European Union Aviation Safety Agency

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

RESEARCH PROJECT EASA.2021.C38, MLEAP

UNIFIED DELIVERABLE PHASE 2

Machine Learning Application Approval

An Agency of the European Union

Disclaimer

DELIVERABLE NUMBER AND TITLE:	MLEAP Unified deliverable
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) interim technical report, May, 2023

```
@techreport
{
        MLEAP interim technical report,
         author    = "MLEAP Consortium",
         title      = " EASA Research – Machine Learning Application Approval (MLEAP) interim technical report ",
         institution = "European Union Aviation Safety Agency",
         year      = "2023",
         type      = " Horizon Europe research and innovation programme report",
         month    = "5",
}
```

APPROVED BY:	AUTHORS	REVIEWER	MANAGING DEPARTMENT
Olivier GALIBERT	Thiziri BELKACEM Arnault IOUALALEN Swen RIBEIRO Noémie RODRIGUEZ Jean-Baptiste ROUFFET	MLEAP consortium	*Project Manager : Michel Kaczmarek* *Quality Manager : Bernard Beaudouin*

DATE: 11 May 2023

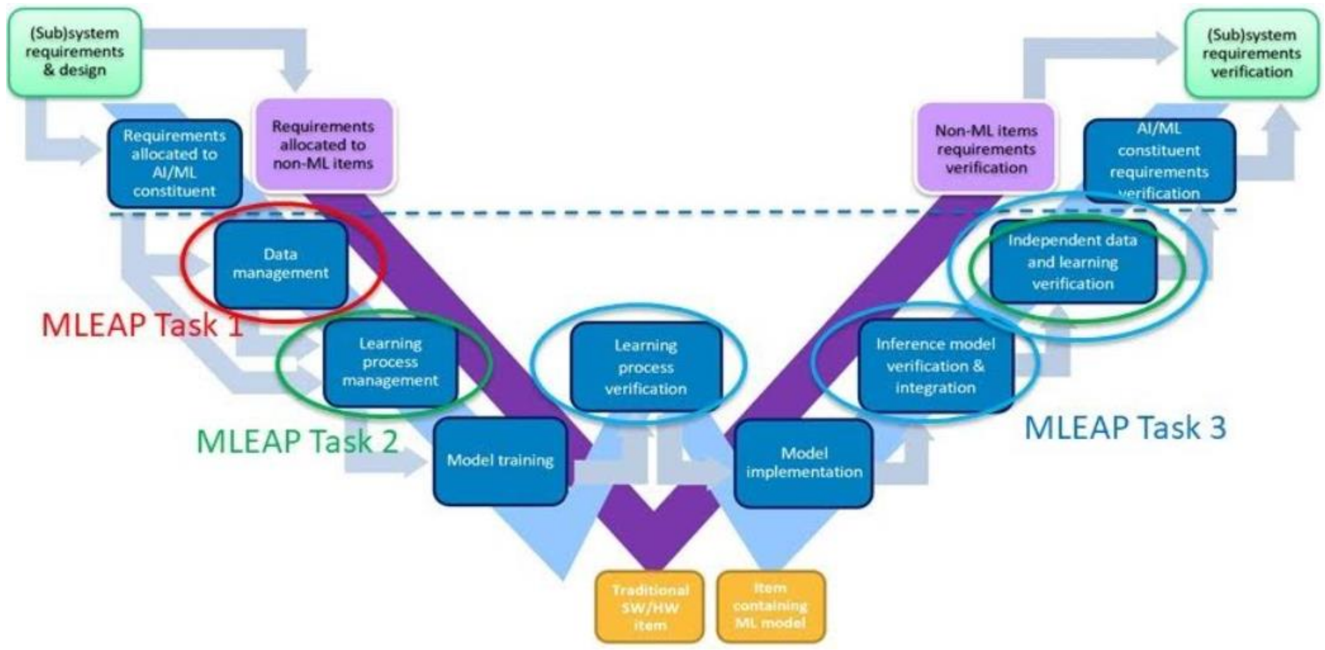
EXECUTIVE SUMMARY

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). ML systems extract and learn behavioural patterns for a given task from data, which are samples of the operational context of the considered task. However, that same learning process can make it harder for those systems 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 a first major deliverable, a Concept Paper 'First usable guidance for level 1 machine learning applications' in December 2021. This first document has been recently updated to a Proposed issue 02 which was published for consultation in February 2023 to cover level 2 AI applications. These iterative versions of the EASA AI concept paper lay down the basis of EASA future guidance for ML applications approval and identifies a number of areas in which further research is necessary to identify efficient and practicable means of compliance with the defined 'AI trustworthiness' objectives. Recently EASA updated its Roadmap 2.0, confirming the framework of learning assurance that serves as reference for the Machine Learning Application Approval (MLEAP) project.

The MLEAP project is one of the projects initiated by EASA towards those goals. This project is funded under the Horizon Europe framework. MLEAP has been tailored to investigate the challenging objectives of the W-shaped process included in the EASA AI Concept paper as per figure below:



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

The MLEAP tasks are:

- Task 1: Data completeness and representativity, with handling of the corner cases

- Task 2: Model development, through the handling of the generalization properties
- Task 3: Model evaluation, in particular in terms of robustness and stability

This interim report offers a set of anticipated concepts for the evaluation and certification of AI-based systems supporting the EASA roadmap deliverables, and help industry stakeholders in planning new strategies for deploying AI in their human and technical organisations.

Report contents

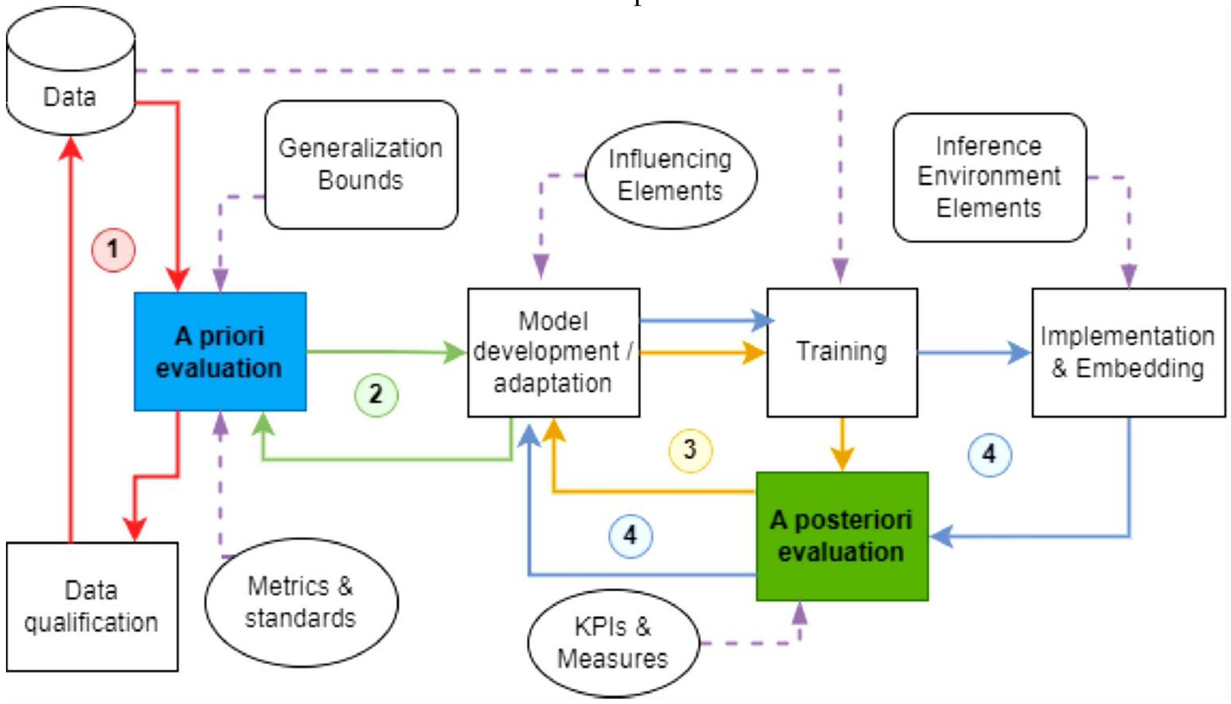
This report, built around six chapters, is the first public intermediate report issued by the MLEAP project. A second and final version will be published in a year.

The first chapter 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. Besides, it highlights the definitions and terminology the work is based upon, and also the performances evaluation metrics used in the following chapters.
The second is dedicated to the use cases description. The selected use cases provide data and models for the evaluation purpose of the different MLEAP tasks. So far, the use cases have served the different tasks analysing the state of the art and the methods selection, as well as their applicability analysis. Finally, they will be used on the following steps to evaluate the project findings.
Chapter 3 is dealing with the data management aspects of the learning assurance aspects. In particular, methodologies for trying to measure or ensure completeness and representativeness are presented and collated in a selection grid. In addition approaches to manage edge cases and corner cases are explored.
The fourth chapter revolves about generalization properties at model development time. It includes model scaling and generalization assessment and evaluation. It concludes by proposing a projection of the different methods presented onto the W-shaped development process.
Chapter 5 then explores the issues of robustness and stability with first a global view of evaluation approaches then a specific overview of formal and analytic methods as applied to models.
The document ends with a global conclusion highlighting the main findings, which is summarised in the next section.

Main results and perspectives

Data quality is a difficult topic, science-wise, because of the inherent cost which comes with doing research in the field. Completeness and representativeness are usually not handled per se, and almost no dedicated tools exist. Thus there is a need to build indicators from more general metrics (such as entropy) or by leveraging different tools (like sample similarity). Intrinsically, the domain is a difficult one, because 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 tradeoff between representativity and diversity, since rare cases need to be amplified to be modelled correctly. Hence, Chapter 3 provides analysis on the requirements for the operational design domain (ODD) to set the expectations for representativity-diversity tradeoff. Hopefully, the array of tools and methods described in the selection grid should give AI developers a chance to document and justify if the tradeoff holds.

The generalizability of trained models, assessment and evaluation is investigated, while analysing methods to avoid along the way under/over-fitting, taking into account the impact of the quality and volume of the data. 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 generalization reached by a training session. Chapter 4 ends with an operational proposal (figure below) on how to project these methods into the W-shaped approach, which should be extended to the whole set of tools and methods presented in the document for the next version.



A General Framework proposed in chapter 4 of this report to project the identified methods on the W-shaped process .

Measuring the quality of the training step takes part in the larger question of the evaluation of the resulting trained and inference models. Such an evaluation is driven by a number of guarantees that need to be gained on the models to ensure an adequate level of confidence in its intended function at a given level of performance. Chapter 5 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, in particular of stability, through an array of analytic or formal methods. Those methods, while sometimes are difficult to put in practice, allow for very powerful analysis of the behaviour of the models, including at runtime, allowing monitoring of the whole system in a live setup. Hence, these evaluation methods on the trained models robustness and performance stability can be leveraged in the pipeline developed in Chapter 4 (figure above), to ensure better performances after implementation.

The main results of EASA IPC ForMuLA report (EASA and Collins Aerospace, 2023) finalised in April 2023 will also be evaluated in the coming phase to assess how it can complement the results achieved so far by MLEAP project.

An updated version of this public report is expected to be published in May 2024, and will allow validation of the applicability of the methods on the aviation use cases we presented in Chapter 2, and focus on the actual operational usability and scaling of the various tools identified.

Description of the consortium

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

Airbus Protect is an Airbus independent subsidiary bringing together expertise in safety, cybersecurity, and sustainability-related services. As a risk management company, the aim of this entity is to offer end-to-end advisory, consulting services, training programmes and software solutions.
Pairing expertise built through large-scale projects with the latest insights from on its own research programmes, it supports customers, partners and their ecosystem 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 1,400 experts based in France, Germany, the UK, Spain and Belgium, to create a centre of excellence to meeting the clients evolving needs. Airbus Protect combines more than 35 years of experience with industry-leading expertise to deliver services in three areas: Cybersecurity, providing a consulting and managed security services to help our clients to establish and maintain persistent cyber resilience; Safe Mobility ensuring the safety of tomorrow's smart mobility solutions and smart cities; Sustainability developing new ways of working, new products and zero-emission energy supplies.
As part of these three pillars of Airbus Protect, the Digital Transformation Office (DTO) is in charge of the development of digital services, from digital transformation to Artificial Intelligence applications proposal. The DTO provides support through the services of Airbus Protect. Its activities concern software engineering for aeronautics, product support, and the research and development of new technologies and AI solutions, for various industrial issues. The DTO implements several Data/AI/engineering projects, including MLEAP:

- SmartPlanif / MaiVA (Maintenance Virtual Assistant): an airline-centric tool, supporting customers by automating/providing an increased level of assistance to activities.
- Climate and energy challenge: aims to provide structured and semantic access to a large amount of data on the climate and energy ecosystem.
- 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 as well as 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 the field of testing, evaluation and metrology. Its action aims in particular to examine new products and assess their impact in order to inform, protect and meet the needs of consumers and national industry. In this context, it carries out measurement, testing, characterization 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 the evaluation of 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 major 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), which aim first and foremost to define standards, 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 in a consensual manner the first AI certification standard

Numalis is a software editing company specialized in the topic of reliability of AI systems. The goal of Numalis is to allow companies to accelerate on the path to adoption of AI, by allowing its design, validation, integration and deployment to be more reliable. Numalis is involved in several industries such as the Defense, the Aeronautic, the Aerospace, the Railway. For them Numalis provides a unique set of tools and expertise in order 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, visualize in a human readable fashion the robustness across the input space and extract explainability components from the system. As robustness and explainability are key components in most software quality models as well for the future EU regulation (the AI Act), Numalis aims at develops also standards at the international level to bring uniformity to processes across all industries. These standards are written in order to bring good practices on the use of formal methods on AI and to that effect, Numalis is currently the editor of ISO/IEC standardization documents (the ISO/IEC 24029 series) related to the assessment of the robustness of neural networks. Founded in 2015 in Montpellier, Numalis employs 18 persons which are mostly PhDs and engineers specialized in formal methods and software development.

CONTENTS

1. Introduction 14
   1.1 The MLEAP project 14
   1.2 Core definitions 14
   1.2.1 Data completeness and representativeness 14
   1.2.2 Generalization guarantees 17
   1.2.3 System robustness and stability 18
   1.3 Evaluation metrics 27
   1.3.1 Regression 28
   1.3.2 Classification 30
   1.4 Document structure 33
2. Use cases description 34
   2.1 Summary of the UCs 34
   2.2 Speech-To-Text for Air Traffic Control 37
   2.2.1 Background and ATC-STT Description 38
   2.2.2 Speech Data 40
   2.2.3 STT Model 40
   2.2.4 Challenges and Issues 42
   2.3 Automated Visual Inspection (AVI) 45
   2.3.1 Description of the AVI application 45
   2.3.2 Provided Data Sets 46
   2.3.3 Model and Approach Description 46
   2.3.4 Challenges 47
   2.4 ACAS Xu 48
   2.4.1 Description 48
   2.4.2 Data 49
   2.4.3 Model 49
   2.4.4 Objective and Challenges 50
   2.5 Conclusion 50
3. Data: representativeness and completeness 52
   3.1 Introduction 52

3.2 Concepts 53
3.2.1 Introduction 53
3.2.2 Generalities on data quality requirements 53
3.2.3 Definitions of completeness and representativeness 54
3.2.4 Considerations on data dimensionality 54
3.3 Technical requirements 56
3.3.1 Introduction 56
3.3.2 Intended behaviour 56
3.3.3 Model architecture 57
3.3.4 Data dimensionality 57
3.3.5 Intended level of autonomy 58
3.3.6 Intended level of performance 58
3.3.7 Intended levels of robustness and resilience 59
3.3.8 Intended levels of stability 59
3.3.9 Methods and tools for assessment 60
3.4 Processes 65
3.4.1 Introduction 65
3.4.2 Data management requirements 65
3.4.3 Data quality improvement (in Data preparation) 65
3.4.4 Data synthesis (in Data preparation) 66
3.4.5 Data sampling (in Data preparation) 66
3.4.6 Labelling (in Data preparation) 67
3.4.7 Pre-processing 67
3.4.8 Methods and tools for assessment 68
3.5 Relation to other data quality requirements 86
3.5.1 Introduction 86
3.5.2 Balance 87
3.5.3 Relevance 88
3.5.4 Diversity 91
3.5.5 Currentness 93
3.5.6 Other aspects of data quality 95
3.5.7 Conclusion 96
3.6 Selection grid 97
3.6.1 Introduction 97
3.6.2 Methods summary 98
3.6.3 Selection grid 102
3.6.4 Technical requirements 104
3.6.5 Processes 105
3.6.6 Other data quality requirements 106

3.7 Preliminary experimentations 107
3.7.1 Experimental setup 107
3.7.2 Selected methods: motivations and expectations 107
3.7.3 Selected methods: Experimental protocols and results 109
3.7.4 Discussions on other methods 125
3.7.5 Experimentation conclusions 126
3.8 Conclusion 126
4. Model development: Generalization properties 128
4.1 Introduction 128
4.2 Background concepts 128
4.2.1 AI modelling 128
4.2.2 Learning process 129
4.2.3 Error metrics 130
4.2.4 Robustness 130
4.2.5 Generalizability 130
4.2.6 Stability 131
4.3 Overfitting and underfitting 131
4.3.1 Overfitting vs Underfitting 131
4.3.2 Under/Over-fitting detection 132
4.4 Generalization in AI 134
4.4.1 Model complexity 134
4.4.2 Generalization Error Bounds 136
4.4.3 Generalization in ML 140
4.4.4 Generalization in DL 142
4.4.5 Domain generalization 145
4.5 Methods to boost generalization 151
4.5.1 Regularization 151
4.5.2 Penalty Methods 159
4.5.3 Network reduction 159
4.5.4 Data Expansion 161
4.6 Methods and tools for generalization evaluation 168
4.6.1 Random labelling 169
4.6.2 Data corruption 169
4.6.3 Evaluation approaches 170
4.7 Limitations of existing methods and discussion 171
4.7.1 Misunderstanding of the generalization bounds 172
4.7.2 Practical issues preventing generalization 172
4.7.3 Expectations from evaluation vs reality 175

4.7.4 ML/DL testing limitation and challenges 177
4.8 Towards application independent ML development pipeline to promote generalizability 178
4.8.1 Research questions and roadmap 179
4.8.2 Generalization Guarantees Selection 180
4.8.3 Methods Applicability Analysis 186
4.8.4 Generic evaluation approach 186
4.9 Conclusion 191
5. Model evaluation: robustness and stability 192
5.1 Introduction 192
5.1.1 Background 192
5.1.2 Scope of the chapter 192
5.2 Robustness and stability 192
5.2.1 Robustness and stability concepts 192
5.2.2 Robustness and stability properties already available 192
5.2.3 General framework of methods to assess robustness and stability 199
5.3 Review of methods and tools 203
5.3.1 Statistical methods 203
5.3.2 Formal methods 208
5.3.3 Empirical methods 217
5.3.4 Selected tools and methods 221
5.4 Conclusion 222
6. Conclusions 223
7. References 224

ABBREVIATIONS

ACRONYM	DESCRIPTION
AI	Artificial Intelligence
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
DL	Deep Learning
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 'First usable guidance for level 1&2 Machine Learning applications'
EDA	Easy Data Augmentation
ELA	Equalized Loss of Accuracy
ERM	Empirical Risk Minimization
GA	Genetic Algorithm
GAN(s)	Generative Adversarial Network(s)
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
KNN	K-nearest-neighbors
KPI(s)	Key Performance Indicator(s) (index)
LDA	Linear Discriminant Analysis
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)
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
PCA	Principal Component Analysis
RBF	Radial Basis Function
RNN	Recurrent Neural Networks
ReLU	Rectified Linear Units
SGD	Stochastic Gradient Descent
SLU	Spoken Language Understanding
SMT	Satisfiability Modulo Theory
STT	Speech-to-text
SVM	Support Vector Machines
TL	Transfer Learning
Tanh	Hyperbolic Tangent Function

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 important or even critical systems poses not-yet solved problems, in particular in the field of aeronautics which is at the core of 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: Generalization properties
- Evaluation: robustness and stability

The aim is to build a reference document on those topics. This is the first public version of the document, a second and final version will be issued by May 2024.

1.2 Core definitions

Those topics happen to have variable definitions from one document to another. In order to clearly scope the boundaries of this document we will first refine which definitions we are going to base our work on.

1.2.1 Data completeness and representativeness

The objective of this present section is to analyse to what extent literature may diverge in the acceptation of the notions, or in the scope they cover. Indeed, the fact that some aspects related to quality attributes are not properly defined in the community considerably hinders the research for appropriate methods, in the present deliverable first, 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.

In (EASA, 2023), the notions are defined as follows in the 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 Concept Paper, which allows the reader of the guidance 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 generalization guarantees for the trained model*" (section C.3.1.3.8).

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

1.2.1.1 Completeness

- EASA 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 literature (e.g. "context of use"), hence not allowing comparison on this aspect.
- In EASA's acceptation, completeness must be considered relatively to: Operational Design Domain (ODD); ML requirements; ConOps; intended application. However, ISO/IEC 25012:2008 only refers to "a specific context of use", which is not strictly defined in the standards nor in 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 provides a definition for 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).

Scientific literature does not always offer an explanation or definition of 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 the assessment of 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 the presence 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 types of missing values: (i) null-based missing values - where missing values are represented by *nulls*; (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 in comparison with a reference population[1]. 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, or 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 notion of 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 quite remotely, by specifying that completeness should be tackled differently depending on specific usage contexts. It seems obvious that a first main step in the definition of operational methods of completeness assessment should encompass the formalized description of what are such specific usage contexts. Notwithstanding the vagueness of the notion of "context of use", the ISO 25012 definition seems coherent with EASA definition.

1.2.1.2 Representativeness

- EASA definition of representativeness is in line with literature and standards, with the exception of EASA covering learning, validation and test data sets, while the other sources only consider training data.
- In literature, representativeness seems sometimes 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). The standard (ISO/IEC CD 5259-2, 202X), in its present state, refers to the degree to which a data set used for training reflects the target population under study. The target population is characterized through an analysis of the data expected in the target conditions of use of the AI system.

Scientific literature does not provide strict definitions, but the works articulate around similar notions whatever 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 provide a definition of representativeness; however, the study shows that it is linked to the distribution of the population in terms of societal and demographic characteristics, a distribution that is 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 the sense that the data sets used for training should be representative of 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 normally present in the population but for which research does not provide enough tools for tracking them.

One should note that the notion of representativeness is linked to the notion of relevance (see Section 3.5.3), 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,

[1] 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.

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 target population of use. It seems then compulsory that the notions be distinguished, and that the intended scope of representativeness is confirmed.

1.2.2 Generalization guarantees

In this document, the following definitions, largely aligned with the literature, are used:

Generalizability: Machine learning model generalization is the capacity of a ML model to keep an acceptable level of performance on unseen input data (during training phase) from the ODD.

Overfitting: Overfitting is a concept in data science, which 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 generalize well, as it learns the noise and patterns of the training data to the point where it negatively impacts the performance of the model on new data.

Underfitting: Underfitting occurs when a model doesn’t have the capacity to generalize well on new data or when the model is not able to create a mapping between the input and the target variable.

In this document, the terminology related to models development and evaluation is based on the following definitions, largely used in the literature, and that we have adapted. More formal descriptions of these concepts can be found in chapter 4.

Operational Domain (OD): As part of the ConOps definition, the OD corresponds to the complete description of the conditions on which the whole system (the AI part included) should operate, along with the operational scenarios (EASA, 2023). 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 properly operate. For every application, the ODD is an abstraction of the operational context and needs to be known in order to state guarantees on performance and safety (Heyn et al., 2022). According to the EASA’s recommendation (EASA, 2023), 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 need to define the technical requirements and initial conditions, under which the system is designed to function.

Model architecture: Represents a set of design features characterizing 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 in statistical learning definitions of data dimensionality (Hastie et al., 2009), this feature can be an influencing parameter on 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), are 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 wide data, which refers to data with numerous features irrespective of the number of observations (similarly, tall data is often used to denote data with a large number of 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 important 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: it refers to the characteristic defining how sophisticated/complicated the model is (Verhagen, 2021). Several ways can be explored to define the model complexity, the main feature relies on the number of trainable variables and the required amount of data needed to train a performing model.

1.2.3 System robustness and stability

Robustness and stability do not necessarily constitute clear and unambiguous notions depending on which document the reader is referring to. In this section we consider both the literature coming from the EASA (and the WG114 [2]) and the one from ISO/IEC JTC 1/SC 42 [3]. By comparing the definition present in which we could first draw the following table of correspondence.

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

[3] For clarity the rest of the document will refer it at ISO/IEC instead of the full denomination ISO/IEC JTC 1/SC 42. When designing another subcommittee of the ISO or the IEC its full name will be used.

1.2.3.1 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 small variations in method parameters and provides an indication of 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 on the stability of the learning algorithms. Stability of the trained model: Refers to keeping input-output relations of the model under small perturbations **Objective LM12**: The applicant should perform and document the verification of the stability of the trained model.	**General definition**: 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 pertaining to 'model stability' and the ones pertaining to 'robustness in adverse conditions'. Robustness of the trained model **Objective LM-13:** The applicant should perform and document the verification of the robustness of the trained model in adverse conditions. Robustness of the inference model

Definition	Stability	Robustness
	Stability of the inference model: **Objective IMP-08:** The applicant should perform and document the verification of the stability 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 largely absent in the ISO/IEC information technology technical sector literature (even outside the subcommittee studying AI). It refers most usually 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 some degree to the performance of the ML model holding even in presence of 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, while 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, whereas ISO/IEC tends to unify them under the same name of robustness. For EASA CP robustness resides in the notion of input adversity, whereas stability is more focused on normal inputs. ISO/IEC views them in the same way since the robustness has to be defined in "any" condition, so it is both true for adverse or normal inputs.

In the rest of the document the term of 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.

1.2.3.2 Corner and Edge Case Concepts

In this section we start with introducing 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 being used in the literature not always in the same ways. This is why in section 1.2.3.2.4 the document will attempt to draw the relationship between the state of the art and the EASA Concept Paper definitions. Corner and edge cases are important to consider since the variety of use cases considered in the MLEAP project (see Chapter 0) will explore several types of them.

1.2.3.2.1 Definitions from the EASA Concept Paper

The EASA concept paper (EASA, 2023) uses several concept related that can be tied to the notion of corner cases. In particular it revolves around the following definitions:

Corner case	Relates to a situation that, considering at least two parameters of the AI/ML constituent ODD, occurs rarely on all of these parameters (i.e. low representation of the associated values in the distribution for those parameters).

Edge case	Relates to a situation that, considering a given parameter of the AI/ML constituent ODD, occurs rarely (i.e. low representation of the associated value 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 in order 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 which is 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 DM-01-01) 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, corner cases data in preparation of 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 that are expressed in the EASA CP relate directly to this notion. For example, IMP-08 (and its anticipated MOC IMP-08-01) mentioned the verification of the inference model in adverse condition, which concerns also edge and corner cases. The requirement LM-13 (and its anticipated MOC LM-13-1), which is its symmetrical for the trained model which specify the same requirement.

1.2.3.2.2 Vocabulary and Concepts for the State of the Art

When dealing with corner cases, several words and concepts are commonly used in the bibliography and it is then necessary to make distinctions between corner cases and other terms used in the literature (Heidecker et al., 2021). First of all, in the context of software and hardware testing, the terms edge case is used. 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 the development (Koopman et al., 2019). The concept of edge cases also covers extreme cases or boundary cases. When edge cases are taken into account by the designers of the system, they become normal cases and are not any longer considered as edge cases.

Next, the terms outlier, novelty, anomaly are often used in ML literature. They are strongly related to corner cases and may have an overlapping meaning. A summary is given in Figure 1. Literature also offers a wide variety of approaches for the detection of anomalies and novelties (Hodge and Austin, 2004; Kumar et al., 2009; Pimentel et al., 2013).

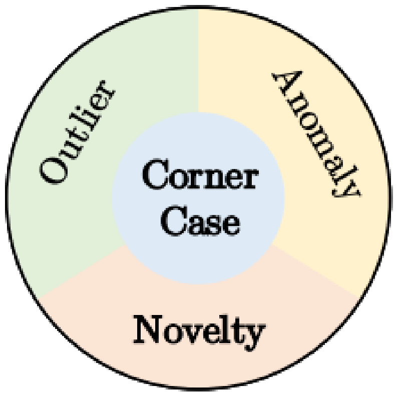


Figure 1. Relations between outliers, anomalies, novelties and corner cases (borrowed from (Heidecker et al., 2021)).

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 agglomeration of anomalies or as 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 artifacts which in turn 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). A categorization of anomalies is given in (Kumar et al., 2009).

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

- A corner case may result from the combination of several normal situations or parameters that coincide simultaneously, 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 cases may result from a deviation from normality that is 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 systematization of corner cases as well (Fingscheidt et al., 2020).
- A corner case set has the potential to 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 2.

- Another definition is based on prediction: Following (Fingscheidt et al., 2019), a corner case arises if there is a relevant object in a relevant context that a modern system cannot predict.

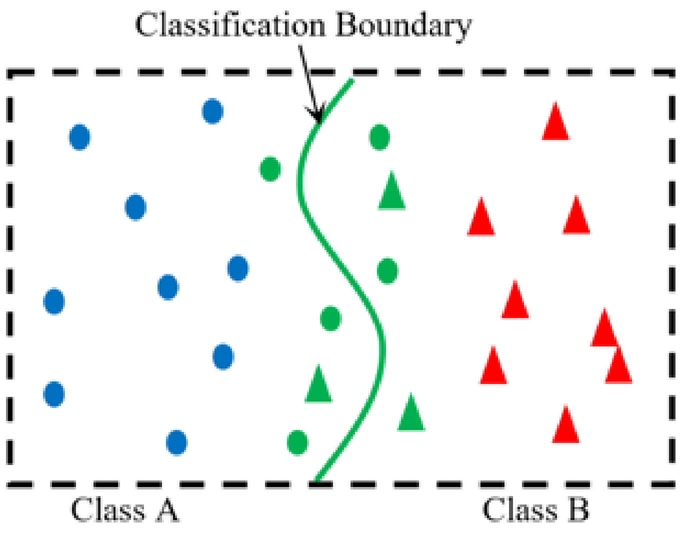


Figure 2 - Illustration taken from (Tinghui Ouyang, 2021) showing corner cases (green) which can be near the classification boundary and include both correct and incorrect classification.

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 the training and validation data of a perception method. In ML, corner cases help to validate, but also to improve the systems by re-training. The recorded data of such a situation can be used as part of the training data for the ML system which can use in the case of an active learning approach.

1.2.3.2.3 Systematization of Corner Cases on Different Levels

Interestingly, the authors of (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 levels. While this work is dedicated to visual perception, it represents a strong basis for corner case systematization in other sectors.

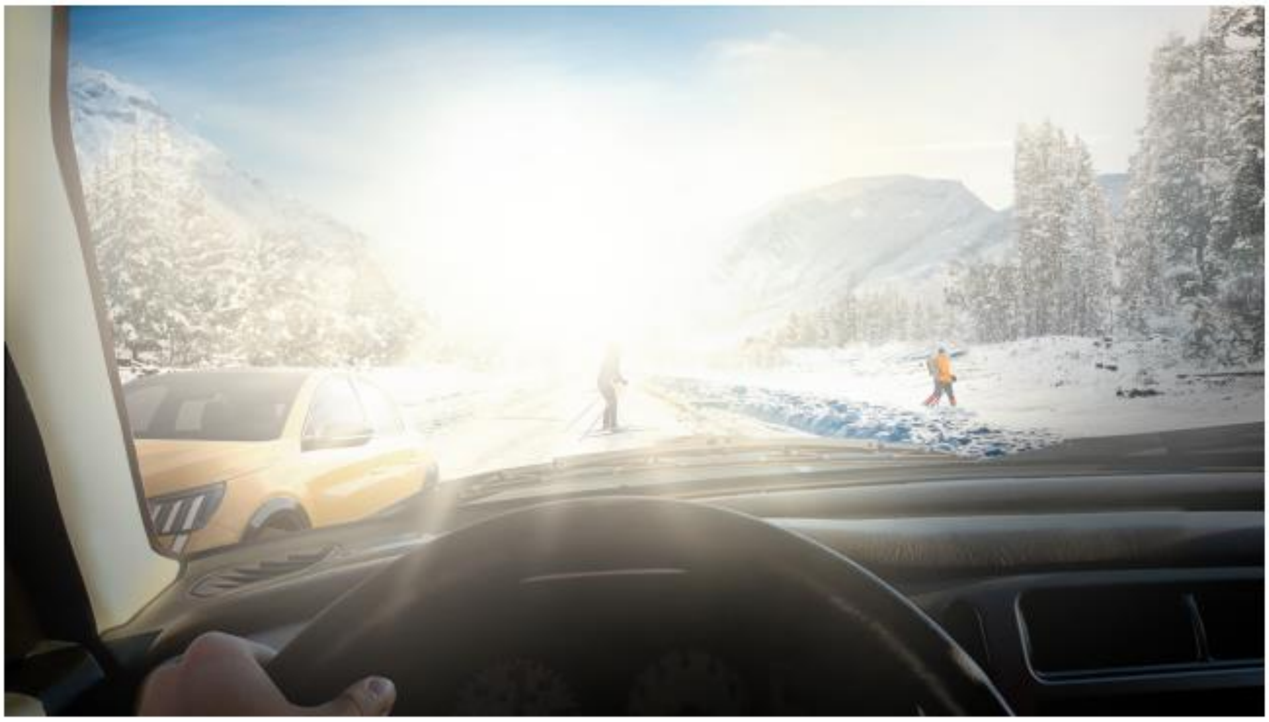


Figure 3. Multiple corner cases: A winter scene with an icy, slippery, reflective road, combined with low winter sun and people on cross-country skis crossing the road (figure borrowed from (Heidecker et al., 2021).)

These levels are based on the type of situation they encompass and ordered by their theoretical complexity of detection. The authors consider corner cases on the scene (see Figure 3 borrowed from (Heidecker et al., 2021)), object (e.g., people on cross-country skis), and domain level (e.g., snowy winter), which they summarize into the content layer. In addition, for corner cases at the scenario level, the authors define the temporal layer, 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 concern single image frames and point clouds (content layer), or multiple consecutive ones (temporal layer.) As the goal is to provide a more comprehensive conceptualization 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.)

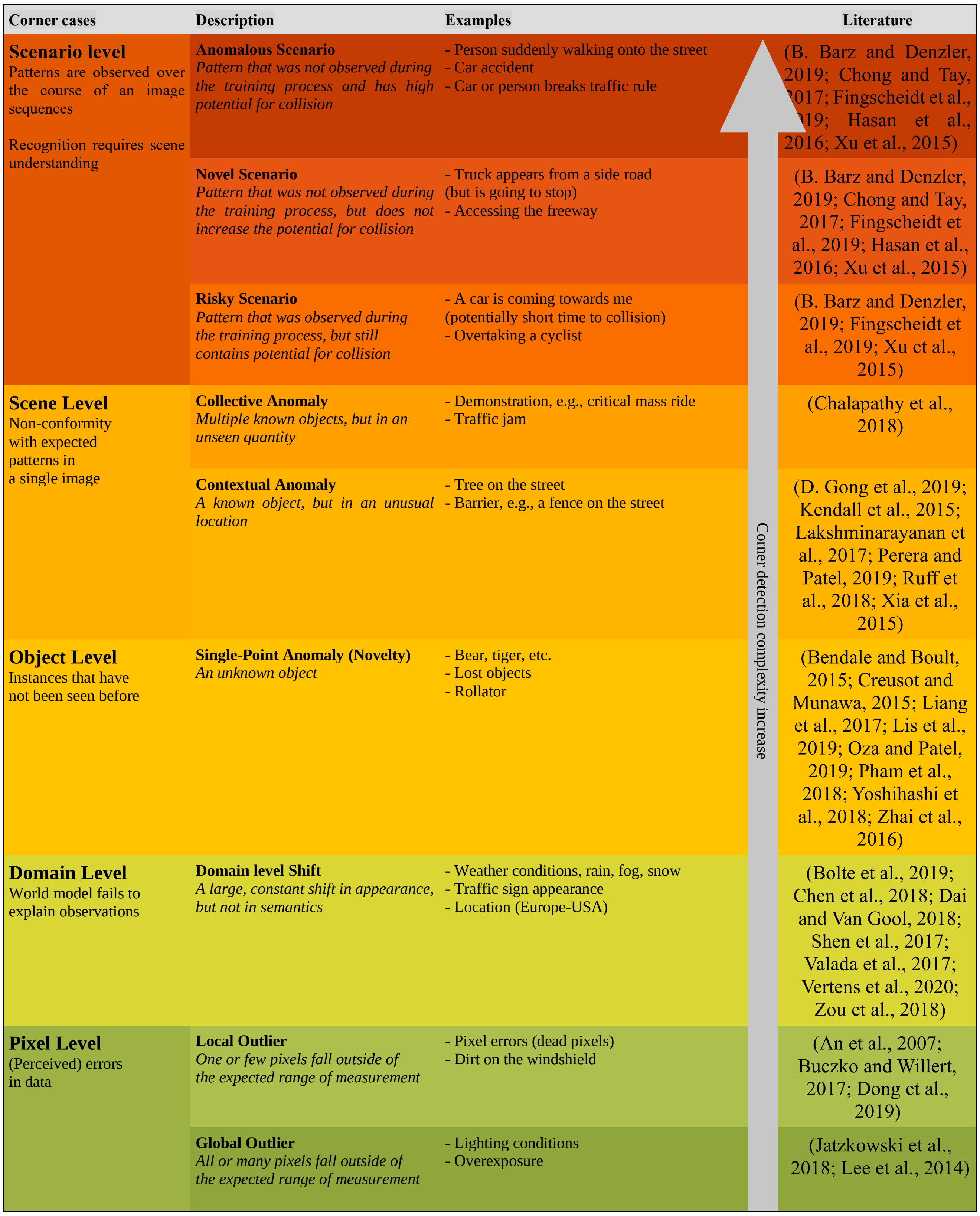
Let us follow the order in Figure 4 (borrowed from (Fingscheidt et al., 2020)) by going from low to high complexity.

