D3.1 Specificability, explainability, traceability, and robustness proof-of-concept and argumentation

Version 1.0

Documentation Information

Contract Number	101069595
Project Website	www.safexplain.eu
Contractual Deadline	31.03.2024
Dissemination Level	PU
Nature	R
Author	Thanh Hai Bui (RISE), Maria Ulan (RISE), Robert Lowe (RISE)
Contributors	Axel Brando (BSC), Jokin Labaien (IKR), Jon Imaz (IKR), Gabriele Giordana (Aiko), Shruthi Gowda (NavInfo), Kobus Grobler (NavInfo)
Reviewer	Ana Adell (IKR)
Keywords	Artificial Intelligence, Explainable AI, Functional Safety

This project has received funding from the European Union's Horizon Europe programme under grant agreement number 101069595.

Change Log

Version	Description Change
V0.1	First draft
V0.2	Reviewed version
V1.0	Final version

Table of Contents

1. Introduction 8
2. Background 9
2.1. Explainable and trustworthy related Standards 9
2.2. SAFEXPLAIN AI Explainability concepts 12
2.2.1. Understandability 13
2.2.2. Comprehensibility 13
2.2.3. Explainability 13
2.2.4. Interpretability 13
2.3. Explainability literature review and categorization 14
2.3.1. Data explainability 14
2.3.2. Model explainability 16
2.4. XAI candidates for use within SAFEXPLAIN 21
2.4.1. Data explainers 21
2.4.2. Model explainers 26
2.4.3. Intrinsic explainable DL models 35
2.4.4. Metaheuristic search 36
3. Specification and Design of FUSA-aware DL solutions 37
3.1. Requirement Engineering for AI 38
3.2. Uncertainty challenges of DL components 39
3.2.1. Data failures (Domain uncertainty) 40
3.2.2. Model failures (Epistemic uncertainty) 41
3.2.3. Conditional occlusion failures (Aleatoric uncertainty) 41
3.2.4. Failure modes 42
3.2.5. DL related risk assessment aspects 43
3.3. AI-FSM aware DL solutions 43
3.3.1. Focused scope 44
3.3.2. XAI usages in “Ph1-DL-related concept specification” 45
3.3.3. XAI usages in “Ph2-DL requirements specification” 45
3.3.4. XAI usages in “PhDM Data management” 46
3.3.5. XAI usages in “PhLM Learning management” 48
3.3.6. XAI usages in “PhIM Inference management” 51
3.4. XAI usages in Operation and Monitoring stage 52
3.5. Relevant metrics 53
3.5.1. Explainability KPI and metrics 53
3.5.2. DL component metrics 53

3.5.3. Dataset metrics 56
3.5.4. Supervisor performance metrics 57
3.6. Structural coverage of DNN 58
3.6.1. Structural coverage of traditional software 58
3.6.2. Structural coverage of DNN 58
4. Improved DL component robustness and online monitorability 59
4.1. Supervisory monitor algorithms 61
4.1.1. Out of Distribution detection 61
4.1.2. Uncertainty Estimators 62
4.1.3. Surrogate models 64
4.2. Decision function 65
4.3. Potential supports for supervision function 66
4.4. Integration with SAFEXPLAIN deployment platform 66
References 67

