nourbakhsh, A., Bang, G., 2019. A framework for anomaly detection using language modeling, and its applications to finance. https://doi.org/10.48550/ARXIV.1908.09156

Nourbakhsh, A., Liu, X., Li, Q., Shah, S., 2017. Mapping the echo-chamber: detecting and characterizing partisan networks on Twitter.

Nourbakhsh, A., Liu, X., Shah, S., Fang, R., Ghassemi, M.M., Li, Q., 2015. Newsworthy Rumor Events: A Case Study of Twitter. 2015 IEEE International Conference on Data Mining Workshop (ICDMW) 27–32.

Novak, R., Bahri, Y., Abolafia, D.A., Pennington, J., Sohl-Dickstein, J., 2018. Sensitivity and generalization in neural networks: an empirical study. arXiv preprint arXiv:1802.08760.

O. Bouissou, M.T., E. Conquet, P. Cousot, R. Cousot, J. Feret, K. Ghorbal, E. Goubault, D. Lesens, L. Mauborgne, A. Miné, S. Putot, X. Rival, 2009. Space software validation using Abstract Interpretation. Data Systems in Aerospace 669.

Ohneiser, O., Helmke, H., Ehr, H., Gürlük, H., Hoessl, M., Mühlhausen, T., Oualil, Y., Schulder, M., Schmidt, A., Khan, A., Klakow, D., 2014. Air Traffic Controller Support by Speech Recognition. https://doi.org/10.54941/ahfe100712

Olorisade, B.K., Brereton, P., Andras, P., 2017. Reproducibility of studies on text mining for citation screening in systematic reviews: Evaluation and checklist. Journal of biomedical informatics 73, 1–13.

Onwujekwegn, G., Yoon, V.Y., 2020. Analyzing the Impacts of Activation Functions on the Performance of Convolutional Neural Network Models.

Osman, M.S., Abu-Mahfouz, A.M., Page, P.R., 2018. A survey on data imputation techniques: Water distribution system as a use case. IEEE Access 6, 63279–63291.

Otles, E., Oh, J., Li, B., Bochinski, M., Joo, H., Ortwine, J., Shenoy, E., Washer, L., Young, V.B., Rao, K., others, 2021. Mind the performance gap: examining dataset shift during prospective validation, in: Machine Learning for Healthcare Conference. PMLR, pp. 506–534.

Ouyang, T., Marco, V.S., Isobe, Y., Asoh, H., Oiwa, Y., Seo, Y., 2021. Improved Surprise Adequacy Tools for Corner Case Data Description and Detection. Applied Sciences 11. https://doi.org/10.3390/app11156826

Owen, M.P., Panken, A., Moss, R., Alvarez, L., Leeper, C., 2019. ACAS Xu: Integrated Collision Avoidance and Detect and Avoid Capability for UAS, in: 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC). Presented at the 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), IEEE, San Diego, CA, USA, pp. 1–10. https://doi.org/10.1109/DASC43569.2019.9081758

Oza, P., Patel, V.M., 2019. C2AE: Class Conditioned Auto-Encoder for Open-set Recognition. https://doi.org/10.48550/ARXIV.1904.01198

P. Malakar, K.K., P. Balaprakash, V. Vishwanath, V. Morozov, n.d. Benchmarking Machine Learning Methods for Performance Modeling of Scientific Applications 2018.

Padilla, R., Netto, S.L., Da Silva, E.A., 2020. A survey on performance metrics for object-detection algorithms, in: 2020 International Conference on Systems, Signals and Image Processing (IWSSIP). IEEE, pp. 237–242.

Paganelli, M., Buono, F.D., Guerra, F., Ferro, N., 2022. Evaluating the integration of datasets, in: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. pp. 347–356.

Palmer, D.D., Hearst, M.A., 1994. Adaptive Sentence Boundary Disambiguation. CoRR abs/cmp-lg/9411022.

Papineni, W.J., K.,. Roukos, S.,. Ward, T.,.&. Zhu, 2002. BLEU: a method for automatic evaluation of machine translation.

Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S., 2019. Continual lifelong learning with neural networks: A review. Neural Networks 113, 54–71.

Pei, K., Cao, Y., Yang, J., Jana, S., 2017. DeepXplore: Automated whitebox testing of deep learning systems, in: Proceedings of the 26th Symposium on Operating Systems Principles. pp. 1–18.

Perera, P., Patel, V.M., 2019. Learning Deep Features for One-Class Classification. IEEE Transactions on Image Processing 28, 5450–5463. https://doi.org/10.1109/TIP.2019.2917862

Peters, B., Kriegeskorte, N., 2021. Capturing the objects of vision with neural networks. Nature Human Behaviour 5, 1127–1144.

Pham, T., G, V.K.B., Do, T.-T., Carneiro, G., Reid, I., 2018. Bayesian Semantic Instance Segmentation in Open Set World. https://doi.org/10.48550/ARXIV.1806.00911

Pimentel, M.A.F., Clifton, D.A., Clifton, L., 2013. A Review of Novelty Detection, Signal Processing 99, 215–249.

Poggio, T., Kawaguchi, K., Liao, Q., Miranda, B., Rosasco, L., Boix, X., Hidary, J., Mhaskar, H., 2017a. Theory of deep learning III: explaining the non-overfitting puzzle. arXiv preprint arXiv:1801.00173.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., Liao, Q., 2017b. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. International Journal of Automation and Computing 14, 503–519.