- The pixel-level corner cases are caused by perceived errors in the data. This type of corner case is at the bottom of Figure 4 because the detection complexity is relatively low. At the pixel level, we distinguish two types of outliers. Global outliers occur when all or many of the 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 the case of domain-level corner cases, the world model fails to explain its observations. Domain-level shifts are typically the cause of this type of corner case where a large and constant shift occurs in the appearance but not in the semantics. Methods for the detection of domain-level corner cases are often related to the field of domain-level adaptation (Bolte et al., 2019).
- At the object level, corner cases arise when instances that were 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 theoretical increase in detection complexity is induced by the need to understand the whole perceived scene. We distinguish two types of corner cases according to the type of understanding necessary for the detection.
    - Contextual anomalies are known objects, which appear in unusual locations of the scene.
    - Collective anomalies are known objects, which appear in an unusual quantity, for example a large gathering of people such as 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 as well, requiring an image sequence for detection. We consider three different cases.
    - Risky scenarios occur when a pattern observed in similar form during training appears again during inference and still contains a potential danger.
    - Novel scenarios occur when a pattern that was 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 approaches can be defined for different type of use case, Sections 3.4.8.8.2 and 3.4.8.8.3 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, 017; Fingscheidt et al., 19; 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)

Figure 4. 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).)

1.2.3.2.4 Comparison Literature and Concept Paper

Table 2 present a comparison between the literature and the concept paper concepts revolving around the notion of corner cases. The definitions are extracted from the literature that was presented in the whole Section 1.2.3.2.2, the reader should refer to the previous sections to match it with its 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 edge cases are taken into account by the designers of the system, they become normal cases and are not any longer considered as edge cases."	Relates to a situation that, considering a given parameter of the AI/ML constituent ODD, occurs rarely (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 literature, as seen in the former paragraphs, a general definition and description is missing. The lack of agreed technical terms and definitions also makes detection of corner cases cumbersome."	Relates to a situation that, considering at least two parameters of the AI/ML constituent ODD, occurs rarely on all of these parameters (i.e. 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 agglomeration of anomalies or as a change in the distribution of an already known process."	"Data which is within the ML Model ODD according to the existing ML model ODD parameters, but which 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 in general due to a lack of characterization of the ML model ODD. It could be integrated to 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 that deviates so much from other observations that it suggests that it was generated by a different mechanism."	"Data which is outside the ML model ODD."

Table 2. Comparison between the literature and the EASA concept paper on topics related to corner cases.

1.3 Evaluation metrics

Machine learning models are evaluated using metrics to analyse their performance on top of the loss function which is used to optimize the model during the training phase. Hence, different metrics can be used to evaluate both: the generalization performances to unseen data during training, and the performances 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 generalizability is assessed based on unseen data samples during training, and the robustness can be assessed based on corrupted data samples. Both generalization and robustness assessment can be performed using the same metrics, that can be divided in the two main problem categories: *classification* and *regression*.

1.3.1 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 y be the predicted (computed) output value (scalar or vector) by the model f, and $\hat{y}$ is the expected output (truth). The classical error value E is computed by $(\hat{y} - y)$, for every instance in the data set of n training samples. As mentioned before, both robustness and generalization performance indicators could be assessed using same metrics. Table 3 shows an overview of the most popular evaluation metrics, based on E, for different ML applications. For these metrics, the generalization 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 generalizing better is the one that returns a good answer in the good ranking). While 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	$Max(\|E_i\|, i = 1 \dots n)$	Max Error is either an absolute or a relative metric calculating 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)	$MAE = \frac{\sum_{i=1}^{n} \|E_i\|}{n}$	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)	$RAE = \sum_{i=1}^{n} \frac{\|E_i\|}{\|\hat{y}_i - \hat{y}_{mean}\|}$ Where $\hat{y}_{mean}$ 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).

Metric	Formula	Description
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)	$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - \hat{y}_{mean})^2}$	
Scatter index (SI)	$SI = \frac{\sqrt{\frac{\sum_{i=1}^{n}\left(y_{\max(\hat{y})} - y_{\max(y)}\right)^2}{n}}}{y_{\max(y)}}$	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)	$PI = \frac{\sqrt{\frac{1}{n}\sum(\hat{y} - y)^2}}{1 + R}$	It is an indicator for the evaluation of predictivity of a model. Lower PI values result in more accurate model predictions (Rifai, 2021).

Table 3. 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. Utilizing a multi-criteria process is not only 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, that corresponds 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.

1.3.2 Classification

Classification applications aim at categorizing 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 5) 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 recognized or the type of errors committed by the classifier.
The confusion matrix is a square matrix where entry e, at 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 can be derived from confusion matrix elements, such as entropy of the histogram represented by the matrix.

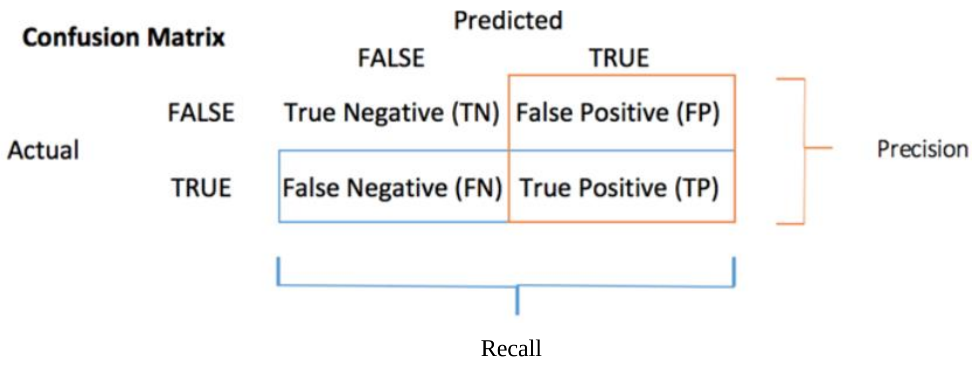


Figure 5. Construction of a confusion matrix[4] for a binary (false, true) classification.

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 4, most of the measures described work with categorical data. Furthermore, the generalizability and robustness of the model can be evaluated using the same metrics, where the well-generalized model is the one able to predict right classes for instances unseen during training, while a robust model is the one that keeps right classification scheme regardless 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. Has an ideal value of 1 and the worst value of *zero*.
Positive likelihood ratio (LR$^+$)	$LR^+ = \frac{TP}{TP + FN}$	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	$F\beta = 1 + \beta 2 \times \frac{precision \times recall}{\beta 2 \times \mathrm{precision} + \mathrm{recall}}$	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

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

Metric	Formula	Description
		$MCC \cong 0$, the model tends to make random predictions instead.
Average precision (AP)	$AP = \sum_r (recall_r - recall_{r-1}) \times precision_r$ Where 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)	$LLE = -\sum_{i=1}^{M} \mathrm{c_i} \log(y_{c_i}) + (1 - \mathrm{c_i}) \log(1 - y_{c_i})$ Where M is the number of classes, $\mathrm{c_i} \in \{0,1\}$ if the current class is the correct one, and 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 penalizes 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	$lift = \dfrac{\frac{TP}{TP + FP}}{\frac{TP + FN}{TP + TN + FP + FN}}$	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)	$MXE = -\frac{1}{n} \sum_{i=1}^{M} \hat{y}_{\mathrm{c_i}} \ln(y_{c_i}) + (1 - \hat{y}_{\mathrm{c_i}}) \ln(1 - y_{c_i})$ Where $\hat{y}_{\mathrm{c_i}}$ and 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 where the output is a probability between 0 and 1. The objective in general is to minimize this value to get a better likelihood.

Table 4. 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.

1.4 Document structure

Following this introduction, which defines the scope of the studies, the next chapter presents the use cases that will be used to illustrate the described methods during the next phase of MLEAP project (from May 2023 to May 2024). Then the three following chapters are the main work on the three data, model and evaluation aspects. The document ends with a global conclusion summarizing the main findings.

2. Use cases description

2.1 Summary of the UCs

In this section, we provide an overall description of the different use-cases chosen for the evaluation part on the MLEAP project. These use-cases may be used as sandboxes for testing some of the methodologies identified in following chapters. Table 5 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. While confidential details have been omitted and replaced by open-source examples (models and data sets), to allow the reproducibility of some results by audience of this deliverable. Last, there is also 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.

With respect to the recommendations in the concept paper by EASA (EASA, 2023), to guarantee compliance with the objectives of the AI reliability guidelines, an overview of 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. Hence, 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 function that the system is made to perform. In machine learning (ML), almost all automated tasks can be designed as a function.
- **System description:** represents a set of functional and technical characteristics that could help for better understanding the structure of the designed system.
- **ML component description:** this part will describe more precisely the ML component (the model and its in/out-puts).
- **Data type:** this could be even text, signal, time series, 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 that are targeted for an intended system performance: noise types, airports with specific checkpoints names, accents, and speech rates.	The data[5] represents input points of the lookup tables from the RTCA SC-147 resources[6]. 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 than for a human inspection
Model architecture	Different architectures: LSTM, CNN, 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 4.4.1) apply for this.	The models are made of simple (shallow) neural network architectures. Hence, methods based on NN architecture, for identifying the model complexity apply for 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.

[5] https://hal.archives-ouvertes.fr/hal-03355299/document
[6] 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 the input situations where the NN and the LUT predictions are different, are considered as 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 sub-system 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; Final softmax layer discriminate, according to the input, what is the heard phoneme	the ML model is composed of several elements more precisely, 45 NNs	Transfer learning using already trained network from ImageNet project (e.g. VGG16); Completion of the transfer learning network with a couple layers with a couple of hidden layers; Final layer is a cross-entropy layer; Adam optimizer 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	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 Typical example using VOTT: The image is not modified, the labelling is supported by

	ATC-STT	ACAS Xu	AVI
	- Not subject to distribution or sharing for non-involved third parties		a metadata 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[7] processes	**System requirements – Complex background noise** PESQ[8] evaluation score representative of operational conditions. PESQ returns[9] 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:** First targeted performance: focus on true positives ~ 80% accuracy. Detection accuracy needs to be tested across different conditions: 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 (lightening and trick).

Table 5. 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.

2.2 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 recognize what a person speaks, and the STT refers to the fact that it can convert speech into text (Mittal and Sharma,

[7] 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 are in place to ensure that applicable safety objectives are met.
[8] PESQ stands for Perceptual Evaluation of Speech Quality. It helps evaluating https://en.wikipedia.org/wiki/Perceptual_Evaluation_of_Speech_Quality
[9] Based on the state-of-the-art of ATC-STT (Lin et al., 2019)

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[10]. 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.

2.2.1 Background and ATC-STT Description

In ATC, the ASR-STT application is more than important to handle several daily challenges and problems (Lin, 2021). Provided that the spoken instruction is transmitted in an analog 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 indicates the evolutions of the flight and traffic in the near-future. These information pieces are highly significant to the air traffic operation and to detect the communication errors that may cause potential safety risks. One of the most known applications is the 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 6. This latter shows the general framework for speech recognition and textual transcription, where main modules that constitute an ASR application can be defined[11] 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 store it to utterance[12] 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 object and structure of lexical 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).

[10] Check HuggingFace models

[11] A set of basic ASR definitions are provided in the WIKI-Speech Recognition HOWTO.

[12] 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.

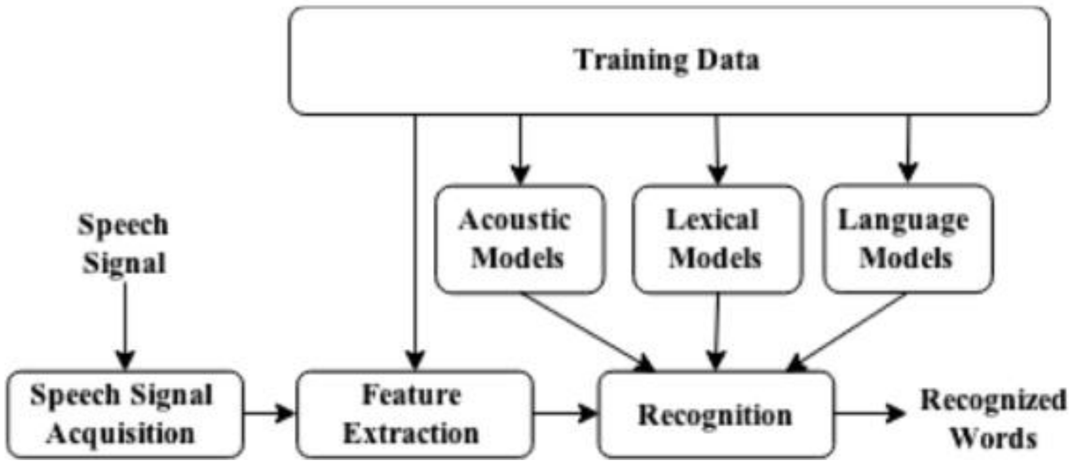


Figure 6. A domain-independent pipeline for ASR applications (Nguyen and Holone, 2015)

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 be understood and correctly interpreted. Known as code-switching (CS) (Mustafa et al., 2022), this challenge in spoken languages needs to be addressed in ASR systems to recognize speech in bilingual and multilingual settings, where the accuracy of the system declines with CS due to pronunciation variation. The diagram in Figure 7 shows the different research directions and taxonomy of models dedicated to different ASR applications in ATC. In 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 Systems (SRS), their 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[13]. 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, where the aim is to correctly recognize all the communicated information pieces, such as those describing movements and manoeuvre instructions received from the AT controllers (ATCO). A more extensive taxonomy of different research objectives and design 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.

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

2.2.2 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 containing:

- 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 analysis on different speech data using several accents. The objective is to enable public sharing of experimental results within the MLEAP's consortium and the EASA's publishable documents. Table 4. 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 6. Open-source ATC-STT data sets with provided text transcriptions for training and testing.

2.2.3 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[14] (Povey et al., 2011). Several automatic speech recognition engines have been built and tested, where the acoustic model is made of a Hidden Markov Model (HMM)[15]. The global architecture, presented in Figure 5, follows three main steps, as follows:

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

[15] Hidden Markov Model is a statistical model based on the Markov chain, for analyzing probabilities of sequences of random variables (Elliott et al., 1995).

- Mel filter-bank and Cepstral coefficients (MFCC)[16] as input features along with i-vectors representations;
- Kaldi with HMM and TDNN[17]/LSTM[18]-HMM chain system used to build the acoustic model;
- The SRILM[19] 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 in order to assign a probability for each word sequence (Chen and Goodman, 1999). The phonetic dictionary comes from the Carnegie Mellon University Pronouncing Dictionary[20] (CMU). In the context of ATC communication, some specific preprocessing is adopted in order to deal with named entities (typically city names) when they do not appear in the CMU dictionary to automatically output the pronunciation of out-of-vocabulary words.
Before decoding, knowledge learned in both AM and LM is combined in a Weighted Finite-State Transducer (WFST) (Mohri et al., 2002) based graph. Decoding consists in 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 further train the HMM. The TDNN/LSTM-HMM system adopts the same general architecture used in traditional speech recognition engines (cf. Figure 7) using hybrid DNN-HMM acoustic modelling. This deep network consists of a particular combination of TDNN[21] 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, and while system running;
2. ***Preprocessing and Feature extraction:*** after splitting the speech into a 2D signal and chopped into 20-25ms sized frames with 10ms shift, it 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.

---

[16] Represents a set of features that correspond to the useful data signals http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
[17] 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.
[18] https://intellipaat.com/blog/what-is-lstm/#no1
[19] http://www.speech.sri.com/projects/srilm/
[20] http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[21] https://kaleidoescape.github.io/tdnn/

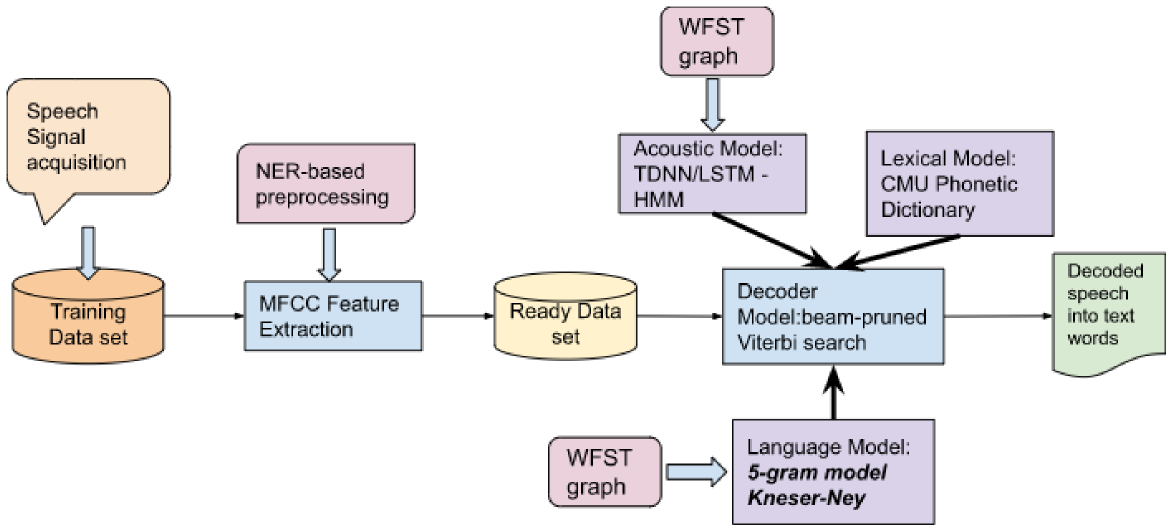

WFST graph
Speech Signal acquisition
NER-based preprocessing
Acoustic Model: TDNN/LSTM - HMM
Lexical Model: CMU Phonetic Dictionary
Training Data set
MFCC Feature Extraction
Ready Data set
Decoder Model:beam-pruned Viterbi search
Decoded speech into text words
WFST graph
Language Model: *5-gram model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

Figure 7. Abstraction[22] 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 in the scope of the MLEAP project, and that will be meant for machine learning applicability approval. The ATC-STT use case aims to correctly transcript the spoken instructions during the interactions between the control tower and the crew.

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

[22] 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.

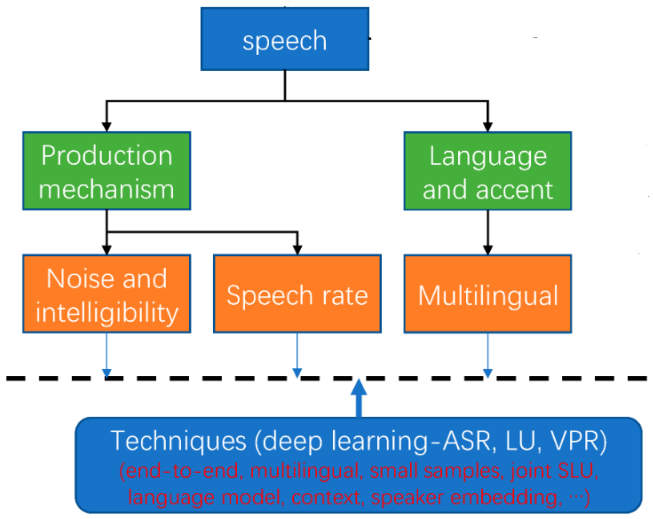

speech
Production mechanism
Language and accent
Noise and intelligibility
Speech rate
Multilingual
Techniques (deep learning-ASR, LU, VPR)
(end-to-end, multilingual, small samples, joint SLU, language model, context, speaker embedding, …)

Figure 8. 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 generalization 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 recognized (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 kind of semi-controlled language)
- General ML/DL issues in evaluation and implementation, including:
    - Misunderstanding and choice of the generalization bounds, the training objectives, and data outliers,
    - Inappropriate regularization 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 in 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 the handling of 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 organization (ICAO), that aims to switch the homophones and near-syllable words. Compared to the pronunciations recommended by the ATC department 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. *Generalization 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, different meanings and uses. Which 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 are ambiguous (e.g. pronounced digits will be tagged as “digit” or something else such as flight number and else?).
9. *Role Recognition.* The role of the speaker is important to better analyse the recognized 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 starts 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 standardized 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 pronunciations of words included in the ATC vocabulary that are different 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 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 a 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.

2.3 Automated Visual Inspection (AVI)

As for the ATC-STT use case, the materials used in the AVI evaluation are provided by an internal Airbus project, and that aims to assist inspectors on the inspection of an airplane 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.

2.3.1 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 21 century, is provided and discussed in (See et al., 2017), showing the main role of AVI and how it enhanced several social, environmental, organizational and industrial tasks. However, when describing AI-based visual inspection, it is frequently about 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 occurring faults make acquisition of appropriate training data difficult, which represents a major challenge for artificial intelligence-based optical inspection. The authors have applied an anomaly detection approach based on a convolutional autoencoder for defect detection during inspection to encounter the challenge of lacking and biased training data. Results indicated the potential of this approach to assist the inspector, but improvements are required for a deployment.

The main goal is to assist inspectors to reduce 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 the 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 the in-service damage detection with two main kinds of damages addressed: lighting strike impacts and dents, as shown in Figure 9:

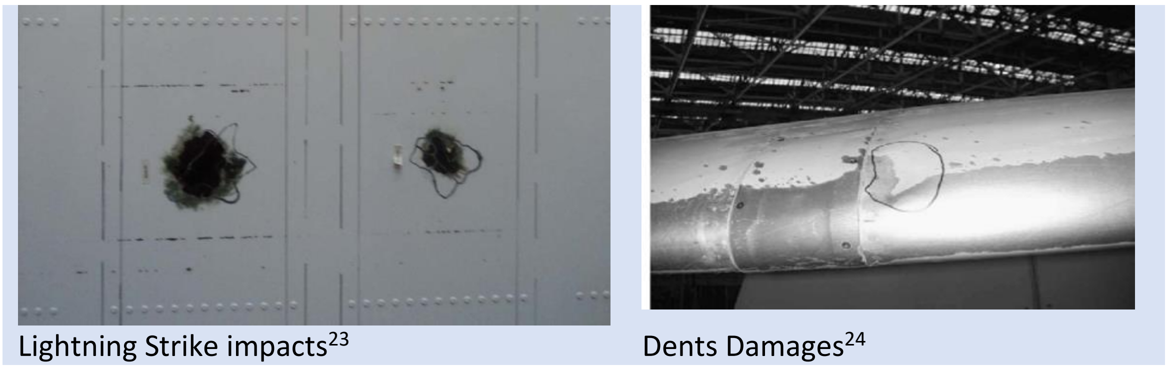


Figure 9. Images showing example damages on an airplane skin.

2.3.2 Provided Data Sets

The Operational Design Domain is defined by the set of all pictures of airframe structures in acceptable lighting and blur conditions. There may be indoors or outdoors pictures. If it is outdoors the weather conditions can influence the lighting & blur state.

Initial input Data details:

- 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[25], 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 weight the images with the reciprocal of their frequency in the training data set to balance the training and avoid overfitting for some classes.

2.3.3 Model and Approach Description

The model selected was built using a transfer learning approach, the architecture is based on VGG16[26] (Simonyan et al., 2014) pre-trained networks on ImageNet[27]. Specifically, a Siamese network is constructed for a multitasking framework, of which the aim is to detect both the damage type (dent

[23] https://www.researchgate.net/figure/Structural-damage-in-the-outer-skin-in-the-Airbus-A400-M-airplane-after-the-lightning_fig8_305817924
[24] https://www.researchgate.net/figure/Wing-skin-metal-dent-examples_fig3_331961295
[25] https://sourceforge.net/projects/vott.mirror/
[26] More technical details can be found in this blog: https://www.mygreatlearning.com/blog/introduction-to-vgg16/
[27] https://www.image-net.org/

impact or lightning strike) and its characterization (corresponding severity level), to further determine its reparability. To do so, the following main steps are implemented:

- Using openCV[28] library for image stain detection. The Siamese model's parts are meant to perform:
  - Stain detection using a binary classification;
  - Detection of the type of damage (multi-class damages classification)
- Completion of the transfer learning network with some layers
- Final layer is a cross-entropy layer
- Adam optimizer is used for the training.
- First targeted performance: >95% accuracy and recall (for the purpose of this study)

2.3.4 Challenges

This use case is considered as high dimension. It will be the opportunity to test different evaluation techniques concerning: the data completeness 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 generalization ability and robustness characteristics. Hence, in the scope of the AVI use case, several issues are addressed and will 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 ;
- No final model has been chosen yet, experiments are still in progress, several models are being compared in order to ;
- Visual inspection related challenges need to be handled too, independently of the object being inspected (e.g. airplanes 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 colors, brightness …)
- A usable visual inspection system needs to be scalable and deployed across different aircraft types, and sizes, which will vary in the engines used, winglets and 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.

[28] https://opencv.org/

- 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, re-classify and archive relevant information. As such, user interfaces should be intuitive to use, while serving a practical purpose.

2.4 ACAS Xu

This use case deals with publicly available data and models developed for flying objects monitoring and directions/decisions definition, 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[29] and activities related to the working group in EUROCAE 2020[30]. 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.

2.4.1 Description

ACAS[31] 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 ACAS Xa for large aircrafts, ACAS Xo for special operations, ACAS Xu for unmanned aircrafts or ACAS sXu for small unmanned aircrafts. The ACAS Xu is an air-to-air collision avoidance system designed for unmanned aircraft (drone), 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 2020, 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 10.

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

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

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

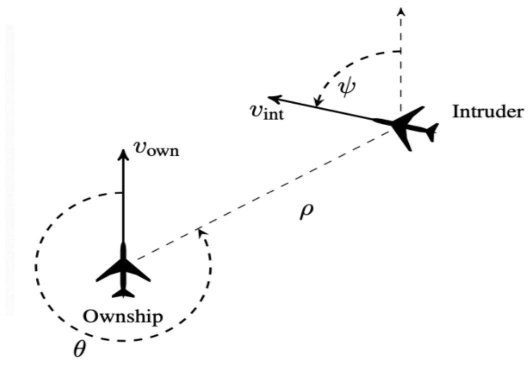


Figure 10. ***Horizontal ACAS Xu geometry***

In

Figure 10, the collaboration between the ownship and intruder is based on a set of parameters that are computed to update the trajectories of the vehicles. The ownship computes six parameters that enable access to the costs tables which give an estimation of the probability to have a collision. The chosen action shall minimize the probability of collision:

- ρ (ft): Distance from ownship to intruder
- θ (rad): Angle to intruder relative to ownship heading
- ψ (rad) : Heading angle of intruder relative to ownship heading direction
- v_{own} (ft/s) : Speed of ownship
- v_{int} (ft/s) : Speed of intruder
- τ (s) : Time until loss of vertical separation

The ACAS Xu system does not need any communication between vehicles. The 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), but also non cooperative traffic, such as vehicles without ACAS Xu (small drones), birds or ground obstacles.

2.4.2 Data

The data consists of different entries of the LUTs from the RTCA SC-147 Minimum Operational Performance Standards (MOPS)[32] 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.

2.4.3 Model

The ML model is broken down into model elements and the architecture is validated (compatibility, consistency, verifiability and conformance 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.

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

Specific ML techniques are defined or selected from the analysis of 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/or the ML requirements process as feedback for clarification or correction. This architecture is identified as an element of 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.

[ACAS-XU instance] The ML model is broken down into XML model elements, as shown in Figure 11. Specifically, it is verified that the union of the ODDs of all ML model elements makes the ML Controller ODD. The ML model description document is created with the NNs architecture and validated. The XML model elements are trained, validated and verified. Each ML model element description is added to the ML model description.

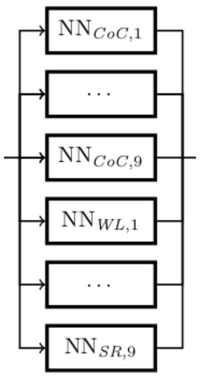


Figure 11. ML model elements of the ACAS Xu system.

2.4.4 Objective and Challenges

In this use-case, the objective is to produce an ML/DL model that is able to completely fit the discrete input LU tables. As opposed to one of the objectives of the experimental part of this project and which is to verify the generalization 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 3). Another issue is that this use case presents a risk of simplification of the generalization 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 generalization 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, generalization and validation on top of the segmentation strategy versus different sub domains of the ODD.

2.5 Conclusion

In this section, we provided a detailed analysis of three different use-cases selected for the evaluation purpose of the different methods and tools that are developed in the scope of the MLEAP project. These use cases address different types of data, which provide an extensive benchmark for the approval and certification of the applicability related to ML/DL models development and evaluation pipelines.
More precisely, 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 models complexity.
Finally, for all use-cases, we provided an overall state-of-the-art review, defining the objectives of the application and the concerned challenges. A high-level ODD definition is also provided, along with the data sets' statistics and models to be evaluate, an overall review of the safety assessment (when applicable), and references to open-source data that could be used for publication purposes.

3. Data: representativeness and completeness

3.1 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 organizing a regulatory framework harmonizing both horizontal (AI-specific) and vertical (Sector-specific) needs. In such a context, the plethora of information and innovations brings as much confusion as progress in the concepts dealt with, and the present document serves the purpose of setting out definitions and identify the topics of interest whose contours are mature enough to design robust requirements.

The chapter is structured as follows. Section 3.2 presents and defines the main concepts to be used, in particular data quality requirements. The aspects 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 3.3).
- Processes of data management, in which activities may affect both attributes (section 3.4).
- Other data requirements that may conflict with requirements on completeness and representativeness (section 3.5).

Each of the three categories listed above include 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 the control of their impact).

Section 3.6 summarizes the information provided in the document, by presenting a first version of the selection grid allowing the identification of methods and tools for the assessment (or control) of the impact of each aspect in the three categories.

Section 3.7 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 usability in a practical manner.

Section 3.8 proposes some conclusions on the topic.

3.2 Concepts

3.2.1 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.

3.2.2 Generalities on data quality requirements

While noting that it is not exhaustive, (EASA, 2023) 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[33]. 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 7 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, offers a list of data quality attributes presented in categories. The series of international standards SQuaRE provides a data quality model in (ISO/IEC 25012, 2008) based 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 standard makes it liable to serve as a reference, since standards rely on international consensus and are mainly designed for

[33] 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, 2023) but does not apply to the present analysis of completeness/representativeness. The term "property" has been selected and remains consistent throughout the chapter.

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 work in progress and not 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.

Table 7. Several examples of data quality attributes.

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

3.2.3 Definitions of completeness and representativeness

In this chapter, in line with the conclusions of Sections 1.2.1.1 and 1.2.1.2, completeness and representativeness (respectively) will be used in the same sense than 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.

3.2.4 Considerations on data dimensionality

At this stage, little to no work could be found that tackles explicitly the notions of completeness and representativeness in their relationship with high-dimensionality data. 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 terms of interesting 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 makes a focus on the challenges relative to dimension reduction, namely the usability and validity of the techniques and the 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 derives mostly from visual inspection of the PCA plots, hence prone to cognitive bias. However, the study does not dwell into a use for representativeness and completeness analysis.

(Liu and Gillies, 2016) present a study on dimensionality reduction, performed to improve the performance of classification while still preserving some properties of the data. The paper argues that inter-class discrimination should minimized rather than maximized, since maximization 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 characterized 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 acceptation of "data efficiency". The second concept, "*well-represented*", is characterized as "*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 concept 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 acceptation 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 high number of dimensions and low number of samples. However, the paper does not provide relevant tools for the assessment of 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 3.5.4.

Literature seems scarce in offering studies that directly deals 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 3.3.4.

3.3 Technical requirements

3.3.1 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 quality and quantity of data required. Most technical requirements discussed hereafter can influence the definition of the ODD, making it a central document in tackling questions relative to completeness and representativeness. In this section, we identify the technical requirements that may impact data completeness or representativeness. Each requirement is presented with its definition and source (if a reference definition could be found), and its relationship with representativeness and completeness. Finally, existing methods and tools for the assessment of 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 at once, 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.

3.3.2 Intended behaviour

3.3.2.1 Definition

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

3.3.2.2 Influence on completeness and representativeness

The intended behaviour of an AI/ML system is first modelled by the operating parameters of the application, 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 that will actually be used by the model. 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. number of features) and their size (i.e. the range of each feature) influence the volume of data required to efficiently train a model, which in turn affects 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 difficult. Finally, even in a given domain, e.g. computer vision, different tasks may have uneven state-of-the-art performance, which may be an indication for the designer to be more or less vigilant about the properties of the data set they use, including completeness and representativeness.

3.3.3 Model architecture

3.3.3.1 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.

3.3.3.2 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[34]. 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.

3.3.4 Data dimensionality

3.3.4.1 Definition

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

3.3.4.2 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 by (Bengio et al., 2013), and 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 preprocessing 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 strategy must also be

[34] Capacity is discussed in more detail in Chapter 4.

used cautiously, as an increase in dimensionality may lead to all samples being at comparable distances in the input space. This phenomenon, called "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 3.5.3, 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 method, as they usually specialize in one particular kind of data (low- or high- dimensional, structured or unstructured).

3.3.5 Intended level of autonomy

3.3.5.1 Definition

In (EASA, 2023), 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*" (highest level of oversight) to "*AI-based system performs non-overridable decisions and actions* " (lowest level of oversight).

3.3.5.2 Influence on completeness and representativeness

As the autonomy of an AI/ML application 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 completeness and representativeness of the data set.

3.3.6 Intended level of performance

3.3.6.1 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 requirement includes function specifics such as: accuracy, fidelity, range, resolution, speed and response time*".

3.3.6.2 Influence on completeness and representativeness

Assuming a model capacity 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.

3.3.7 Intended levels of robustness and resilience

3.3.7.1 Definition

The definitions of robustness and resilience from (EASA, 2023) 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 pertaining to '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*".

3.3.7.2 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 by 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.

3.3.8 Intended levels of stability

3.3.8.1 Definition

EUROCAE WG 114 and SAE G-34 are working on a definition of stability as the ability "*to provide equivalent response within the neighbourhood of an input*". This definition is in line with the definition of "model stability" in (EASA, 2023), which distinguishes "*model stability*" from "*algorithm stability*". In this work, stability will be considered exclusively from the "*model stability*" perspective.

3.3.8.2 Influence on completeness and representativeness

In this chapter, stability is distinguished 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 to 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 with the collection of a larger volume of quality samples[35].

3.3.9 Methods and tools for assessment

3.3.9.1 Methods related to technical requirements

A number of 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 completeness. Despite their subsequent limitations, these approaches enable a dynamic, model-driven process of data set improvement with a minimal number of additional samples[36] (those needed to improve model coverage).

An added advantage of using the model for the characterization of the training data set, 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), 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 characterization 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 on 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.

3.3.9.1.1 Feature set characterization

A more favored angle is to use the model as a tool to characterize 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 set for each class to predict. The limit of a class is defined by the extrema of the features ranges. Beyond these boundaries, the model either changes class or is incapable of taking a decision. The authors hypothesize that it is possible to use the trained model as a generator to explore the feature set and use it to characterize 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.

[35] Chapter 4 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.
[36] 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.

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 on rare cases, but it must be so while respecting the larger distribution of phenomena of interest in operating condition.

In a supervised setting, this method is more of a fine-grained model performance diagnostic tool. To be relevant as a data set characterization tool, it is preferable to apply it on unsupervised models. Nonetheless, the chosen model remains a proxy with lots of potential caveats that must be anticipated and kept in mind when performing analysis.

3.3.9.1.2 Performance monitoring

Another approach, by (Tae and Whang, 2021) is called Slice Tuner, a tool for maximizing 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 Equalized Error Rate over all slices, which can be extended to other forms on constraints on the learning process, making Slice Tuner a seemingly flexible tool. Beside 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 the characterization of the completeness and representativeness of the data set.

3.3.9.1.3 Neuron activity monitoring

The approach described by (Pei et al., 2017), called DeepXplore, is tailored for neural networks and aims at monitoring its 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 an input activating all neurons (i.e. full neuron coverage) has been fully exploited by the model. Conversely, low input coverage shows a weakness of the model and might be used to identify low quality samples, i.e. samples containing either little to no information (which could indicate a sample too hard, or 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, characterizes a data set through the behaviour of a model. This is done via two observations:

- *Major function regions* are characterized by ensuring that the activation values of the neurons 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 value is binned and Deepgauge monitors that the entire spectrum of bins is used.