List of figures

Figure 1: Elements of an ML system (ISO/IEC 23053) 10
Figure 2: Data prototypes extracted from a dataset of snowy road conditions 22
Figure 3: Extracted prototypes and critisms by ProtoDash (courtesy [29]) 22
Figure 4: WADS Point cloud data profiling report using yData 23
Figure 5: Visualizations of data annotation distributions (subset of PASCAL_VOC test dataset) 24
Figure 6: Feature interactions between sepal length and sepal width (IRIS dataset) 25
Figure 7: t-SNE 2D plot of MNIST digit image dataset, color dots representing different digits 25
Figure 8: ProtoPNet example (courtesy [106]) 26
Figure 9: ProtoTree example (courtesy [107]) 27
Figure 10: ALE plots for 2 trained models (Logistic Regression and Gradient Boosting) on IRIS dataset, selected input features are petal length and petal width 27
Figure 11: PDP (Orange) and ICE (blue) plots of the dependence of bike rental number predictions on temperature and humidity 28
Figure 12: Permutation Feature Importance plot, Random Forest model, IRIS dataset 29
Figure 13: LIME provides local explanations as simple linear classification approximates. 29
Figure 14: LIME explanation of a dog image. Image is segmented by superpixel regions. The yellow region represents the most influencing region of the model prediction (of “dog”) 30
Figure 15: SHAP explanation of a dog image 30
Figure 16: Illustration of the LRP procedure (courtesy of [107]) 31
Figure 17: CAM maps overlaying on input image, VGG16 CNN classification 32
Figure 18: EigenCAM applied on YOLOV8 revealing where the model is focusing when detecting a train and some cars. 33
Figure 19: Anchors explainers applied for InceptionV3 model trained on ImageNet. From left to right: Original image, anchors superpixel segments (slic function) and segment that constitutes the anchor 33
Figure 20: CLIP-Dissect and MILAN results on ResNet-50 explanations 34
Figure 21: CLIP-Dissect and MILAN results 34
Figure 22: probabilistic disentanglement of the uncertainty sources for a supervised learning system (i.e. the predictive AI-based system). Note that only the last two types of uncertainty depend on the selected predictive system or model. 39
Figure 23: DL uncertainties 40
Figure 24: Alternative evolution of the scenario categories resulting from ISO21448 activities [REF] 44
Figure 25: DL requirement argument pattern 46
Figure 26: Data argument pattern 47
Figure 27: Model dependency. 48
Figure 28: Heatmaps showing the test scores w.r.t. model hyperparameter settings (a simple SVC model with 2 hyperparameters gamma and C). Methods used from left to right: Successive Halving, Random Search and GridSearch in finding hyperparameters 50
Figure 29: Model performance vs input parameters plot. The plot shows number of detected objects (measured by if IoU>0) by input parameters: time of day and weather conditions 51
Figure 30: SAFEXPLAIN reference safety architecture pattern (D2.2) 60
Figure 31: Multiple supervisors architecture to support L1DM 60
Figure 32: Variational AutoEncoder as Anomaly detection supervisor. 62
Figure 33: Schematic view of three different uncertainty models [161] 63
Figure 34: Detection of body parts and person by PASCAL-VOC YOLOv7 64
Figure 35: Decision Tree surrogate model on person detection w.r.t. bodypart detections (trained to approximate PASCAL-VOC YOLOv7 65

Figure 36: Use soft voting as ensemble method to combine predictions from 3 DL models (Logistic regression, SVC, DT), dataset used MNIST..... 66

Acronyms and Abbreviations

- ADAM - Adaptive Moment Estimation
- AI – Artificial Intelligence
- AI-FSM – Artificial Intelligence Functional Safety Management
- CNN – Convolutional Neural Network
- CAM – Class Activation Map
- DL – Deep Learning
- DNN – Deep Neural Network
- DT – Decision Tree
- EDA - Exploratory Data Analysis
- FUSA – Functional Safety
- GLM – General Linear Model
- GMM – Gaussian Mixture Model
- IoU – Intersection over Union
- L1 norm – Manhattan distance
- L2 norm – Euclidean distance
- ML – Machine Learning
- MLP – Multilayer Perceptron
- MSE – Mean Squared Error
- OF – Optical Flow
- OM – Operation and Monitoring stage
- ODD – Operational Design Domain
- OOD – Out of Distribution
- PCA – Principal Component Analysis
- PhDM – Data Management phase
- PhLM – Learning Management phase
- PhIM – Inference Management phase
- RF – Random Forest
- ReLU – Rectified Linear Unit
- SGD – Stochastic Gradient Descent
- V&V – Verification and Validation
- XAI – Explainable AI
- YOLO – You Only Look Once

Executive Summary

This document reports the interim results (as of M18) of SAFEXPLAIN's FUSA-aware DL solution within the scope of WP3-Deep Learning. The work has been performed in alignment with WP2 (AI-FSM lifecycle, reported in D2.2 and reference architecture patterns, reported in D2.1). While explainable Artificial Intelligence (XAI) is a fast-growing area of research, a systematic approach of how one can use it to support FUSA compliant (in this specific context, it refers to AI-FSM compliance) solutions is lacking. The document starts with defining project-relevant concepts, gathering relevant standards/guidelines, best practices and state of the art XAI techniques. Gap analysis activities have been performed (continuously and interactively together with WP2) to determine where, how, and which available XAI techniques can be utilized to support the engineering process of dependable DL component as well as the required techniques/algorithms to support the architecture patterns in Operation and Monitoring stage.