Popoola, O.P., Wang, K., 2012. Video-Based Abnormal Human Behavior Recognition – A Review, IEEE Trans. on Systems, Man and Cybernetics, Part C (Applications and Reviews) 42, 865–878.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlíček, P., Qian, Y., Schwarz, P., Silovský, J., Stemmer, G., Vesel, K., 2011. The Kaldi speech recognition toolkit. IEEE 2011 Workshop on Automatic Speech Recognition and Understanding.

Prabakaran, D., Sriuppili, S., 2021. Speech Processing: MFCC Based Feature Extraction Techniques- An Investigation. Journal of Physics: Conference Series 1717, 012009. https://doi.org/10.1088/1742-6596/1717/1/012009

Prechelt, L., 1994. PROBEN1: A set of benchmarks and benchmarking rules for neural network training algorithms. Fakultaet fuer Informatik, Universitaet Karlsruhe.

Priyadarshini, I., Puri, V., 2021. Mars weather data analysis using machine learning techniques. Earth Science Informatics 14, 1885–1898.

Pulina, L., Tacchella, A., 2010. An Abstraction-Refinement Approach to Verification of Artificial Neural Networks, in: Touili, T., Cook, B., Jackson, P. (Eds.), Computer Aided Verification. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 243–257.

R., C.P., Cousot, 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages.

R., E., 2017. Formal verification of piece-wise linear feed-forward neural networks. Automated Technology for Verification and Analysis.

R., H.C.A., 1969. An axiomatic basis for computer programming. Communications of the ACM 12.

R. R. Selvaraju, D.B., M. Cogswell, A. Das, R. Vedantam, D. Parikh, 2016. Visual explanations from deep networks via gradient-based localization.

R. S. Olson, J.H.M., W. La Cava, P. Orzechowski, R.J. Urbanowicz, 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison 10.

Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I., 2022. Robust Speech Recognition via Large-Scale Weak Supervision.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., Sohl-Dickstein, J., 2017. On the expressive power of deep neural networks, in: International Conference on Machine Learning. PMLR, pp. 2847–2854.

Rajendran, J., Irpan, A., Jang, E., 2020. Meta-learning requires meta-augmentation. Advances in Neural Information Processing Systems 33, 5705–5715.

Raju, A., Tiwari, G., Rao, M., Dheram, P., Anderson, B., Zhang, Z., Bui, B., Rastrow, A., 2021. End-to-End Spoken Language Understanding using RNN-Transducer ASR. ArXiv abs/2106.15919.

Ramsey, C.A., Hewitt, A.D., 2005. A methodology for assessing sample representativeness. Environmental Forensics 6, 71–75.

Ranzato, Z.M., F., 2019. Robustness Verification of Support Vector Machines. Springer International Publishing.

Rao, Q., Frtunikj, J., 2018. Deep learning for self-driving cars: Chances and challenges, in: Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems. pp. 35–38.

Raskutti, G., Wainwright, M.J., Yu, B., 2014. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. The Journal of Machine Learning Research 15, 335–366.

Ratner, A.J., Ehrenberg, H., Hussain, Z., Dunnmon, J., Ré, C., 2017. Learning to compose domain-specific transformations for data augmentation. Advances in neural information processing systems 30.

Razavi, S., Jakeman, A., Saltelli, A., Prieur, C., Iooss, B., Borgonovo, E., Plischke, E., Piano, S.L., Iwanaga, T., Becker, W., others, 2021. The future of sensitivity analysis: an essential discipline for systems modeling and policy support. Environmental Modelling & Software 137, 104954.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., Lampert, C.H., 2017. icarl: Incremental classifier and representation learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2001–2010.

Redman, T.C., 2018. If your data is bad, your machine learning tools are useless. Harvard Business Review 2.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 779–788.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Refaeilzadeh, P., Tang, L., Liu, H., 2009. Cross-validation. Encyclopedia of database systems 5, 532–538.

Reid, M., 2010. Generalization Bounds, in: Sammut, C., Webb, G.I. (Eds.), Encyclopedia of Machine Learning. Springer US, Boston, MA, pp. 447–454. https://doi.org/10.1007/978-0-387-30164-8_328

Renggli, C., Rimanic, L., Gürel, N.M., Karlaš, B., Wu, W., Zhang, C., 2021. A data quality-driven view of mlops. arXiv preprint arXiv:2102.07750.

Rice, M., Li, L., Gu, Y., Wan, M., Lim, E., Feng, G., Ng, J., Jin-Li, M., Babu, S., 2018. Automating the Visual Inspection of Aircraft.

Rifai, A.I., 2021. ITISE 2018: Data Mining Applied for Performance Index Prediction in Highway Long Segment Maintenance Contract.

Riley, R.D., Debray, T.P., Collins, G.S., Archer, L., Ensor, J., van Smeden, M., Snell, K.I., 2021. Minimum sample size for external validation of a clinical prediction model with a binary outcome. Statistics in medicine 40, 4230–4251.

Rodríguez Gálvez, B., Bassi, G., Thobaben, R., Skoglund, M., 2021. Tighter expected generalization error bounds via wasserstein distance. Advances in Neural Information Processing Systems 34, 19109–19121.

Roelofs, R., 2019. Measuring Generalization and overfitting in Machine learning. University of California, Berkeley.

Rohlfs, C., 2022. Generalization in Neural Networks: A Broad Survey. arXiv preprint arXiv:2209.01610.

Rong, K., Khant, A., Flores, D., Montañez, G.D., 2021. The Label Recorder Method: Testing the Memorization Capacity of Machine Learning Models, in: Machine Learning, Optimization, and Data Science: 7th International Conference, LOD 2021, Grasmere, UK, October 4–8, 2021, Revised Selected Papers, Part I. Springer-Verlag, Berlin, Heidelberg, pp. 581–595. https://doi.org/10.1007/978-3-030-95467-3_42

Rosario, B., 2001. Latent Semantic Indexing : An Overview 1 Latent Semantic Indexing : An overview INFOSYS 240 Spring 2000 Final Paper.