- *Corner case regions* identification consists in checking 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.

3.3.9.1.4 Human-in-the-loop approaches

(Kiela et al., 2021) describe a tool specifically aimed at addressing the problematic 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 in order 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 very precise 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. This functionality was introduced by Dynatask (Thrush et al., 2022), allowing any user to incorporate new tasks in the framework.

3.3.9.2 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 3.3.3, 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[37], 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 a neuron switch 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).
- Activation patterns extend transitions of a single neuron to all neuron in the network. Thus, a specific activation pattern can be mapped for each input going through the network.

[37] 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.

- Dichotomies is a dual formulation of expressiveness. Transitions and Activation patterns focus on expressiveness relative to an input, while dichotomies focus 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 3.4.8.8 for more.

Other strategies such as ensemble learning, fine-tuning of pre-trained models or regularization also affect the capacity of a model and the volume of data required to reach a given level of performance, which affects the conditions to be met to ensure the completeness and representativeness of the data set[38]. 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 be able to quantify the difficulty of a task, which in turn would enable the sizing of the model's capacity, but such 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[39], cleanlab[40] (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 performance. Further testing of whylogs and cleanlab is required to get more insight on 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).

---

[38] 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.

[39] https://whylabs.ai/

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

3.3.9.3 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 set rather than devising good data curation solutions[41] (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 alters 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.

[41] 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.

3.4 Processes

3.4.1 Introduction

Most learning systems[42] require data in order to identify patterns used to infer behaviours, which means data need to be collected beforehand to enable the learning and evaluation of the system.
Therefore, several processes must or may be performed before the data are usable, each of which constituting 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, 2023). 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.

3.4.2 Data management requirements

3.4.2.1 Definition

Data management (as defined in (EASA, 2023)) is the process of designing the specification of the required data set.

3.4.2.2 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.

3.4.3 Data quality improvement (in Data preparation)

3.4.3.1 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.

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

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.

3.4.3.2 Influence on completeness and representativeness

Data augmentation is at its core a strategy to improve the completeness and representativeness on 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 3.4.2, 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.

3.4.4 Data synthesis (in Data preparation)

3.4.4.1 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.

3.4.4.2 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 3.4.2 and 3.4.3, i.e. to ensure these new samples preserve the completeness and representativeness of the data set.

3.4.5 Data sampling (in Data preparation)

3.4.5.1 Definition

Data sampling covers the methods and tools used to select a subset of datapoints from a larger set of datapoints 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.

3.4.5.2 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 3.5.2 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 set, but this should be done in accordance with the capacity of the chosen model architecture, to ensure sufficient overall volumetry.

3.4.6 Labelling (in Data preparation)

3.4.6.1 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.

3.4.6.2 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, loose annotation consistency across annotators (due to fuzzy guidelines, 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.

3.4.7 Pre-processing

3.4.7.1 Definition

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

3.4.7.2 Influence on completeness and representativeness

Certain preprocessing operations may reduce the amount of information available for learning. While it is acceptable and sometimes necessary to encourage generalization 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.

3.4.8 Methods and tools for assessment

3.4.8.1 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 to 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 generalization: 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 on possible weaknesses in the data set's properties and be exploited as a more precise feedback for the development of the model.

3.4.8.2 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 to be met will require ensuring 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 important 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 3.3.9.3) can provide a coarse-grain monitoring of the data set after collection, possibly motivating additional efforts.

More in-depth methods for characterizing data sets exist, with their limitations. (Asudeh et al., 2019) describe an approach to exhaustively identify the combinations of features patterns represented in a data set. The method works best with low-dimensional, categorical feature sets, though the authors indicate that it can scale to higher-dimensional, continuous inputs by preprocessing 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 datapoints (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 their age.
- Hence, the root is a pattern of only variable features (here, 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 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. It therefore exhibits the opposite weakness of Pattern breaker, being 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 maximizing 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 standardized 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 on the individual impact of each sample on the utility function.

3.4.8.3 Data quality improvement

As discussed in 3.4.3, data quality improvement encompasses data augmentation and data imputation. Data augmentation may rely either on "non-learnable methods"[43], 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 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 more 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 quantify 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 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, along with visual assessment on 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:

[43] « non-learnable » and « learnable » methods are concepts developed in Sections 4.5.4.3 et 4.5.4.4.

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

Imbalance is characterized at 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 characterize sub-types, as it requires additional annotation effort 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.

3.4.8.4 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 undersampling (i.e. deleting samples in overrepresented classes). However, undersampling 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 undersampling are especially visible if the number of samples in the majority class is modest in absolute value. From this observation, the authors favor oversampling using data synthesis, of which they identify 2 types:

- structural methods favor the synthesis of example 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 neighbors 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.

3.4.8.5 Data sampling

While the limitations of undersampling (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 in computing Shannon entropy for every sample of Female vs. Male (and can be applied on 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. The idea is, for a set of k-dimensional samples, 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 it the case for most method 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 cost and delay. A similarly limited but complementary approach is proposed by (A. Wang et al., 2020), that specializes 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 in 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 neighbors 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 in 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 on pre-COVID-19 surveys to weight 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), who exploits reference demographic statistics to weight households (by size), response method (cellphone 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 realized 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.

3.4.8.6 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. 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[44] (Northcutt et al., 2021). Similarly (Sánchez et al., 2019) present a framework to explore the missing values in a data set, according to defined axis such as labels, features or time. In their case study, they highlight the importance to distinguish the sample-level analysis from the class-level analysis.

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

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 Kappas (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 problematic 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.

(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 conceptualized 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 specialized to be extended outside the domain of linked databases.

3.4.8.7 Preprocessing

As preprocessing transforms data and assuming the original data set has been curated for completeness and representativeness, a first consideration is to apply the same methods for curation after each preprocessing step. The remainder of this section describes preprocessing 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 Open Street map) to better match a reference data (from a national cartographic institute). In a first phase, preprocessing steps are applied to align both data sets: the formats and coordinates systems are harmonized, 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 optimized, so the matching operation is repeated for several combinations of values of both parameters. 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 preprocessing 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 been already 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 for the improvement or assessment of 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.

3.4.8.8 Corner case and edge case detection for ML models

3.4.8.8.1 Prediction based methods for the vision domain

Following the definition introduced in 1.2.3.2.2 prediction-based approaches can be found mainly at 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 is to describe 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).

Hereafter, this section introduces first a corner case detection method based on unpredictable situations detection and then summarizes methods based on surprise adequacy.

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

In (Fingscheidt et al., 2019), the authors identify corner cases as technically unpredictable situations. However, it is important to note that not each 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) relevant object or class in 3) a relevant location.

More precisely:

1. The method describe 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 predict 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 12).
2. 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 semantics segmentation, the training protocol from (Chen et al., n.d.) can be used.
3. Finaly 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).

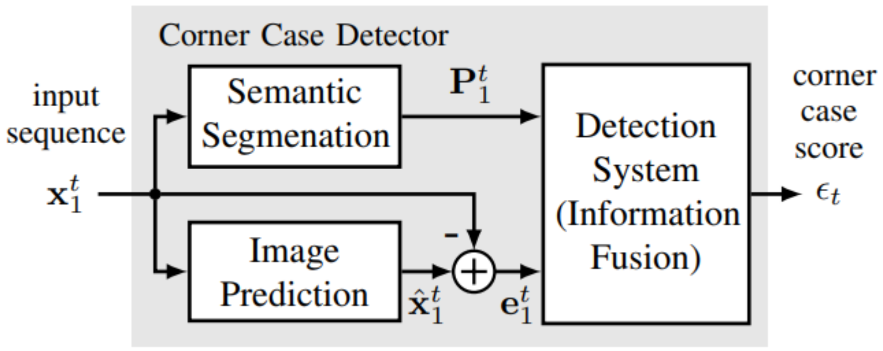


Figure 12. High-level block diagram of the corner case detector (borrowed from (Fingscheidt et al., 2019)).

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

Following (Ouyang et al., 2021), a good way to evaluate surprise adequacy is to study neurons behaviour in terms of a given deep learning (DL) model. The authors emphasize 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 3.3.9.2 for more. While, compared with those metrics reflecting independent behaviours of testing and training sets, in (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 $A_N(Te)$:

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

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 of corner case study, the proposed method can be utilized as a tool in recognition of corner case data.

The tool related above is:

- In python: dnn-tip 0.1[45].

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

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

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 $x_a = \underset{D(x_i)=C_x}{argmax}\|\alpha_N(x) - \alpha_N(x_i)\|$
- $dist_a = \|\alpha_N(x_a) - \alpha_N(x_b)\|$ 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[46].

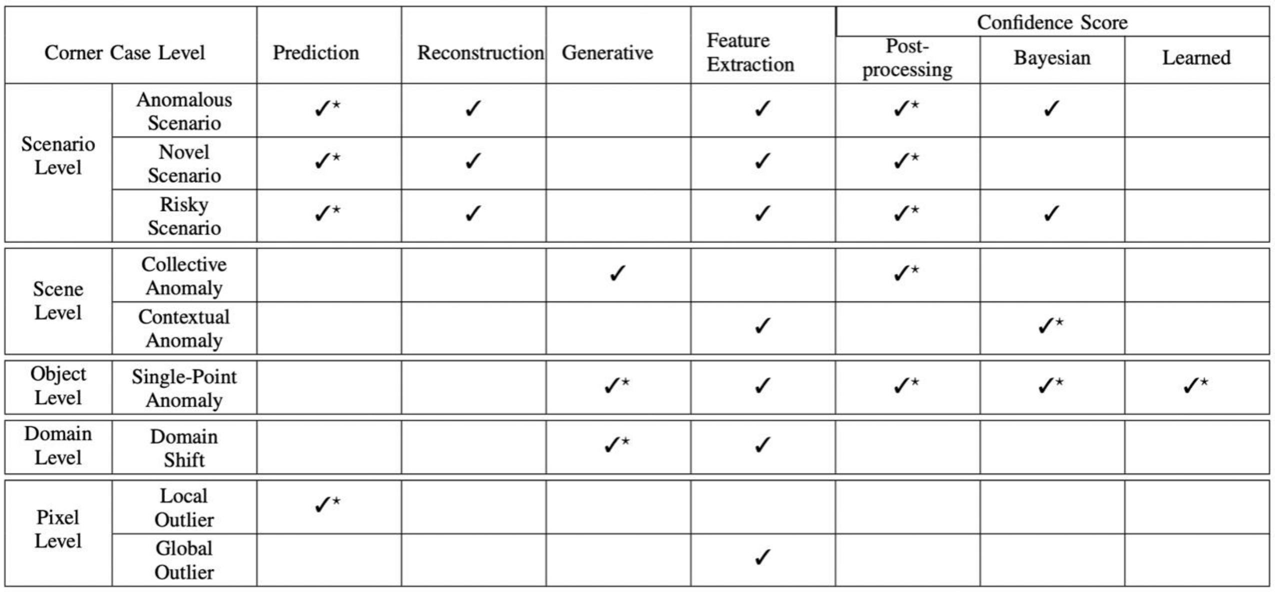
Summary of Methods Available

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 maneuver 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)

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

Corner case level	Methods available
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) • 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 13 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 13 – The summary of approaches to detect corner case (Breitenstein et al., 2021).

3.4.8.8.2 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 transcribes, “*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, signaling 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-centered 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 defense 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[47], 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.

3.4.8.8.3 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:

- **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

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

- **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).

3.4.8.8.4 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 problem, which is not always the case), but would have to justify its choice under the subjective opinion they used.

In all the techniques available it can be noted that test-based methods have (at least until now) the highest degree of generalization to other use case 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, 2023) 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 characterization of the attributes of the OD and ODD, as well as a description of their distribution.

3.4.8.9 Conclusions

In general, and as is the case in Section 3.3, 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 work 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 variation (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) highlights 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.

3.5 Relation to other data quality requirements

3.5.1 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 to 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.

3.5.2 Balance

3.5.2.1 Definition and assessment of the value

The standard (ISO/IEC CD 5259-2, 202X) defines that balance refers to 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), through 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:

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

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 logics 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[48] 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.

3.5.2.2 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 of 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 representativeness of the data set,

[48] 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.).

since representativeness implies that the data set matches the statistical distribution of the phenomena in operational conditions.

3.5.2.3 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 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 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.

3.5.2.4 Impact and observability

Representativeness may be strongly impacted by the decisions made to enhance balance, 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.

3.5.3 Relevance

3.5.3.1 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 in 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[49] (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) presents 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 for a use of the output labelled data in ML, since it spots both 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, a 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 generalized to all configurations of AI use, since it requires a vast quantity of different users to generate and combine data.

3.5.3.2 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, 2023) also highlights indirectly 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,

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

the document does not warn about some pitfalls linked to the subtle balance between the three attributes, and that trade-offs must be found.

The curation of a data set in view of increasing its relevance has an impact on data dimensionality, which may in turn affect completeness and representativeness. Indeed, since relevance enhancing 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, datapoints tend to become equidistant in the resulting space as there is always neighbor 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 fulfill completeness and representativeness requirements, but not too many in order to both match relevance requirements and avoid high-dimensional data "curse".

3.5.3.3 Methods and tools for assessment

Literature is scarce about methods for the assessment of 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 no 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) summarize theorems highlighting why compositional functions may prevent the 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).

3.5.3.4 Impact and observability

Enhancing relevance of the data may impact both representativeness and completeness, and a trade-off should be found by the data scientist. 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.

3.5.4 Diversity

3.5.4.1 Definition and assessment of the value

3.5.4.1.1 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 following 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.

3.5.4.1.2 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 Trusworthy 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 system 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 of 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 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 difference. 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).

3.5.4.1.3 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, use 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 following of the chapter.

3.5.4.2 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 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.

3.5.4.3 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 any mention of 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 3.5.4.1 may be applied.

3.5.4.4 Impact and observability

Data diversity is defined in terms of discriminative power of the data, and the quest for diversity may alter completeness and representativeness. However, in this acceptation, 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.

3.5.5 Currentness

3.5.5.1 Definition and assessment of the value

From 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 done 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; 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 $X = A/B$, where 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 $X = A/B$, where A is the number of data records[50] that fall within the required age range, and B the total number of data records ina 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, summarizes 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) defines 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 consider 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 of the appropriateness of the concept under exploration.

3.5.5.2 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 intended conditions of operation of the system, 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.

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

3.5.5.3 Methods and tools for assessment

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

3.5.5.4 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.

3.5.6 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 standardization 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 to 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 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.

3.5.7 Conclusion

Ensuring all data quality attributes of a data set may strongly impact completeness and representativeness. Figure 14 summarizes the interdependences highlighted by literature, and the way it may affect the two attributes.

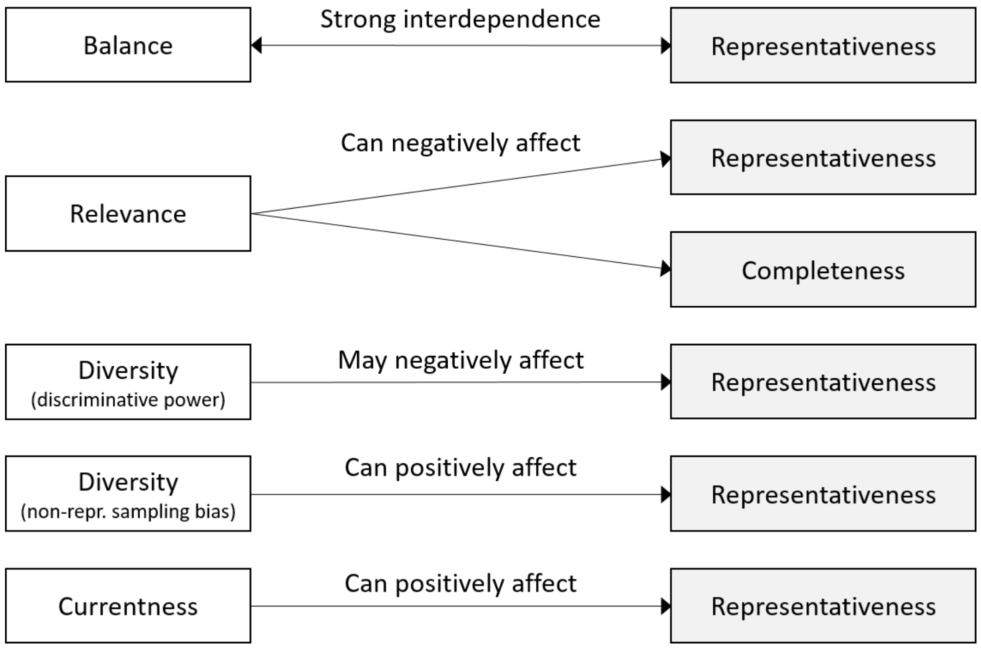


Figure 14. Interdependence between other DQRs and data completeness/representativeness.

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 than 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 literature are that there is no standard method to reach a perfect state, where all attributes may be respected. Overall, literature seems to 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 logics, 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 a best strategy for the implementation of a trade-off.

3.6 Selection grid

3.6.1 Introduction

This section offers an operational synthesis of the methods discussed in the previous sections. A first table recapitulates all the methods presented, along with a recommendation on whether they should be integrated to 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 discussions of the document: 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[51] 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 DQR. It seems reasonable to envision that, in context where no tools or methods seem to exist, DQRs may be presented as attention point, 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: 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, cited papers may not correspond to the selected works, and work that require exploratory testing were not considered substantial enough to change a conclusion stating that no method stood out during the analysis.

[51] 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.).

3.6.2 Methods summary

Bibliographical references	Document references	Decision
(Mani et al., 2019)	3.3.9.1.1	Selected. A ponderation of the four metrics will probably be explored, as strict Equivalence partitioning may be too strong a condition.
(Tae and Whang, 2021)	3.3.9.1.2	Selected. Extension of Slice Tuner to unstructured data is unclear and could be explored.
(Pei et al., 2017)	3.3.9.1.3	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)	3.3.9.1.4	Discarded. Adversarial examples are too narrow a problematic, and manual addition of samples is too tedious for the solution to be explored.
(Raghu et al., 2017)	3.3.9.2	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, with is not directly link to the objectives of the project, similar to Dynabench.
(Schouten et al., 2009)	3.4.8.1	Discarded. R-Indicator would require adaptations to be usable in the context of assessing the completeness and representativeness of a data set, which is out of the scope and time frame of the project.
(Cabitza et al., 2021)		Selected.
(Catania et al., 2022)		Selected. The approach described mixes ML good practice 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)	3.4.8.2	Selected. Despite its limitations in terms of data dimensionality, it is the only exhaustive method found.
(Paganelli et al., 2022)		Discarded. The MLEAP project has no 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 prioritized 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)	3.4.8.3	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 is too complex and slightly off-topic for the later stages of the MLEAP project. The testing of other methods should be prioritized.
(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)	3.4.8.4	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)	3.4.8.5	Discarded. The method is complex and its results perspective are unclear. Other methods should be prioritized.
(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. Testing may be limited to the comparison of train vs validation vs test sets, due to the contrastive nature of the method.
(Brubaker et al., 2021)		Discarded. The method may require too much adjustment w.r.t the constraint of the project. Other methods should be prioritized.
(Kohut et al., 2012)		Discarded. The method may require too much adjustment w.r.t the constraint of the project. Other methods should be prioritized.
(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.

Bibliographical references	Document references	Decision
(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.
(Sánchez et al., 2019)	**Error! Reference source not found.**	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)	3.4.8.7	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)	3.5.2.3	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, and is rather a position paper on balance.
(Dickinson et al., 2012)		Selected, although a part of the method consists in documenting data without being explicit on the criteria for selection of the elements to document.
(Van Vleck et al., 2007)	3.5.3.1, 3.5.3.3	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 do not seem realistic in real settings of ML design.
(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 CoD limitation of representativeness is still unclear.
(Surace et al., 2019)		Discarded. The positive impact of 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)	3.5.4.1, 3.5.4.3	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

Bibliographical references	Document references	Decision
(Leavy, 2018)		Discarded. The work does not focus on the link with representativeness, and do 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.
(Iphar et al., 2015)	3.5.5.3	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.

3.6.3 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 (3.3.9.1.1)
(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 (3.3.9.1.2)
(Pei et al., 2017)	Any. The method rests on observation of neuron activation at learning time. It is however restricted to neural networks, and it is not clear whether complex architecture such as attention networks would respond well, which may indirectly limit the spectrum of use cases covered.	Not related to a particular factor (3.3.9.1.3)
(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 (3.3.9.2)
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 state 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 the comparison of 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 testing of the method.	Not related to a particular factor (3.4.8.1)
(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 categorize quantitative data.	Data Management requirements (3.4.8.2)
(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 (3.4.8.3)
(Catania et al., 2022)	The method uses PCA and other variations, like (Caiafa et al., 2020), but for other purposes.	
(Dourado Filho and Calumby, 2022)	Images. Exploratory work may investigate 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 (3.4.8.4)
(Mountrakis and Xi, 2013)	Images. Exploratory work may investigate extension to other types of data, especially unstructured ones.	Data sampling (3.4.8.5)
(Sánchez et al., 2019)	Most data types seem compatible with the method (axis are explicitly flexible and include labels, features and time). Exploratory work may be performed to confirm this.	Labelling (**Error! Reference source not found.**)
(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-dimensionality data.	Balance (3.5.2.3)
(BSA, 2021)	Virtually any; the document does not present a limitation depending on the type of data. An exploration of the adaptability of the framework to high-dimensionality may be relevant.	Diversity (3.5.4.3)
(Iphar et al., 2015)	The method seems to be applicable to any type of data. The paper seems however to consider currentness of samples, but not currentness of the values of specific dimensions. An exploration would be required on the feasibility of assessing currentness of high-dimensionality data.	Currentness (3.5.5.3)

3.6.4 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 (3.3.2)	C / R	The more complex the task, the more data points will be required to guarantee C/R.	Yes	Volume of data	No methods or tools. Some methods may be tailored for certain AI tasks through the type of data available.
Model architecture (3.3.3, 3.3.9.1.1, 3.3.9.1.3)	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 (3.3.4, 3.4.8.2, 3.5.3.3)	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 (3.3.5)	C / R	The level of autonomy can impact the required amounts of data needed to ensure robustness, resilience and adaptability of the system.	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 (3.3.6, 3.3.9.1.2, 3.4.8.1)	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 (3.3.7)	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 (3.3.8)	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.

3.6.5 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 (3.4.2, 3.4.8.2)	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 get general insight on the data set. Whylogs and Cleanlab are two off-the-shelf tools for the observation of data sets.
Data quality improvement (3.4.3, 3.4.8.3)	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 (3.4.4)	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 (3.4.5, 3.4.8.5)	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. Without external information on the real-life distributions of the phenomena to capture, no method enables the absolute assessment of a data set.
Labelling (3.4.6, **Error! Reference source not found.**)	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 (3.4.7)	C / R	Activities reducing the amount of information may affect R and C.	No	Correspondence between the function and the task	No preprocessing method enabling the assessment of C&R stood out upon analysis.

3.6.6 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 (3.5.2)	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 (3.5.3)	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) (3.5.4)	R	If diversity enhancement is performed in view of maximizing 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) (3.5.4)	R	If diversity enhancement is performed in view of controlling 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 (3.5.5)	R	Currentness is one of the prerequisites for data representativeness.	No	/	Ensure currentess, e.g. by computing a probability that the data remains up-to-date in a certain time range (Iphar et al., 2015).

3.7 Preliminary experimentations

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

3.7.1 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 much 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 on 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[52] organized 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, it was replaced by the STT task of the REPERE campaign, organized by LNE between 2012 and 2014. This data set contains 60h of recordings of French TV channels BFMTV and LCP. The sample durations range from 1min to 1h but mostly hover around 15mn. It is mostly news and debates with clearer sound than what should be expected in the ATC-STT use case, but this should not influence the experiments, especially at this early stage.

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

3.7.2 Selected methods: motivations and expectations

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

3.7.2.1 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 on the completeness of a

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

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 homogeneously occupy the entire plot. 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. Getting more insight on this type of round-trip engineering step was the main objective of the experiments.

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 to learn 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 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.

3.7.2.2 Graph-based analysis

The method proposed by (Asudeh et al., 2019) relies on traversing the tree-like graph of 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, it seemed easy to implement comparatively to other methods 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.

3.7.2.3 Entropy-based analysis

The characterization 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.

3.7.2.4 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 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).

At this stage of the MLEAP project, only the approach by (Cabitza et al., 2021) could be tested. (Mountrakis and Xi, 2013) rely on another similarity metric and will be tested in later phases.

This method was 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 speech is difficult to process on its own. Most assessment methods identified would work best on vectorized information. Building vectorized representation of speech is a non-trivial task. As the overall method was already implemented and yields a single similarity value, it was seen as a reasonable testbed for the assessment of pre-trained speech-embedding, i.e. learned vectorized representation of speech data.

A possible difficulty would be the scalability of the approach to large data sets. Indeed, the time needed to encode embeddings for long samples such as those in the REPERE data set remained a potential shortcoming of the method, along with the ability to compute the metric at scale.

3.7.3 Selected methods: Experimental protocols and results

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

3.7.3.1 PCA-based analysis

3.7.3.1.1 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 ran directly on the data set. The results are presented in Figure 15.

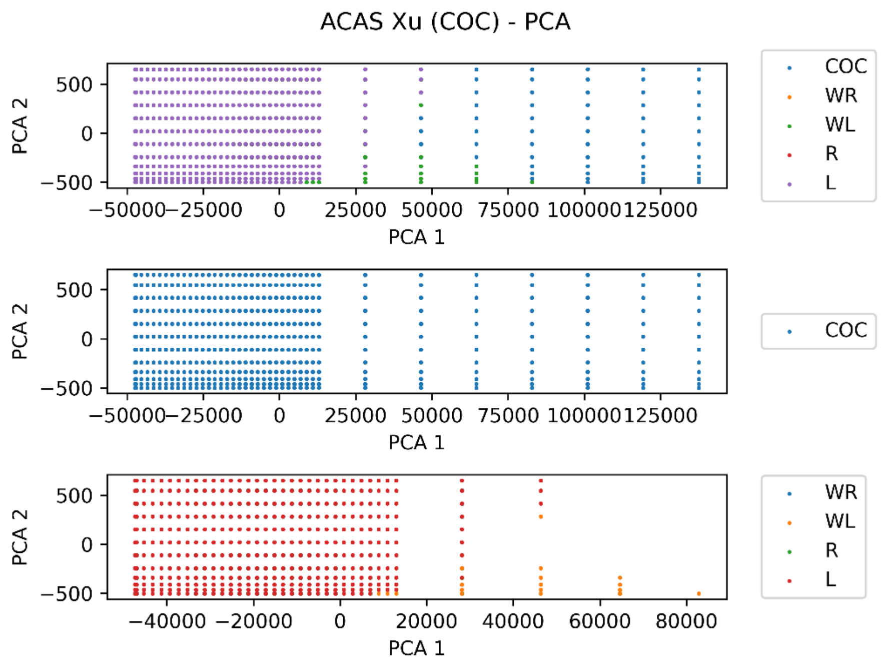


Figure 15. Results of a 2-components PCA applied on a subset of the ACAS-Xu data set. Only samples with input state COC were used.

As a reminder, the ACAS-Xu data set is comprised of eight features, including the current state of the aircraft, expressed with the same class than those to predict, hence the plot representing the *Clear Of Conflict (COC)* class (i.e. it is the input state and not the expected label)[53].

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) and 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.

Combining with Figure 16, 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.

[53] 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.

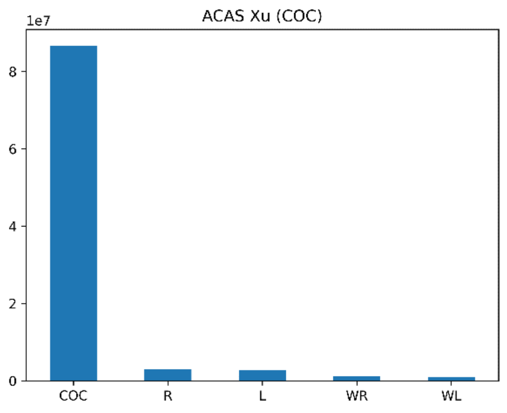


Figure 16. Distribution of the output label on the ACAS-Xu samples with input state COC.

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 intuitive explanation for it called for complementary work, to assess whether they were artifacts 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 which the PCA will be computed. Figure 17 shows the histogram of explained variance for the ACAS-Xu data set (still on the *COC* label).

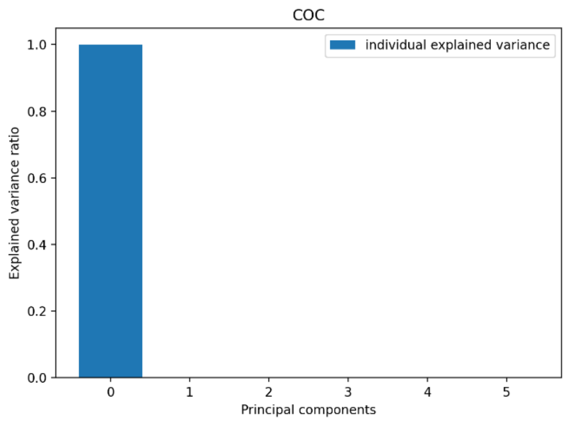


Figure 17. Explained variance ratio of each component of the PCA. The PCA was performed on the subset the ACAS-Xu samples with input state COC

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 18, 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*.

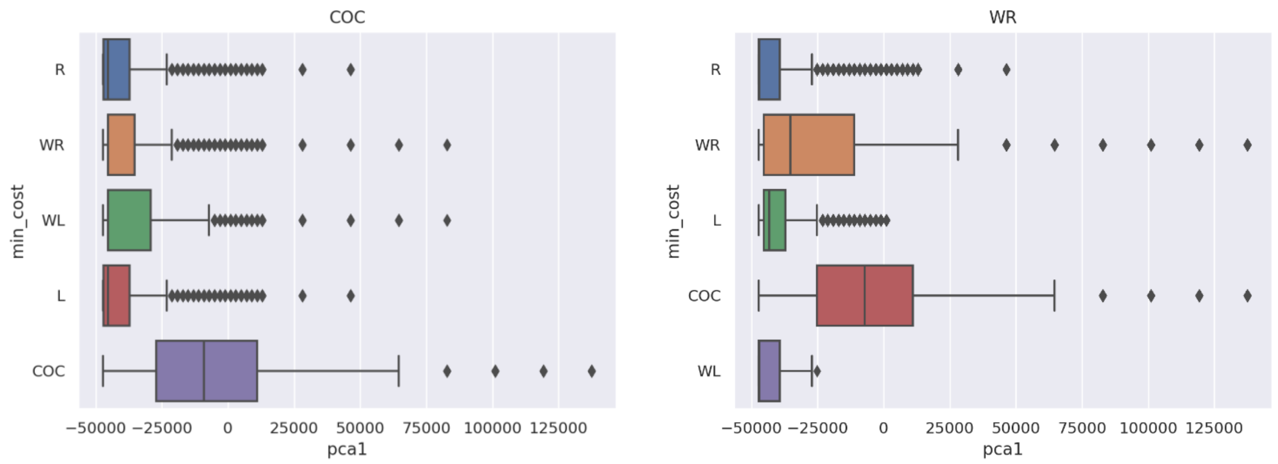


Figure 18. On the left, boxplot of the distributions of values taken in the first component of the PCA of the ACAS-Xu subset with input state COC, in function of the output states. On the right, the same plot for the subset of input state WR.

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 are 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 value below the first quartile and 25% 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, 2023). The boxplots on Figure 18 confirm the imbalance between classes, 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 amount 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 on 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[54] was chosen. This data set is comprised of 13 910 measurements from 16 chemical sensors utilized 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 on the data set characteristics.

First, the distribution of the different gases and the histogram of explained variance were plotted and are shown in Figure 19 and Figure 20, respectively.

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

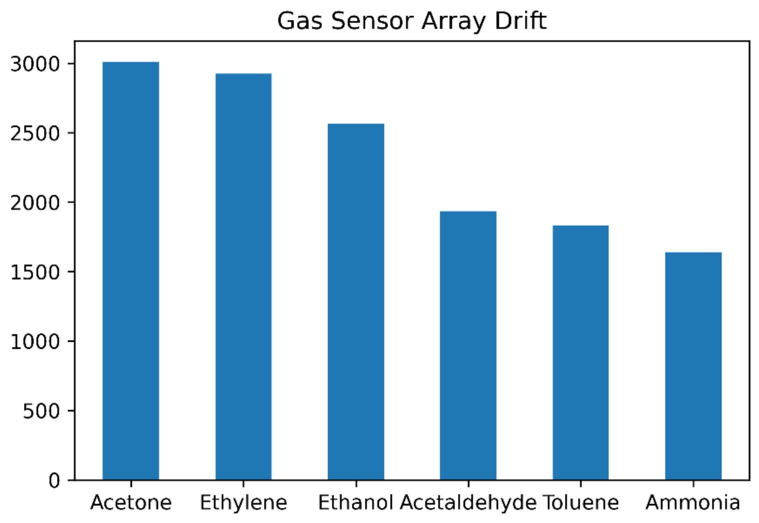


Figure 19. Distribution of the labels in the Gas Sensor Array Dataset.

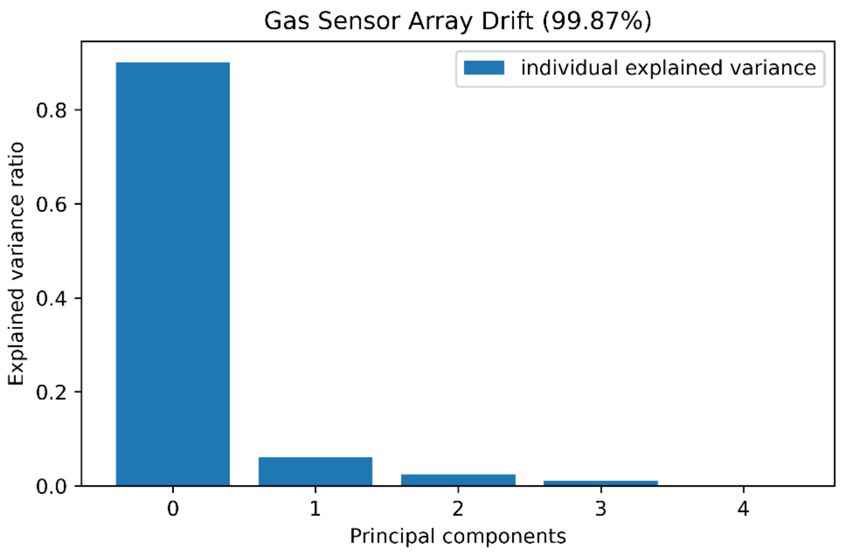


Figure 20. Explained variance ratio of each component of 5-component PCA performed on the Gas Sensor Array Dataset.

Though not completely balanced, the data set has a globally homogeneous distribution of classes, with a 30% difference between the most represented and less represented class. 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 on Figure 21.

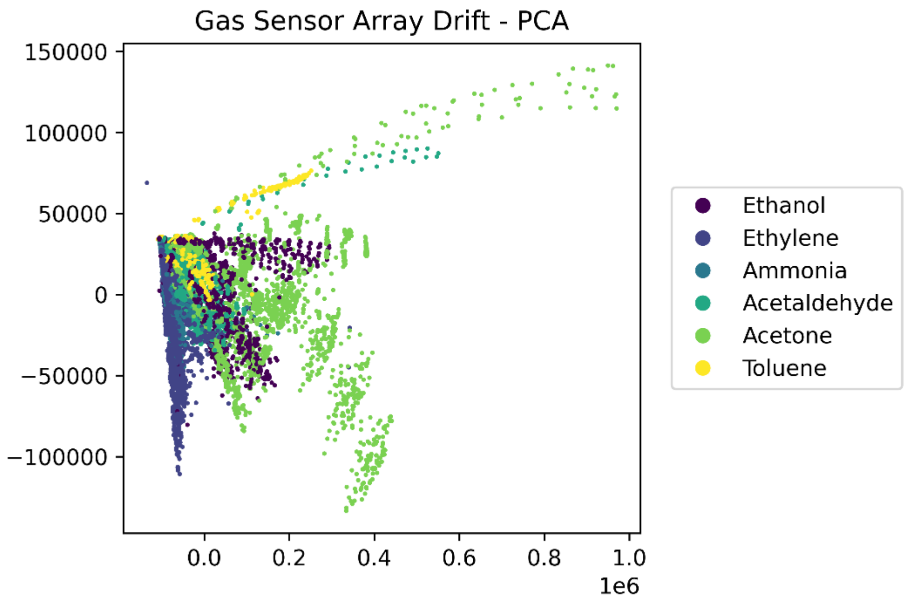


Figure 21. PCA score plot of the Gas Sensor Array Dataset.

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, the 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 8.

Table 8. 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.