Noting that it is an active and ongoing core activity of the project, the content of the document should not be considered complete. Instead, it captures the current understandings, which are constantly evolving (and converging) at a rapid pace through collaboration and exchange of ideas among researchers from different disciplines. Despite diligent efforts to maintain alignment, we anticipate discrepancies and gaps in understanding at this stage. Indeed, these will serve as good motivation for subsequent gap closing activities, ultimately leading to a final, comprehensive deliverable at the end of the project.

Existing relevant standards are also in a rapid development phase, facilitating knowledge sharing through various collaboration channels. This ensures that the work remains up-to-date and relevant to the community.

1. Introduction

To be able to deploy in safety critical systems, the Deep Learning-based (DL-based) software functions need to follow the specifications as defined in AI-FSM safety lifecycle (documented in D2.1[1]), where explainability and traceability of requirements are mandated.

This document reports the interim results of tasks "*T3.1 - Specification of dependable DL components*", "*T3.2 - Design of dependable DL components*" and "*T3.3 - Improved DL component robustness and online monitorability*" within the SAFEXPLAIN project. The aim is to report on our exploratory research into how existing approaches, with focus on explainability and trustworthiness, addresses different challenges of DL component (model) engineering and compliance with AI-FSM lifecycle specification (D2.1). The proposed approach will devise explainability by design (either intrinsically by modified DL architecture or extrinsically via external Explainable AI supporting tools). The traceability is managed within the AI-FSM via *AI_Traceability_Matrix* and related argument patterns.

This document is organized as follows:

- Section 2 describes SAFEXPLAIN defined relevant concepts related to AI explainability, state of the art reviews of relevant standards and explainable AI techniques. The section also describes several representative techniques that have been preliminarily evaluated to validate the potential usage assumptions, i.e. support building and operating AI-FSM aware DL solutions (discussed in the subsequent sections).
- Section 3 describes a systematic approach to leverage these Explainable AI techniques to support the specification and architecture of AI-FSM aware DL solutions, in compliance with AI-FSM development lifecycle[1] as well as safety architecture patterns (D2.2[2]) in Operation and Monitoring stage (OM stage).
- Section 4 focuses on the related algorithms to support safety architecture patterns in Operation and Monitoring stage. The algorithms consider the "known" area as the safe operations of DL solutions specified within AI-FSM lifecycle and enable the supervision components in OM to manage the risks resulted from the system operating outside of its known/certified areas or if the residual uncertainty may violate control requirements.

The following terminology are used within the document (in alignment with D2.1):

- AI/ML/DL: refers to Artificial Intelligence, Machine Learning and Deep Learning respectively. Whereas AI is the broadest concept, enabling machines to mimic human behaviours, ML is a subset of AL where the system can learn from the data with statistical methods, and DL is a subset of ML that uses multiple-layered neural networks to analyse large datasets and learn from them.
- DL algorithm: Architecture design of a DL module, including all required building blocks, such as layer definition and declaration, layer connectivity, optimizer, loss function and parameter definition.
- DL model: Refer to the trained DL model, i.e. together with trained parameters (model weights)
- Dataset: Collection of data points, together with relevant annotations (labels), that can be used for different purposes: training, validation and testing of a DL algorithm/model. The terms also be used for production data during the OM stage.
- Learning type[3]: Supervised learning, Unsupervised learning, Semi-supervised learning, Reinforcement learning, Transfer learning
- DL tasks[4]: Regression, Classification, Clustering, Anomaly detection, Dimensionality reduction, generative, others.
- Traceability: DL traceability should be ensured traceability including in relation to datasets, processes and decision made during the lifecycle to enable analysis of the DL outcomes and responses to the context and consistent with the state of the art.

- Phases/stages: Refers to (i) AI-FSM compliant development lifecycle phases/stages, and (ii) Operation and Monitoring (OM) stage.
- Explainers: Explainable AI (XAI) techniques that are used to extract explanations from a dataset or a trained DL model. The explainers are of two main categories: data explainers and model explainers.
- DL component: The terms DL component within this document is used to refer to a specified DL model (together with all required specification artifacts and explainers).

While various technical variations are mentioned, the focuses of this deliverable are:

- Tasks of target DL models: Object detection, semantic segmentation (special subclass of classification).
- Algorithms of target DL models: CNN, MLP.
- Learning type of target DL models: Supervised learning.
- Datasets: (Timeseries) image datasets, synthetic datasets.

2. Background

2.1. Explainable and trustworthy related Standards

This section presents a review of several existing standards relevant to Explainable and Trustworthy AI.