Rostami, M., 2021. Lifelong domain adaptation via consolidated internal distribution. Advances in Neural Information Processing Systems 34, 11172–11183.

Roy, P.P., Roy, K., 2008. On some aspects of variable selection for partial least squares regression models. QSAR & Combinatorial Science 27, 302–313.

Ruder, S., 2017. An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M., 2018. Deep One-Class Classification, in: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, pp. 4393–4402.

Russo, D., Zou, J., 2016. Controlling bias in adaptive data analysis using information theory, in: Artificial Intelligence and Statistics. PMLR, pp. 1232–1240.

S. Katz, M.K., C. Strong, K. Julian, 2021. Generating probabilistic safety guarantees for neural network controllers. arxiv.

S. M. Lundberg, S.-I.L., 2017. A unified approach to interpreting model predictions.

S. Wang, S.J., K. Pei, J. Whitehouse, J. Yang, 2018. Efficient Formal Safety Analysis of Neural Networks. International Conference on Neural Information Processing Systems 32, 6369-—6379.

Sadeghzadeh, A., 2020. Linear Parameter-Varying Embedding of Nonlinear Models with Reduced Conservativeness 53.

Sáez, J.A., Luengo, J., Herrera, F., 2016. Evaluating the classifier behavior with noisy data considering performance and robustness: The equalized loss of accuracy measure. Neurocomputing 176, 26–35.

Salamon, J., Bittner, R.M., Bonada, J., Bosch, J.J., Gómez Gutiérrez, E., Bello, J.P., 2017. An analysis/synthesis framework for automatic f0 annotation of multitrack datasets, in: Hu X, Cunningham SJ, Turnbull D, Duan Z. ISMIR 2017 Proceedings of the 18th International Society for Music Information Retrieval Conference; 2017 Oct 23-27; Suzhou, China.[Suzhou]: ISMIR; 2017. International Society for Music Information Retrieval (ISMIR).

Samanta, P., Chaudhuri, B.B., 2013. A simple real-word error detection and correction using local word bigram and trigram, in: Proceedings of the 25th Conference on Computational Linguistics and Speech Processing (ROCLING 2013). The Association for Computational Linguistics and Chinese Language Processing (ACLCLP), Kaohsiung, Taiwan, pp. 211–220.

Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., Aroyo, L.M., 2021. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. pp. 1–15.

Sameena Shah, Dietmar Dorr, Khalid Al-Kofahi, Sisk, J., n.d. Systems and methods for determining atypical language.

Sampson, G., 1987. Parallel distributed processing: Explorations in the microstructures of cognition.

Sánchez, R.Á., Iraola, A.B., Unanue, G.E., Carlin, P., 2019. TAQIH, a tool for tabular data quality assessment and improvement in the context of health data. Computer methods and programs in biomedicine 181, 104824.

Santos, M.S., Pereira, R.C., Costa, A.F., Soares, J.P., Santos, J., Abreu, P.H., 2019. Generating synthetic missing data: A review by missing mechanism. IEEE Access 7, 11651–11667.

Santurkar, S., Tsipras, D., Ilyas, A., Madry, A., 2018. How does batch normalization help optimization? Advances in neural information processing systems 31.

Saquib, Z.U.H., Salam, N., Nair, R.P., Pandey, N., 2011. Voiceprint Recognition Systems for Remote Authentication-A Survey.

Sarker, I.H., 2021. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. SN Comput Sci 2, 420. https://doi.org/10.1007/s42979-021-00815-1

Scheirer, W.J., de Rezende Rocha, A., Sapkota, A., Boult, T.E., 2013. Toward Open Set Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 35, 1757–1772. https://doi.org/10.1109/TPAMI.2012.256

Schelter, S., Rukat, T., Biessmann, F., 2021. JENGA-A Framework to Study the Impact of Data Errors on the Predictions of Machine Learning Models., in: EDBT. pp. 529–534.

Schölkopf, B., 2022. Causality for machine learning, in: Probabilistic and Causal Inference: The Works of Judea Pearl. pp. 765–804.

Schouten, B., Cobben, F., Bethlehem, J., others, 2009. Indicators for the representativeness of survey response. Survey Methodology 35, 101–113.

See, J.E., Drury, C.G., Speed, A., Williams, A., Khalandi, N., 2017. The Role of Visual Inspection in the 21st Century. Proceedings of the Human Factors and Ergonomics Society Annual Meeting 61, 262–266. https://doi.org/10.1177/1541931213601548

Seeger, M., 2002. PAC-Bayesian generalisation error bounds for Gaussian process classification. Journal of machine learning research 3, 233–269.

Sena L. H., M.E., Bessa I.V.,. Gadelha M.R.,. Cordeiro L.C., 2019. Incremental Bounded Model Checking of Artificial Neural Networks in CUDA.

Seo, S., Suh, Y., Kim, D., Kim, G., Han, J., Han, B., 2020. Learning to optimize domain specific normalization for domain generalization, in: European Conference on Computer Vision. Springer, pp. 68–83.

Setiawan, B.D., Serdült, U., Kryssanov, V., 2021. A machine learning framework for balancing training sets of sensor sequential data streams. Sensors 21, 6892.

Shahinfar, S., Meek, P., Falzon, G., 2020. “How many images do I need?” Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. Ecological Informatics 57, 101085.