3.7.3.1.2 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 situation would be a typical illustration of the balance/coverage dilemma discussed earlier in the chapter.

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

More information on the constitution of the data set were obtained, and complementary analysis (such as feature-component correlation or normalization 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 component), and did reveal unexpected trends in the data set related to its completeness.

3.7.3.2 Graph-based analysis

3.7.3.2.1 Experimental protocol

As no reference implementation was available, the development of the necessary software had to be performed in-house. Several iterations were necessary, to validate the correct operation of the tools. The general idea of the method is to explore the 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 as the same root pattern with only undetermined feature, 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 to this example is presented in Figure 22. 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 male 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.

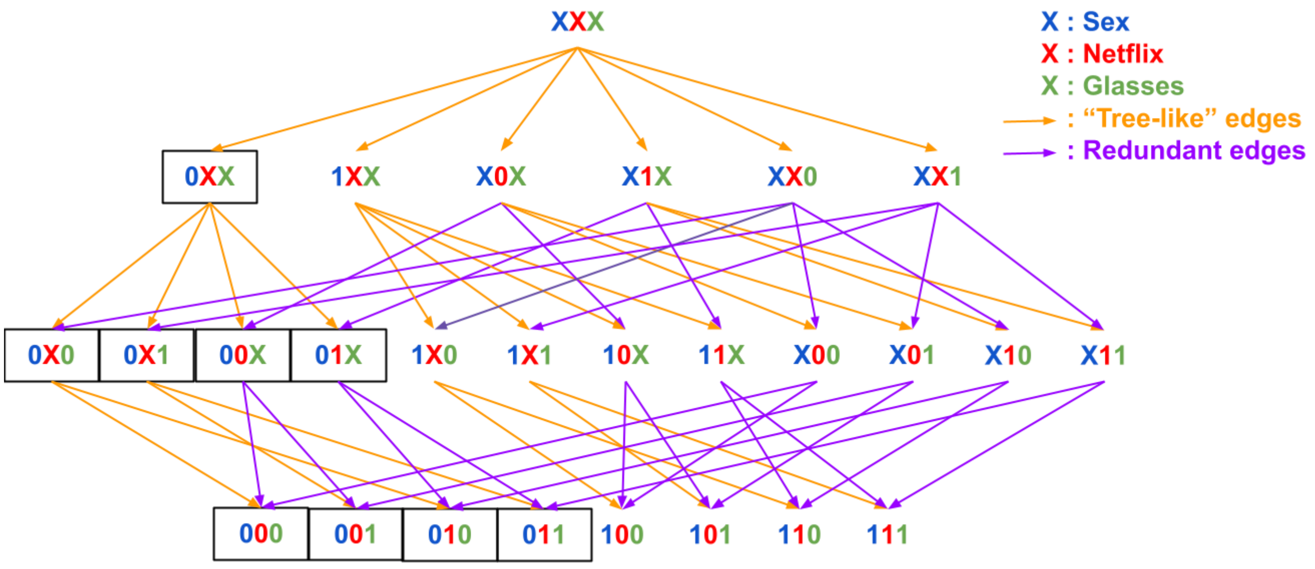


Figure 22. Graph representation of the example

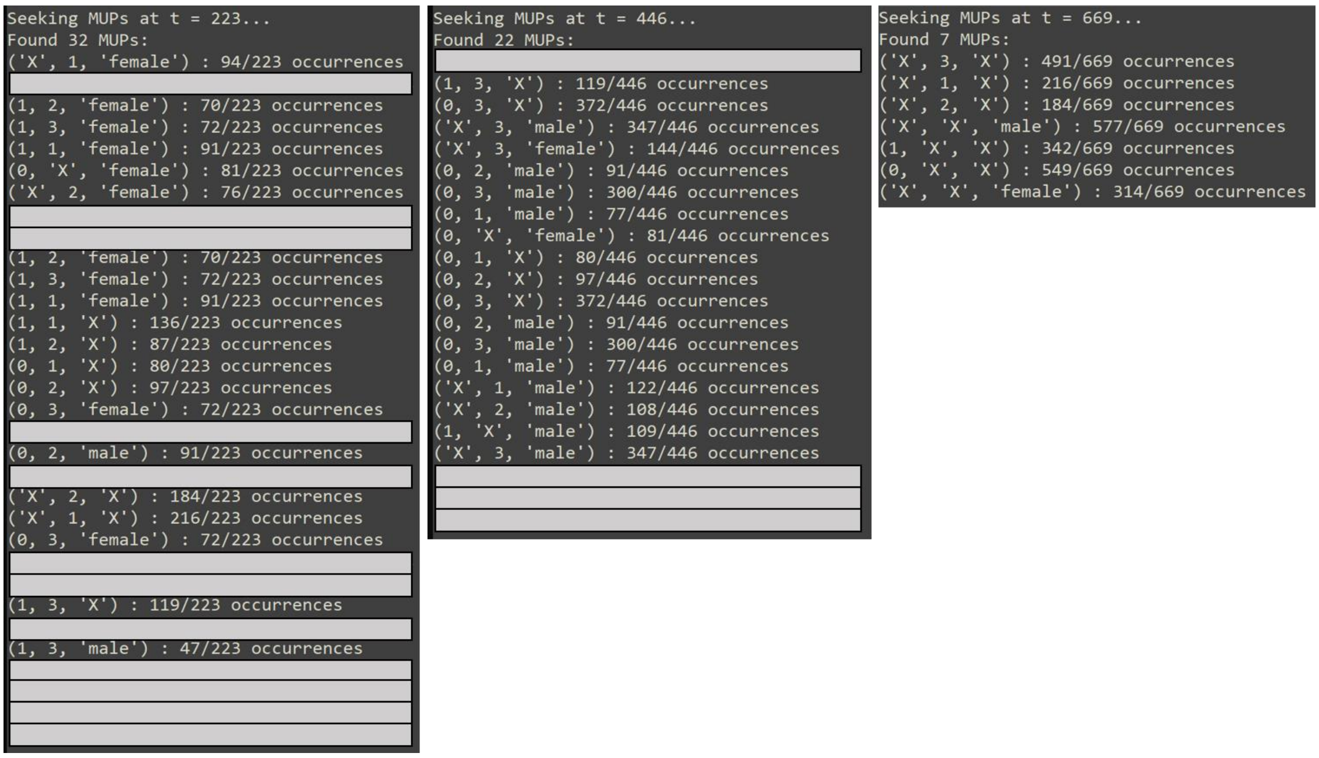
The first version of the implementation included a complete building of the connection graph with separate traversal algorithms, to allow the checking of the behaviour of both the population and traversal algorithms separately and by hand if needed. This is in opposition with the original approach that 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[55].

The Titanic data set is a public data set comprised of various information about the 891 passenger of the RMS Titanic that sunk on April 15, 1912. It is a simple and popular classification data set for small

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

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 23, 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 23. 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 $t = 669$, 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 survivor 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 clear 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 of a second version of the implementation that would include pruning. Some preprocessing had to be performed. Indeed, the method is designed to work only with qualitative data, quantitative data had to be discretized. 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. At 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.

3.7.3.2.2 Results from the MLEAP perspective

Despite seemingly light experimentations, the interest of the method was confirmed. It has the potential to be a useful tool for a 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).

3.7.3.3 Entropy-based analysis

3.7.3.3.1 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[56]. This particular data set was selected because as any such data set, it is easy to handle and thus to allow 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 annotation 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 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).

The results of computing Shannon entropy on each super-class (label-wise entropy) are shown in Figure 24. The image-wise entropy of each 20 super-classes of the data set is visible in Figure 25.

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

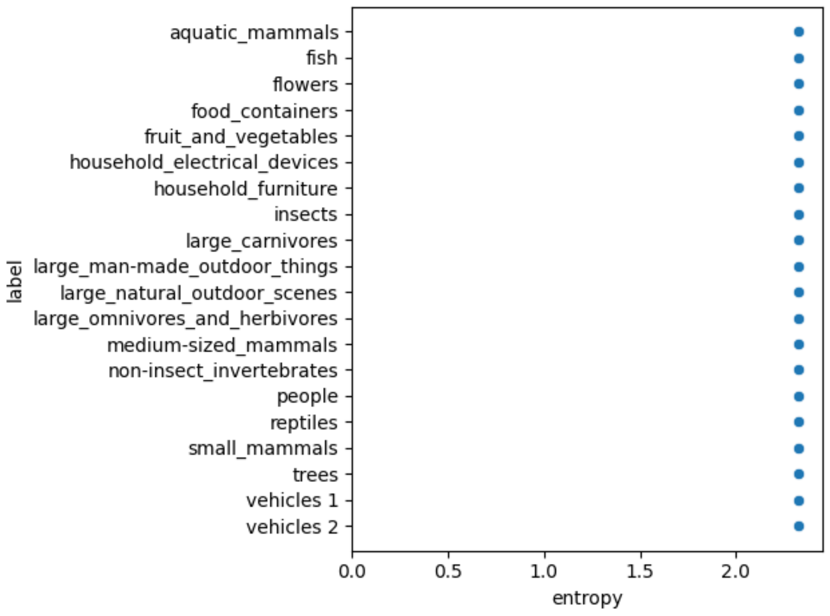


Figure 24. Entropy value of each super-class of the CIFAR-100 data set.

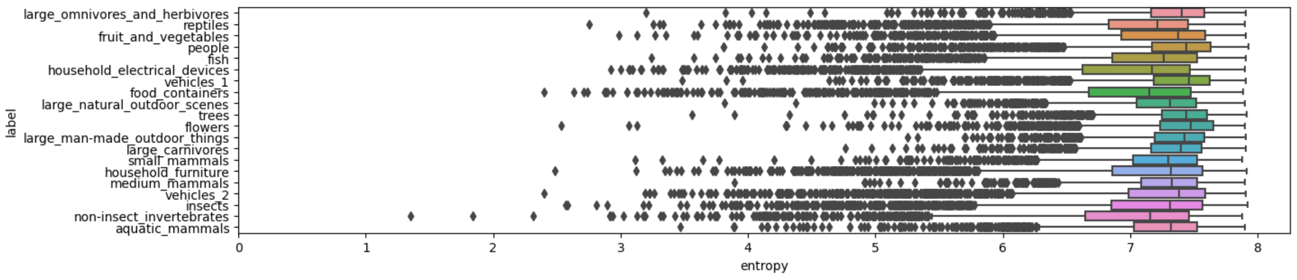


Figure 25. Boxplots of the distribution of the entropy values of every images in each 20 classes of the CIFAR-100 data set.

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 are 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 value below the first quartile and 25% 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, 2023). Figure 24 is merely a visual confirmation of the balance of the super-classes: entropy has 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 in defined as:

$$log_2(n)$$

where 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 25 shows a little more variability among the images of each class: *large man-made outdoor things* or *medium mammals* have notably less 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 colors 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 26 shows the image-wise entropy for the super-classes of group A (left, team “bipbip”) and B (right, team “roseau”).

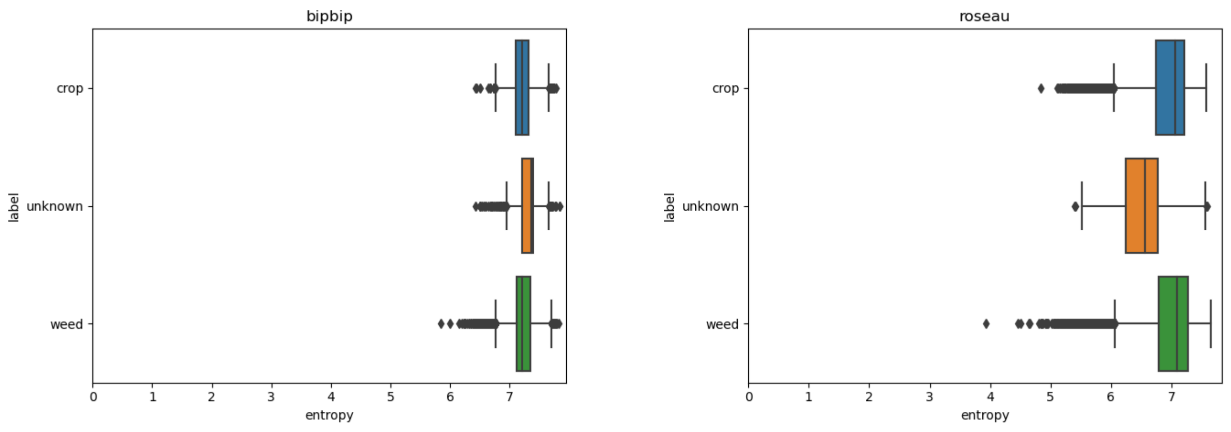


Figure 26. On the left: boxplots of the distributions of the entropy values of the images of group A (team “bipbip”) in each classes of the ROSE data set. On the right: boxplots of the distributions of the entropy values of the images of group B (team “roseau”) in each classes of the ROSE data set.

The medians of *crop* and *weed* are close, while *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. 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 9 presents the dispersion of the image sizes for both groups. The large variability explains the number of outliers display in the boxplots.

Table 9. 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 27 shows the label-wise entropy of both groups.

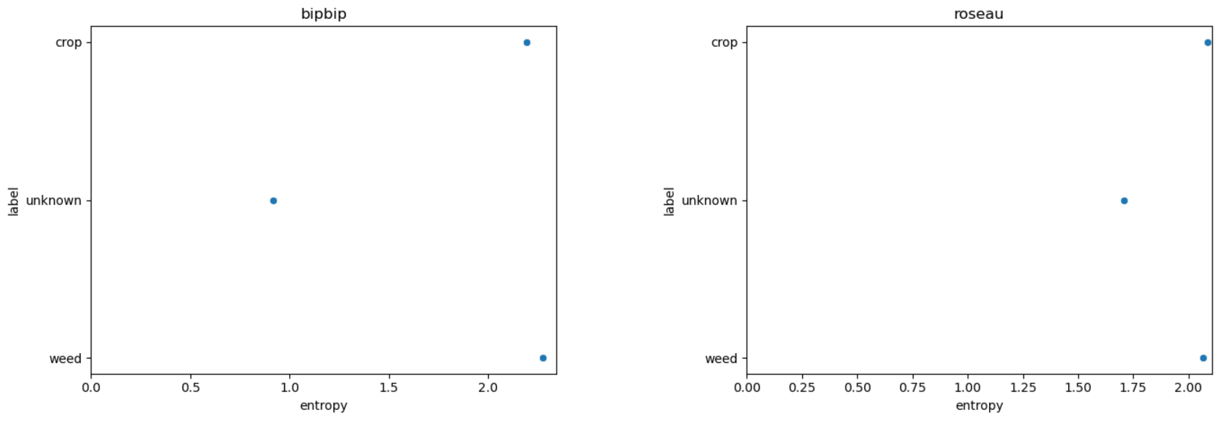


Figure 27. On the left, group A's entropy values for each label. On the right, the same plot for group B.

.

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.

3.7.3.3.2 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 colors (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 performance of the system. If such 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.

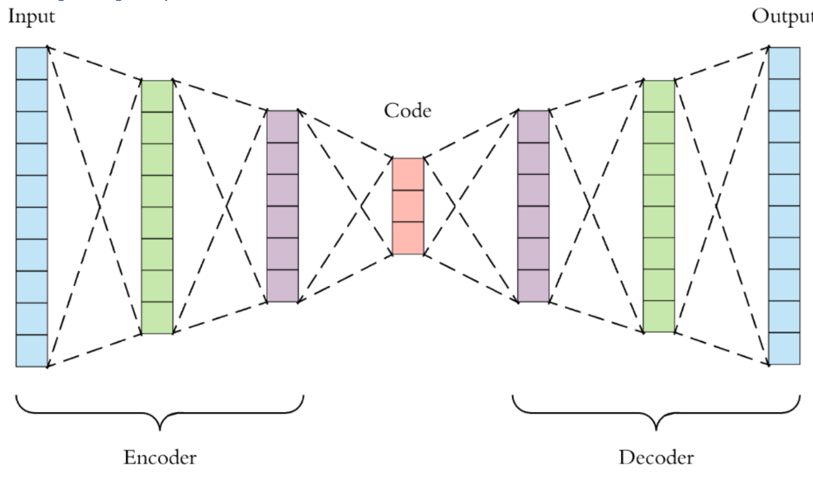
3.7.3.4 Sample-wise similarity analysis

3.7.3.4.1 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 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 informations 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 specifically identifying 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, and thus transforms an embedding into a text sequence through increasingly large hidden layers. Training both modules jointly increases the overall performance on the task. Figure 28 illustrates the general principle of the encoder-decoder architecture.

Figure 28. General principle of the Encoder-Decoder architecture. The "Code" is the learned embedding space.



Some of these models are trained on extremely large data set with learning strategies favoring 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 task with good performance. As these pre-trained embeddings are formed on sub-units of speech, there is no problem of Out-of-Vocabulary samples.

Few off-the-shelf pre-trained models with retrievable embeddings (i.e. separable Encoder and Decoder) were found during the research step. Two candidates were identified:

- End-to-End SLU[57] (Lugosch et al., 2019)
- Wav2Vec 2.0[58] (Baevski et al., 2020)

Wav2Vec 2.0 being provided by Facebook and accessible via Hugging Face, it may be perceived as a more appealing resource, easier to deploy and with better support. Moreover, Wav2Vec2 is probably more accurate in absolute value, as it relies on Transformers while E2ESLU is based on their predecessors, RNN. However, since the evaluation protocols are different for both 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. Nonetheless, the embeddings of E2ESLU have less dimensions (256 vs 512), which makes computation less costly. In the meantime, the performance of E2ESLU embeddings would 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.

The preliminary experiments were very simple: a subset of three audio samples from the Fluent Speech Commands Dataset[59] was duplicated to emulate two identical datasets (called A and B respectively). Each file is then loaded and its associated embedding is generated.

Speech embeddings (whether E2ESLU or Wav2Vec2) are 3-dimensional, with the second dimension of variable length, depending on the duration of the input sample. On the other hand, the metric of the reference paper (i.e. (Cabitza et al., 2021)'s), called Degree of Correspondence (DoC) requires 2-dimensional data (due to its internal use of the K-nearest-neighbours (KNN) algorithm) in matrix, i.e. with a fixed shape. Therefore, the embeddings underwent a two-steps 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 set are identical. The difference seemed too high to be imputable to variations caused by the internal KNN algorithm. Moreover, the algorithm was ran 2 more times, yielding scores of 0.88 and 0.66. A series of sanity checks was 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 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 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 use the Wav2Vec2 despite its potential limitations.

[57] Hereafter abbreviated *E2ESLU*, implementation available at https://github.com/lorenlugosch/end-to-end-SLU
[58] Implementation available at https://github.com/huggingface/transformers
[59] https://fluent.ai/fluent-speech-commands-a-dataset-for-spoken-language-understanding-research/

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 ran between data set A and data set B, yielding 0.96 and confirming an irreducible variability that, although small, must be taken into account in further experiments.

Finally, the choice of speech embeddings as the representation for DoC and other similarity assessment was explored. To do so, two data sets of three commands were assembled. Table 10 shows the commands chosen.

Table 10. 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 data set has a semantically similar counterpart 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 systems such as Wav2Vec may capture a wide variety of phenomena, and investigating semantic could provide directions for further experimentations.

The sanity check of comparing data set A against itself yielded a DoC score of 0.85. DoC for data set A against B yielded 0.0. This experiment shows the wide intrinsic variability of the DoC score, making it a tool to use very cautiously. Despite this variability, it seems clear that Wav2Vec embeds no or insignificant semantic information.

3.7.3.4.2 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 difficult in of itself 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.

Further experiments could be devised, taking this variability into account and exploiting embeddings capturing the semantic of the input signal. In this case, the main focus would be to enrich the metric in order to enable the analysis of what elements in the embeddings are identified as similar during computation. The DoC could also be compared with more classical metrics (such as cosine) to validate its interest and assess whether its computation is worth the computational effort. Nonetheless, it will not be considered a priority in the next phases of the MLEAP project.

3.7.4 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 feedbacks 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 of 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 endeavor 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 focusing on tabular data. The tool proposes an interface synthesizing 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 preprocessing 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 GoTo solution. 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 in (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 will be performed directly on one or more MLEAP use case since it does not require exploration, as opposed to software tools. It could not be applied on ACAS XU due to the priority given to manipulating data sets and implementing tools.

3.7.5 Experimentation conclusions

The outcome of these preliminary experiments is positive overall. Many methods have been implemented and validated on small data sets. In addition, some methods have been tested on high scale data set, allowing for the identification of technical challenges. This important groundwork offers a solid testbed for the continuation of experimentations by enabling a smoother testing of the remaining methods, along with the deepening of the analysis already implemented, either through complementary experiments or 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 characterization of the data set (beyond its description) through simple metrics and tools prior to investigate 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 to get 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, comparison of two data sets is easy and may be more insightful but 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.

3.8 Conclusion

In light of all the elements discussed in this document, it appears that 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.

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 a variety of approaches, from devising data-quality-aware objective function 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 preliminary experimentations empirically confirmed this combination of scarce information, complex operationalizability and lack of focus on the specific issues of completeness and representativeness. Putting up a framework for such assessment would require significant engineering and expert knowledge. While it is probable that such 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.

4. Model development: Generalization properties

4.1 Introduction

This part is about evaluating Model Generalization in AI. It aims to review existing methods of machine learning (ML) and deep learning (DL) generalization guarantees and evaluation, distinguishing between methods from the two fields. Problems and limitations are analysed, and then new evaluation methods development is presented. This section includes a comprehensive overview and state-of-the-art analysis of existing methods for ML and DL models in terms of generalization process and bounds. In this section, we first present the issues of generalization and the known ML/DL-related problems of *underfitting* and *overfitting*. 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 generalizing. Furthermore, we comprehensively review the available methods and tools to evaluate generalization bounds. Finally, we identify the barriers to the tractability of the objective of quantification of generalization guarantees for a given AI model, and provide a generic development and evaluation pipeline, dealing with the identified limitations to promote generalization, after training and model implementation.

4.2 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 the second concept paper by EASA[60] (EASA, 2023).

4.2.1 AI modelling

Using and implementing artificial intelligence (AI) applications involves creating a mathematical model function f. The latter is then fitted to some training data, in a given context, to be able to process additional data and hence make predictions. Several models have been defined for AI applications. There are two main categories: (1) ML models, which could be supervised[61] or not, and which are 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 top of each other. DL uses supervised and unsupervised strategies to learn multi-level representations and features in hierarchical architectures, for several different tasks, such as classification or regression. A detailed description can be found in (Sarker, 2021).
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 $\emptyset = D_{train} \cap D_{test} = D_{train} \cap D_{val} = D_{test} \cap D_{val}$.

[60] https://www.easa.europa.eu/
[61] 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.

F is a *set of possible models* produced by the learning algorithm, sometimes called *hypothesis space*. 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 29.

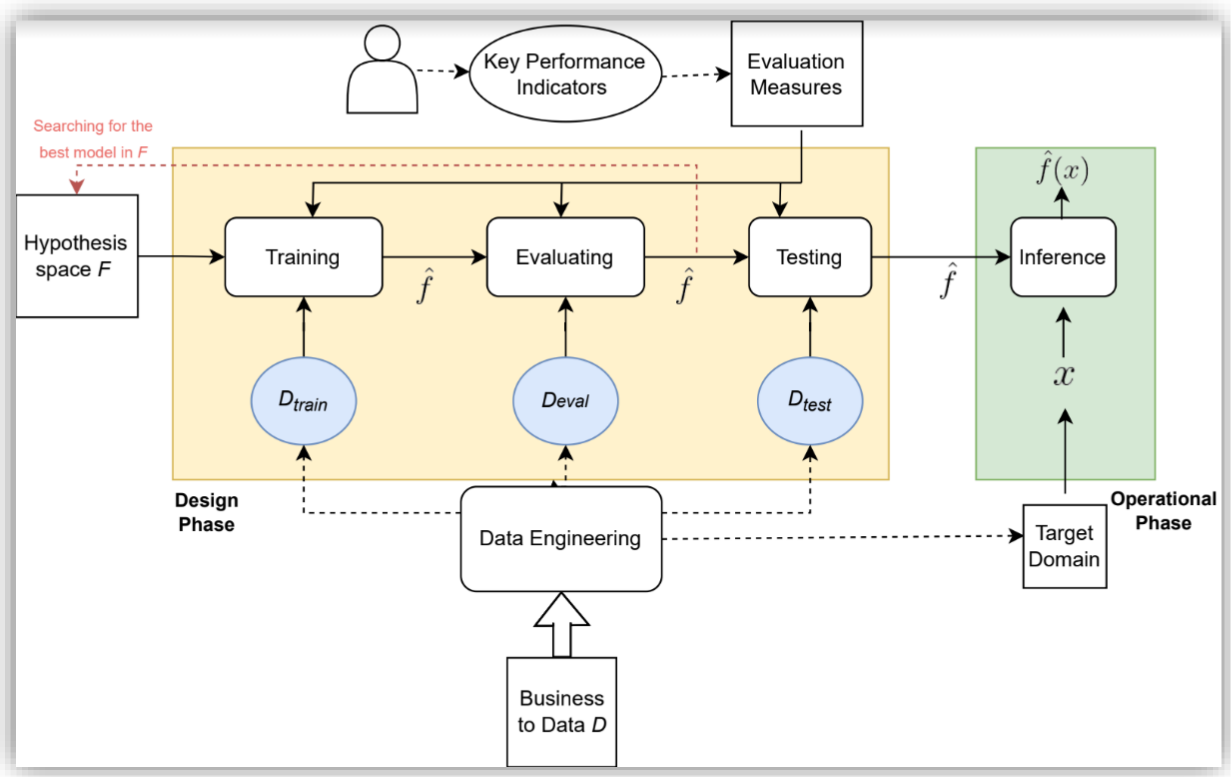


Figure 29. From the design to the operational[62] *phase of machine learning modelling and production.*

1. *The design phase:* which includes the ML model (algorithm) choice, the training on a dedicated data set, evaluation of the best model, and finally testing its performance. In this phase, it is very common that a model can 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. Here, the trained model is simply used to make predictions for new inputs (inference).

In both phases, design and operational, we assume that a same data engineering pipeline is used to process data samples. The evaluation measures (cf. section 5.3) can be derived from the key performance indicators that are provided by the domain application expert.

4.2.2 Learning process

The learning process aims at going through a learning algorithm F and repeatedly updating the model's parameters. The goal of a (supervised) learning algorithm (Cluzeau et al., 2020) F 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

[62] 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.

$(x, f(x))$, 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}:

$$D_{train} = \{(x_i, f(x_i)) : 1 \leq i \leq n_{train}\}$$

the goal of the training algorithm F is to generate a function

$$\hat{f}(D_{train}) : X \to Y$$

that approximates f "well", as measured by some error metric. $\hat{f}(D_{train})$ is the result of learning algorithm F trained on data set D_{train}.

Hence, the learning process aims to approximate the best function $\hat{f} \in F$ with the suitable parameters, (weights and biases, referred to as hyper-parameters θ) that best perform the ongoing task.

4.2.3 Error metrics

The pointwise quality of the approximation of f by $\hat{f}$ is measured by using a predefined choice of *error metric(s)* $m : Y \to R_{\geq 0}$ expecting that

$$\sum m\, F(D_{train}), f(x)$$

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 differences $m(y_1, y_2) = |y_1 - y_2|$ (respectively $(y_1 - y_2)^2$).

These metrics are useful to identify whether the model can keep high performances in front of data changes and perturbations (robustness aspect), and how it would behave in new data instances that have not been evaluated during training (generalizability aspect).

4.2.4 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 function in the presence of abnormal or unknown inputs, and to provide 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 refer also to the generalization 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. Which 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 section 5, which provides a more detailed definition and analysis of the "*robustness*" as a quality of AI applications.

4.2.5 Generalizability

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 generalize the learned knowledge to a new context or environment. The most important success indicator is to produce a model that can perform well on an unseen data set $X \neq D_{train}$ (*out-of-sample*[63] examples), and not simply memorize the *in-sample*[64] examples of the subset D_{train}. The memorization aspect is used to describe an overfitting behaviour of the learning process. The next sections will describe more these aspects and show how to recognize the lack of generalization.

4.2.6 Stability

In the scope of analysing a model robustness, the *stability* concerns the learning algorithm and the model itself. According to (Cluzeau et al., 2020), the performance stability can be observed based on the behaviour of the trained model $\hat{f}$ when it deals with 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 to 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) the model stability, associated to the 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 well and which types of changes in environment will degrade performance. The stability characteristic of a model is important 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 as well as the trained model. This notion is further developed in chapter 5.

4.3 Overfitting and underfitting

4.3.1 Overfitting vs Underfitting

The overfitting and underfitting are two aspects that show that the ML/DL model is not learning well, or that it is not able to perform on unseen data as well as it does on training samples. According to (Cluzeau et al., 2020), on 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 very sensitive to it (i.e. variance increases). These two facts yield a specific and different evolution of error values. Figure 30 shows variance and bias tradeoff that optimizes the learning errors.

[63] 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).

[64] According to the CoDANN paper (Cluzeau et al., 2020), the in-sample data includes all the training samples. According to the CoDANN paper (Cluzeau et al., 2020), the in-sample data includes all the training samples.

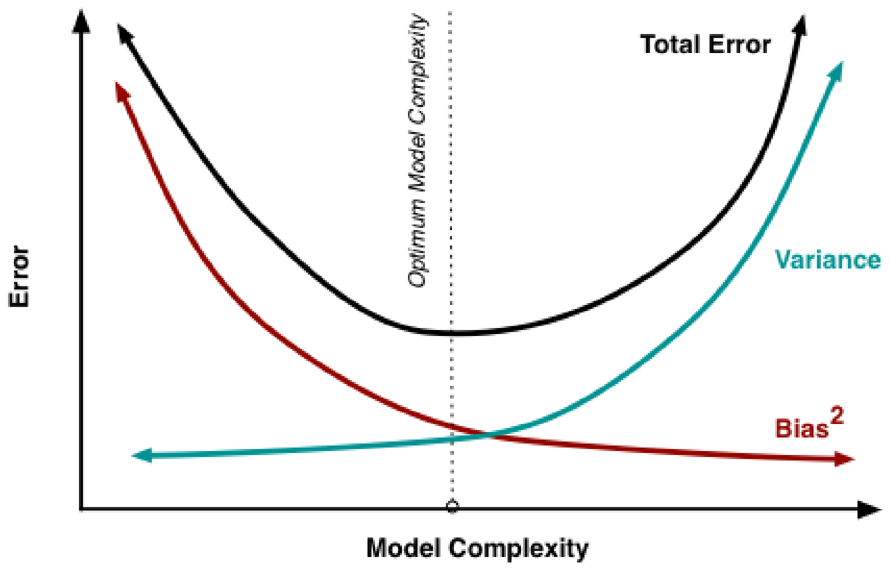


Figure 30. Bias-variance trade-off[65], for better error optimization, based on the model's complexity.

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 F chooses among a finite set of models, such that: if F contains a single model, the variance will be zero, but the bias might be large, if the single model does not approximate well the target; in contrast when F contains different models, the bias should be smaller, since there is more capacity to find a model that approximates $\hat{f}$ well, but the variance will be non-zero.

Both overfitting and underfitting come with risks: overfitting will lead to models that do not generalize well (may not perform well in unseen contexts), while underfitted models will not achieve a satisfactory trade-off[66]. 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).

4.3.2 Under/Over-fitting detection

As explained above, overfitting occurs when an algorithm reduces error through the memorization 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 generalizing the same performances, from observed data during training, to unseen data during testing (Ying, 2019). This happens mostly 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

[65] https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

[66] Note that current aviation standard on AI (Liao et al., 2022), intends to refer to "bias variance optimisation" rather than "bias variance trade-off" Note that current aviation standard on AI (Liao et al., 2022), intends to refer to "bias variance optimisation" rather than "bias variance trade-off"

generalizing the same performances, from observed data during training, to unseen data during testing (Ying, 2019). This happens mostly 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 not a sufficient model capacity or sufficient training to fully learn the exact relationship between input and output spaces, whether through memorization or not. To determine a model's generalization 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 generalization. Here, we first define the aspects related to the training algorithm, then bridge the relationship with the resulting model ability to generalize:

- ***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), namely,

$$C_f = sup_D \, I(f \mid D_{train})$$

  Note that the maximum amount of information that an model may transfer from a data set, w.r.t. a hypothesis, is the number of bits required for a model to memorize a one-to-one mapping between each *feature-label* pair in that data set. Hence, the capacity aspect has a high impact on the model's generalization.

- ***True risk (loss):*** is the aggregated expected loss values over the different data samples x of the test data set: $R_{D_{test}}(f)$ is estimated by a sampling of the test data sets.

- ***Overfitting:*** is diagnosed by comparing the losses of an model 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 $R_{D_{test}}(f)$ exceeds the empirical risk $\hat{R}_{D_{train}}(f)$ (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 $f \in F$ such that

$$R_{D_{test}}(f) > \hat{R}_{D_{train}}(f)$$

  Where:

$$R_{D_{test}}(f) = \frac{1}{m} \sum_{j=1}^{m} l(f, x_j) \; < \; E$$

  for some fixed scalar $\mathcal{E} > 0$ for any data distribution D_{test}, and

$$\hat{R}_{D_{train}}(f) = \frac{1}{n} \sum_{i=1}^{n} l(f, x_i)$$

- ***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 is used in order to bridge the gap between the observed evolution in simple ML models compared to 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 memorize all the training set, including irrelevant signals (noise), leading to overfitting;.

  To deal with this problem, one way 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 in order to restrict the search space of the models.

4.4 Generalization in AI

Generalization 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 found in past observations. This is called generalization (C. Zhang et al., 2021): finding rules consistent with available data that apply to instances we have yet to encounter. Thus, there are a variety of theories proposed to explain generalization. Uniform convergence[67] [G. H 1918], margin theory (Wei et al., 2019), and algorithmic stability[68] (T. Liu et al., 2017) are some of the important conceptual tools to reason about generalization. In addition, the capacity of a model can be evaluated to help predict generalization. When the complexity of a model is very high, regularization introduces algorithmic tweaks intended to reward models of lower complexity.

4.4.1 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, optimization process, and data complexity. In deep learning, the model complexity is concerned with the network architecture and how

[67] 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 $f_N, f_{N+1}, \dots$ differ from f by no more than ε at every point x in E.

[68] Captures stability of the hypothesis output by the learning algorithm in the normed space of functions from which hypotheses are selected.

complicated problems the model can express (Bianchini and Scarselli, 2014; X. Hu et al., 2020). While 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; while in logistic regression, it is investigated using several metrics, such as the perspectives of Vapnik-Chervonenkis (VC) theory[69] (Tempo et al., 2013), Rademacher complexity (Kakade et al., 2008), Fisher Information matrix (Bulso et al., 2019), and the razor of model (Balasubramanian, 1997).

Model complexity can be categorized 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 that are two classic measures of expressive capacity in machine learning.

2) *Effective complexity*: also known as *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 parameterizations [89]. It is based on two different aspects:
   - *1)* General measures of effective complexity that design quantitative measures for effective 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.

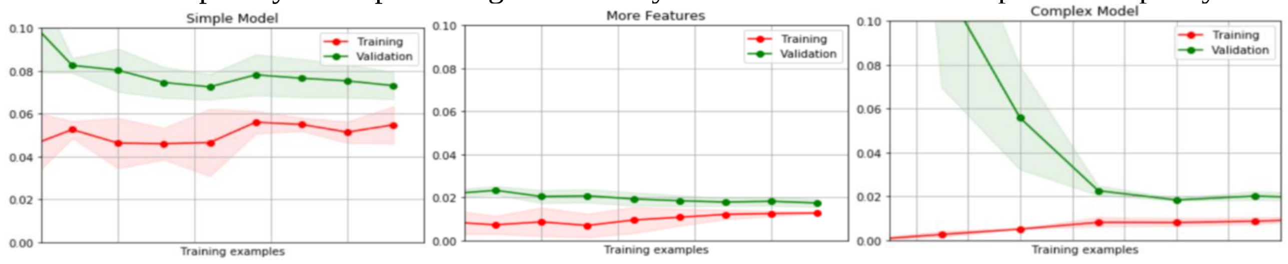


Figure 31. Learning curves[70] showing the error evolution of different models with different complexities

Figure 31 shows how a model complexity impacts the training. We can see that adding more features reduces the error on both training and validation data. But when complexity is increased again with even more features, training error improves but validation error does not, which means that the model is beginning to over-fit. A trade-off needs to be made:

[69] 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.

[70] https://www.pico.net/kb/overfitting-variance-bias-and-model-complexity-in-machine-learning/

- Using fewer features reduces model complexity. However, inadvertently removing key features that are important 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, increase model complexity.

High-Capacity Low-Reality Phenomenon

The *effective complexity*, reflecting the model's characteristics that makes it practical enough to address target problems, and the *expressive capacity*, reflecting its ability to express those problems, are highly related and need to 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, as well as deep neural networks, can. Such that, 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 that is as high as the deep model's accuracy. 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 optimization algorithms, and explain the model efficiency and results (Kalimeris et al., 2019). Furthermore, effective model complexity can be defined based on the relationship between the model and target data. 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 utilization and model performance, according to every use case's technical requirements and characteristics.