ISO/IEC TR 5469:2024[5] (Artificial intelligence Functional safety and AI systems) is recently published, its guidelines are recognized as relevant for SAFEXPLAIN and have been incorporated into the AI-FSM lifecycle (D2.1), reference architecture (D2.2) and the AI-FSM compliant design of DL component within the scope of this deliverable.

ISO 21448[6] focuses on the safety of the intended functionality (often known as SOTIF). The standard address non failure related risks from functional insufficiencies or potential misuse of the human. The standard is for road vehicles' systems and considered as complementing Road vehicles FUSA standard ISO 26262[7], however its guidance is applicable also for other domains and have been as inspiration for the Verification and Validation (V&V) strategy within SAFEXPLAIN project.

ISO/IEC 22989[3] (Artificial intelligence concepts and terminology) establishes the terminology for AI and describe the concepts in the field of AI. It defines trustworthiness via several concepts:

- AI robustness: Ability to maintain level of performance under any circumstances. Within the scope of this document, this is reflected in the DL robustness assessments via correlation between the expected performances against input variations (ODD, operational scenarios parameters and input disturbances)
- AI reliability: Ability of a system to perform its required functions under stated conditions. The reliability aspect is addressed throughout this document via assessment and assurance of all required properties with regards to the defined ODD/operational scenarios.
- AI resilience: Ability to recover operational condition quickly following an accident. Resiliency of the FUSA-aware DL design is provided by redundancy and other monitoring mechanisms in operational safe architecture.
- AI controllability: Property of an AI system that an external agent can intervene in its functioning. This is realized in the safe architecture.
- AI explainability: the important factors influencing a decision can be expressed in a way that humans can understand. This is the key topic of FUSA-aware DL specification and design. Explainability techniques are used in all applicable steps within AI-FSM lifecycle and also being used to support "*L1 Diagnostic and monitoring mechanisms*" within *Supervision Components* in

Operation and Monitoring stage (reference safety architecture pattern for safe AI-based system, D2.2[2]).

- AI predictability: property of an AI system that enables reliable assumptions by stakeholders about the output. This is realized within this work by the usage proposals of surrogate models both in the AI-FSM phases and in OM stage.
- AI transparency: informing stakeholders about the detail and appropriate information. This is realized via the application of data explainers and model explainers throughout all different phases with involved stakeholders.
- AI bias/fairness: refers to the idea that different cases call for different treatment. Unwanted bias may result in unexpected behaviours of the system. Within this document, the bias will be controlled via assessments of data balance properties, V&V strategy and related XAI techniques to find interactions between expected model performance against different dimension of input scenarios.

ISO/IEC 23053[4] (Framework for AI Systems Using ML) defines system components and proposes the reference elements of an ML system as in Figure 1. Related terminologies are used throughout this deliverable.