Shailaja, P., Padmanabhan, S., 2022. A Survey on Autonomous Damage Detection on Aircraft Surfaces using Deep Learning Models, in: 2022 6th International Conference on Computing Methodologies and Communication (ICCMC). IEEE, pp. 1135–1140.

Sharma, A., Kumar, S., 2023. Machine learning and ontology-based novel semantic document indexing for information retrieval. Computers & Industrial Engineering 176, 108940. https://doi.org/10.1016/j.cie.2022.108940

Sharma, G., Umapathy, K., Krishnan, S., 2020. Trends in audio signal feature extraction methods. Applied Acoustics 158, 107020.

Sharma, Sagar, Sharma, Simone, Athaiya, A., 2017. Activation functions in neural networks. towards data science 6, 310–316.

Sharon Y. Li, n.d. Automating Data Augmentation: Practice, Theory and New Direction. The Stanford AI Lab Blog. URL https://ai.stanford.edu/blog/data-augmentation/ (accessed 4.24.20).

Shen, J., Qu, Y., Zhang, W., Yu, Y., 2017. Wasserstein Distance Guided Representation Learning for Domain Adaptation. https://doi.org/10.48550/ARXIV.1707.01217

Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., Cui, P., 2021. Towards out-of-distribution generalization: A survey. arXiv preprint arXiv:2108.13624.

Shi, Y., Yu, X., Sohn, K., Chandraker, M., Jain, A.K., 2020. Towards universal representation learning for deep face recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6817–6826.

Shi, Z., Zhang, H., Chang, K.-W., Huang, M., Hsieh, C.-J., 2020. Robustness Verification for Transformers. https://doi.org/10.48550/ARXIV.2002.06622

Shinde, P.P., Shah, S., 2018. A Review of Machine Learning and Deep Learning Applications, in: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). pp. 1–6. https://doi.org/10.1109/ICCUBEA.2018.8697857

Shiomi M, H.N., Sakamoto D, Kanda T, Ishi CT, Ishiguro H., 2011. Field Trial of a Networked Robot at a Train Station. International Journal of Social Robotics 3.

Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. Journal of big data 6, 1–48.

Shorten, C., Khoshgoftaar, T.M., Furht, B., 2021. Text data augmentation for deep learning. Journal of big Data 8, 1–34.

Shrestha, A., Mascaro, G., Garcia, M., 2022. Effects of stormwater infrastructure data completeness and model resolution on urban flood modeling. Journal of Hydrology 607, 127498.

Shwartz-Ziv, R., Tishby, N., 2017. Opening the black box of deep neural networks via information. arXiv preprint arXiv:1703.00810.

Shyam, R., Singh, R., 2021. A Taxonomy of Machine Learning Techniques. Journal of Advancements in Robotics 8, 18–25p.

Siebert, J., Joeckel, L., Heidrich, J., Nakamichi, K., Ohashi, K., Namba, I., Yamamoto, R., Aoyama, M., 2020. Towards guidelines for assessing qualities of machine learning systems, in: International Conference on the Quality of Information and Communications Technology. Springer, pp. 17–31.

Simão, F.A., Waterhouse, R.M., Ioannidis, P., Kriventseva, E.V., Zdobnov, E.M., 2015. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics 31, 3210–3212.

Simonyan, K., Vedaldi, A., Zisserman, A., 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Singh, G., Gehr, T., Püschel, M., Vechev, M., 2018. An Abstract Domain for Certifying Neural Networks. Programming Languages.

Sinha, G., Shahi, R., Shankar, M., 2010. Human computer interaction, in: 2010 3rd International Conference on Emerging Trends in Engineering and Technology. IEEE, pp. 1–4.

Šmídl, L., 2011. Air Traffic Control Communication.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S., 2018. X-Vectors: Robust DNN Embeddings for Speaker Recognition, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5329–5333. https://doi.org/10.1109/ICASSP.2018.8461375

Soenjaya, A.L., 2013. On Strong and Weak Convergence in $ N-$ hilbert Spaces. Journal of the Indonesian Mathematical Society 79–87.

Sohai, F.S., 2022. A MACHINE LEARNING FRAMEWORK FOR AUTOMATIC SPEECH RECOGNITION IN AIR TRAFFIC CONTROL USING WORD LEVEL BINARY CLASSIFICATION AND TRANSCRIPTION (PhD Thesis). Rowan University.

Song, H., Kim, M., Park, D., Shin, Y., Lee, J.-G., 2022. Learning from noisy labels with deep neural networks: A survey. IEEE Transactions on Neural Networks and Learning Systems.

Souyris J., D.D., 2007. Experimental Assessment of Astrée on Safety-Critical Avionics Software. Proceding of Internation Conference on Computer Safety, Reliability, and Security 4680.

Spasic, I., Nenadic, G., others, 2020. Clinical text data in machine learning: systematic review. JMIR medical informatics 8, e17984.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15, 1929–1958.

Steinke, T., Zakynthinou, L., 2020. Reasoning about generalization via conditional mutual information, in: Conference on Learning Theory. PMLR, pp. 3437–3452.

Subbaswamy, A., Adams, R., Saria, S., 2021. Evaluating model robustness and stability to dataset shift, in: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 2611–2619.

Sun, K., Nielsen, F., 2019. A Geometric Modeling of Occam’s Razor in Deep Learning. arXiv preprint arXiv:1905.11027.

Sun, Y., Huang, X., Kroening, D., Sharp, J., Hill, M., Ashmore, R., 2018. Testing Deep Neural Networks. https://doi.org/10.48550/ARXIV.1803.04792

Surace, S.C., Kutschireiter, A., Pfister, J.-P., 2019. How to avoid the curse of dimensionality: Scalability of particle filters with and without importance weights. SIAM review 61, 79–91.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9.