4.4.2 Generalization Error Bounds

Generalization bounds are statistical tools that take as input various measurements of a predictor on training data, and output a performance estimate for unseen data — that is, they estimate how well the predictor generalizes to unseen data. Knowing the generalization 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 risk of its hypotheses, given randomly selected training samples, a generalization 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 risk that is no greater than some value ε:

$$P_D\left(\left|l_{out}(\hat{f}, m) - l_{in}(\hat{f}, D, m)\right| < \varepsilon\right) > 1 - \delta$$

l_{out} is the out-of-sample loss (error) value, and l_{in} is the in-sample loss. $\delta \epsilon (0, 1)$ is a probability tolerance, and ε is the generalization gap tolerance. The *Probably Approximately Correct* (PAC)-learning setting is then defined by the inverted form (Cluzeau et al., 2020):

$$P_D > 1 - \delta\colon\ G(\hat{f}, D) < \varepsilon(\delta, |D|, bounds)$$

The bounds represent several involved theoretical aspects, and they all manifest an intuitive idea: models that are more complex are more prone to overfitting and lesser generalization (Cluzeau et al., 2020). 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 $\hat{f} \in F$:

$$G(\hat{f}, D) \leq \sqrt{\frac{func(model\ class\ F\ complexity) + \log(1/\delta)}{\|D_{train}\|}}$$

$$G(\hat{f}, D_{train}) \rightarrow 0 \quad as \quad \|D_{train}\| \rightarrow \infty$$

In the following, we review some of the most used bounds in ML, grouped following the taxonomy proposed in CODANN-1 document (Cluzeau et al., 2020).

4.4.2.1 Data-independent, algorithm-independent

This is the first bounds class to be derived by the Vapnik-Chervonenkis theory (VC) (Tempo et al., 2013), and 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:

$$VC_{dim} = \tilde{O}(d * \dim(\theta))$$

Where, θ is the set of trainable parameters (weights and bias), and d (depth) is the number of layers of the network. Note that the notion of VC-dimension does not take into account properties of a particular data set, which is important in practice for a better estimation of generalization bounds (Dar et al., 2021).

4.4.2.2 Data-dependent, algorithm-independent

Using data set information, such as the structure and size, could help for better tightening of generalization bounds. 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 VC-dimension, the Rademacher complexity takes data distribution into consideration and therefore 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). Generalization bounds that are data-dependent and algorithm-independent scale inversely with data margin[71] (which is a training-data dependent quantity). These bounds are directly proportional to the Rademacher complexity and can explain good generalization for classification, in some cases of overparameterization (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.

[71] 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.

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 are always too high, so the practical guidance they can provide is weak. Recently, (D. Li et al., 2022) have investigated the domain generalization problem providing a novel learning-theoretic generalization bound that bounds unseen domain generalization performance in terms of the model's Rademacher complexity. This analysis suggests that domain generalization should be achieved by simply performing regularized Empirical Risk Minimization (ERM) with a leave-one-domain-out cross-validation objective.

4.4.2.3 Data-dependent, algorithm-dependent

Methods of this class take into account the distributional properties of the learning algorithm. 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 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.

4.4.2.4 Overview of recent bounds

In several cases (Dziugaite and Roy, 2017), the values of the generalization 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 D_{val} can then be used to optimize the model better during training epochs. Besides, bias and variance need to be estimated and minimized in the train data set. The analysis in (Cluzeau et al., 2020) suggest 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} in order to train the model in the subsets separately. Note that bootstrap sampling uses random sampling with replacement. This means that it is very much possible for an already chosen observation to be chosen again. The resulting subsets $D_{i=1\ldots k}$ will help estimating the variance for the different subsets separately.
2) *Jackknife* (Miller, 1974): consists in 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 produces the most different trained instance. 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 generalization behaviour while large ones fail.

Another taxonomy of the different theoretical and empirical generalization bounds is provided in (Valle-Pérez and Louis, 2020), where the models are grouped based on assumptions about the data, about the algorithm or even according to the dependence of their capacity on the training set. Moreover, in DL models, the classical aforementioned bounds 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 models generalization:

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[72] (Nakkiran et al., 2021) shows that it can be fully explained by classical VC-generalization bounds. This is by an application of 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, if a complex model can be replaced by a simpler one, up to some small admissible error, it might be possible to obtain stronger generalization bounds on the simple model. Hence, the objective is to perform a model reduction process to reduce the complexity of the model in a low resources setting (Choudhary et al., 2020).
3) *Based on Model Distillation* (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[73] (Nakkiran et al., 2021) shows that this latter can be fully explained by classical VC-generalization bounds. This is by an application of analytic VC-bounds for modelling double descent in classification problems, using empirical results for several learning methods.
4) *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, if a complex model can be replaced by a simpler one, up to some small admissible error, it might be possible to obtain stronger generalization bounds on the simpler model. Hence, the objective is to perform a model reduction process to reduce the complexity of the model in a low resources setting (Choudhary et al., 2020).
5) *Based on Model Distillation* (Hsu et al., 2021): differently than the compression-based methods (Meir and Fontanari, 1993), the objective of distillation is, given a high complexity network with poor generalization bounds, one can distil it into a network with nearly identical predictions but with lower complexity and vastly better generalization bounds. After training and optimizing the bounds of the distilled network, the purpose is then to carry the good generalization bounds of this network back to the original network. A complete evaluation of the different theorems can be found in (Hsu et al., 2021).
6) *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 optimized (Dziugaite and Roy, 2017) and shown that one can bound the generalization gap of a two-layer-stochastic neural network. Another method (Nagarajan and Kolter, 2019), namely *Deterministic PAC-Bayesian* provides generalization bounds computation method for DL models via generalizing 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 generalize to the interactions between the matrices on test data.
7) *Statistical guarantees* (V. N. Vapnik, 1999): in this class, the generalization bounds are defined based on statistics either from data, especially for classes of estimators that are used in practice

[72] The double descent means that multilayer neural networks can be trained to achieve zero training error, while generalizing well on test data.
[73] The double descent means that multilayer neural networks can be trained to achieve zero training error, while generalizing well on test data.

(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 generalization 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 (with respect to the data distribution) and the local smoothness of the objective function along the SGD path, as well as the sensitivity of the loss function to perturbations to the final output. Recently, Taheri et al. (Taheri et al., 2021) have introduced a "scale regularization" approach, a general class of regularized least-squares estimators. The main strategy is to disentangle the parameters of a model into a ‘‘scale’’ and a ‘‘direction’’ – similarly to introducing polar coordinates – which allows data engineers to focus the regularization on a one-dimensional parameter. The scale regularized 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 regularization. Exemplified in the l1-regularization, this method provides a guarantee for the squared prediction error which decreases essentially in the number of training samples.

8) *Geometry analysis bounds*: in previous studies (Russo and Zou, 2016), it was well known that the generalization 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 generalized beyond the dependencies between input and output information. Other research interests (Rodríguez Gálvez et al., 2021) have developed bounds that are based on Wasserstein distance. More specifically, it introduces full-data set, single-letter, and random-subset bounds, and their analogous in the randomized subsample setting from Steinke and Zakynthinou (Steinke and Zakynthinou, 2020). Moreover, when the loss 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 subgaussianity 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 generalizability (Cluzeau et al., 2020) related to theoretical guessing of the generalization performance. Hence, the generalization bounds associated with the theoretical methods could be larger than the values in practice. Therefore, practical approaches, such as *Regularization* (cf. Section 4.5.1) in its different forms (e.g. *batch normalization* and *early stopping*) and *domain generalization* (cf. Section 4.4.5) 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 generalization, and showed surprising failures of some measures.

4.4.3 Generalization 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).

This can be formalized by generalization guarantees. 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.

4.4.3.1 Function of Data Set Characteristics

Considering the generalization aspect as a function of a target data set means that we need to capture relationships between the conception and design context, compared to the target context. In (Barbiero et al., 2020), the generalization 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 generalization. In addition to several predictable correlations in different data samples, there are weak associations between the generalization ability of ML models and metrics related to dimensionality, such as the *curse of dimensionality* that might impair generalization in machine learning.
The *curse of dimensionality* denotes a variety of phenomena hindering data analysis if a large number of variables need to be considered at the same time (Bellman, 1966; Bittner, 1962). This aspect prevents ML models from generalizing and includes several 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 generalization problem. Extrapolation comes from computational geometry. If we 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, a ML model will interpolate using known data to obtain a prediction; while the same model will extrapolate for test points placed outside of the convex hull. An example is presented in Figure 4, where data points of $\mathbb{R}^2$ are presented.

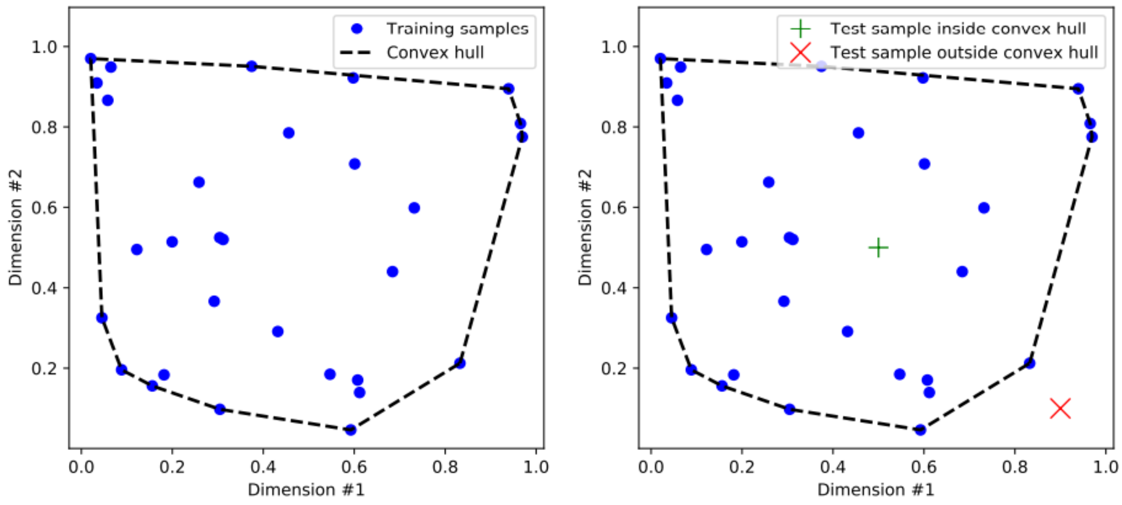


Figure 32. Illustrating the convex hull aspect of intra/extrapolation of training and testing data samples in a two-dimensional space (Barbiero et al., 2020).

4.4.3.2 Function of Model Characteristics

For a 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 optimization and evaluation of the measures of performance, such as accuracy and precision, the ability of a model to generalize 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 generalization ability provides some guarantees on the model's applicability to new unseen data, especially in real-world situations.
As discussed in 4.4.1, the model's complexity and effective capacity are two main factors defining its ability to generalize the learned evidence to unseen data samples. Besides, the bias-variance trade-off and model's number of trainable parameters and data effectiveness are other criteria that considerably impact the robustness of a 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 Generalization Error (true error $\hat{R}(f)$) as defined in Section 4.3.2. Other measures like norm-based control and sharpness could also be used to assess the model's ability to generalize.

4.4.4 Generalization in DL

Deep neural networks (DNNs) trained in effective data sets exhibit good generalization behaviour, even when the number of parameters is significantly larger than the amount of training data (Neyshabur et al., 2014; C. Zhang et al., 2021). In those settings, the objective function has multiple global minima[74], all minimizing the training error, but not all of them generalize well (Neyshabur et al., 2017). Picking the wrong global minima can lead to bad generalization behaviour, which makes the training insufficient for learning. Different methods are used to minimize the training error for better optimization, such as the initialization, update rules, learning rate, and training stopping conditions, all of which lead to different global minima with different generalization abilities. For example, Path-SGD (Neyshabur et al., 2015a) is an optimization algorithm that is invariant to the rescaling of weights and showed better generalization behaviour over the classical SGD (stochastic gradient descent) algorithm (Bottou and others, 1991), in the training of different evaluated DNNs. In addition, smaller batch[75] sizes for training with the SGD algorithm generalize 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 generalization, i.e. that the test error is close to the training error, even when minimizing training error. This corresponds to the maximum number of examples with which one can obtain small 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 generalize.

---

[74] 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 minima is called so since the value of the loss function is minimum there, globally across the entire domain of the loss function.
[75] 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.

4.4.4.1 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 has been observed when learning on the MNIST[76] data, using an increasing number of channels, as shown in figure 30.

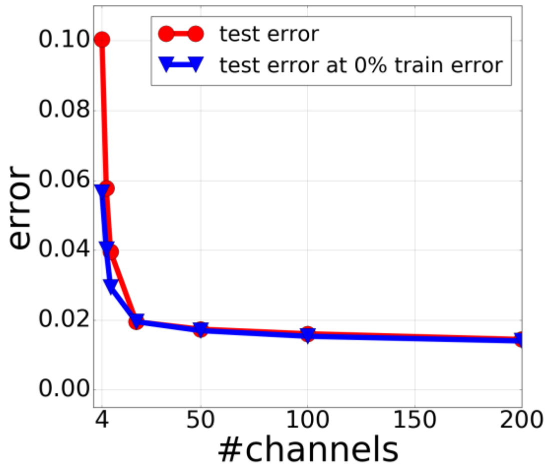


Figure 33. Illustration of the test error evolution depending on the number of channels used in a NN (Brutzkus and Globerson, 2019)

In practice, over-parametrized settings are used 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 not enough, since NNs having significantly more parameters than samples can perfectly fit even random labels, without generalizing (Kawaguchi et al., 2017).(Kawaguchi et al., 2017). Moreover, measuring complexity in terms of the number of parameters cannot explain the reduction in generalization error as the number of hidden units increases (Neyshabur et al., 2015b). Recently, another study (Liu, 2021) have shown that sparse NNs can even generalize better than their dense counterparts, and proposed different efficient approaches that can yield sparse neural networks with good generalization bounds.

4.4.4.2 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[77] activation function. Hence, the capacity can be bounded based on the l_1 norm of the weights of hidden units of every layer 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 generalization ability of a DL model:

1) l_2 norm-based capacity

[76] http://yann.lecun.com/exdb/mnist/

[77] 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.

$$\frac{1}{\gamma_{margin}^2} \prod_{i=1}^{d} 4\|W_i\|_F^2$$

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)

$$\frac{1}{\gamma_{margin}^2} \left| \sum_{j \in \prod_{k=0}^{d}[h_k]} \left| \prod_{i=1}^{d} 2W_i[j_i, j_{i-1}] \right| \right|^2$$

Where $\prod_{k=0}^{d}[h_k]$ is the Cartesian product over data sets $[h_k]$.

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 bounds can be found in (Neyshabur et al., 2017), where an empirical investigation of the appropriateness of different complexity 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 generalization 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 new capacity in order 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 capacity of the network. The reported results in (Neyshabur et al., 2017) have shown a gap between the complexity of models learned on real and random labels for all compared norms, with the difference in the increase in capacity between true and random labels being most pronounced for the l_2-norm and l_2-path norm.

4.4.4.3 Uniform Stability

Stepping away from complexity measures corresponding to different models of the hypothesis class, another way is to evaluate the generalization by considering properties of the training algorithm (C. Zhang et al., 2021). Uniform stability[78] of an algorithm measures how sensitive the algorithm is to the replacement of a single example. However, it is solely a property of the algorithm, which does not take into account details of the data or the distribution of the labels. The weakest stability measure is directly equivalent to bounding generalization error and does not take the data into account.

[78] 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 [88]–[90]. 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 generalization aspect of DL models to make a deep understanding of it, where the main question is *"What practices promote generalization? And what does it measure?"*. Conducted experiments in (C. Zhang et al., 2021), on both the CIFAR10[79] and ImageNet[80] data sets, for image labelling have shown how DNNs fit random labels, through several randomization tests. The main outcome is that the effective capacity of neural networks is sufficient for memorizing the entire data set, and even optimization on random labels remains easy, the training time increases only by a small constant factor compared with training on the true labels. Furthermore, randomizing labels is solely a data transformation, leaving all other properties of the learning problem unchanged.

4.4.5 Domain generalization

The main 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 minimize empirical risk on a single domain often fail to generalize when applied to other unseen domains (Bayasi et al., 2022), due to domain shift.

In the previous sections, we described some widely used methods that can help identify *a priori* the ability of the model to generalize to unseen data samples. Apart from 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 take into account the target application features and the domain characteristics. When it comes to *domain generalization*, 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 generalize 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 both training and target application (Chung et al., 2018). A more detailed taxonomy of different domain generalization methods is highlighted in Figure 34.

[79] http://www.cs.toronto.edu/~kriz/cifar.html
[80] https://www.image-net.org/

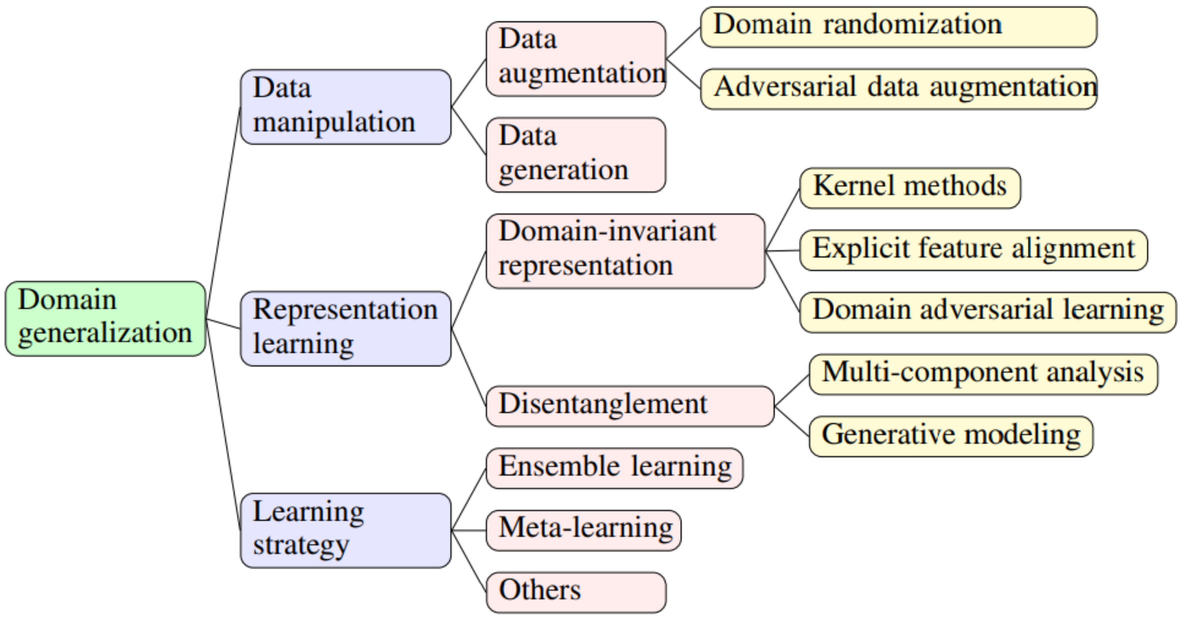


Figure 34. Taxonomy of domain generalization state of the art (J. Wang et al., 2022).

As shown in Figure 34, several solutions can be adopted to achieve a cross-domain generalization 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 generation of new instances (Gordon et al., 2020) or 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 generalization can be found in (Zhou et al., 2021a).
In this document, we study the domain generalization 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 think about 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 the one of the training domain, and the training process is thus adjusted.

4.4.5.1 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 minimize 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 consists in 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 minimize 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 distributions mismatch between training and target domains, by minimizing 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, in the aim of defining the domain generalization 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 that are invariant to different marginal distributions (Blanchard et al., 2011). This process allows a model to discover stable patterns across source domains, which generalize better to unseen domains. While the *single-source* setting assumes that training data is homogeneous (Hendrycks and Dietterich, 2019), and does not require domain labels for learning and thus they apply to multisource scenarios as well. Independently of the domain specificity, the general Out-of-Distribution (OOD) generalization problem addresses every challenging setting where the testing distribution is unknown and different from the training. Hence, the OOD entities need to be handled carefully to ensure the model's robustness (Shen et al., 2021).

4.4.5.2 Learning adaptation

Another way to think about the domain generalization problem is to focus on exploiting the general learning strategy of ML/DL, to promote domain generalization capabilities. To do so, the main paradigms[81] 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 multiple instances of the same model with different initialization 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* uses 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 generalization, exemplar-SVMs select the top-K exemplar classifiers that give 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 specializing 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,

[81] 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*).

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 Normalization. Domain-Specific Neural Networks* aim to learn set of neural networks, each specializing 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 Normalization.* In batch normalization (Ioffe and Szegedy, 2015), the statistics are computed on-the-fly during training and their moving averages are stored in buffers for inference. For a domain generalization purpose, batch normalization has served for mixing statistics of multiple source domains, in order to learn generalizable 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 is used to improve 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, consists in 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 generalization, a general strategy is to divide the multi-source domains into meta-train set and meta-test set (Rajendran et al., 2020). The motivation behind applying meta-learning to domain generalization is to expose a model to domain shift during training with a hope 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, there are two main components to 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) in such a way that the test error on the meta-target domain can be reduced, which is often achieved by bi-level optimization. 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 generalize well to novel testing domains, the model agnostic training procedure (D. Li et al., 2018) simulates train/test domain shift during training by synthesizing 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 generalization 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 generalization.

- ***Self-supervised learning.*** (Jing and Tian, 2020) is often referred to as learning with free labels generated from data itself. This can be achieved by teaching a model to predict the transformations applied to the inputs. In domain generalization, self-supervised learning can be applied to both single and multi-source scenarios without requiring any domain labels. In single pretexts (i.e. single source scenarios), in addition to optimizing 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. While in the 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 images 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 generalization 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 generalization.

- ***Transfer learning.*** (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 application. This is due to 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 generalization (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 generalization methods based on TL do not require having the target data for the model fine-tuning phase in the downstream tasks. We assume to have no access to the target data, thus focusing more on model generalization. In (Zhou et al., 2021a), a theoretical comparison between the domain generalization objectives and TL intentions is provided. One of the common features of TL and domain generalization 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 generalization 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 set and then fine-tuning the model not only improve its performance on downstream tasks, it 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 generalize to new tasks containing only a few samples with supervised information. However, the main limitation of this approach is that the empirical risk minimization is unreliable (Wang and Yao, 2019) due to the low number of training examples for risk minimization. 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 in (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 be able to classify classes covered by them. This approach is however less practical since it still requires the complete knowledge of the target domain's possible classes. 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 case, an open problem is the development of incrementally learning systems capable of assimilating more and more concepts over time from a stream of data. When the acquisition of incrementally available information from non-stationary data distributions is continuous, catastrophic forgetting[82] or interference of information is likely to happen (Parisi et al., 2019). This limitation represents a major 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 on which information becomes incrementally available over time, e.g. Traffic management applications

[82] 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.

(Nallaperuma et al., 2019). LL-learning models learn continuously while retaining previously learned experiences, which is different from domain generalization (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 in 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 through a set of classifiers and a data representation simultaneously. 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 a sequential learning of several domains (D. Li et al., 2020), inspired from LL-learning, where accumulated experience means that learning the n^{th} thing becomes easier than the *first* thing. Applied to domain generalization, this means that the performance at domain n depends on the previous $n-1$ learned problems. Thus, backpropagating through the sequence means optimizing 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 generalization problem in a different way, aiming to explore causal variables for prediction and became more practical recently (Schölkopf, 2022).. Other methods based on stable learning have been recently used (Cui and Athey, 2022). Compared with domain generalization and causal learning, stable learning motivates another way of incorporating causal inference with machine learning, which significantly relaxes the requirements for multiple environments. Such that, 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 generalization in (Shen et al., 2021).

4.5 Methods to boost generalization

Several solutions for a better generalization have been proposed in the literature. The impact of each identified generalization strategy has been empirically investigated by several studies (Ying, 2019), showing how each of them could help the targeted application. In this section, based on the taxonomy by (Kukačka et al., 2017), we summarize the most used solutions, showing results from some comparative studies, then discuss their strengths and weaknesses. The impact of every generalization strategy has been empirically investigated by several studies (Ying, 2019), showing how each of them could help the targeted application. In this section, based on taxonomy by (Kukačka et al., 2017), we summarize the widely used solutions, showing results from some comparative studies, then provide a relative discussion about the benefits of each of them.

4.5.1 Regularization

Regularization is a method to avoid high variance and overfitting as well as to increase generalization (Müller, 2012). In deep learning especially, regularization has a broader definition: *regularization is a* *technology aimed at improving the generalization ability of a model* (Tian and Zhang, 2022). Widely used in deep learning, it allows a better generalization to unseen data, even when training on a finite and small training data set, or with an inappropriate optimization method. It encompasses any modification made to a learning algorithm with the intention 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.

4.5.1.1 Data-driven

The quality of a trained model depends on the quality and volume of the training data. It is possible to employ regularization 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 larger, possibly infinite, augmented data set (DeVries and Taylor, 2017). Both approaches (a) and (b) are somewhat independent and may be combined. Regularization via data relies on transformations with (stochastic) parameters (Kukačka et al., 2017). The latter is a function that 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 categorize the data-based methods according to the properties of the used transformation and of the distribution of its parameters:

1) ***Stochasticity of the transformation parameters***. 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 larger, 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, 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 preprocessed 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 as part of the architecture; or the target output space to help the model's fast learning (in this case, it is used only during 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 or specific 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.*** It 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). In the latter, for example, multiple augmented variants of a sample can be classified and the result is then aggregated over them.

6) ***Batch Normalization***[83]. Is an operator that normalizes 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 regularization, batch normalization is usually found to improve the generalization performance (Santurkar et al., 2018).

Note that the data-driven regularization methods may affect ability to meet representativeness and completeness (cf. Section 3) objectives from EASA CP (EASA, 2023). Those two objectives should be assessed after data-driven regularization. Several other data-driven methods for regularization are cited and classified in (Kukačka et al., 2017), please, refer to this reference to know more about how one can exploit data processing and transformations to let the DL model learn and generalize better.

4.5.1.2 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 constructed to have certain properties or match certain assumptions in order to have a regularizing 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 generalize better:

1) ***Assumptions about the mapping.*** 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 minimize the number of trainable parameters. Reusing a certain trainable parameter in several parts of the network is referred to as weight sharing (Xie et al., 2021). This usually makes the model less complex than using separately trainable parameters. An example are convolutional networks (Li et al., 2016). Here weight sharing does not only reduce the number of weights that need to be learned; it also encodes the 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 right activation function is quite important. There are several activation functions, 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 problem i.e. gradient tending toward zero, sigmoid and tanh functions are sometimes[84] avoided. ReLU function corrects this behaviour, making it a widely used function

[83] A complete explanation can be found in: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/

[84] 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)

(Agarap, 2018), performing 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 at the same time. The different tasks can help each other to learn mutually useful feature extractors, as long as the tasks do not compete for resources (e.g. network capacity) (Ruder, 2017).

4.5.1.3 Based on the training objective

Another way to make a model generalize 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 $R_{D_{test/train}}(f)$ of a model f reflects the learning state, during the training and/or testing phase, and in some cases, it can give some obvious assumptions about the data distribution. The training objective is to learn the model parameters that minimize the cumulated error values. The common form is:

$$\arg\min_{\theta} \frac{1}{|D_{train}|} \sum_{(x_i,y_i)\in D_{train}} R_{D_{train}}\big(f(x_i,y_i)\big) + \varphi(\dots)$$

where $\varphi(\dots)$ is a regularization term added to control the impact of the accumulated error.
Generalization 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 regularizing effect, thanks to an added term $\varphi(\dots)$. An example is Dice coefficient optimization (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.

4.5.1.4 Based on the optimization

While training a DL model, the optimization algorithm finds the values of the model's parameters θ (weights and bias) that minimize 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, regularization through optimization aims to find out the optimizer 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 optimization algorithms, in the context of deep neural networks. Each epoch consists of one forward pass and one backpropagation pass, over all of the provided training samples, in a full batch learning process. The true gradient ∇_θ is obtained by computing the gradient value of each training case independently, then summing together the resulting vectors, in order to update farther the model parameters. Hence, SGD is an iterative optimization 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[85] 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 optimization methods are derived from the SGD algorithm, to make the model learn more efficiently (fewer data and shorter training time):

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. Where the weights of a network are initialized and then adjusted repeatedly during the training of the network. Several initialization 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 initialization (Sampson, 1987)
    2) Orthogonal weight matrices (Vorontsov et al., 2017)
    3) Data-dependent weight initialization (Cachi et al., 2020)

    Another (complementary) option is *pre-training* of 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 perform a *fine-tuning* pass in the target context to make the model perform better on the actual objective starts. One important aspect of this approach is that pre-training a model on a different task of the same domain may lead to learning useful 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 rather to data-based method.

3) ***Early Stopping.*** This is a technique that aims to stop the training process at a good 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 35 (right), or even gets worse

[85] 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.

because of the start of noise learning (Raskutti et al., 2014). Also, it is a widely used regularizer in neural networks starting from the 1990s. As shown in Figure 35 (left), where the blue line shows the testing error and the green line shows the training error. If the model continues learning after the red dashed line, the testing error will increase while the training error will continue decreasing.

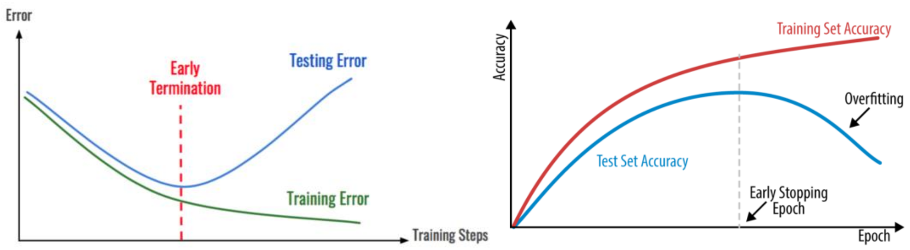


Figure 35. Illustration of the early stopping strategy based on error (left[86]) and accuracy (right[87]) evolution during training in test and training sets.

However, the best training step for stopping is not easy to define. If we stop learning too early, the model could underfit the data, and 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. To do so, we can track the accuracy on validation set instead of test set in order to determine when to stop training. The early stopping method has been used in several effective DL models and has shown its effectiveness to promote generalization (Caruana et al., 2000; Wu and Shapiro, 2006).

4) ***Dropout.*** 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, but also 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-time.

It is not clear which of the methods merely speeds up optimization and which actually help the generalization. One needs to perform several optimization tests and comparative analysis in order to find the best match between the model complexity and the optimization algorithm. An empirical study by (C. Zhang et al., 2021) compared different regularization techniques: data augmentation (aug), weight decay (wd), batch normalization (BN), and dropout. Figure 36 shows the performance evolution of a trained Inception[88] model, using different regularizers. Since the Inception architecture uses a lot of batch normalization 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 normalization layers removed. Figure 36a shows the training and testing accuracy on ImageNet. Figure 36b compares the learning curves of

[86]https://medium.com/analytics-vidhya/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting-dce6de4081c5
[87] https://datascience.stackexchange.com/questions/32306/in-which-epoch-should-i-stop-the-training-to-avoid-overfitting
[88] 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.

the two variants of Inception on CIFAR10, with all the explicit regularizers turned off. Each curve corresponds to one of the regularization methods, where the shaded areas are the cumulative best 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.

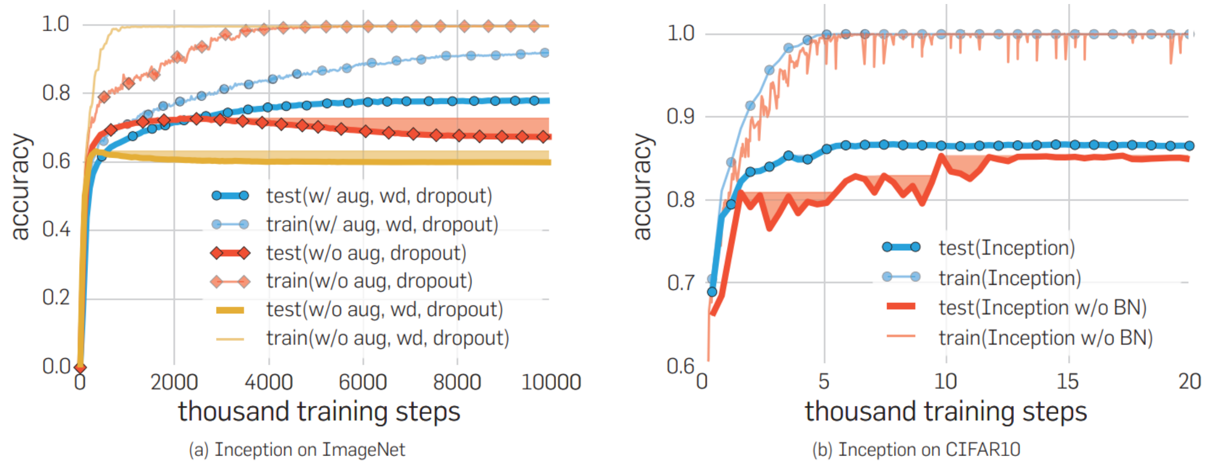


Figure 36. Effects of different regularizers on generalization performance, in terms of accuracy (C. Zhang et al., 2021).

This evaluation shows that a good match between different regularizers can lead to better performances. In this experiment, there are two main observations to retain:

(a) Early stopping could potentially improve generalization when other regularizers are absent.
(b) Early stopping is sometimes not necessarily helpful (e.g. CIFAR10), while batch normalization consistently stabilizes the training process and improves generalization.

4.5.1.5 Boost the generalization with regularization

All the regularization methods and generalization 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 generalization, as shown in Figure 37.

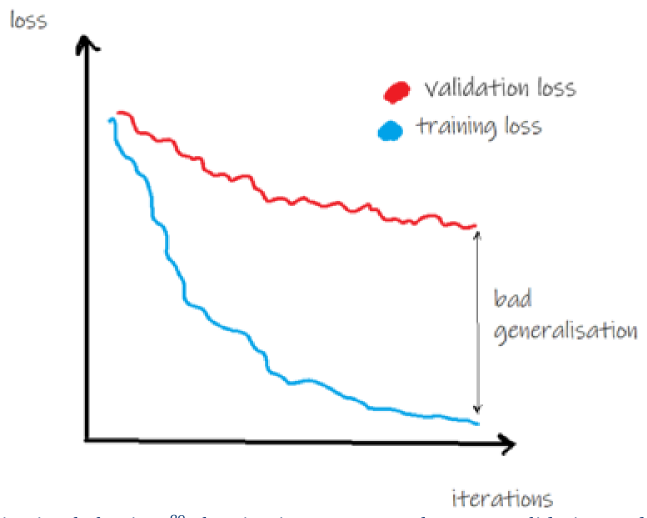


Figure 37. Illustration of a bad generalization behaviour[89] *showing important gap between validation and training loss errors.*

Regularization can be as simple as shrinking or penalizing large coefficients (cf. Section 4.5.1.3), where an added term $\varphi(\dots)$ is often used to calibrate weights and can be defined differently. Regularization can also impose a penalty on the model's complexity or smoothness (cf. Section 4.5.1), allowing for good generalization even when training on a finite data set or with an inadequate iteration. Recently, a comparative study (Tian and Zhang, 2022) of different existing regularization methods for ML/DL applications has discussed how to choose a regularization for a specific task. Such new regularization techniques can be constructed by extending and combining existing regularization techniques.

Even if the methods are different, the objective of regularization remains the same and is to improve the generalization ability of a ML/DL application (Müller, 2012). For instance, in the data-driven methods, the optimization of the data representation and the leveraging of latent content on the data, along with the identification of the right location and function for data transformation is the strength of the regularization by data processing. The advantage of these methods is that the model will master the data as well as the target application domain. However, to avoid the opposite effect of these methods on the generalization, we need to pay attention to data biases (e.g. ensure a balanced data distribution over different mini-batches while using batch normalization) which will be harmful to the data generalization. Model-driven approaches rely on building 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 important 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 optimization algorithms can also be leveraged to promote the models' generalization. In fact, these methods, such as early-stopping and weight-update functions, rely on the training process itself, rather than the model's architecture or its inputs. They enable to preserve the optimal state of the model that is more likely to generalize.

[89] Source: https://towardsdatascience.com/generalization-regularization-overfitting-bias-and-variance-in-machine-learning-aa942886b870

4.5.2 Penalty Methods

An overfitted model tends to memorize all the data features while training, even if some of them are noise. In order to limit these cases, two possible solutions can be adopted:

1) Select only the useful 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 preprocessing 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 important 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 useless. 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 common approach is to limit the effect of noise by minimizing the cost function of the model. To do this, a "*penalty term*", called regularizer, can be added to the cost function used during training, as shown in the formula of section 4.1.3. Hence, regularization 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 regularization 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 regularization term φ is used in general to express a given assumption about the elements of the data set and the target application domain. For example, one of the classical regularizers is weight decay (Goodfellow et al., 2016)

$$\varphi(\theta) = \lambda \frac{1}{2} \|\theta\|_2^2$$

where λ is a weighting term controlling the importance of the regularization over the consistency. θ are the weights of the model being trained.