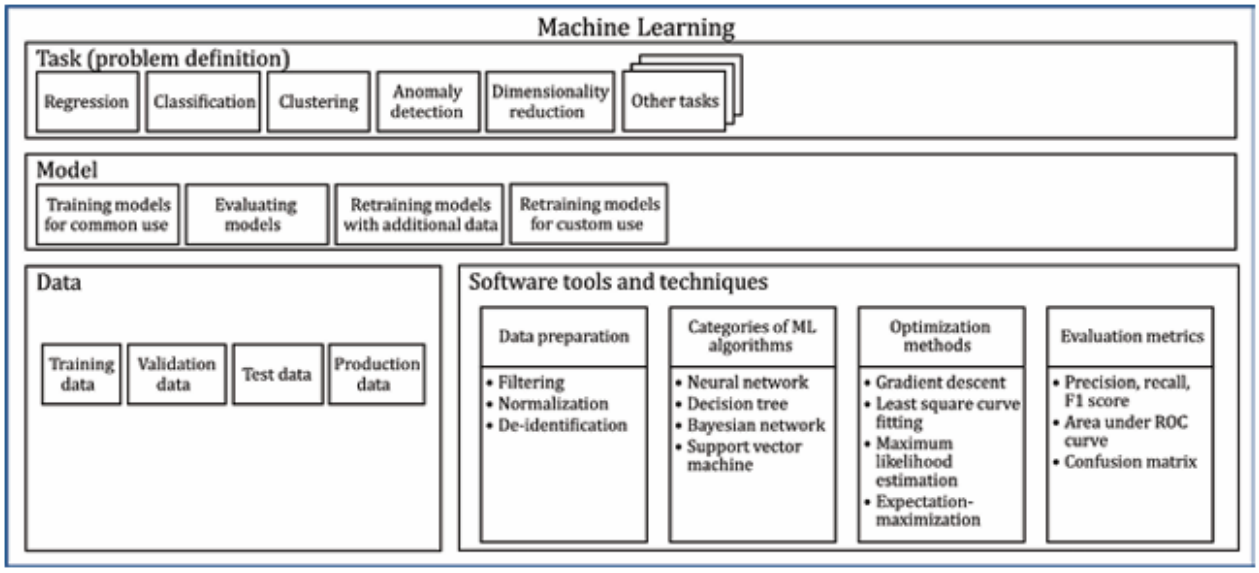

Figure 1: Elements of an ML system (ISO/IEC 23053)

IEEE P7001[8] "Transparency of Autonomous Systems" aims to set out 'measurable, testable levels of transparency, so that autonomous systems can be objectively assessed and levels of compliance determined'. The standard classifies three target audience:

- Designers: Need to self assess the transparency of their DL-based system
- Expert stakeholders: Require transparency as part of their work, including certification/regulatory bodies, incident/accident investigators and other expert advisors in administrative actions or litigation.
- End users and society: Benefit directly from the increased transparency.

ISO/IEC TR 24027:2021[9] (Information technology — Artificial intelligence (AI) — Bias in AI systems and AI aided decision making). This technical report addresses bias and fairness related to AI-based decision making. ISO/IEC TR 24027 defines bias and fairness as follows:

- Bias: systematic difference in the treatment of certain objects, people, or groups in comparison to others. The bias can be sourced from Human cognitive, data or engineering decisions. From the AI-FSM process and artifact perspective, the biases can be seen as follows:
  - Requirement bias: Human cognitive bias, omission of important characteristics, contextual assumption. Bias treatment strategy: Involve relevant stakeholders during the requirement specification steps.
  - Data bias: Non-representative sampling, label's definition and generation, omission of important data characteristics, unintentional changes during pre-/post-processing, redundant encodings, historical and societal biases, data collection biases. Bias treatment strategy: Measure bias/fairness, evaluate annotations and annotation process, identify sources of bias. This type of biases is addressed within the scope of PhDM phase and with the help of Data explainers.
  - DL Model bias: Manual designed model features, informativeness bias between different input data grouped by labels, interaction between models, expressiveness. Bias treatment strategy: Testing with different test scenario settings, evaluation and measurements, tuning.
- Fairness: A concept that is distinct from but related to bias. It can be characterized by the effects of an AI system on individuals, groups of people, organizations, and societies that the system influences.

Recommended bias/fairness assessments include the following metrics: confusion matrix.

ISO/IEC TR 24028:2023[10] (Information technology – Artificial intelligence – Overview of trustworthiness in Artificial Intelligence). The technical report suggests that explainability is an important component of a transparent AI system. Explanations shall be categorized by the aims, context, stakeholders needs, types of understanding sought and mode of explanation.

Aims of explanations: (i) a causal understanding of how a result is arrived at; (ii) an epistemic understanding of the knowledge on which the result is based; or (iii) a justificatory understanding of the grounds in which the result is offered as being valid.

ISO/IEC TR 24029-1:2021[11] (Artificial Intelligence (AI) — Assessment of the robustness of neural networks) is a technical report providing background information about existing methods to assess the robustness of AI models: (i) statistical methods, (ii) formal methods, and (iii) empirical methods. The report also provides robustness metrics available for statistical methods, which inspires the metric selection in Section 3.5.2 DL component metrics.

The National Institute of Standards and Technology (NIST) have produced a report NISTIR 8312 [12] describing four key facets of explainability in relation to AI systems:

1. **Explanation**: A system delivers or contains accompanying evidence or reason(s) for its outputs and/or processes.
2. **Meaningful**: A system provides explanations that are understandable to the intended consumer(s).
3. **Explanation Accuracy**: An explanation correctly reflects the reason for generating the output and/or accurately reflects the system's process.
4. **Knowledge Limits**: A system only operates under conditions for which it was designed and when it reaches sufficient confidence in its output.

Moreover, it is concluded that an AI system may be better able to meet system requirements if it utilizes more than one type of explanation (both for the same user type, e.g. safety standards certifier, operational user, and for different users). ***Explanation*** is constituted by the three remaining properties 2-4.