T. St. John, P.M., 2019. MLPerf: A Benchmark for Machine Learning.

Tabassi, E., 2023. Artificial Intelligence Risk Management Framework (AI RMF 1.0) (No. NIST AI 100-1). National Institute of Standards and Technology (U.S.), Gaithersburg, MD. https://doi.org/10.6028/NIST.AI.100-1

Tae, K.H., Whang, S.E., 2021. Slice tuner: A selective data acquisition framework for accurate and fair machine learning models, in: Proceedings of the 2021 International Conference on Management of Data. pp. 1771–1783.

Taheri, M., Xie, F., Lederer, J., 2021. Statistical guarantees for regularized neural networks. Neural Networks 142, 148–161.

Talbot, J., Ting, D., 2022. Statistical Schema Learning with Occam’s Razor, in: Proceedings of the 2022 International Conference on Management of Data, SIGMOD ’22. Association for Computing Machinery, New York, NY, USA, pp. 176–189. https://doi.org/10.1145/3514221.3526174

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C., 2018. A survey on deep transfer learning, in: International Conference on Artificial Neural Networks. Springer, pp. 270–279.

Tan, H.H., Lim, K.H., 2019. Vanishing gradient mitigation with deep learning neural network optimization, in: 2019 7th International Conference on Smart Computing & Communications (ICSCC). IEEE, pp. 1–4.

Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S., 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. Advances in Neural Information Processing Systems 33, 6377–6389.

Tatman, R., VanderPlas, J., Dane, S., 2018. A practical taxonomy of reproducibility for machine learning research.

Tempo, R., Calafiore, G., Dabbene, F., 2013. Statistical Learning Theory, in: Randomized Algorithms for Analysis and Control of Uncertain Systems. Springer, pp. 123–134.

Thrush, T., Tirumala, K., Gupta, A., Bartolo, M., Rodriguez, P., Kane, T., Rojas, W.G., Mattson, P., Williams, A., Kiela, D., 2022. Dynatask: A Framework for Creating Dynamic AI Benchmark Tasks. arXiv preprint arXiv:2204.01906.

Tian, B., Y.,. Pei, K.,. Jana, S.,. Ray, 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. Proceedings of the 40th international conference on software engineering.

Tian, Y., Pei, K., Jana, S., Ray, B., 2017. DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars. https://doi.org/10.48550/ARXIV.1708.08559

Tian, Y., Zhang, Y., 2022. A comprehensive survey on regularization strategies in machine learning. Information Fusion 80, 146–166.

Tinghui Ouyang, Y.S., Vicent Sanz Marco, Yoshinao Isobe, Hideki Asoh, Yutaka Oiwa, 2021. Corner Case Data Description and Detection 19–26.

Tolstikhin, I.O., Seldin, Y., 2013. PAC-Bayes-empirical-Bernstein inequality. Advances in Neural Information Processing Systems 26.

Too, E.C., Yujian, L., Njuki, S., Yingchun, L., 2019. A comparative study of fine-tuning deep learning models for plant disease identification. Computers and Electronics in Agriculture 161, 272–279.

Torens, C., Durak, U., Dauer, J.C., 2022. Guidelines and Regulatory Framework for Machine Learning in Aviation, in: AIAA Scitech 2022 Forum. p. 1132.

Trinh, T.X., Ha, M.K., Choi, J.S., Byun, H.G., Yoon, T.H., 2018. Curation of datasets, assessment of their quality and completeness, and nanoSAR classification model development for metallic nanoparticles. Environmental Science: Nano 5, 1902–1910.

Tsai, C.-F., Sung, Y.-T., 2020. Ensemble feature selection in high dimension, low sample size datasets: Parallel and serial combination approaches. Knowledge-Based Systems 203, 106097.

Turney, P.D., Pantel, P., 2010. From Frequency to Meaning: Vector Space Models of Semantics. Journal of Artificial Intelligence Research 37, 141–188. https://doi.org/10.1613/jair.2934

UK-Government, 2015. The Pathway to Driverless Cars: A Code of Practice for testing. UK Government.

Upadhyaya, P., Farooq, O., Abidi, M.R., Varshney, Y.V., 2017. Continuous Hindi speech recognition model based on Kaldi ASR toolkit, in: 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). IEEE, pp. 786–789.

Uzair, M., Jamil, N., 2020. Effects of Hidden Layers on the Efficiency of Neural networks, in: 2020 IEEE 23rd International Multitopic Conference (INMIC). pp. 1–6. https://doi.org/10.1109/INMIC50486.2020.9318195

V. Tjeng, R.T., K. Xiao, n.d. Evaluating Robustness of Neural Networks with Mixed Integer Programming. The International Conference on Learning Representations.

Valada, A., Vertens, J., Dhall, A., Burgard, W., 2017. Adapnet: Adaptive semantic segmentation in adverse environmental conditions, in: 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 4644–4651.

Valle-Pérez, G., Louis, A.A., 2020. Generalization bounds for deep learning. arXiv preprint arXiv:2012.04115.

Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. Journal of machine learning research 9.

van Dyck, L.E., Gruber, W.R., 2020. Seeing eye-to-eye? A comparison of object recognition performance in humans and deep convolutional neural networks under image manipulation. arXiv preprint arXiv:2007.06294.

Van Ginneken, B., Kerkstra, S., Meakin, J., n.d. [Online] https://grand-challenge.org.