Several other error regularizers 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

4.5.3 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 same good results. This regularization approach is inspired by the noise reduction to prevent overfitting. Known also as pruning (Ying, 2019), network reduction is proposed to reduce the heavy inference cost of deep models in a low-resources settings. Pruning is a theory used to reduce classification complexity by eliminating less meaningful, or irrelevant data, and finally to prevent overfitting and to improve the classification accuracy. During pruning, according to a certain criterion (rules), redundant weights (inter-neuron connections) are removed and important weights are kept to best preserve the accuracy of the model (Liu et al., 2018), a fine-tuning pass is then performed to optimize the resulting reduced network.

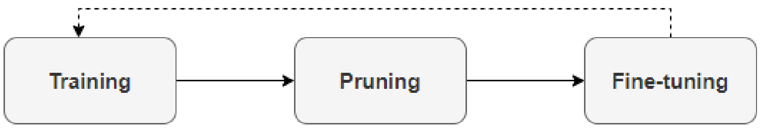


Figure 38. The three stages NN pruning pipeline.

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 rule to a model description, such as encoding length restriction;
- post-pruning splits the training set into two subsets: growing set and pruning set. Post-pruning 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. $\theta \leftarrow trainToConvergence(f_{\theta}(X))$
    3. $M \leftarrow 1^{|\theta|}$
    4. For $i \in \{0, \dots N\}$ do
           a. $M \leftarrow prune(M, score(\theta))$
           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 convergence. Afterwards, for each parameter or structural element in the network, a score is computed, and the network is pruned based on these scores. The process of pruning and fine-tuning is often iterated several times for gradually reducing the network's size without losing performance. This pipeline is shown in Figure 38:

1. **Pruning**: where the objective is to gradually reduce the number of trainable parameters in the network. 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). While other methods (You et al., 2019) are based on global scores. They aim at comparing scores of a group of parameters to other parts in 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 methods 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**[90]**.** Since the pruning reduces the accuracy of the network, this latter is trained further (fine-tuned) to recover. This step aims in general 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 trained weights from before pruning. Alternative proposals include rewinding the network to an earlier state (Frankle et al., 2019) or reinitializing the network entirely (Zhang et al., 2022).

3. **Truncation.** (Nevzorov et al., 2022) is another method for network reduction and which is similar to the classical dropout (cf. section 4.5.1.1). In this method some neurons are excluded during training, according to a position-based probability. By defining a training direction in the network, neurons that are positioned at the beginning of each layer tend to be better connected (have higher weights) so not excluded by the Truncation (Nevzorov et al., 2022), and 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 a network accuracy on the number of neurons in a layer has been implemented, after one cycle of training, while performing some regularization, without losing accuracy.

4.5.4 Data Expansion

4.5.4.1 Data quality and volume qualification

A crucial issue in machine learning projects is to determine how much training data is needed to achieve a specific performance goal (w.r.t. industrial requirements and theoretical evaluation metrics. Cf. Section 1.3), and how to qualify this data set (identify characteristics that describe the target application). The 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:

[90] http://d2l.ai/chapter_computer-vision/fine-tuning.html

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, (Cluzeau et al., 2020). In this latter, a set of classical measures along with discussions highlighting the required modifications and adaptations that is needed to bring to them, in order to cope with ML/DL development and release. These requirements are:
   1) *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 statistic errors (e.g. zero mean). In addition, the following errors need to be minimized:
      - *Capture error:* noisy, biased, or distorted collection of data, 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:* which is 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, "*misfielded*" values and incorrect references (Laranjeiro et al., 2015). Hence, one of the central preprocessing steps is improving accuracy of data by trying to predict and fill missing values in data sets (Ma et al., 2007).
   2) *Entity Resolution:* is about recognizing 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.
   3) *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.
   4) *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.
   5) *Timeliness:* Confidence that the data is applicable to the period of intended use.
   6) *Completeness:* 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 in (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 an improved marketability, but of socially acceptable, beneficial machine learning applications. Another study by Chen et al. (H. Chen et al., 2021) have 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* that 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 associate 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 lead to data noises, and incorrectly labelled data lead to label noises; and *variety* which. Hence, these qualities need to be assessed on the data preparation step since they affect a good or poor model performance (more details about the data quality assessment are provided in section 3).

2) ***Data volume***. As for data quality, the amount of data that should be used to train a 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 a 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 4.4.1) model complexity has an important impact, as well as the complexity (cf. Section 4.4.2) in the model's generalization estimation. Hence, we can choose one (or more) method from the existing two classes, with respect to the target application objectives:
   1) *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 of percent (100%) more rows than columns (have more examples than 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 the 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.
   2) *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 in (Balduzzi et al., 2021) to tie the empirical generalization 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.

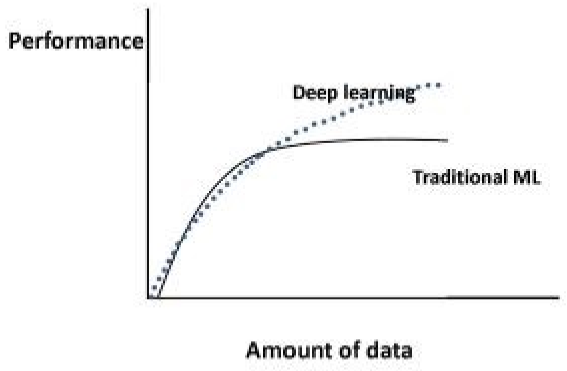


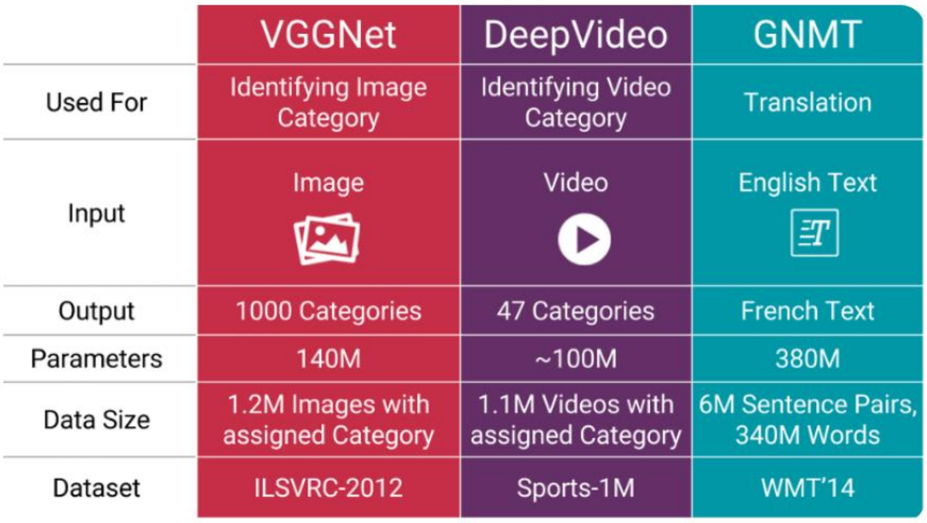
Figure 39. Difference between ML and DL w.r.t. the impact of the amount of data on the performance evolution during training (Zhu et al., 2016).

- *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 39 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).(Hassaballah and Hosny, 2019). Specifically, for traditional machine learning algorithms, performance grows according to a power law and then reaches and settle 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 certain accuracy level a priori. It is based on observations of the *learning curves* to show that the common behaviour is: the performance keeps increasing with data size according to a power law.

4.5.4.2 Importance of data

Data choice and preparation are two important steps in the development of DL applications. However, when the target application requires a complex model, there is a need for an important amount of data from the target domain. In particular deep learning applications require considerable amounts of data for training[91]. Figure 40 shows some examples of DL applications w.r.t. the amount of training data sizes used to train 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.

[91] https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/



	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 40. Three example applications show the complexity of the used DL model, the data type, and the sizes of the training sets[92].

In real-world applications, collecting such a volume of data is difficult and sometimes impossible. In some domains, collecting new data is either not feasible or requires comparably much 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 overothers, 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 some other labels in the same data set, which can lead to a data bias problem.

For most[93] of the aforementioned problems, data augmentation (expansion) is one of the common solutions to reach the required data amount for DL models 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 small changes to the original qualified data, or using deep learning models to generate new data points for the training (cf. section 3). Formally, given a data set $X = \{(x_i, y_i), i = 0, \dots, |X|\}$, a transformation function tf produces a new data example associated to every input sample x_i: $tf(x_i, \omega) = x_i^t$, 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. 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 augmented at training time so that there is no need to store the augmented data (Lemley et al., 2017). In offline augmentation, data is augmented in preprocessing phase and stored for later use (Lei et al., 2017).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 series of data transformations).

[92] https://www.datarobot.com/blog/introduction-to-dataset-augmentation-and-expansion/

[93] 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.

4.5.4.3 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 41. Where a series of transformation functions (TF) are defined and applied to different data entities, under a supervision of a human expert, then the qualified augmented data set is used for DL models training.

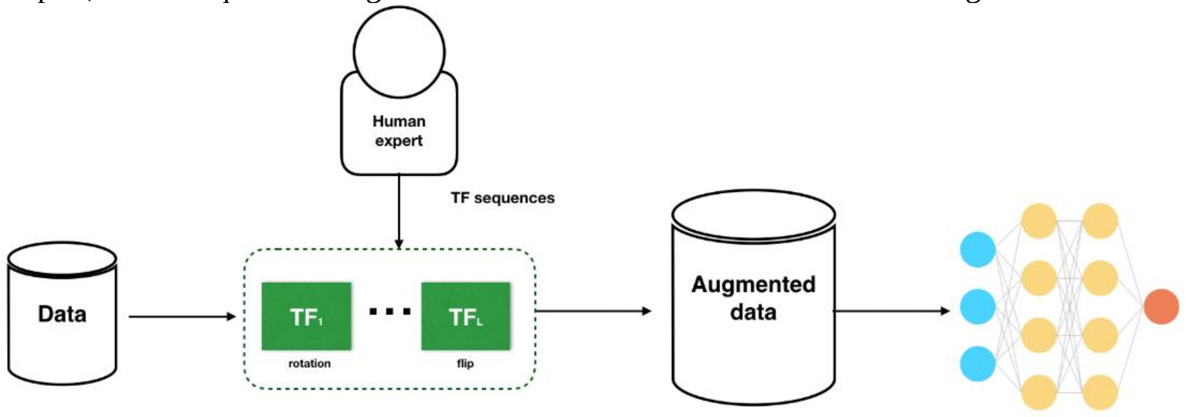


Figure 41. Description of the heuristic data augmentation pipeline[94] which applies a deterministic sequence of transformation functions tuned by human experts.

In image processing (Shorten and Khoshgoftaar, 2019), one of the most used heuristics is *geometric transformation* (Goodman, 2022), where the main 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 color augmentation which is based on the color space transformation. It is also known as photometric transformation, it can be made by transferring colors between images (Xiao et al., 2019) or applying color perturbations for 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 is widely used to produce new images by transferring the style of an 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 together such that the image G is transformed to look like the image C, in the style of the image S. Figure 42 shows an example of a style transfer result:

[94] https://salmenzouari.medium.com/automating-data-augmentation-f2bdf1f1c0da

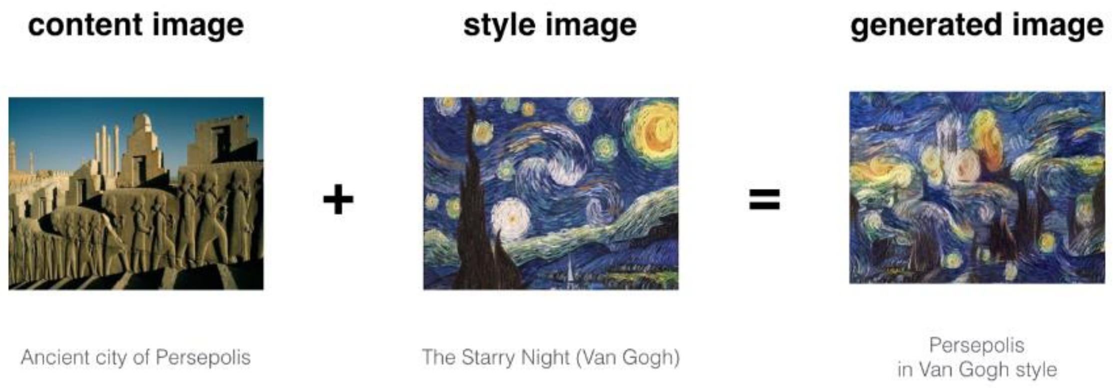

content image
style image
generated image
Ancient city of Persepolis
The Starry Night (Van Gogh)
Persepolis in Van Gogh style

Figure 42. Example of style transfer data augmentation in images processing[95].

Apart from computer vision, several methods for natural language processing (NLP) data augmentation 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 11; and 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 to 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.

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 11. Examples of Easy Data Augmentation techniques

The main limitation of the non-learnable data augmentation methods is that the human is 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

[95] Source: https://towardsdatascience.com/neural-style-transfer-on-real-time-video-with-full-implementable-code-ac2dbc0e9822

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 not available.

4.5.4.4 Learnable methods

Learnable data augmentation is promising, in that it enables to search for more powerful parameterizations and compositions of transformations. Perhaps the biggest difficulty with automating data augmentation is how to search over the space of transformations. This can be expensive due to the large number of transformation functions and associated parameters in the search space. The learnable data augmentation methods are based on advanced models that are 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 directly optimize for validation accuracy on the end model. 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 they could be 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 that are 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, (3) the discriminator that should distinguish the augmented images from real ones. Another case study of text data augmentation is training sentence classifier 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. Using the fine-tuned model and given a class label, new sentences for the class are generated. The sentence classifier has then shown better performances after training on the generated data.

4.6 Methods and tools for generalization evaluation

The objective is to quantify the generalization ability of an ML/DL system. In this section, we describe existing methods in three main approaches, which could be used together or separately, depending on the generalization evaluation objective.

4.6.1 Random labelling

This method aims to evaluate, a priori, the ability of the model to generalize (C. Zhang et al., 2021). The main idea is to create a copy of the training data where each label is replaced independently by a random label chosen from the set of valid labels (e.g. a dog picture labelled "dog" might thus become a dog picture labelled "airplane"). The objective of randomization is to break any relationship between the training instances, then run the learning algorithm both on the original true label data and on the randomized data with identical settings and model choice. By design, no generalization is possible on the randomized data. The model is fitted to random labels and compared to how it behaves on the natural data against the random data. The hypothesis is that if it turns out to be the same in both cases, it cannot even distinguish learning from natural data (where generalization is possible) from learning on randomized data (where no generalization is possible).

4.6.2 Data corruption

Another way to evaluate how a model could generalize better is to compare the learning process and performance evolution on original data, and on data with corrupted samples (added noise). In (C. Zhang et al., 2021) different corruption methods have been compared, as shown in Figure 43:

1) ***Partially corrupted labels:*** independently with probability *p*, the label of each image is corrupted as a uniform random class.
2) ***Shuffled pixels***: a random permutation of the pixels is chosen and then the same permutation is applied to all the images in both the training and testing sets. This function breaks the structure of the input original images.
3) ***Random pixels:*** a different random permutation is applied to each image independently.
4) ***Gaussian:*** A Gaussian distribution (with matching mean and variance to the original image data set) is used to generate random pixels for each image.

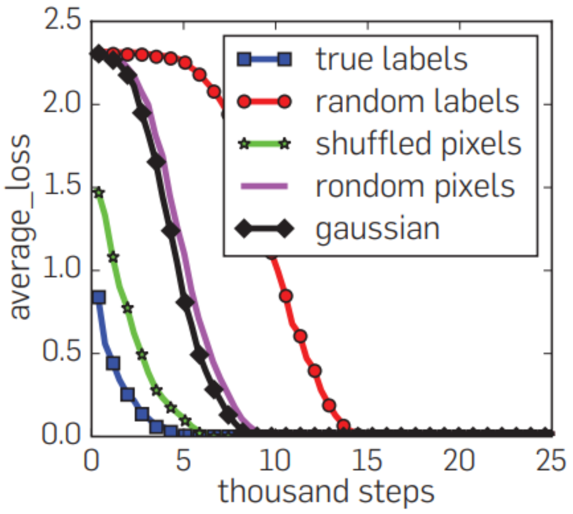


Figure 43. Comparison of the average loss evolution of the Inception model on the CIFAR10 data set under various settings[96].

[96] https://dl.acm.org/doi/fullHtml/10.1145/3446776

4.6.3 Evaluation approaches

4.6.3.1 Machine learning testing

The “massive” or “big bang” testing aims to run a set of scenarios to analyse the behaviour’s correctness of the whole system after putting all modules together (Soares et al., 2022), known also as “integrating testing” it enables to test the correct interaction between several modules integrated together in the same system, of which an ML/DL module can be part. In ML/DL, the testing helps spot problems in models that regular evaluation metrics might miss (Zhang et al., 2020). The aim of testing in ML/DL systems can target different elements such as:

- Testing properties, including correctness, robustness, and fairness of the ML module, using different tools such as DeepXplore (Pei et al., 2017) and Themis (Galhotra et al., 2017) ;
- Testing components where the data, the learning program (the core of the ML/DL module), and used frameworks, are tested using several ML development tools such as TensorFlow[97] and Scikit-learn[98] ;
- Software workflow and whole application scenarios (e.g., test generation and test evaluation).

A more complete ML testing taxonomy is reviewed in (Zhang et al., 2020). Based on the fact that testing is the common practice for software quality assurance, the same policy could be applied for ML/DL systems. Hence, while evaluation metrics are used to tell the performance of the model on test data sets, model testing focuses on checking the expected behaviour of the model and its corresponding module. This testing is needed since there could be unexpected events in production. Several forums of the ML community are still discussing these issues. However, testing on DNN-based software, such as DeepGini (Feng et al., 2020), is significantly different from conventional software. While conventional software depends on programmers to manually build up the business logic, DNNs are constructed based on a data-driven programming paradigm. Therefore, a lot of effort is required to obtain oracle information, which usually requires expensive human implication to label the testing data that will be used in the quality assurance process. To deal with these costing issues, several ML-testing methods can be used:

- ***Adversarial attacks:*** it aims to confuse the model to make incorrect predictions (Das et al., 2020). Rather than letting this confusion happen in a production environment, a model can be tested with adversarial examples to increase its robustness prior to deployment.
- ***Data integrity and bias***: it refers to the data quality (cf. section 4.5.4.2) in terms of balanced samples. Note that data collected from different sources might reflect human bias that can be modelled during training. During the evaluation, bias can be missed because it focuses mostly on performance and not the behaviour of the model given the role of the data in this case. Hence, several methods (Tomalin et al., 2021) to reduce the bias on data can be used.
- ***Spot failure modes:*** Failure modes can occur in the production of ML/DL models. These can be due to performance bias failures, robustness failures or model input/output failures. The latter can be even intentional, due to a system intruder, or unintentional due to an unsafe model output (incorrect or with a very low confidence). A more detailed failure modes analysis, related to ML systems, can be found in (Kumar et al., 2019). Whatever it is their source, these failures should be taken into account, to ensure a more complete testing/evaluating benchmark. However, some of them can be missed by evaluation, for instance, a trained model with an accuracy of 90% means

[97] https://www.tensorflow.org/
[98] https://scikit-learn.org/stable/

that the model is struggling to generalize with the 10% of the data. In this case, a verification on the remaining 10% of data needs to be performed to figure out the elements that the model is not able to learn with the used architecture/configuration, and hence try to integrate them into the objective function, the application description, or the evaluation process (metrics and benchmark).

4.6.3.2 Discussion

As described by the different evaluation metrics, the confidence in the performance of a 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 specialized 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, a combination of several metrics, such as MAE, MSE, RMSE and R-squared coefficient (cf. section 1.3) is required to validate the model's effectiveness.
Furthermore, adopting a set of metrics does not negate the occurrence of certain common issues, namely overfitting and bias. As such, an analysis that uses ML should also consider additional techniques, such as the use of independent test data sets and threshold values (Gossmann et al., 2018; 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 5. 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 types of analyses provide indicators on the generalization ability of the model 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, 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 include any functions of model responses, including those that may vary over a spatio-temporal domain, objective functions such as a production or cost function in cost-benefit analysis, or an error function in model calibration.

4.7 Limitations of existing methods and discussion

In this chapter, we have reviewed several existing methods for generalization evaluation, mentioning the strengths and weaknesses of each of them. In this part, we analyse the main limitations of the existing methods and pipelines, for ML/DL models analysis and generalization evaluation benchmarks. This analysis will provide the main resources to develop a rather compact methodology in terms of analysis and leveraging of the different elements (data preparation, training pipeline, evaluation, and analysis) allowing a rigorous workflow for the development of a certifiable trustworthy AI.

4.7.1 Misunderstanding of the generalization bounds

To generalize well beyond the training set, ML/DL models undergo analysis based on many hypotheses concerning what the triplet of a *model*, *optimization algorithm*, and *data properties* could generate. The good combination of the elements of this triplet could help provide an AI solution that generalizes better. To do so, hypotheses about the model's complexity, effective capacity and data properties are made. This yields a set of theoretical and empirical analysis of the generalization ability. However, despite the prominent role of complexity measures in studying generalization, the empirical evaluation of the existing measures (cf. Sections 4.4.2 and 4.6) is usually limited to a few models, often on toy problems. A measure can only be considered reliable as a predictor of generalization gap if it is tested extensively on many models at a realistic scale (Jiang et al., 2020). In this study, the findings are surprising. For instance, it has been verified that one can easily capture false correlations, between generalization and complexity and that do not reflect more causal insights about generalization, using some complexity measures. Hence, an empirical verification on the use-case needs to be performed. Another limitation is that many norm-based measures (Bartlett et al., 2017) could negatively correlate with generalization specifically when the optimization procedure injects some stochasticity.
An experimental study (C. Zhang et al., 2021) has shown that conventional generalization bounds, based on uniform convergence or uniform stability are inadequate for over-parameterized deep neural networks. Hence, several empirical studies have been performed to show how generalization 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 generalize 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 most correct generalization performance). As an explicit demonstration of the looseness of the existing bounds, several previous studies (Hsu et al., 2021; Jiang et al., 2020) have shown that calculated bounds for a feed-forward NN architecture achieving a small test errors represent an important gap in different observations.
Despite these limitations, the extensive analysis by (Jiang et al., 2020) have shown that measures related to the optimization procedures such as the gradient noise and the speed of the optimization can be predictive of generalization. Besides, sharpness-based measures like PAC-Bayesian bounds (McAllester, 2003) perform the best overall and seem to be promising candidates for further research.

4.7.2 Practical issues preventing generalization

In the previous sections, we have reviewed the importance of the generalizability 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' generalization and that could prevent them from being released. There are two major sources of performance limitations: (1) overfitting that results from too much learning of a model or learning a complex model with insufficient data. The model will then perform well on the training data set and poorly on new data; (2) underfitting, that results from too little learning or training a model with too small capacity to learn the task. The model will then fail to sufficiently learn how to solve the problem and perform poorly on a training data set, and does not perform well on test samples. In addition to the overfitting and underfitting issues, several studies have investigated the main issues that can lead ML/DL models to under/over-fit the data. In the following, we review the common mistakes and pitfalls that lead to a lack of performance in ML/DL applications. This list is not exhaustive:

- ***Inappropriate training objective.*** The loss function plays an essential role in the training of ML/DL model. It essentially 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*[99]*.*** In statistics, an *outlier* is an observation point that is distant from other observations. In training data, the presence of several special cases in the data set should be handled carefully. One outlier could skew the distribution of results if it is ignored during training. In the observation of 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).
- ***Regularization without standardization.*** Standardization[100] is a data preprocessing technique. Regularization is used to improve model performance and aims to tune the function by adding additional penalty term in the error function. However, when the data is not standardized or normalized, the regularization alone could not be enough to boost the model ability to generalize. To better help the regularization, it is needed to rescale the features using standardization and/or normalization, which will, together with regularization, make the parameters in learning algorithms more likely to converge to smaller values, resulting in better performances (Dauphin and Cubuk, 2020).
- ***Inappropriate data representation.*** In ML/DL, models are built by observing and interpreting the data. 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.
- ***Inappropriate data in training and/or test.*** As discussed in this chapter, the quality of the data used in developing any ML/DL application is very important to produce performant models (Gupta et al., 2021). Nevertheless, the following(Gupta et al., 2021). Nevertheless, these issues can lead to a poor generalization 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 that contains very few actual values, and is instead composed of mostly zeros, undefined, or missing values.

[99] Another important 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 (eg. 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 with two or more parameters working together and make a significant impact on the behavior of the system. More analysis about the corner and edge cases is provided in the chapter related to Task 3 of the MLEAP project.

[100] Standardization (Ali et al., 2014) typically means rescaling data to have a mean of 0 and a standard deviation of 1 (unit variance). Feature scaling is one of the most important data preprocessing step in machine learning. Algorithms that compute the distance between the features are biased towards numerically larger values if the data is not scaled.

    - Inadequate data. Data that is either incomplete or biased.

  To cope with data completeness, representativeness and volume adequacy, the reader can refer to chapter 3 for a more detailed analysis of the state-of-the-art, relating to 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 ability to generalize is highly correlated with its complexity as well as data qualification. More complex models will, in general, perform better on large training data than simpler ones. These models have more parameters that can be adjusted during training to get a good fit for the desired result. Therefore, their error rate when measured on the training data set will usually be lower. 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). If these random effects are not present in evaluation and test data sets, then the model will perform poorly. To figure out if this is the case, the model's error rate on a validation data set is a best indicator:
    - A high error rate on both training and validation data indicates that the model may be too simple to faithfully capture any relationships present in the data. In this case, the model seems to have 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 regularize complexity measure (Jiang et al., 2020) by adding it a regularizer and directly optimizing it, but this could fail due to: (1) The complexity measure that could change the loss landscape in non-trivial ways and make the optimization more difficult; (2) The existence of implicit regularization of the optimization algorithm. This makes it hard to run a controlled experiment since one cannot simply turn off the implicit regularization. The consequences should not be ignored, in (1), if the optimization fails to optimize the measure, no conclusion can be made about the causality; in (2), if optimizing a measure does not improve generalization it could be simply due to the fact that it is regularizing the model in the same way as the optimization is regularizing it implicitly.

    - ***Inappropriate evaluation metrics***. Evaluation metrics are used to assess the model's evolution during training and in the validation set as well. Hence, the metrics to be used must correspond to the objective of the application (classification or regression). In addition, the mean performance by the evaluation needs to be handled carefully during the metrics selection. For instance, in the case of recommending safe city locations to tourists, true positives should be prioritized. In this case, 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 1.3) to ensure a more rigorous evaluation and verification.
    - ***Bad global minima***[101]**.** Picking the bad global minima can lead to bad generalization behaviour, especially for models trained with SGD (S. Liu et al., 2020). To avoid the wrong global minima, regularization plays a central role, where complex models do not simply "get lucky" with the training data, but the regularization makes simple models fit the data as well as the global optima, and it also clears the way to make them discoverable by local methods, such as SGD**.**

[101] 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.

- ***Misuse of the model-driven regularizers***. The misunderstanding of the assumptions about the input-output mappings that could be applicable only to the training data set as a limited sample, or mismatching different tasks in the multi-task learning-based methods that could lead to a competition-based learning process.
- ***Bad match between the model complexity and the optimization algorithm***. As for the evaluation metrics, not all the optimizers can lead to better results. For example, early stopping is useless when the error does not stabilize, and the wrong data augmentation function will harm the model's performance instead of providing a larger 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 neither trainable nor non-trainable augmentation methods.
- ***The violation of data (training-validation) independence assumption.*** Inaccurate data oversampling[102], 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 measured not representative compared to the full input domain defined by the ODD.
- ***Adapting a large 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[103]. This latter considers methods of constructing approximations that converge to the desired function with an increasing number of 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 both weak and strong convergences of the Hilbert space (Soenjaya, 2013).

4.7.3 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. An analysis about how the system would interact with the user, through additional testing of the corresponding module can be made, and an overall idea is then drawn about the performances after the release.

In the industry, some AI technologies can be meant to replace (or highly assist) a human being in several tasks and decision takings. Hence, these technologies are supposed to have a behaviour as close as possible to that of an expert of the concerned industrial domain, and this, in terms of decision-making and choice of the different actions necessary to perform a task. This is why the evaluation phase is so

[102] It aims to duplicate some examples from the minority class (data samples) in the original dataset, to overcome the data imbalance problem.

[103] 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.

crucial in the certification[104] 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 objects recognition (Ho-Phuoc, 2018) have shown that the best DL model trained for this task still cannot recognize 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 properly represent actual observations – despite having 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. It is important that the model evaluation involves appropriate performance indicators. More precisely, the metrics and performance thresholds that are chosen need to reflect adequately human judgement. Standard metrics, like Mean Absolute Error (MAE) and accuracy, are sometimes not enough to reflect the model's ability to generalize, or the domain specific (business) key performance indicators (KPIs). For example[105], the enhanced use of email cannot be enough to measure the effectiveness of spam filter 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[106] 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 key question that needs to be answered is: *what constitutes a good AI/ML model?* A proper answer to this question depends on various factors, including the fact that a good model optimally performs and describes the phenomenon being addressed. One possible option (Naser and Alavi, 2021) is the use of 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 individual metrics. Furthermore, several key engineering safety principles need to be handled, along with ML safety issues due to dependability[107] limitations (Mohseni et al., 2021) such as generalization errors. Several research activities are launched in ML safety investigation and implementation, mainly with the perspective of enhancing performance

[104] 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.

[105] https://vitalflux.com/different-success-metrics-for-ai-ml-initiatives/

[106] Ability of the model to bring an answer and which is relevant to the input question.

[107] 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.

and robustness, by adapting robust network architectures (Djolonga et al., 2021) or robust training process (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 really important to promote AI adoption in the industry (Gordon et al., 2021). The relation between ML models' performance and user-facing performance is indicative of a larger 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, where most of them are based on generalization 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). In order 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% of accuracy is acceptable? And how the remaining 15% of uncertainty would affect the user experience and tolerance?). The proposed tool enables to systematically select an objective function to optimize during classifier evaluation, in addition to insights into how to design user feedback in interactive classification systems.

In the same line of the technical performances evaluation of ML/DL models, the safety of the systems integrating these models needs to be evaluated, in most of critical applications of AI, including the wide area of aeronautics. In this case, AI systems are submitted to a consistent evaluation and validation pipeline (Brunton et al., 2021). Recently, the outcomes (Alecu et al., 2022) of one of the most promising projects in aeronautics, called DEEL[108] (DEpendable Explainable Learning), have provided an overview of several analysis required for the assessment of ML models' capacity to comply with existing safety standards. Such that 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, and so on. Two practical examples from the railway and automotive industries have been explored, showing that ML performances are currently far from those required by safety objectives. Several techniques aimed at 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.

4.7.4 ML/DL testing limitation and challenges

As described in Section 4.6.3.1, a massive testing[109] approach can be adopted to make sure that the trained model and/or the module that includes a trained AI model, has 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 as much as possible system failures related to the implemented software. Hence, a continuous and resource-consuming testing is required;

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

[109] 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 testing schemes should include “*performance evaluation*” indicators and scenarios to test the correct behaviour of the ML/DL model and the system module that embed it and/or modules that are related to it as well;

Testing in software development identifies explicitly which parts of the source code fails and it provides a relatively coherent coverage measure (e.g., lines of code covered). It is not that simple when it comes to the ML/DL model, where other criterion 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 important 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) *Identify defects and flaws* during development and in 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 outcomes, due the use of stochastic training algorithms (e.g. SGD). This latter can provide two different models for two training times at the same data set, due random local/global minima and random initialization of a same network connections. Hence, a small difference on the predicted scores can have an important impact on the results (e.g. predicted probability for a given class), which prevent models from reproducing the same results (or output scores) after (re)training;
3) The unclear/insufficient tests coverage, due to 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 difficult 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 robustness (cf. Section 4.2.4) and generalizability (cf. Section 4.2.5) of the trained model. Unless being combined with a well-defined evaluation pipeline, we believe that the ML/DL testing is not enough to build a safe AI module properly.

4.8 Towards application independent ML development pipeline to promote generalizability

As mentioned before, the main objective of this section is to analyse the ML/DL state-of-the-art concerning model development and training to promote the generalization after testing. So far, we analysed several methods and common practices allowing, on one hand, optimal training and on the other hand a consistent evaluation, by emphasizing the expectations of the target application. However, in the previous section we identified several limitations of existing approaches preventing the produced models from generalizing their performance to unseen data sets, and farther from being released. In order to come with recommendations and solutions to cope with those limitations, we revisit the pipeline for ML/DL models development, training, evaluation and enhancement after implementation in the target system. In the following, we will first explain the main research questions addressed so far and how we aim to deal with each issue, providing the identified ways that can lead to answers in industry. Further, we propose a use-case independent approach, including selected methods, for the generalization assessment and evaluation.

4.8.1 Research questions and roadmap

How to 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 memorized too. 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 is incapable of capturing the variability of the data, which is the result of using a model with a low capacity to describe a given problem. The common point is that both overfitting and underfitting lead to deterioration of the generalization 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 generalize better. Although the original model may be too large to generalize well, regularization 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 generalize better, independently of the generalization type (domain-based, multi-tasking, OOD-based):

- The regularization 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 optimization algorithm that aims to converge to the best global minima. Hence specific initialization and warm-up or pre-training options could be adopted, or by using specific functions in the weight update (e.g. weight sharing) 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 involved in the performance boosting of an ML/DL model that is not able to generalize to unseen instances during training.

How to bridge the gap between experimentation and industrial expectations?

To know if the model generalizes effectively, several performance evaluation metrics have been proposed for both regression and classification applications. A lot of work is done so far 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. Such that 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 find out if the model is performant enough. Industrial development pipeline focuses on a set of performance and robustness indicators (KPI) that is 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 is not adapted to the target domain. As we can understand, an infinite number of 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 generalizability weak and hence an adverse (limited) user experience.

To bridge the gap between the empirical and industrial processes, we need to leverage the evaluation metrics in the way that they reflect the targeted performances, and integrate the KPIs in the training objectives and the evaluation pipeline as well:

- To evaluate the model results in a more rigorous way, one interesting conclusion by Caruana et al. (Caruana and Niculescu-Mizil, 2004) is that *"Different performance metrics yield different tradeoffs that are appropriate in different settings. No one metric does it all, and the metric optimized to or used for model selection does matter."* With this in mind, recent works in ML/DL models evaluation recommend the utilization 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 need to be considered seriously to prevent adverse user experiences after the ML/DL application release.

How to cope with common data processing and evaluation mistakes?
In the previous sections, we briefly reviewed the different pitfalls that could lead to weak model robustness. We believe that the evaluation process of ML/DL applications should include the evaluation and adaptation of traditional mismanagement and manipulations undertaken in the model building pipeline, data analysis and/or pre-processing, and the evaluation process as well, to have a better return on investment on the results of the built model, in terms of generalization for the target application. Besides, despite being insufficient to address all the potential ML/DL model errors and failures, ML validation approaches provide important elements to construct model evaluation and testing scenarios, such as the data qualification and identification of the expected behaviour (i.e. *What kind of outputs is expected for every kind of input?*).

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, benefits from some software engineering best practices, such as the scenarios building for 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:

- To construct a more rigorous pipeline, we provide a generic framework for data preparation, model training and evaluation pipelines including most of the best practices we identified during the state-of-the-art analysis, in addition to suggestions on how to deal with common limitations in existing processes.

4.8.2 Generalization Guarantees Selection

The objective is to provide assurance that the trained model will perform well on unseen data. The generalization performance assessment aims to:

- Ensure a minimum gap between the generalization error and the test error;
- Provide generalization guarantees on the full domain of application;

It is a concern that has to be taken into account during the full development of the machine learning solution and each step of the model development will have to demonstrate the level of performance according to the different criteria of the target application. 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 that gives the most optimal approximation of the expected value is considered consistent. However, some issues should be handled carefully, including:

- The training data that may be noisy so that there is no consistent hypothesis at all.
- The real target function that may be outside the hypothesis space and has to be approximated.
- A rote[110] 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, generalization 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;
- The training process and the loss function selection;
- The trained model robustness and stability that indicate how the generalization can be achieved.

Hence, in order to choose the suitable bounds, the target data and algorithm need to be considered.

4.8.2.1 Data impact

A strong assumption made at the beginning is that the data sets available are issued from the same distribution than the real world. Section 3 details elements that enable to provide recommendations that help to verify that the operational domain concerned by the model is properly covered by the data sets. Besides, the volume of data needed to properly train the model is also key. *A priori* approach can help estimate this volume according to the complexity of the task to be performed (cf. section 4.5.4.1), 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.