NISTIR 8312[12] also provides considerations for Risk management with regards to potentially revealing of proprietary information. This trade-off is detailed with reference to the four above-mentioned facets of explainability.

1. *Risk explanation*: detailed (e.g. traceable) explanations can potentially reveal proprietary details, i.e. trade secrets (two-way interaction might even expose information about the operational user). A trade-off, therefore, resides in how many explanations (XAI approaches) to use – more explanations increase explainability, but at the increasing cost/risk of exposing proprietary information. Nevertheless, explanations, at some level, are necessary for compliance to regulations such as given by the Fair Credit Reporting Act (FCRA) and GDPR (Article 13).
2. *Risk meaningful*: similar to Risk "explanation", more meaningful explanations allow for greater ability to meet standards or regulatory authorities but at the cost/risk of exposing proprietary details.
3. *Risk accuracy*: if explanations are inaccurate representations of the trained model (model risk), they are not only not useful but potentially dangerous explanations (if relied upon). The Federal Reserve System[13] note that if the model is erroneous and the explainer is erroneous this may lead to poor decision making (developer, user). If the explainer is inaccurate, we cannot know whether an output is erroneous, or an explanation is erroneous.
4. *Risk knowledge limits:* As above, if the limits of the systems are well clarified, the inner working (proprietary information) may be exposed.

It has been pointed out ([14]) that the use of counterfactual explanations (finding conditions where the model prediction qualitatively changes) can be sufficient for explanations and do not necessarily reveal the inner workings of the algorithm so can protect proprietary details.

NISTIR 8312 also refers to the accuracy-explainability trade-off in relation to categorizing explainable AI methods as intrinsic (built in self-explainability) or post hoc (using separate XAI algorithms).

2.2. SAFEXPLAIN AI Explainability concepts

Various explainability concepts are used in the literature and related works, however the related definitions and usages are sometimes ambiguous. Within SAFEXPLAIN, we decided to study the most common usages of the concepts and define here below the relevant explainability concepts that will be used throughout the project to avoid confusion.

In general, for a particular audience and task performed by a DL model, explainability is the ability to generate human understandable reasons for the model predictions along with the internal workings of the system. Explainability concepts are thus subjective and dependent on the target audience. In SAFEXPLAIN, the key target audience are AI developer, data analysts, domain and FUSA experts. Extracted AI explanations are also used by machine audience in the Operation and Monitoring stage (in the form of runtime monitoring using extracted explanations).

Within SAFEXPLAIN context, explainability refers to the availability of a process and its methods to provide the sequence of mappings from a blackbox's (DL components) behaviours to the knowledge level that is acceptable by a human being (an audience). Noted that sometimes the explanations are also used for another module (machine) as its audience.

The complexity level of the chosen process and the acceptance level of its output (explanations) will thus define the explainability level of the target DL components. During the project, evaluation criteria and related metrics will be developed to measure explainability level. Initial ideas of such evaluation criteria are discussed in Section 3.5 Relevant metrics.

2.2.1. Understandability

Understandability (or equivalently, intelligibility) denotes the characteristic of a DL model to make a human understand its function – how the model works – so that it can be conceived in a generic way the process by which the model generates the output from the input, without any need for further explaining its internal structure or the algorithmic means by which the model processes data internally. Since it is subjective and dependent on human understanding, the provided examples are sensitive to personal perspective: different human audience may require different level of explanations of the same example model.

Note that the level of understandability depends on the complexity of the model for human to follow, i.e. understandability is related to the number and type of associations between the model structure and their input/output values. Some examples of Understandability metrics can be found in [15], e.g. objectively measure the time consumed by test persons to complete a task related to understanding how the model deriving the output, and use that to validate hypotheses if a proposed quantitative metric has significant impact on understandability.

2.2.2. Comprehensibility

Comprehensibility refers to the ability of a DL algorithm to represent its learned knowledge in a human understandable fashion. This notion of model comprehensibility stems from the postulates of Michalski [16], which stated that "*the results of computer induction should be symbolic descriptions of given entities, semantically and structurally similar to those a human expert might produce observing the same entities. Components of these descriptions should be comprehensible as single 'chunks' of information, directly interpretable in natural language, and should relate quantitative and qualitative concepts in an integrated fashion*". Given its difficult quantification, comprehensibility is normally tied to the evaluation of the model complexity.

2.2.3. Explainability

Refers to the ability to provide understandable explanations of a DL model global behaviour or a specific prediction (local behaviour) via using extra supporting tools/techniques (referred to as explainers). Examples of the explanations provided by the explainers can be human understandable rules, or important features in the data, etc.