Van Royen, F.S., Asselbergs, F.W., Alfonso, F., Vardas, P., Van Smeden, M., 2023. Five critical quality criteria for artificial intelligence-based prediction models. European Heart Journal 44, 4831–4834.

Van Vleck, T.T., Stein, D.M., Stetson, P.D., Johnson, S.B., 2007. Assessing data relevance for automated generation of a clinical summary, in: AMIA Annual Symposium Proceedings. American Medical Informatics Association, p. 761.

Vapnik, V., 1999. The nature of statistical learning theory. Springer science & business media.

Vapnik, V., Izmailov, R., 2020. Complete statistical theory of learning: learning using statistical invariants, in: Conformal and Probabilistic Prediction and Applications. PMLR, pp. 4–40.

Vapnik, V.N., 1999. An overview of statistical learning theory. IEEE transactions on neural networks 10, 988–999.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention Is All You Need.

Vecchi, E.M., Baroni, M., Zamparelli, R., 2011. (Linear) Maps of the Impossible: Capturing Semantic Anomalies in Distributional Space, in: Proceedings of the Workshop on Distributional

Semantics and Compositionality. Association for Computational Linguistics, Portland, Oregon, USA, pp. 1–9.
Veregin, H., 1999. Data quality parameters. Geographical information systems 1, 177–189.
Vertens, J., Zürn, J., Burgard, W., 2020. HeatNet: Bridging the Day-Night Domain Gap in Semantic Segmentation with Thermal Images. https://doi.org/10.48550/ARXIV.2003.04645
Vetter, W., V.,. Zielke, T.,. von Seelen, 1997. Integrating face recognition into security systems. In: Audio- and Video-based Biometric Person Authentication Audio-and Video-based Biometric Person Authentication (AVBPA).
Volpi, R., Murino, V., 2019. Addressing model vulnerability to distributional shifts over image transformation sets, in: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7980–7989.
Vorontsov, E., Trabelsi, C., Kadoury, S., Pal, C., 2017. On orthogonality and learning recurrent networks with long term dependencies, in: International Conference on Machine Learning. PMLR, pp. 3570–3578.
Vrbančič, G., Podgorelec, V., 2020. Transfer learning with adaptive fine-tuning. IEEE Access 8, 196197–196211.
W. Ruan, M.K., M. Wu, Y. Sun, X. Huang, D. Kroening, 2019. Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance 28.
Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.J., 1989. Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing 37, 328–339. https://doi.org/10.1109/29.21701
Wang, A., Narayanan, A., Russakovsky, O., 2020. REVISE: A tool for measuring and mitigating bias in visual datasets, in: European Conference on Computer Vision. Springer, pp. 733–751.
Wang, C., Wu, Y., Liu, S., Li, J., Qian, Y., Kumatani, K., Wei, F., 2021. Unispeech at scale: An empirical study of pre-training method on large-scale speech recognition dataset. arXiv preprint arXiv:2107.05233.
Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., Yu, P., 2022. Generalizing to unseen domains: A survey on domain generalization. IEEE Transactions on Knowledge and Data Engineering.
Wang, J., Xu, C., Yang, X., Zurada, J.M., 2017. A novel pruning algorithm for smoothing feedforward neural networks based on group lasso method. IEEE transactions on neural networks and learning systems 29, 2012–2024.
Wang, Q., Ma, Y., Zhao, K., Tian, Y., 2022. A comprehensive survey of loss functions in machine learning. Annals of Data Science 9, 187–212.
Wang, R., Yang, L., Zhang, B., Zhu, W., Doermann, D., Guo, G., 2022. Confidence Dimension for Deep Learning based on Hoeffding Inequality and Relative Evaluation. arXiv preprint arXiv:2203.09082.
Wang, R.Y., Ziad, M., Lee, Y.W., 2006. Data quality. Springer Science & Business Media.
Wang, S., Yu, L., Li, K., Yang, X., Fu, C.-W., Heng, P.-A., 2020. Dofe: Domain-oriented feature embedding for generalizable fundus image segmentation on unseen datasets. IEEE Transactions on Medical Imaging 39, 4237–4248.
Wang, Wenqi, Wang, R., Wang, L., Wang, Z., Ye, A., 2019. Towards a Robust Deep Neural Network in Texts: A Survey. https://doi.org/10.48550/ARXIV.1902.07285
Wang, Wei, Zheng, V.W., Yu, H., Miao, C., 2019. A survey of zero-shot learning: Settings, methods, and applications. ACM Transactions on Intelligent Systems and Technology (TIST) 10, 1–37.

Wang, Y., Yao, Q., 2019. Few-shot learning: A survey.

Wang, Z., Cheng, X., Sapiro, G., Qiu, Q., 2020. A dictionary approach to domain-invariant learning in deep networks. Advances in neural information processing systems 33, 6595–6605.

Wang, Z., Loog, M., van Gemert, J., 2021. Respecting domain relations: Hypothesis invariance for domain generalization, in: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, pp. 9756–9763.

Watson, D.S., Wright, M.N., 2021. Testing conditional independence in supervised learning algorithms. Machine Learning 110, 2107–2129.

Weber, E., Gut, D., 2005. A survey of weeds that are increasingly spreading in Europe. Agron. Sustain. Dev. 25, 109–121. https://doi.org/10.1051/agro:2004061

Wei, C., Lee, J., Liu, Q., Ma, T., 2019. On the Margin Theory of Feedforward Neural Networks.

Wei, J., Zou, K., 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196.