4.8.2.2 Model impact

At this stage, we aim to leverage some functions to help select potential models for the target task learning, having in mind its ability to generalize. Theoretical generalization bounds are great support to identify the right hypothesis to be selected. A summary table can be found in (X. Li et al., 2018).
Here, the adequacy between the targeted function and the complexity of the algorithm is an important step and will assess the risk for the model to overfit or underfit. It is in this step that the hyperparameters of the models are defined using the validation data set for example using bias-variance tradeoff or cross validation techniques. Whatever the adapted method is, the objective is the same and should ensure that:

[110] 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.

- The model will have high stability regarding the data selected for the training phase (cf. section 5.2);
- 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 generalization error;
- The metrics are properly identified to measure the performance of the model;
- The loss function is properly selected and can help to minimize the cost;
- The training strategy is identified thanks to generalization bounds minimization.

In terms of theoretical evaluation of generalization, we have previously reviewed different classes of methods to correctly assess model's generalization error (cf. Section 4.4.2). 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) is a measure of the capacity/complexity of a learnable function set. To make sure the chosen method can correctly reflect the model performance, a common practice is to adapt the one that is recommended to the model's architecture. Bounds based on uniform convergence provides guarantees that holds for any hypothesis and bounds (with probability $1 - \delta$) the true 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 with respect to different datasets. The main difference with the bounds based on uniform convergence is the incorporation of regularization term and removal of the complexity argument; it can be used when hypothesis classes are difficult to analyse with classic complexity argument (such as support vector machines where VC Dimension is infinite). The robustness-based generalization 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 larger class of regularizers than stability and its geometric interpretation makes adaptation to non-standard settings possible such as non-i.i.d. data. For instance, Chris Rohlfs (Rohlfs, 2022) proposed to categorize generalization under several levels of abstraction, including sample generalization where test cases are drawn from the same population as training data set, distribution generalization where test cases are drawn from new populations, and domain generalization where the input-output relationship has changed. Another criterion is confidence dimension (R. Wang et al., 2022) which is used to measure deep model's generalization ability and to theoretically calculate the upper bound of generalization based on the conventional VC-dimension and Hoeffding's inequality.

As a result, in table 4, we show a classification of commonly used bounds that have been selected, w.r.t. several model architectures, based on their applicability analysis (cf. sections 4.8.2.1 and 4.8.2.2), general formulation and genericity.

We plan to follow and assess the selected bounds (difficulty to calculate, applicability limitations and assumptions validation…) for the subsequent evaluation steps of this project through use cases.

Algo.	Ref.	Bound
CNN	(Lin and Zhang, 2019)	$R_{\mathcal{D}}(F_C) \le \hat{R}_{S,l_\eta}(F_C) + \mathcal{O}\left(\left(\frac{\|X\|_F \mathcal{R}_C}{\eta}\right)^{\frac{1}{4}} n^{-\frac{5}{8}} + \sqrt{\frac{\ln(1/\delta)}{n}}\right)$

Algo.	Ref.	Bound
RNN	(Chen et al., 2019)	$R(f_t) \leq \hat{R}(f_t) + \tilde{O}\left(\frac{L \times Complexity}{\sqrt{m}} + B\sqrt{\frac{\log(1/\delta)}{m}}\right)$
NN for classification	(P. Jin et al., 2020)	$\mathcal{E}(f) \leq \frac{\sqrt{d}.(1-\rho_\tau)}{\min(\delta_0, \kappa\delta_\tau)} = \alpha(\tau).CC(\tau)$
NN	(Alquier, 2021)	Catoni's bound (PAC Bayes) $\mathbb{P}_S\left(\forall\rho \in \mathcal{P}(\Theta), \mathbb{E}_{\theta\sim\rho}[R(\theta)] \leq \mathbb{E}_{\theta\sim\rho}[r(\theta)] + \frac{\lambda C^2}{8n} + \frac{KL(\rho\|\pi) + log\frac{1}{\epsilon}}{\lambda}\right) \geq 1-\epsilon$
	(Alquier, 2021) (McAllester, 1998)	Mc Allester's bound $\mathbb{P}_S\left(\forall\rho \in \mathcal{P}(\Theta), \mathbb{E}_{\theta\sim\rho}[R(\theta)] \leq \mathbb{E}_{\theta\sim\rho}[r(\theta)] + \sqrt{\frac{KL(\rho\|\pi) + \log\frac{1}{\epsilon} + \frac{5}{2}\log(n) + 8}{2n-1}}\right) \geq 1-\epsilon$
	(Alquier, 2021) (Seeger, 2002)	Seeger's bound $\mathbb{P}_S\left[\forall\rho \in \mathcal{P}(\Theta), \mathbb{E}_{\theta\sim\rho}[R(\theta)] \leq kl^{-1}\left(\mathbb{E}_{\theta\sim\rho}[r(\theta)]\left\|\frac{KL(\rho\|\pi) + \log\frac{2\sqrt{n}}{\epsilon}}{n}\right.\right)\right] \geq 1-\epsilon$
	(Alquier, 2021) (Tolstikhin and Seldin, 2013)	Tolstikhin and Seldin's bound $\mathbb{P}_S\left[\forall\rho \in \mathcal{P}(\Theta), \mathbb{E}_{\theta\sim\rho}[R(\theta)] \leq \mathbb{E}_{\theta\sim\rho}[r(\theta)] + \sqrt{2\mathbb{E}_{\theta\sim\rho}[r(\theta)]\frac{KL(\rho\|\pi) + \log\frac{2\sqrt{n}}{\epsilon}}{2n}} + 2\frac{KL(\rho\|\pi) + \log\frac{2\sqrt{n}}{\epsilon}}{2n}\right] \geq 1-\epsilon$
Fully connected NN	(Arora et al., 2018)	$\hat{L}_\gamma(f_A) + \widetilde{O}\left(\sqrt{\frac{c^2 d^2 \max_{x\in S}\|f_A(x)\|_2^2 \sum_{i=1}^{d}\frac{1}{\mu_i^2 \mu_{i\rightarrow}^2}}{\gamma^2 m}}\right)$
Two class classifier	(Anthony, 2004)	$er_P(f) < er_S(f) + \sqrt{\frac{8}{m}\left((n+k-1)\ln\left(\frac{2emk}{n+k-1}\right) + \ln\left(\frac{4}{\delta}\right)\right)}$
Supervised learning	(Neu and Lugosi, 2022)	$\lvert\mathbb{E}[\text{gen}(W_n, S_n)]\rvert \leq \sqrt{\frac{4H(P_n)\mathbb{E}\left[\left\|\bar{l}(.,Z)\right\|_*^2\right]}{\alpha n}}$
SVM (Pac Bayes)	(Tang et al., 2012)	$\frac{1}{n}\sum_{i=1}^{n} l\big(\rho_h w(x_i, y_i)\big) + 3B_l\sqrt{\frac{\log\frac{2}{\delta}}{2n}}$ $+ \frac{2L\Lambda}{n}\sqrt{\sum_{i=1}^{n} k(x_i, x_i)} \times \begin{cases}\sqrt{e}(4\log c)^{1+\frac{1}{2\log c}}, if\ p^* \geq 2\log c \\ (2p^*)^{1+\frac{1}{p^*}} c^{\frac{1}{p^*}}, otherwise\end{cases}$
γ-uniformly stable learning algorithm	(Feldman and Vondrak, 2018)	$\underset{\bar{s}\sim\mathcal{P}^n}{\mathrm{E}}\left[\left(\mathcal{E}_{\bar{s}}^{\leftarrow\mathcal{P}}[L]\right)^2\right] \leq \gamma^2 + \frac{1}{n}$

Algo.	Ref.	Bound
DNN	(Arora et al., 2018)	$\tilde{O}\left(\sqrt{\frac{hd^2 \max_{x\in S}\|x\| \prod_{i=1}^{d}\|A^i\|_2^2 \sum_{i=1}^{d}\frac{\|A^i\|_F^2}{\|A^i\|_2^2}}{\gamma^2 m}}\right)$
	(Hardt et al., 2016)	$\mathbb{E}\left[R[\overline{w}_T]\right] \leq \mathbb{E}\left[R_S[w_*^S]\right] + \frac{DL}{\sqrt{n}}\sqrt{\frac{n+2T}{T}}$
	(Lei et al., 2022)	$R_D\left(A(S)\right) \leq \Omega + \sqrt{4\Omega\Delta} + 8\Delta + \epsilon,$ where $\Omega = \epsilon + \sqrt{\frac{1}{2(1-\eta)m}\log\frac{1}{\delta}},$ $\Delta = \eta \log\frac{e}{\eta} + \frac{1}{m}\log\frac{2}{\delta}$
CNN	(Arora et al., 2018)	$L_0(f_{\tilde{A}}) \leq \hat{L}_\gamma(f_A) + \widetilde{O}\left(\sqrt{\frac{c^2 d^2 \max_{x\in S}\|f_A(x)\|_2^2 \sum_{i=1}^{d}\frac{\beta^2(\lceil\kappa_i/s_i\rceil)^2}{\mu_i^2\mu_{i\rightarrow}^2}}{\gamma^2 m}}\right)$
penalty linear regression models	(Montiel Olea et al., 2022)	$\overline{\mathcal{W}}_r(\mathbb{P}_n, \mathbb{P})^r \leq \frac{C}{\sqrt{n}},$ $C \coloneqq \left(180\sqrt{d+2} + \sqrt{2\log\left(\frac{1}{\alpha}\right)}\right)\mathrm{diam}\left(\sup(\mathbb{P})\right)^r$

Table 4: *Theoretical generalization bounds*

4.8.2.3 Generalization evaluation

The generalization error can be expressed as follow:

$$GE\left(f_{D_{train}}\right) = \int_{x,y} l\left[f_{D_{train}}(x), y\right] p(x,y)\, dx\, dy \qquad (x,y) \in (X \times Y)$$

Where $f_{D_{train}}(x)$ is the predicted output provided by the trained model and $p(x,y)$ the distribution on the domain (which is not fully known) defined by the ODD.
To approximate this error, we introduce the test error defined by:

$$R_{D_{test}}\left(f_{D_{train}}\right) = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} l\left[f_{D_{train}}(x_i), y_i\right]$$

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[111].
We can apply this approach for the different metrics which are relevant versus the targeted function with the objective 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:

$$P(m_i > perfo_i/x) > Req\ ,\forall\ x\ \in X$$

Nevertheless, it is expected to provide guarantees that the model is properly generalizing locally, meaning that the probability to obtain a correct result is not impacted by a special area of the full domain (for example while being close to a decision boundary or in a corner case). This aspect is developed in Chapter 5 where robustness and stability are analysed in depth. We can note the data representativeness and completeness are helpful in the identification of the area of interest for this analysis.

Finally, the assessment of the generalization ability of a trained model has to be done locally and globally regarding the parameters used to define the ODD. The global performance of a model is not a guarantee that the model is performing at an acceptable level in all conditions, especially for the one having a low probability of occurrence. This is further developed in section 4.8.4.2.

Figure 44, is an illustration machine learning pipeline and some activities useful for the generalization concerns

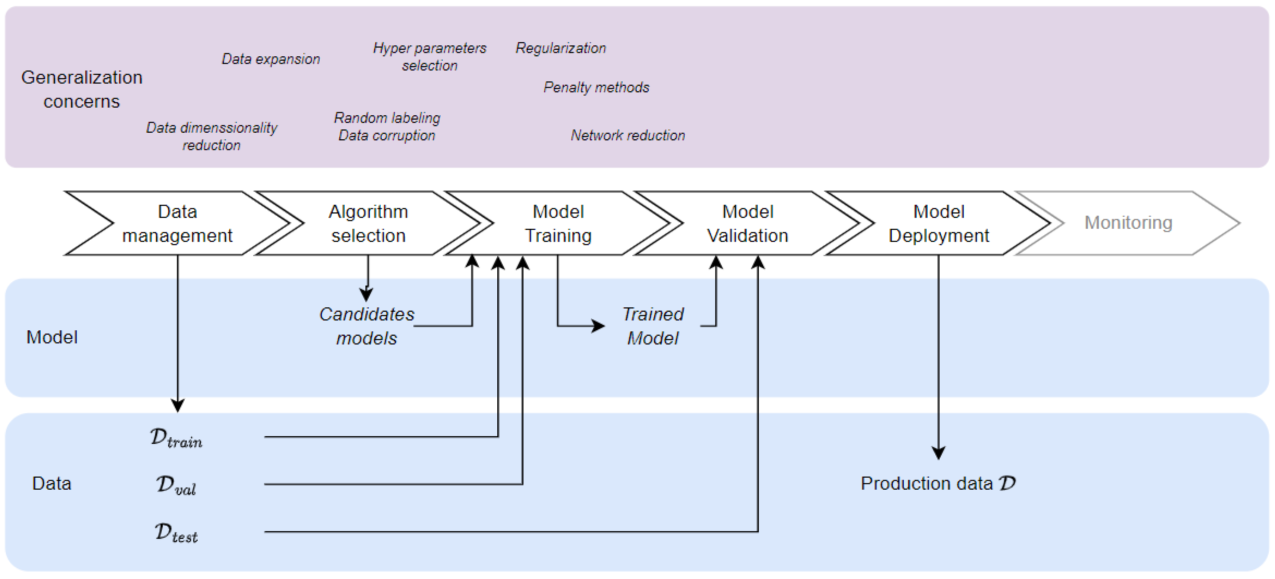


Figure 44. Illustration of the machine learning model development[112]

[111] 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 analyzing several methods to detect out of distribution samples
[112] Based on the ML pipeline in a nutshell: https://www.altexsoft.com/blog/machine-learning-metrics/

4.8.3 Methods Applicability Analysis

Generalization guarantees aim to provide evidences that the trained model will provide correct outputs (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 real target domain, and adapted to the expectation purpose (see Section 3);
- Ensure that the right adequacy between the hypothesis space selected and the targeted function;
- Ensure that the trained model is correctly optimized;
- 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 generalization performance of the trained model using statistical theory results such as Chernoff Bound to estimate the Generalization 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 generalization error and the test error is at most* ϵ.

  As this bound is growing exponentially with the tightness of the difference and the guarantee[113] *around*, we will use bounds identified in the literature and applicable to the use cases;

- Dive into the local generalizability of the trained model using for example data set margin calculations to assess the thickness of decision boundaries and area to focus on the local generalization performance assessment. This analysis will potentially provide guarantees depending on the input (using what is presented in Chapter 3 and 5);
- Investigate information bottleneck framework and multi-view to bound generalization error through input compression bound (Shwartz-Ziv and Tishby, 2017; Yan et al., 2023).

4.8.4 Generic evaluation approach

4.8.4.1 Pipeline definition

Following the general pipeline for the development and release of machine learning and deep learning models, as defined in Figure 29 of Section 4.2.1 and Figure 44 of Section 4.8.2.3, and the requirements discussed previously (for data preparation, generalization guaranty and verification pipelines), we consider the training, evaluation, and implementation steps, to provide a workflow promoting generalizability of implemented models. Putting ahead the best practices and potential means to avoid common mistakes (cf. Section 4.7.1), we define a domain-independent approach for ML/DL applications design and evaluation. Figure 45 shows the different steps and loops that we suggest building an appropriate data optimization, training-evaluation and performance verification framework, to cover most of the ML-related steps of EASA's W-shaped process.

[113] *for example "we need closely 50 000 data to statistically guarantee a difference of* 10^{-2} *with a 90% probability"*

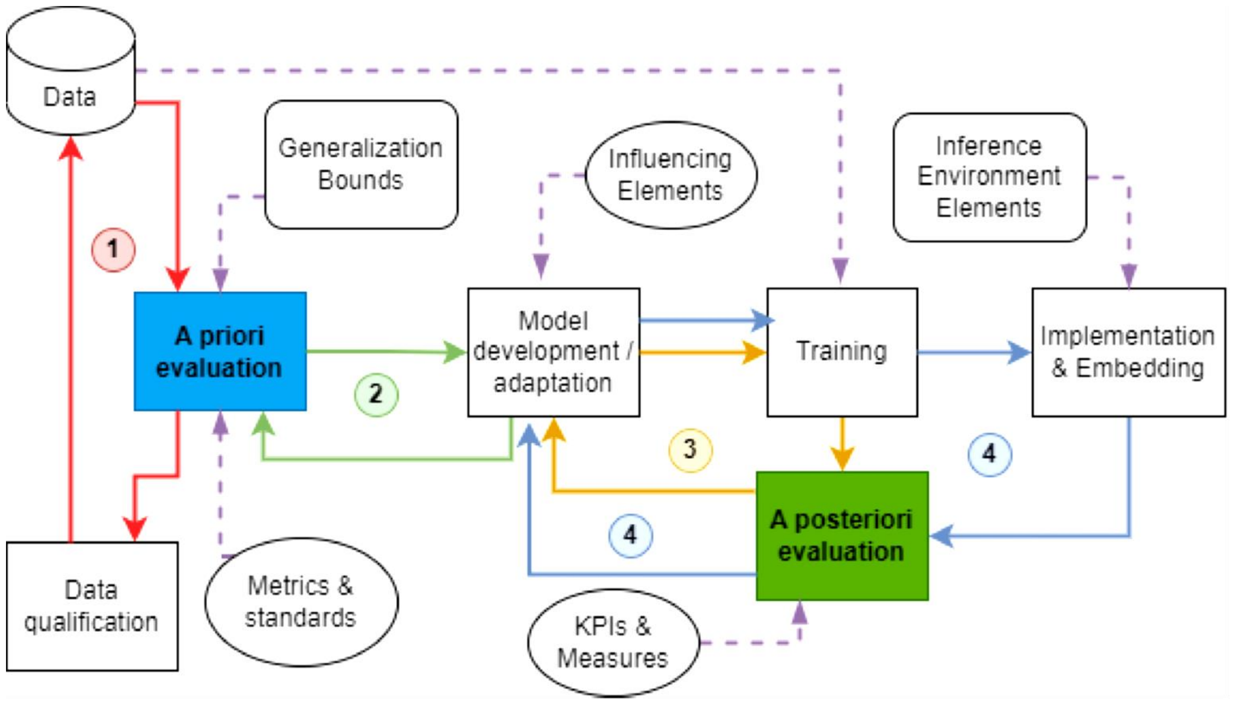


Figure 45. A general framework for ML/DL models development and evaluation, including a priori evaluation of data, metrics, and generalization bounds, and a posteriori evaluation after training, which is built around the model optimization and verification before/after implementation.

Figure 45 shows three main loops, loop 1 corresponds to data optimization and qualification, loop 2 to the model design optimization, and loop 3 corresponds to the training of the model and its optimization before putting it into production.

(1) After the data is collected, it needs to be evaluated and qualified based on a set of metrics and practices (cf. section 4.5.4.1) w.r.t. the target application. This evaluation includes two main steps:
  a. the assessment of the minimal size of data set needed. This should be performed early enough to ensure that data collection activities are launched if needed;
  b. the data quality evaluation, where the completeness and representativeness are some of the features that will be verified. Targeted performances could also be analysed, independently of the models and the training algorithm (derived from section 3).

  After these two evaluation steps, a set of operations, like data augmentation, processing, cleaning can be performed when needed. Finally, the data is then divided into at least three sets, for training, testing and evaluation;

(2) Once the model architecture is chosen, it should be developed and adapted to deal with the target application constraints, such as : data constraints (e.g. data size and type, alignment and independency between training and testing sets …); the mappings between the inputs and outputs; as well as the generalization bounds (cf. Section 4.4.2) that provide signals on the ability of the trained model to generalize to unseen data after training; in addition to the abstraction of targeted performance through a well combined set of evaluation metrics (cf. Section 1.3). Furthermore, some known elements that could influence the models results/performance should be anticipated. For instance, several regularization and model adaptation methods (cf. Section 4.5) can be used in the model level, and other methods (e.g. data expansion and evaluation) can be used to better construct the training data sets (cf. Section 3). 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) need to be included in the input features description and/or representations learning;

(3) When loop (2) is completed, the model and the data are qualified for running. Then, the model is trained on the optimized data set. To do so, a benchmark including a set of industrial KPIs, evaluation measures, and acceptability criteria (e.g. threshold values on every metric) need to be defined for the target task to be performed effectively. *A posteriori* evaluation of the trained model is then performed to ensure that it meets the industrial objectives and the evaluation standard as well. In this crucial step, measures defined in Section 5.3 can be adapted to match the target application. Adapted loss functions and objective functions should to be used as recommended in the state of the art (Q. Wang et al., 2022; Zhu et al., 2022).In this crucial step, measures defined in Section 1.3 can be adapted to match the target application. Adapted loss functions and objective functions need to be used as recommended in the state of the art (Q. Wang et al., 2022; Zhu et al., 2022);

(4) The post implementation evaluation aims to verify the trained model and its ability to bear the complete process on which it is implemented (i.e Embedding system, submodule, whole system providing intermediate results …). This verification, under influencing elements of the system and the implementation environment is crucial. It is needed to verify:
   a. If performances are considerably deteriorated, the influencing factors should be taken into consideration in the adaptation loop. Hence, we could get back to model adaptation step, training, and evaluation;
   b. Otherwise, these elements can be included before training (in step 3), to ensure good performance after implementation, and less impact on the released model.

Note that this process is designed for an offline model training setting. It applies to both supervised and unsupervised model development, and can be farther extended to online training settings (e.g. lifelong learning frameworks). Hence, depending on the target application and the task being addressed, our evaluation of ML/DL models is two folds:

A. ***A priori evaluation.*** Where the listed development and design pitfalls (cf. Sections 4.7.1 and 4.7.2) and mishandlings of the task (cf. Section 4.7.3) should be evaluated, in order to prevent harmful learning and development process of the model, as well as the generalization guarantees evaluation and bounds identification. Finally, with respect to the target system requirements, a hypothesis on the performance requirements of the ML/DL model should be made;
B. ***A posteriori evaluation***. Where a set of technical metrics (cf. Section 1.3) 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 generalization 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 and then allow to identify ways to optimize the model better.

4.8.4.2 Target application definition

Based on the pipeline defined in the previous section (cf. figure 43), the targeted application needs to be defined in terms of data inputs, processors, and expected outputs. Hence, for a data set D, we need to define a function T_f representing a series of transformations applied to every instance $d \in D$ to put it into an independent input space X and a dependent output space Y:

$$D \xrightarrow{T_f} (X, Y), \quad with$$
$$T_f(d) = (x, y), \quad and \;\; x \in X \;\; and \;\; y \in Y$$

This transformation function can include a series of data analyses that will be performed to identify whether there is a need for a data augmentation (generation and expansion), after assessing the quality and quantity of data in our possession (cf. Section 4.5.4). This step will give some evidence of how the data will be processed, to fit into the model's architecture. A representation computation method will be then used in order to transform the data set as a part of the global function T_f. Next, we define the important elements that constitute an AI pipeline designed for the target application T:

- Given a model $f \in F$, the input space X, and the output space Y, we define the target application T as a system of 8 elements:

$$T = \begin{cases} f \in F\,, & (1) \\ X:\; x_i = [x_i^0, \dots, x_i^n], & (2) \\ Y:\; y_j = [y_j^0, \dots, y_j^m], & (3) \\ f: X \xrightarrow{f(x)} Y\,, & (4) \\ M = \{m_1, \dots, m_k\}, & (5) \\ B = \{b_1, \dots, b_l\}, & (6) \\ b_t\big(m_t \circ f(x_t)\big) \in \{0, 1\} & (7) \\ E = \{e_1, \dots, e_z; (e_i \odot x_i) = x'_i\} & (8) \end{cases}$$

In this system:

(1) Corresponds to a mapping strategy (trained model), part of 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 formalized in the definition 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 values (features) associated with every sample in X;

(3) Is the definition of the space Y, it provides a set of results and/or consequences emerging from the inputs in X, and every element $y_j \in Y$ can be viewed as a vector of resulting observations (output values, classes…);

(4) Corresponds to the mapping from the input to the output space, performed by the trained model f;

(5) Is a set of SMART[114] indices that provide signals in the effectiveness of f while performing the mappings in (4). Hence, we define M as a set of metrics matchingmatch the objectives of the application. Hence, for f to be effective it should obey all the elements $m_t \in M$ corresponding to evaluation measures (cf. Section 1.3 for different possibilities) and KPIs as well. This should provide standards of accurate mapping in T (check correctness and relevance of the mappings), and that should be valid whatever the elements of both the spaces X and Y. This phenomenon corresponds to the "*Generalization*" aspect that we need to quantify and qualify as well;

(6) Represents a *benchmarking* scheme of the industrial KPIs. Every element b_t corresponds to a validity/acceptance factor and/or a target performance index threshold value, as intended

[114] SMART (Haughey, 2014) objectives are: specific, measurable, achievable, relevant and time-bounded.

in the ground truth. Hence, for each metric $m_t \in M$, there exist an element $b_t \in B$ which is a validity criteria (e.g. a minimum threshold value, or indicator p ϵ [0,1] that could even represent a probability tolerance) that enables to correctly interpret the computed value by the measure;

(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 $b_t(m_t \circ f(x_t))$ 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 sub-system including the ML/DL model under development;

(8) It provides a set of phenomena and/or conditions and circumstances that directly impact the inputs, and hence the model predictions after implementation. This could represent the set of environment elements (e.g. blurring, noise, signal errors, interference… several implementations can be found in ISO 24029-2[115]) 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 data set.

Note that, this function can be considered in two different ways (regarding the real environment)

a. Training time:
    i. Used as an "*identity function*": if the inputs are perfect (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,
b. Implementation time: after embedding in the target system, the function $\odot$ represents the introduction of real-life factors E that could harm the model performance. Hence, we believe that environmental factors should be included in the model evaluation and training process as well.

4.8.4.3 Evaluation

Based on the pipeline defined in Section 4.8.4.1 and the target application definition of Section 4.8.4.2, the resulting evaluation pipeline includes the key steps to achieve most of the objectives of an AI application. For each of the different steps, the targeted performance (hypothesis on performances of the ML/DL component) is defined formally and will be checked empirically in the different use-cases of the MLEAP project (Chapter 0).

Note that, one of the main issues that could impact the robustness/generalization of a model is the reproducibility of the results (Tatman et al., 2018), for the same inputs processed in equivalent circumstances but in different times. Hence, in the evaluation pipeline of section 4.8.4.1, one of the main objectives is to ensure that the model would not produce unexpected behaviours by adapting different behaviours in the same conditions. Hence, this aspect is evaluated in loop (4) of the process, to make sure that the model is still performant after its implementation in the target system. In the literature, few work (McDermott et al., 2021; Olorisade et al., 2017) has been done on this aspect so far. Mainly, in deep learning models, the reproducibility is an important concern, as several factors can

[115] https://www.iso.org/obp/ui/#iso:std:iso-iec:24029:-2:dis:ed-1:v1:en

prevent the reproducibility of the results of the same model, including random initializations and the hardware features. Although recent studies (Chen et al., 2022) have investigated some solutions to promote results reproducibility in DL, there is no specific approach that define away to design and develop DL models to guarantee reproducible results.
Finally, the theoretical hypothesis on performance requirements corresponding to the different models and applications, under investigation on the MLEAP project, will be checked and developed more on the next phases of the project, to give more precisions about applicable methods to bridge the gap between experimentation and target system requirement.

4.9 Conclusion

In machine learning and deep learning, the key to success of any approach is to provide a model that is robust enough and that could generalize well on unseen data, during training, without a drop in performance when moving from one context to another. 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 generalizing from a large number of training examples using no domain-specific knowledge. In the past few years new methods have been developed for applying domain-specific knowledge to formulate valid generalizations from single training examples. Other techniques were designed to solve problems related to areas with little qualified data, as well as training algorithms to accelerate and improve model learning. When it comes to generalization evaluation, it is highlighted that the classical approach that aims at using a set of technical metrics to assess the model's performance is limited in terms of capturing performance aspects related to the industry (KPIs) and reproducibility. Hence, the generalization evaluation is ultimately more crucial than originally thought.
In this section, we have made a detailed analysis of the state-of-the-art of ML/DL generalization evaluation and provided our main observations about the cited methods, concerning the generalization assessment, issues detection, and methods of improvement. Finally, we provided a generic target domain definition and a generalization evaluation protocol in order to take into account the best practices for ML/DL models development, independently from the target domain characteristics. The proposed pipeline in this section (cf. Figure 45) is mainly focused on generalization assessment and how to conduct the development in a way to reduce the drop on model performance after implementation. This process will be developed more in the next phases of the project, based on findings related to the data evaluation and preparation (Chapter 3), to take into account elements that could harm the model performance (loop (1) on Figure 45), and consider recommendation about robustness and performance stability evaluation (Chapter 5), to validate the model after its implementation (loop 4 of Figure 45).

5. Model evaluation: robustness and stability

5.1 Introduction

5.1.1 Background

Methods to evaluate the robustness and the stability of ML models are moving rapidly from academic prototypes to more industrial tools. These methods can be classified roughly into three categories: statistical, formal and empirical methods. They can also be combined. Statistical approaches usually rely on a mathematical testing process and are able to illustrate a certain level of confidence 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 will we study the approaches developed by the tools and methods available and the target properties as well as their domain validity.

5.1.2 Scope of the chapter

This chapter aims at studying the relationship between concepts developed in the EASA CP and ISO/IEC concepts around the robustness and the stability of machine learning models.
First, the chapter is relying on the terminology associated to the robustness and stability concepts detailed in Section 1.2.3.1, specifically between the EASA CP, the CoDANN documents and ISO/IEC literature. The chapter studies how to align related concepts from all sources in order to support the MLEAP program with the on-going standardization work. Then, this chapter aims at analysing 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.

5.2 Robustness and stability

5.2.1 Robustness and stability concepts

Robustness and stability concepts were introduced in Section 1.2.3 as components of the trustworthiness of a ML system. These two concepts have obvious relation with the model performance, but also the way it has been trained (see chapter 4 for more) and the data that has been used (see chapter 3 for more). In this chapter we aim at detailing what is currently available to assess the robustness and stability beyond the performance metric that are used initially (see section 1.3 for more).

5.2.2 Robustness and stability properties already available

From the literature, several properties can be tied to robustness and stability. This section aims at listing (non-exhaustively) properties available and their associated criteria[116]. For the whole chapter, we use the following terms (from the CoDANN documents):

- X is the input space;
- n is the size of a data set;

[116] 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).

- $x \sim \mathcal{X}$ 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;
- $\mathcal{F}$ is 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.

5.2.2.1 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 are easier to specify and verify. Local robustness properties are specified with respect to 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 to it are still classified with the same outcome. Since local stability or robustness does not entail local performance, the monitoring of 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, for example, 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).

5.2.2.2 [Model] Bias (CoDANN-1 and EASA CP)

Model Bias is defined as an error from 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 4.3).
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.

5.2.2.3 [Model] Variance (CODANN-1)

It is defined as an error from 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 4.3).
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 $\bar{f_n}$:

$$var(\mathcal{F}, n, x) = \mathbb{E}_{D \sim \mathcal{X}^n}\left[\left(\hat{f}^{(D)} - \bar{f_n}(x)\right)^2\right]$$

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, the variance of a learning algorithm can be interpreted as a measure of its fluctuations around the average model and thus reflects how stable it is.

5.2.2.4 Learning algorithm stability

The LM-11 objective from EASA CP refers to the learning algorithm stability by pointing out to the 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. The 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 is changing.
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 model, because of the random nature that can occur during training. It would rather be defined using the gap in performance between the two evaluated models, by using metric such as bias, variance, robustness, relevance. For that evaluation a threshold should be define.

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:

$$|D_{train} \cap D'_{train}| > N \Rightarrow \|P(\mathcal{F}(D_{train}), T) - P(\mathcal{F}(D'_{train}), T)\| < \varepsilon, \text{ where } \varepsilon \in \mathbb{R}_+.$$

5.2.2.5 Stability (ISO/IEC)

5.2.2.5.1 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 hold 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 of the input space?”).
A stability property allows to check if a machine learning model remains performant in the presence of noisy inputs. In order to be usable, a trained model stability property requires that the input space presents some regularity properties. It is not adapted for a chaotic system for example, 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 useful to use the trained model stability property to compare different machine learning models. For example, it is possible to measure for each system their maximum stable space around the same input.
The property is then being implicitly extended to the inference model stability by ISO/IEC 24029-2 (ISO/IEC, 2022b) by the Deployment requirements expressed by the document. 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 an equivalent to the inference model stability of the EASA CP.

5.2.2.5.2 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 the transformation of 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 5.3.2.1.
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 and 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.

In order to be a fair comparison, the machine learning models need to have performed the same tasks, the criterion needs to 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 done.
To be applicable, a trained or inference model stability criterion relies on pre-existing information of the expected output of the machine learning model. This information can be known by the user or can be determined by other means (using simulation or solvers systems). It is well-suited to assess the stability over a 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:

$$\text{Given } x \in D \text{ and } \delta \in R_+ \text{ then } \|x' - x\| < \delta \Rightarrow \hat{f}(x') = \hat{f}(x), \text{ where } x' \in X.$$

5.2.2.6 Sensitivity (ISO/IEC)

5.2.2.6.1 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. In order to assess the robustness on 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 which can influence that variation of the output. This analysis is then compared to a pre-existing understanding of the expected

performance of the system. 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 is used over an ODD to prove 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.

5.2.2.6.2 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 5.3.2.1.
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 what are the sensitivity thresholds to be checked.
- A sensitivity criterion may be used to compare different machine learning model architectures or trained models. 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 has the same objective to be proven.

A sensitivity criterion is especially well-suited for machine learning models performing interpolation or regression tasks. For these kinds of tasks, it often allows a 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 $\varepsilon \in R_+$.

5.2.2.7 Relevance (ISO/IEC)

5.2.2.7.1 Relevance property

The relevance property is first introduced in ISO/IEC 24029-1 (ISO/IEC, n.d.). This property helps to evaluate the decision of a model, contributing to its development explainability as well as its stability and robustness evaluation. A relevance property on neural network (and here by extension to a machine learning model) expresses an ordering of the impact of the inputs on the outputs. For each output a relevance property expresses the individual impact of each input on the result obtained for this output. For each output the individual 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 a variety of methods to evaluate the impact of each input. 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 relevance property results, both of which could be considered as acceptable depending on the experts. This case should be taken into account in the comparison protocol in order to be resolved. For example, a protocol may use an evaluation system (as for an inter-annotator agreement (Mathet et al., 2015)) in order to resolve the situation. This stems from the fact that interpretability methods used to build a relevance 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 can be done by a human. For (ISO/IEC, n.d.) in these cases the justification of the output of the machine

learning model should be understood and checked, whereas in EASA CP the objectives of explainability are applicable in every cases (i.e. not only by tasks done by a human). A relevance property is essential in order to assert if the performance of the system can be assured for the correct reasons. If that is the case then the robustness of the system can be justified and not just asserted. This verification may be done manually by a human operator or automatically using references that have been checked before.

5.2.2.7.2 Relevance criterion

A relevance criterion expresses a relevance property over an ODD which requires demonstration of a link between each input and the outputs. For that, it requires a method to quantify the influence of each input. Formal methods relying on symbolic calculus, logical calculus or computational methods can be used to achieve such a goal. Examples of formal methods available to check a relevance criterion are provided in 5.3.2.1

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 at least the methodology to evaluate the results if they cannot be defined *a priori*.
- A relevance criterion allows to compare 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 in order to identify 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 on a variety of tasks, as long as the result can be analysed by a user. A relevance criterion can be used for example on 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.

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 $[r_1, r_2, \ldots r_n]$ 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, $R(f, S) = [r_1, r_2, \ldots r_n]$. From the vector $[r_1, r_2, \ldots r_n]$ we considered the fully ordered vector $[r'_1, r'_2, \ldots r'_n]$ which is a permutation of the initial vector. A relevance criterion is an evaluation of acceptability of the vector $[r'_1, r'_2, \ldots r'_n]$ regarding an expected vector $[r^*_1, r^*_2, \ldots r^*_n]$. This acceptability can be formalized in several ways, for example by using the Hamming distance or the transposition distance.

5.2.2.8 Reachability (ISO/IEC)

5.2.2.8.1 Reachability property

Reachability property is first introduced in ISO/IEC 24029-2 (ISO/IEC, 2022b). A reachability property on a machine learning model expresses the multi-step performance of the model in conjunction with its operating environment. A reachability property 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 that 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 it is possible for the agent to reach a particular set of states.