2.2.4. Interpretability

Refers to a passive characteristic of a model referring to the level at which a given model makes sense for a human observer. This feature is also expressed as transparency. Models belonging to this category can be also approached in terms of the domain in which they are interpretable, namely, algorithmic transparency, decomposability and simulatability. Each of these classes contains its predecessors:

- Simulatability: denotes the ability of a model of being simulated or thought about strictly by a human, hence complexity takes a dominant place in this class. Again, endowing a decomposable model with simulatability requires that the model must be self-contained enough for a human to think and reason about it as a whole.
- Decomposability: stands for the ability to explain each of the parts of a model (input, parameter, and calculation). It can be considered as intelligibility. The added constraint for an algorithmically transparent model to become decomposable is that every part of the model must be understandable by a human without the need for additional tools.
- Algorithmic Transparency: can be seen in different ways. It deals with the ability of the user to understand the process followed by the model to produce any given output from its input data.

The main constraint for algorithmically transparent models is that the model must be fully explorable by means of mathematical analysis and methods.

2.3. Explainability literature review and categorization

Various methods have been established to address different aspects of explainability. Categorization can be organized based on the subject they pertain to, the approach to transparency they employ, the form of representation they utilize, the scope of their application, the intended audience they serve, and the techniques they apply. The subject of explainability can refer to the data or model. Transparency can be characterized as simulatability (an understanding of the functioning of the method), decomposability (understanding of the individual components), and algorithmic transparency (visibility of the algorithms). Explanations can be represented in different forms such as text/tabular data, graphs and images. The scope of the explainability method can be local or global, whereas the audience for whom the explainability is tailored might include AI developers, safety experts, or end-users. Lastly, the technique employed can be intrinsic, which is integrated within the model, or post-hoc, which is applied after the model has been developed.

Table 1: Categories of Explainability methods

Category	**Applicability**
Subject	Data, model
Transparency approach	Decomposability, simulatability, algorithmic
Representation	Text/tabular, graph, image, numerical values
Scope	Local, global
Audience	AI developer, data analyst, domain expert, FUSA expert, end user, machine
Technique	Intrinsic, posthoc, antehoc

Algorithmic approaches for interpretability and explainability are required to be applied at different phases of the AI-FSM lifecycle including: Data Management – to assess required distributions and quality of data; Learning Management – to assess whether the selected model is optimal and optimized for the specific task(s) within the ODD/Operational scenarios; Inference Management – to assess whether the performance of the model can generalize, within defined safe boundaries, to making accurate predictions for real world data. The phases themselves are not independent as poor model performance may owe to the quality of the data as much as the design or optimization of the model itself, for example. Similarly, failure of the model to generalize well to the real-world use case (deployment) may owe to failings of the model or the dataset on which it was trained on or a combination thereof.

Algorithms designed to provide transparency and explainability to black box models also may provide levels of detail of explanation suitable for different audience such as developers, safety assurance certifiers, domain experts.

2.3.1. Data explainability

A standard approach to evaluate the quality of the datasets on which the black box DL models are trained or otherwise finetuned, is to undertake Exploratory Data Analysis (EDA). This can take the form of simply evaluating whether there is a distribution skew of data instances per class of data, or missing/corrupt data

instances, or can evaluate the features inherent in the data to assess potential (feature explanation). Here the task is to understand the feature (dimension/subspace) either from a domain perspective (expert knowledge via EDA) or from a model perspective.

Many tools exist for undertaking a number of EDA based analyses of data, e.g. pandas profiling (now referred to as yData profiling[17]), SweetViz[18], Google Facets[19]. EDA tools provide a list of statistical characteristics of a dataset. Data visualizations are also a powerful tools to support data analysts: Parallel Coordinate Plots (PCP), PCA, t-SNE[20], UMAP[21]

Data Readiness Levels (DRL[22]) are an ongoing development of a systematic framework used to assess the quality and preparedness of data for making informed decisions, similar to Technology Readiness Levels used in evaluating technology maturity. This framework helps in evaluating the usability, reliability, and completeness of data across different stages, from initial collection to its application in real-world scenarios. DRLs provide organizations with a structured approach to identify gaps, improve data management practices, and ensure that the data is fit for its intended use.