Wendlandt, L., Kummerfeld, J.K., Mihalcea, R., 2018. Factors Influencing the Surprising Instability of Word Embeddings, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). Association for Computational Linguistics. https://doi.org/10.18653/v1/n18-1190

Willemink, M.J., Koszek, W.A., Hardell, C., Wu, J., Fleischmann, D., Harvey, H., Folio, L.R., Summers, R.M., Rubin, D.L., Lungren, M.P., 2020. Preparing medical imaging data for machine learning. Radiology 295, 4–15.

Williams, L., 2006. White-Box Testing. PDF): 60–61 69.

Wilson, A.C., Roelofs, R., Stern, M., Srebro, N., Recht, B., 2017. The marginal value of adaptive gradient methods in machine learning. Advances in neural information processing systems 30.

Witten, I.H., Frank, E., Hall, M.A., Pal, C.J., 2016. Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques, 4th ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Wu, H., Shapiro, J.L., 2006. Does overfitting affect performance in estimation of distribution algorithms, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. pp. 433–434.

Wu, T. (Sherry), Ribeiro, M.T., Heer, J., Weld, D.S., 2019. Errudite: Scalable, Reproducible, and Testable Error Analysis, in: Proc. Association for Computational Linguistics (ACL).

Wu, X., Lv, S., Zang, L., Han, J., Hu, S., 2019. Conditional bert contextual augmentation, in: International Conference on Computational Science. Springer, pp. 84–95.

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J., 2016a. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. https://doi.org/10.48550/ARXIV.1609.08144

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., others, 2016b. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

X. Sun, ., H. Khedr, Y. Shoukry, 2019. Formal Verification of Neural Network Controlled Autonomous Systems. International Conference on Hybrid Systems: Computation and Control 22, 147—156.

Xia, Y., Cao, X., Wen, F., Hua, G., Sun, J., 2015. Learning Discriminative Reconstructions for Unsupervised Outlier Removal. 2015 IEEE International Conference on Computer Vision (ICCV) 1511–1519.

Xiao, Y., Decencière, E., Velasco-Forero, S., Burdin, H., Bornschlögl, T., Bernerd, F., Warrick, E., Baldeweck, T., 2019. A new color augmentation method for deep learning segmentation of histological images, in: 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019). IEEE, pp. 886–890.

Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Wang, L., Chen, Z., Xiao, A., Chang, J., Zhang, X., others, 2021. Weight-sharing neural architecture search: A battle to shrink the optimization gap. ACM Computing Surveys (CSUR) 54, 1–37.

Xie X, C.T., Ho J, Murphy C, Kaiser G, Xu B., 2009. Application of Metamorphic Testing to Supervised Classifiers.

Xu, D., Ricci, E., Yan, Y., Song, J., Sebe, N., 2015. Learning Deep Representations of Appearance and Motion for Anomalous Event Detection, in: British Machine Vision Conference.

Xu, F., Zou, Z., Yin, J., Cao, J., 2012. Parametric identification and sensitivity Analysis for Autonomous underwater vehicles in diving plane. Journal of Hydrodynamics 24, 744–751.

Xu, H., Mannor, S., 2012. Robustness and generalization. Machine Learning 86, 391–423. https://doi.org/10.1007/s10994-011-5268-1

Xu, T., Chen, W., Wang, P., Wang, F., Li, H., Jin, R., 2021. Cdtrans: Cross-domain transformer for unsupervised domain adaptation. arXiv preprint arXiv:2109.06165.

Xu, Z., Li, W., Niu, L., Xu, D., 2014. Exploiting low-rank structure from latent domains for domain generalization, in: European Conference on Computer Vision. Springer, pp. 628–643.

Xu, Z., Saleh, J.H., 2021. Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. Reliability Engineering & System Safety 211, 107530. https://doi.org/10.1016/j.ress.2021.107530

Yamaguchi T., K.Y., Brain M.,. Ryder C.,. Imai Y., 2019. Application of Abstract Interpretation to the Automotive Electronic Control System. Verification, Model Checking, and Abstract Interpretation 11388.

Yan, L., Dodier, R.H., Mozer, M., Wolniewicz, R.H., 2003. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic, in: Proceedings of the 20th International Conference on Machine Learning (Icml-03). pp. 848–855.

Yan, W., Zhu, J., Zhou, Y., Wang, Y., Zheng, Q., 2023. Multi-view Semantic Consistency based Information Bottleneck for Clustering. arXiv preprint arXiv:2303.00002.

Yan, X., Lau, R.Y.K., Song, D., Li, X., Ma, J., 2011. Toward a Semantic Granularity Model for Domain-Specific Information Retrieval. ACM Trans. Inf. Syst. 29. https://doi.org/10.1145/1993036.1993039

Yang, Y., Urolagin, S., Niroula, A., Ding, X., Shen, B., Vihinen, M., 2018. PON-tstab: protein variant stability predictor. Importance of training data quality. International journal of molecular sciences 19, 1009.

Yao, L., Chu, Z., Li, S., Li, Y., Gao, J., Zhang, A., 2021. A survey on causal inference. ACM Transactions on Knowledge Discovery from Data (TKDD) 15, 1–46.

Yates, C., Christopher, R., Tumer, K., 2020. Multi-fitness learning for behavior-driven cooperation, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 453–461.

Yelle, L.E., 1979. The learning curve: Historical review and comprehensive survey. Decision sciences 10, 302–328.

Yin, D., Kannan, R., Bartlett, P., 2019. Rademacher complexity for adversarially robust generalization, in: International Conference on Machine Learning. PMLR, pp. 7085–7094.