5.2.2.8.2 Reachability criterion

A reachability criterion expresses a reachability property over a given set of initial states. For a deterministic environment it can be checked using methods described in Section 5.3.2.1.3. For a stochastic environment, the criterion expresses a probability of reaching a set of states. This probability can be determined using methods in Section 5.3.2.1.4.
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 be used to determine the set of 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 performance of the network 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.
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 is met if:

$$\forall x \in S, \qquad n \in \mathbb{N}, \qquad \mathcal{E}^n\left(x, \hat{f}^n(x)\right) \notin Z$$

5.2.2.9 Discussion

The EASA CP and ISO/IEC concepts are overlapping in part but are not entirely aligned. EASA CP is considering a "fine-grained" notion of robustness by assessing the stability of the training algorithm, the robustness of the trained model and of the inference model, while the ISO/IEC is almost only focused on the trained and inference model.
The CODANN-1 document would use the notions of bias and variance for example as one way to evaluate the stability and robustness of a model. 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.
The concept of relevance introduced in the ISO/IEC standards however is not present in the EASA CP (neither in the CoDANN IPC reports) and can help expand the notions around robustness.
In conclusion, we can say that 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
Reachability (from CODANN-1)	Reachability

Table 12 – table of relevant concepts from either EASA CP or ISO/IEC

5.2.3 General framework of methods to assess robustness and stability

5.2.3.1 EASA CP framework

Robustness and stability are verified through different requirement depending on the level of the system to assess. Stability is viewed as a property stemming from the learning process, it implies that the outcome of the process has to handle properly the trade-off between bias and variance. Different methods can be deployed to assess bias and variance at this stage. Robustness is a property of the model, the ML component and/or the system against the adverse conditions in the ODD. On each level, different approach 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 IMP-08 with requirement-based robustness testing, including input perturbations and edge cases tests cases	ODD monitoring, outliers, infeasible corner cases and novelty	Performance monitoring

Table 13 – table describing the different concepts of robustness and stability and the actions related to their assessment

5.2.3.2 ISO/IEC framework

5.2.3.2.1 AI Life cycle

In ISO/IEC literature the life cycle of AI is organized as described in Figure 46. It is composed of several phases starting with the Inception of the AI project to the Retirement of the AI system. CoDANN documents are mostly focusing on phases starting at the Inception phase up to the Operation and monitoring phase. The EASA CP has instead a wider spectrum which overlaps most of the life cycle described in Figure 46.

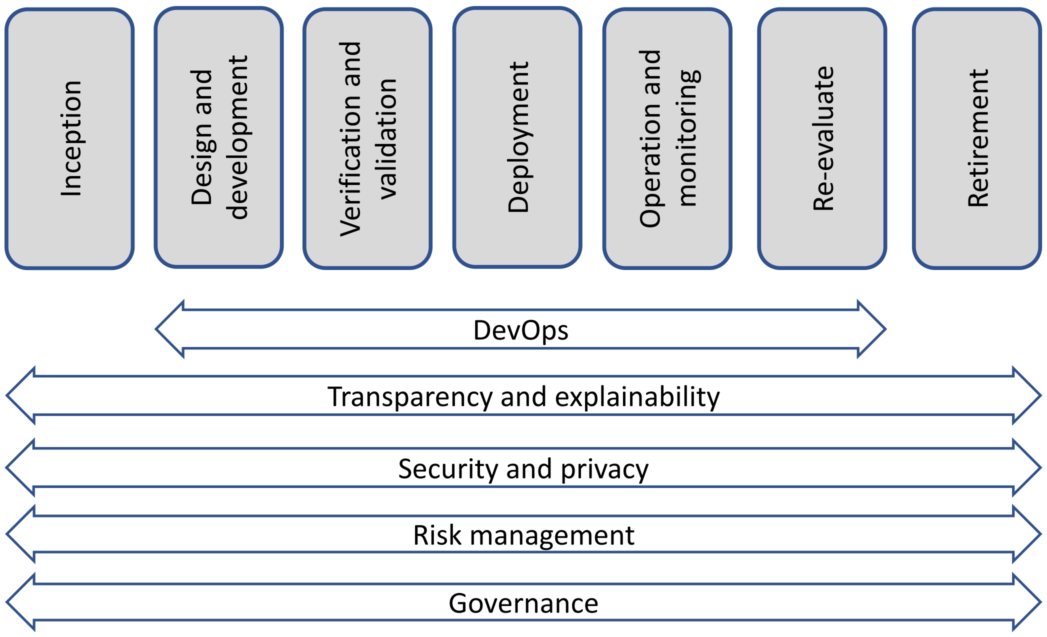


Figure 46 – ISO/IEC AI life cycle available in ISO/IEC 22989 (ISO/IEC, 2022a)

5.2.3.2.2 Correspondence between AI life cycle frameworks regarding robustness

Following the description given in 5.2.3.1 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 regroup the training and internal selection of a trained model. The verification and validation step is 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. While the Deployment phase does not seem to be explicitly covered in CoDANN-1. CoDANN-2 (EASA and Daedalean, 2021) is considering in its chapter 3.2.2 the impact of the transformation of a deep neural network toward a specific hardware and imply the fact that it would probably require to retrain the model. This is also taken into account in the notion of inference model used by the EASA CP. CoDANN-2 is advocating an approach where the trained model adapted to hardware is a separate model that needs to go through every step of verification. The EASA CP is aiming at generalizing the objectives to avoid being prescriptive on one approach, which considers the inference model as a derived model 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. To ensure that it respect still the same properties the evaluation protocol should be also applied on the inference model. ISO/IEC is more aligned with EASA CP, by considering the transformation of a trained model as a new step in the process to deploy a pre-validated model.

Table 14 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 architecture aspects, whereas in the EASA CP this notion appears through the distinction between ML model, ML constituent and AI-based system robustness. 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 a ML component all the way to the deployment and the monitoring of the model into 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 14 – Correspondence between the EASA CP robustness concepts and the ISO/IEC life cycle phase.

5.2.3.2.3 *Robustness assessment*

In the ISO/IEC TR 24029-1, robustness is assessed through a process that takes place 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 focusses on the robustness of the outcome. The process is decomposed into 6 steps.

1) The first step is a statement of the robustness goals. During this initial step, the targets to be tested for robustness are identified. 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 is 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 methods 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 consists in describing 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. It is possible to perform tests using a real-world experiment or a simulation, and potentially a combination of these approaches.
4) After completion, tests 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.

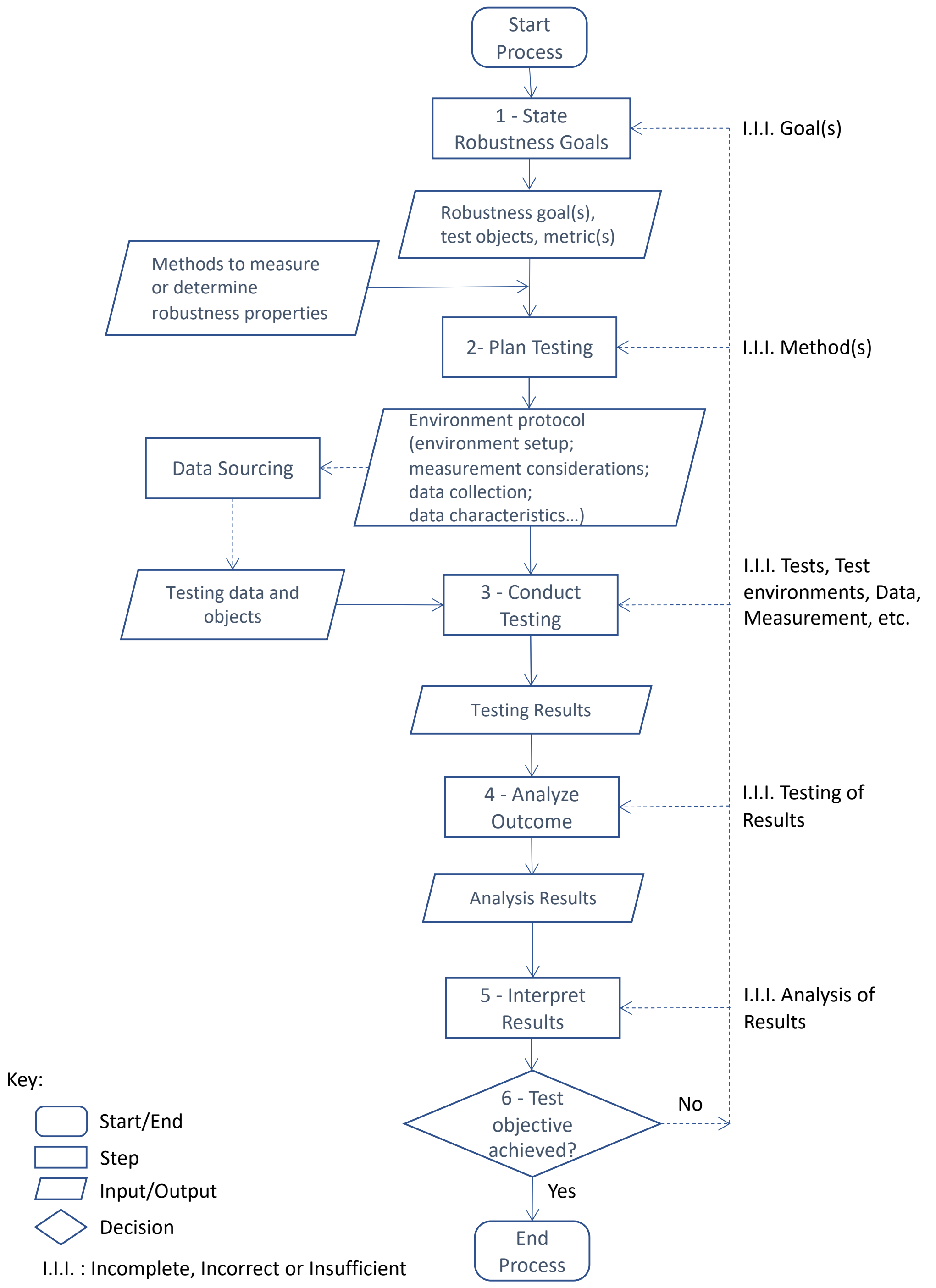

Start Process
1 - State Robustness Goals
I.I.I. Goal(s)
Robustness goal(s), test objects, metric(s)
Methods to measure or determine robustness properties
2- Plan Testing
I.I.I. Method(s)
Environment protocol (environment setup; measurement considerations; data collection; data characteristics…)
Data 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 47 – ISO/IEC 24029-1 generic workflow to assess robustness from (ISO/IEC, n.d.)

When the final test objectives are not met, an analysis of the process is conducted and the process returns to the appropriate preceding step, in order to alleviate deficiencies, e.g., add robustness goals, modify or add metrics, add consideration of different aspects to measure, re-plan tests, etc.

One important step that is not overly detailed in the 24029-1 is Data sourcing. It is a process selecting, producing and/or generating the testing data and objects that are needed for conducting the testing. It can sometimes include considerations of legal or other regulatory nature, as well as practical or technical issues. This step could constitute a serious 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 failsafe preventing them. In any case the testing protocol contains the requirements and the criteria necessary for data sourcing. Data sourcing issues and methods are covered extensively in Chapter 3.

5.3 Review of methods and tools

5.3.1 Statistical methods

5.3.1.1 Method

In order to evaluate the robustness or the stability of a ML system it is possible to apply a statistical methodology. In short, the general method consists of choosing the data set to evaluate the ML system on, as well as the metrics that will be calculated. For that, the general method will select one or several metrics that are presented in Section 1.3 in order to considered them together. These metrics will then be applied on the system using the testing data in order to assess the robustness or the stability. This section describes the available statistical methodology to perform the steps 2 and 3 described in Section 5.2.3.2 as to plan and conduct the testing. Performing a testing protocol is not unique to machine learning models and considerations include the testing environment set-up, what and how to measure, 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 on a reference data set and then on one or several data sets representative of the targeted changes of circumstances. For each of those, if the performance drop on the reference test set is sufficiently low the system is deemed robust. Section 1.3 present a detailed list of available metrics for the purpose of applying statical methods.

5.3.1.2 Corpus amplification

Corpus amplification, or augmentation, is a statistical method used to ensure a 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 keep identical performance when rotating both image and reference.
- Adding different types of noise or other forms of input transformation or degradation[117] which are reasonable in the considered use-case and hence the system should be robust to.

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 4 for the use of those methods at training time.

[117] 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.

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 difficult to put in place 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 the 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 principled 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

The approach, in evaluation, is usually to plot the performance of the system as a function of the noise level and, where a level of performance is deemed to be inadequate, to have an expert examine examples of noisy inputs so as 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, which are presented later in the document, analytically cover the whole noise space to ensure every potential input is tested.

5.3.1.3 Corner and edge case detection using test based methods

Corner cases and edge cases are by essence more complex cases that can be hard to test exhaustively because they are located on the limit of the ODD (see Section 1.2.3.2 for more). But corner case detection is not limited to ML system, in classical software validation their handling is necessary. Some test methods for DNNs corner case detection were in fact directly inspired by software test methods. In classical software testing, one can find two main sorts of methods (Ahamed, 2010):

- White-box test methods (Williams, 2006), for which one need to own the source code. In this case, tests mostly focus on code coverage;
- Black-box test methods, based on the executable and without knowledge of the source code. In this case, tests mostly focus on functional properties.

These methods coming from software engineering have been adapted to DNNs as developed hereafter. Section 5.3.1.3.1 deals with white box approaches for neural networks while the methods related to black box testing are discussed in Section 5.3.1.3.2.

5.3.1.3.1 White Box Testing

In (Pei et al., 2017) the authors states 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 also by 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 is not very useful for DNN-based software as the logic is not encoded using control flow. DNNs are built in a different way that traditional software, by automatically learn its logic from the data rather than expressing the human intent through a control flow graph. They operate also differently since they do not rely on the control flow to change behaviour but rather 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[118], 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 DNNs have 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 synthesize test cases that maximizes neuron coverage in DNN based systems like autonomous cars. They also demonstrate that different realistic image transformations like changes in contrast, presence of fog, etc. can be used to generate synthetic tests that increase neuron coverage. This approach is implemented into the DeepTest[119] tool.

In (Yu et al., 2022) the authors propose a differential testing strategy to automatically verify inconsistent behaviour of DNNs, which can track inactive neurons and triggered them in order to maximize neuron coverage. Then they use an optimization 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. It is done to improve the coverage and to find adversarial inputs. This approach is implemented into the Adapt[120] tool.

5.3.1.3.2 Black Box Testing

[118] https://github.com/peikexin9/deepxplore
[119] https://github.com/ARiSE-Lab/deepTest
[120] https://github.com/kupl/adapt

In (Tian et al., 2017) an overview of the black box testing approaches is provided, which present 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 are not able to 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 uses diversity metrics, to then use them to construct corner case detection cases.
Another recent line of work has explored the possibility of 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.
Also metamorphic testing (Chen et al., 2020; Zhou et al., 2004) is a way of creating test oracles in settings where manual specifications are not available. Metamorphic testing has been used in the past for testing both supervised and unsupervised machine learning classifiers (Murphy et al., 2008; Xie X, 2009).
Finally, surprise adequacy methods, as described in Section 1.2.3.2.3, can also be considered as black box testing methods.
In this section, the academic papers are referencing only prototypes in their work. These prototypes are not always available to be tested but some of them are.

5.3.1.4 Applicability analysis

In this section we study the applicability of the methods presented in the previous sections to the properties detailed in Section 5.2.2. Corner case and edge case detection are not considered here since they relate more to performance properties (see Section 1.3 for more) than stability and robustness properties. The choice of properties reflects both the current literature on the topic from the standardization effort (e.g. (ISO/IEC, 2022a)) and the EASA CP (EASA, 2023). 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 to the best of our knowledge.

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 15 – Applicability of the different metrics and methods on the robustness and stability property

5.3.1.5 Tools maturity and scalability

Statistical methods presented in 5.3.1 dispose of a quite wide variety of tools available to implement different metrics. It is also entirely possible to implement them directly in any programming language. Here are some examples of common 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 »[121]: AUC, ROC, rmse, mae..
    - Package « bmrm »[122] hinge loss
    - Package « vcd »[123] cohen's kappa
    - Package « caret »[124] confusion matrix
- Using Python, with the library scikit-learn[125] which contains already almost all metrics
- Using Matics[126] (LNE)
- Proprietary solutions also exist such for example as the ones provided by:

[121] https://cran.r-project.org/web/packages/Metrics/Metrics.pdf
[122] https://cran.r-project.org/web/packages/bmrm/bmrm.pdf
[123] https://cran.r-project.org/web/packages/vcd/vcd.pdf
[124] http://cran.nexr.com/web/packages/caret/caret.pdf
[125] https://scikit-learn.org/stable/search.html?q=metrics
[126] https://www.lne.fr/fr/logiciels/lne-matics

  - Tibco (Data Science[127])
  - Microsoft (Azure [128])
  - Salesforce (Tableau[129])
  - SAS (Model Studio[130])
  - …

Concerning the evaluation of corner cases most of the most advance tool available rely on a white-box testing approach, such as:

- DeepXplore[131]
- DeepTest[132]
- Adapt[133]

5.3.2 Formal methods

5.3.2.1 Methods

This section uses the description provided by ISO/IEC 24029-1 (ISO/IEC, n.d.) and 24029-2 (ISO/IEC, 2022b) to introduced the different methods available to apply formal methods to neural networks. Indeed, most of the methods described in these documents were designed primarily for neural networks. Here we are considering machine learning models in a broader sense. While for many of the methods described there is no major problem to be adapted to other machine learning techniques, this work is mostly not yet done.

5.3.2.1.1 Solver

Neural network models are known to be often large, non-linear, non-convex and beyond the reach of general-purpose tools such as linear programming solvers or existing satisfiability modulo theories (SMT). However, some advances have been made to use 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 then use these constraints 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 certain non-linear activations (such as hyperbolic functions including sigmoid and tanh) are too complex to encode precisely. The same issues would be identified for example for SVM with non-linear kernel (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 optimize the robustness constraint. If the bounds

[127] https://www.tibco.com/fr/products/data-science
[128] https://azure.microsoft.com/fr-fr/
[129] https://www.tableau.com
[130] https://support.sas.com/en/software/model-studio-support.html
[131] https://github.com/peikexin9/deepxplore
[132] https://github.com/ARiSE-Lab/deepTest
[133] https://github.com/kupl/adapt

on the robustness constraint satisfy the requirements, the property is proven. SMT solvers pose the verification problem as a constraint satisfiability question that either holds or 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).

5.3.2.1.2 Abstract interpretation

Abstract interpretation is a kind of formal method that relies on a theory which constructs controlled approximations. It is often used to prove complex program properties (R. and Cousot, 1977). Abstract interpretation has taken a larger role in the software verification and validation community, especially in the context of safety-critical software, such as embedded software for aircrafts (Souyris J., 2007), cars (Yamaguchi T., 2019), spacecrafts (O. Bouissou, 2009). It aims at tackling 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 is used to provide 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 input based on the robustness specification. This region can be represented exactly or approximately using certain 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, such that every layer is sequentially applied to the input region. The input region is transformed into an output region containing all outputs reachable from the input region. Depending on the layer, this can introduce approximations (outputs that are unreachable from the input region).
- Finally, an output region captures all possible outputs of the network for input perturbations that are 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 effective verification.

The principles of abstract interpretation are as follows. First, the approach usually considers all the possible executions of a program through the use of 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 at least 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 done by defining two specific functions: one called the abstraction α (going from the concrete to the abstract domain) and the other called the concretization γ (going from the abstract to the concrete

domain). These functions have some specific properties that are 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 the all the concrete executions, which is also tractable (by construction). It is then possible to prove properties on the abstract domain and have them transferred automatically on the corresponding concrete traces represented by the abstraction. The main difficulty with an abstract interpretation is to define a simple enough 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, in order to represent 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 is the result of 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 network. 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.

5.3.2.1.3 Reachability analysis in deterministic environments

Reachability-based machine learning verification techniques combine the outputs of the solvers described in Section 5.3.2.1.1 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 5.3.2.1.1 can be used to determine the possible control outputs of the network in the region it defines. Using this information along with the environment model, it is possible to determine an overapproximation of the range of possible next states from any given cell. By repeating this for all cells in the initial state region over multiple time steps, an overapproximation of the set of reachable states can be determined (K. Julian, 2019). Another approach to this problem is to encode an overapproximation of the environment dynamics as constraints in a mixed-integer program and using the mixed-integer verification technique from Section 5.3.2.1.1 to solve for an overapproximation the output reachable set (C. Sidrane, 2019).

5.3.2.1.4 Reachability analysis in non-deterministic environments

When the environment is stochastic, the solvers in Section 5.3.2.1.1 can be combined with techniques in probabilistic model checking to determine the probability of reaching a set of states. Similar to the technique described in Section 5.3.2.1.3, 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 certain 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 when using a machine learning system (S. Katz, 2021).

5.3.2.1.5 Model checking

Model checking is a method to prove that a formal expression of a theory is valid under a certain interpretation. More detailed descriptions can be found in (ISO/IEC/IEEE, 2017) and in (Ehrig H., 1985). 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. A theory can either be expressed by sentences of a predicate logic or expressed by data patterns. Machine learning models are

algorithms designed for discovery and use of 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 on machine learning to prove relationships among different sorts of sets which obey some relationship.
Example 1: The 'theory of family'(Manna Z., 1993) obeys the interpretation which implements the membership of persons belonging to a family. Thus, two arbitrary persons are be proven to be members of the family or not. Then the sentence 'one person is parent of the other person' has to be checked for all available pairs of persons.
Example 2: Model checking has been used in (Sena L. H., 2019) in order to prove the existence of adversarial inputs for a machine learning. The theory is the language constituted by the letters and the weights and biases description of the machine learning. The interpretation is constituted by the label attached to the image of the letter. It is possible to compute a distance between every possible pair of letters in the alphabet. Then, the model can be checked in order to ensure the predicate that every distance is greater than a specific threshold fixed by the user. User predefined predicates are checked through machine learning against a theory.

5.3.2.2 Applicability

In this section we study what methods can prove the properties presented in section 5.2.2 by using the method introduced in (ISO/IEC, n.d.). For each, we review the literature associated and the possible limitation of each given method.

5.3.2.2.1 Using uncertainty analysis to prove interpolation model stability

Uncertainty analysis is a general method used to measure the variability of any kind of mathematical functions. For that the methods tends to search for inputs that can cause significant drift in the behaviour. For 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 measuring the variation of response of the network in order 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 response of the neural network is abnormal and non-robust behaviour is to be expected. In (Montaño and Palmer, 2003), a method is described indicating the impact or importance of the input variables on the output, independently of the nature (quantitative or discrete) of the variables. 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 is also used to detect some conditions under which a machine learning model can operate in a non-robust manner. Another advantage is that is can also allow to trace the cause of such a behaviour by checking the source of the uncertainty change in the computation.

5.3.2.2.2 Using solvers to prove a maximum stable space property

Several example of use of solvers to compute a maximum stable space property have been published recently. For example, in (Huang X., 2016), an SMT solver is used to prove the absence or the existence of 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 as a logical formula the property needed, and the capacity of a generic solver to prove this kind of properties on a machine learning model.

5.3.2.2.3 Using optimization techniques to prove a maximum stable space property

Common optimization techniques are also able to verify a machine learning model, whereby any satisfiability problem is converted to an optimization problem. It then becomes possible to apply conventional optimization techniques, such as the Branch and Bound algorithm(A. H. Land, 1960), to solve it.
In the case of a maximum stable space property, it can be expressed as a Boolean formula on linear inequalities. As an example, the property can express that the output of a neural network must not become greater than a certain value. This requirement is then translated into a new (Boolean) layer in the network. Then the model is analysed in order to find a solution of the optimization problem, consequently the solution of the satisfiability problem is also solved by checking the sign of the solution. In (Bunel, 2017), the construction of such an optimization problem from a satisfiability problem on a neural network is described, combining a classical gradient descent to find a local minimum, as well as a branch and bound optimizer to determine the global optimum.
In the field of optimizations 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.

5.3.2.2.4 Using abstract interpretation to prove a maximum stable space property

Abstract interpretation can easily be used in order to verify a maximum stable space property. It requires to first express the part of the input space to be check using an abstract domain. Then an abstract interpretation verifier can infer using the abstract object the over-approximation of the model over the considered input space. The abstract computation will contain all possible outputs of the system over the input space. For a classifier model for example, the abstract computation will produce for each class the maximum and minimum values possible for the model. Then, two cases are possible: either one of the class outputs is greater than any other (no overlap is detected), or there is no class that always takes over the decision. Several works explore how to compute such a property for neural networks in (A. Boopathy, 2018), (Singh et al., 2018) and (Gehr, n.d.), or for SVM in (Ranzato, 2019).

5.3.2.2.5 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 rather 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 generalized to prove (local) relevance properties. For abstract interpretation to be applied, one needs to first generalize this approach in order to considered 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[134]. 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 5.3.2.1.2, but it applies the abstraction on the gradient space instead on just the input space.

5.3.2.2.6 Using optimization 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 great attention to tackle 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 been employed for the reachability analysis of nonlinear systems yet such as the one used in ML. Finding an exhaustive method to address LPV embedding of nonlinear system on a systematic basis is still an ongoing research topic; most of the available methods are ad-hoc approaches with an inherent lack of generality and/or have shortcomings in addressing predominant issues regarding the constitution of the embedding (A. Kwiatkowsku, 2008; Sadeghzadeh, 2020).

5.3.2.2.7 Using solvers and abstract interpretation to prove a reachability property

By combining both solvers and abstract interpretation it is possible to prove a reachability property. A detailed approach is described in section 5.3.2.1.3 for deterministic environments and 5.3.2.1.4 for non-deterministic environments.

5.3.2.2.8 Overview

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
Solver	N/A	++	++	+	+	N/A	+
Abstract interpretation	N/A	++	++	+	+	++	++
Optimization techniques	N/A	++	++	+	+	+	+

Table 16 – Applicability of the different techniques on the robustness and stability property

While formal techniques allow to tackle 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 is not properly defined yet.

[134] This work has been carried out from 2021 to 2022 through the ESA tender AO10403.

5.3.2.3 Tools overview

5.3.2.3.1 Saimple

Saimple is a proprietary tool based on abstract interpretation to verify neural networks and support vector machine. It uses several proprietary abstract domains developed by Numalis[135]. It can be used to verify both robustness and relevance properties. It can also be used to check reachability, by building around it the necessary loop infrastructure of such use cases.
The product can currently handle convolutional, residual, recurrent architectures of neural networks. In addition, detectors are currently in beta-version in the product. It handles most of the ONNX specification of neural network (with the notable exception of custom layers). Saimple can also process support vector machines with linear, quadratic, cubic and RBF kernel.
The tool has SaaS capability and can be used either through a web interface or through its REST API.

5.3.2.3.2 PyRat

PyRAT is proprietary tool based on abstract interpretation to verify neural networks[136]. 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. To verify networks, PyRAT propagates the abstract domains through them in order to obtain the whole set of reachable values from an initial configuration. 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.

5.3.2.3.3 ERAN

ERAN is an academic tool based on abstract interpretation to analyse neural networks[137]. ERAN stands for *ETH Robustness Analyzer for Neural Networks*. It refers to the work introduced in (Singh et al., 2018) and (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.

[135] https://www.saimple.com/
[136] https://list.cea.fr/en/page/development-environments/
[137] https://github.com/eth-sri/eran

- 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.

5.3.2.3.4 Marabou

Marabou is academic tool based on an SMT solver to compute bounds on neural networks[138]. It is a framework that formerly verifies 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 further enhance scalability. 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 NNet 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:

- 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.

5.3.2.3.5 MIPVerify

MIPVerify is academic tool based on MILP solver to compute bounds on neural networks[139]. 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.).

5.3.2.3.6 DNNV

DNNV is a framework for verifying Deep Neural Networks. DNNV takes as input a neural network[140], 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.

[138] https://neuralnetworkverification.github.io/
[139] https://github.com/vtjeng/MIPVerify.jl
[140] https://github.com/dlshriver/dnnv

5.3.2.3.7 ReluVal

ReluVal is one of the first systems proposed for formally verifying properties of feed-forward fully-connected neural networks with ReLU activations[141], 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 has been further improved with the integrations of the optimizations first proposed in Neurify.

5.3.2.3.8 Nnenum

Nnenum is an exact verifier for sequential neural networks supporting fully-connected and convolutional network with ReLU activation functions[142]. 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 overapproximations with an improved path enumeration verification method for case splitting on the ReLU function.

5.3.2.3.9 $\boldsymbol{\alpha}, \boldsymbol{\beta}$-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[143]. As mentioned, it 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.

5.3.2.3.10 Saver

SAVer (**S**VM **A**bstract **Ver**ifier) is an abstract interpretation-based tool for proving properties of SVMs[144] (Ranzato, 2019), in particular it aims at proving robustness or vulnerability properties of classifiers.

5.3.2.4 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 by "+" or "++" the ability of the tool to deliver results to assess each property, and by N/A if the property cannot be assessed by this method. Assessment is based on the ability of the underlying method used by the tool, the documentation provided and the general knowledge we have of the tool.

[141] https://github.com/tcwangshiqi-columbia/ReluVal
[142] https://github.com/stanleybak/nnenum
[143] https://github.com/Verified-Intelligence/alpha-beta-CROWN
[144] 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 17 – 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 being able to process 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.

5.3.3 Empirical methods

5.3.3.1 Methods

This section presents methods described first in the ISO/IEC 24029-1 (ISO/IEC, n.d.).

5.3.3.1.1 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 5.2.3.2, Figure 47 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 useful 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 of 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.

5.3.3.1.2 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 a priori testing and a posteriori testing method. 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 47 in Section 5.2.3.2 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 references produced by the annotators varies 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 a 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 useful 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 a fully reliable measure of performance and subjective a posteriori testing is often more precise. Similarly, during a navigation task, it is possible to accept several trajectories to move from one place to another. 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 the case of the evaluation of the quality of a human-machine interaction, the evaluation is done a posteriori because it is not possible to know the how the interaction will generalize across the population before widespread interaction takes place. 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.

5.3.3.1.3 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 a set of benchmarking rules that describe and standardize ways of setting up testing, documenting this setup, measuring results 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).

5.3.3.2 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 18 – 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 recognized. 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 it often happen that outliers, incorrect labelling, or even incorrect data are 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 standardize the way benchmarks can be assembled, annotated, traced and validated in order 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 19 – 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[145], with more than thousands of entries.
- The Opensky datasets[146] with more than dozens of thousands of commercial and non-commercial flights.

5.3.4 Selected tools and methods

[145] https://av-info.faa.gov/dd_sublevel.asp?Folder=%5CAirOperators
[146] https://opensky-network.org/datasets/metadata/

Following the assessment done of each method and the maturity of the available tools, the next phase of the MLEAP project 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. 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 both 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.

5.4 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 realization of such a methodology in a tractable way.

As several methods are available the document regroups them into 3 categories: statistical, formal or empirical ones. In each several metrics and techniques are available with a varying degree of 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 a point to implement some of the requirements described in those standardization reference documents.

Statistical methods might be the most straightforward way to analyse these properties. However, they require lots of preparation work in order to setup the right data sets. Also, any attempt to sample in an exhaustive way is immediately limited by the high dimensionality of the input space. In terms of tools many libraries are providing the necessary functions to evaluate statistical metrics. However, few tackle the data issues associated to such methodologies.

Formal methods, while being promising in overcoming this limitation, suffer in part 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. These methods offer stronger properties in terms of robustness and stability; however, they are often limited in the scope on what they can prove over the input space.

Finally, empirical methods might be considered as the most practical one in the sense that 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 to assess to the same level of confidence the properties required on the system. Empirical methods either rely on a concrete assessment evaluated by some human experts which tends to reach a consensus both in term of methodology and the assessment obtain through it. But it can also propose some benchmarking using common and widely spread data sets as points of references. Their use might be considered to applications of low-level criticality depending on the objective the system has to meet.

Overall, any meaningful evaluation of robustness and stability would benefit from a combination of techniques. In this way, the process would cover as much as possible the input space while maintaining tractability.

6. Conclusions

This document presents the current state and latest achievements of the MLEAP project. So far, tools and methodologies needed to instantiate several steps of the EASA's W-shaped development approach (EASA, 2023) have been investigated. The axes being explored focus on:

- Data completeness and representativity, with handling of the corner cases
- Model development, through the handling of the generalization properties
- Model evaluation, in particular in terms of robustness and stability

Data quality is a difficult topic, science-wise, because of the inherent cost which comes with doing research in the field. Completeness and representativeness are usually not handled per se, and almost no dedicated tools exist. Thus there is a need to build indicators from more general metrics (such as entropy) or by leveraging different tools (like sample similarity). Intrinsically, the domain is a difficult one, because 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 tradeoff between representativity and diversity, since rare cases need to be amplified to be modelled correctly. Hence, Chapter 3 provides analysis on the requirements for the ODD to set the expectations for representativity-diversity tradeoff. Hopefully, the array of tools and methods described in the selection grid should give AI developers a chance to document and justify if the tradeoff holds.

The generalizability of trained models, assessment and evaluation is investigated, while analysing methods to avoid along the way under- or overfitting, taking into account the impact of the quality and volume of the data. 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 generalization reached by a training session. Chapter 4 ends up with an operational proposal on how to project these methods into the W-shaped approach, which should be extended to the whole set of tools and methods presented in the document for the next version.

Measuring the quality of the training step takes part in the larger question of the evaluation of the model. We present multiple approaches in Chapter 5, from pure performance measures with empirical, data-based approaches to the validation of explicit properties, in particular of stability, through an array of analytic or formal methods. Those methods, while sometimes difficult to put in practice, allow for very powerful analysis of the behaviour of the models, including at runtime, allowing monitoring of the whole system in a live setup.

An updated version of this public report is expected to be published in May 2024, and will allow validation of the applicability of the methods on the aviation use cases we presented in Chapter 2, and focus on the actual operational usability and scaling of the various tools identified.

Finally, for further information on the MLEAP achievements, and to give your feedback, do not hesitate to contact the ai@easa.europa.eu.

7. 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.

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. Modelling 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.

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., 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 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. Modelling 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.

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 modelling 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.

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, M., Li, X., Zhao, T., 2019. On generalization bounds of a family of recurrent neural networks. arXiv preprint arXiv:1910.12947.

Chen, S.F., Goodman, J., 1999. An empirical study of smoothing techniques for language modelling. 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 biomedicaI 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.

Das, N., Park, H., Wang, Z.J., Hohman, F., Firstman, R., Rogers, E., Chau, D.H., 2020. Massif: Interactive interpretation of adversarial attacks on deep learning, in: Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems. pp. 1–7.

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

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 Behaviour 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., 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, 2023. EASA Concept Paper: First usable guidance for Level 1&2 machine learning applications (proposed issue).

EASA, Collins Aerospace, 2023. Formal Methods use for Learning Assurance (ForMuLA).

EASA, Daedalean, 2021. Concepts of Design Assurance for Neural Networks (CoDANN) II.

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.

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.

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.

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.

Feng, Y., Shi, Q., Gao, X., Wan, J., Fang, C., Chen, Z., 2020. Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks, in: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 177–188.

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., 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.
Galhotra, S., Brun, Y., Meliou, A., 2017. Fairness testing: testing software for discrimination, in: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. pp. 498–510.
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

Gossmann, A., Pezeshk, A., Sahiner, B., 2018. Test data reuse for evaluation of adaptive machine learning algorithms: over-fitting to a fixed'test'dataset and a potential solution, in: Medical Imaging 2018: Image Perception, Observer Performance, and Technology Assessment. SPIE, pp. 121–132.

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.

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 Behaviour: 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.

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

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.

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.

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 modelling 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/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.

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 Modelling 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.

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.

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., 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

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.

Koehrsen, W., 2018. Overfitting vs. underfitting: A complete example. Towards Data Science.

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, R.S.S., Brien, D.O., Albert, K., Viljöen, S., Snover, J., 2019. Failure modes in machine learning systems. arXiv preprint arXiv:1911.11034.

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, 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., 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.

Liao, G., Su, Y., Ziani, J., Wierman, A., Huang, J., 2022. The privacy paradox and optimal bias-variance trade-offs in data acquisition. ACM SIGMETRICS Performance Evaluation Review 49, 6–8.

Lin, S., Zhang, J., 2019. Generalization bounds for convolutional neural networks. arXiv preprint arXiv:1910.01487.

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.

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.

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 Behaviour 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.

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.

Montiel Olea, J.L., Rush, C., Velez, A., Wiesel, J., 2022. On the generalization error of norm penalty linear regression models. arXiv e-prints arXiv-2211.

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. Modelling 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 modelling, 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

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 Modelling of Scientific Applications 2018.

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 Behaviour 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.

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.

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 modelling 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.

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.

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 behaviour 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.

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, 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 modelling. 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.

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

Soares, E., Sizilio, G., Santos, J., da Costa, D.A., Kulesza, U., 2022. The effects of continuous integration on software development: a systematic literature review. Empirical Software Engineering 27, 78.

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 Modelling 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.

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.

Tang, L., Zhao, Z., Gong, X., Zeng, H., 2012. On the generalization of PAC-Bayes bound for SVM linear classifier, in: Contemporary Research on E-Business Technology and Strategy: International Conference, ICETS 2012, Tianjin, China, August 29-31, 2012, Revised Selected Papers. Springer, pp. 176–186.

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.

Tomalin, M., Byrne, B., Concannon, S., Saunders, D., Ullmann, S., 2021. The practical ethics of bias reduction in machine translation: Why domain adaptation is better than data debiasing. Ethics and Information Technology 1–15.

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.

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 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 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.

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.

Verhagen, M.D., 2021. Identifying and Improving Functional Form Complexity: A Machine Learning Framework (SocArXiv No. bka76). Center for Open Science. https://doi.org/10.31219/osf.io/bka76

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.

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 behaviour-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.

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, J.M., Harman, M., Ma, L., Liu, Y., 2020. Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering 48, 1–36.

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