Ying, X., 2019. An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series 1168, 022022. https://doi.org/10.1088/1742-6596/1168/2/022022

Yoon, C., Hamarneh, G., Garbi, R., 2019. Generalizable feature learning in the presence of data bias and domain class imbalance with application to skin lesion classification, in: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp. 365–373.

Yoshihashi, R., Shao, W., Kawakami, R., You, S., Iida, M., Naemura, T., 2018. Classification-Reconstruction Learning for Open-Set Recognition. https://doi.org/10.48550/ARXIV.1812.04246

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? Advances in neural information processing systems 27.

You, Z., Yan, K., Ye, J., Ma, M., Wang, P., 2019. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. Advances in neural information processing systems 32.

Yu, J., Duan, S., Ye, X., 2022. A White-Box Testing for Deep Neural Networks Based on Neuron Coverage. IEEE Transactions on Neural Networks and Learning Systems 1–13. https://doi.org/10.1109/TNNLS.2022.3156620

Yu, Z., 2021. Fair Balance: Mitigating Machine Learning Bias Against Multiple Protected Attributes With Data Balancing. arXiv preprint arXiv:2107.08310.

Z. Wang, Q.Z., C. Huang, n.d. Efficient Global Robustness Certification of Neural Networks via Interleaving Twin-Network Encoding 2022.

Z. Zhong, B.R., Y. Tian, 2010. Understanding Spatial Robustness of Deep Neural Networks.

Zecchin, M., Park, S., Simeone, O., Hellström, F., 2024. Generalization and Informativeness of Conformal Prediction. arXiv preprint arXiv:2401.11810.

Zeiler, M.D., Fergus, R., 2013. Stochastic pooling for regularization of deep convolutional neural networks. arXiv preprint arXiv:1301.3557.

Zeng, Z., Liu, Y., Tang, W., Chen, F., 2021. Noise is useful: Exploiting data diversity for edge intelligence. IEEE Wireless Communications Letters 10, 957–961.

Zhai, S., Cheng, Y., Lu, W., Zhang, Z., 2016. Deep Structured Energy Based Models for Anomaly Detection. https://doi.org/10.48550/ARXIV.1605.07717

Zhan, X., Liu, H., Li, Q., Chan, A.B., 2021. A Comparative Survey: Benchmarking for Pool-based Active Learning., in: IJCAI. pp. 4679–4686.

Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O., 2021. Understanding deep learning (still) requires rethinking generalization. Communications of the ACM 64, 107–115.

Zhang, G., Zhu, A.-X., 2018. The representativeness and spatial bias of volunteered geographic information: a review. Annals of GIS 24, 151–162.

Zhang, K., Liu, G., Lv, M., 2022. RUFP: Reinitializing unimportant filters for soft pruning. Neurocomputing 483, 311–321.

Zhang, X., Cui, P., Xu, R., Zhou, L., He, Y., Shen, Z., 2021. Deep stable learning for out-of-distribution generalization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5372–5382.

Zhang, Y., Zhou, L., 2019. Fairness assessment for artificial intelligence in financial industry. arXiv preprint arXiv:1912.07211.

Zhao, F., 2017. Hanging on Every Word: Natural Language Processing Unlocks New Frontier in Corporate Earnings Sentiment Analysis. .

Zhong, G., Wang, L.-N., Ling, X., Dong, J., 2016. An overview on data representation learning: From traditional feature learning to recent deep learning. The Journal of Finance and Data Science 2, 265–278.

Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y., 2020. Random erasing data augmentation, in: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 13001–13008.

Zhou, K., Liu, Z., Qiao, Y., Xiang, T., Loy, C.C., 2021a. Domain generalization: A survey.

Zhou, K., Yang, Y., Qiao, Y., Xiang, T., 2021b. Domain adaptive ensemble learning. IEEE Transactions on Image Processing 30, 8008–8018.

Zhou, L., Fu, A., Yu, S., Su, M., Kuang, B., 2018. Data integrity verification of the outsourced big data in the cloud environment: A survey. Journal of Network and Computer Applications 122, 1–15.

Zhou, Z.-H., 2018. A brief introduction to weakly supervised learning. National science review 5, 44–53.

Zhou, Z.-H., 2012. Ensemble methods: foundations and algorithms. CRC press.

Zhou, Z.Q., Huang, D., Tse, T.H., Yang, Z., Huang, H., Chen, T.Y., 2004. Metamorphic Testing and Its Applications.

Zhu, S., An, B., Huang, F., 2021. Understanding the Generalization Benefit of Model Invariance from a Data Perspective. Advances in Neural Information Processing Systems 34, 4328–4341.

Zhu, X., Vondrick, C., Fowlkes, C.C., Ramanan, D., 2016. Do we need more training data? International Journal of Computer Vision 119, 76–92.

Zhu, X., Yang, Q., Zhao, L., Dai, Z., He, Z., Rong, W., Sun, J., Liu, G., 2022. An Improved Tiered Head Pose Estimation Network with Self-Adjust Loss Function. Entropy 24, 974.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., Fidler, S., 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, in: Proceedings of the IEEE International Conference on Computer Vision. pp. 19–27.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q., 2020. A comprehensive survey on transfer learning. Proceedings of the IEEE 109, 43–76.

Zou, Y., Yu, Z., Kumar, B.V.K.V., Wang, J., 2018. Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training. https://doi.org/10.48550/ARXIV.1810.07911

EASA

European Union Aviation Safety Agency

European Union Aviation Safety Agency

Konrad-Adenauer-Ufer 3
50668 Cologne
Germany

Mail	EASA.research@easa.europa.eu
Web	https://www.easa.europa.eu/en/research-projects/machine-learning-application-approval

An Agency of the European Union