A particular subset of data explainabilty techniques is feature extraction, encompassing both domain-specific and model-based approaches. Domain-specific methods leverage expertise in the domain and insights from EDA to extract and identify significant features. For instance, a binary classifier distinguishes pixels that appear similar but denote different entities[23]. On the other hand, model-based feature engineering employs various mathematical models to ascertain the inherent structure of a dataset. Clustering and dictionary learning serve as examples of such model-based methods[24].

Implementing standards for documentation could bridge the communication gap between those who create and use datasets, potentially mitigating these issues. Several proposals have been introduced to encourage standardized dataset documentation, including Datasheets for Datasets[25], Dataset Nutrition Labels[26], and Data Declarations for Natural Language Processing[27]. These initiatives propose different frameworks for documenting essential information about datasets, such as their development history, contents, collection methods, and any legal or ethical concerns.

Techniques for summarizing data are frequently employed when maintaining the entire training dataset is costly or not feasible. One of these methods is prototype selection[28]. To summarize a dataset, one can identify a limited set of representative samples, referred to as prototypes, which provide a quick overview of the wider dataset. ProtoDash[29] selects representative and diverse prototypes from large datasets, effectively capturing the essence and variety within the data. It optimizes for relevance and diversity, ensuring the selected prototypes are both indicative of the overall data distribution and sufficiently distinct from each other. This method is particularly useful for data summarization, visualization, and enhancing model explainability by providing a compact, insightful subset of the original dataset. Contrastive explanations method (CEM)[30] uses unsupervised network to learn from datasets and provide data explanation (e.g. using the provided prototype patterns) for dataset or specific data point.

Variational Autoencoders (VAEs) are used where models that can generate new data points similar to the training data. The latent space in VAEs, being a lower-dimensional representation of the data, can serve as a compact and interpretable summary of the dataset's underlying structure. In other words, VAE can reconstruct the input features from the output prediction, and Latent distribution at the bottleneck provides a visual explanation (multivariate normal distribution of feature representation) [31]. Disentangled Inferred Prior Variational Autoencoder (DIPVAE)[32] learns high-level independent features from images that possibly have semantic interpretation.

2.3.2. Model explainability

Even when the data has been cleaned and readied for training with explainability techniques as outlined in the previous section, the training process can remain difficult. Model explainability refers to the techniques to enhance explainability of DL models either by intrinsic (interpretable models by design), post-hoc (using model explainers) or antehoc (hybrid approach, where explainers are incorporated into the model designs to extract required explanations for some important subpart of it).

2.3.2.1. Intrinsic

Intrinsic explainability focuses on designing/developing models that are inherently easier to comprehend. We sometimes refer to models applying this concept as self-explaining models or interpretable models.

One strategy for constructing explainable models involves choosing from a group of inherently interpretable or "white-box" modelling techniques. A fundamental method within this category is the linear regression model, which estimates the target by calculating a weighted sum of the input features[33]. To incorporate possible interactions between features, generalized linear models are used[34]. Decision tree models serve as another illustration, segmenting the data repeatedly based on specific threshold values within the features[35]. Decision sets[36], rule sets[37], k-nearest neighbours and Naive Bayes data mining algorithms[38] are also examples of intrinsic approaches.

By combining an inherently interpretable modelling approach with a blackbox method, it's possible to build a high-performing model that also offers explanations for its decisions. An illustration of this approach is the Self-Explaining Neural Networks[39]. This strategy aims to generalize a basic linear classifier by employing neural networks to identify relevant features, determine their weights, and outline how these elements contribute to the final decision. This involves the use of concept encoders, a mechanism for adjusting parameters based on input, and aggregators to describe three neural networks. The resulting hybrid model combines the clarity and simplicity of a linear model with the powerful, adaptable capabilities of a black-box model.

A model designed for explainability can be trained to justify its own predictions. The Teaching Explanations for Decisions (TED) framework[40] is utilized to enhance the training dataset by incorporating a set of features, the outcome, and the rationale behind that outcome, termed as an explanation, within each example. TED is a framework aimed at improving explainability by using explanations relevant to the specific domain included in the training dataset to predict both the labels and the explanations for new cases. It assimilates the explanations given by users for the input/output combinations and produces explanations for new input/output scenarios.

Modifying the structure of models can also enhance their explainability. For instance, in a convolutional neural network (CNN) it is possible to steer the representations produced by the filters in the higher layers to identify components of objects rather than merely recognizing patterns[41]. This is achieved by integrating a specific loss function into the feature maps of a standard CNN. This loss function gives preference to the identification of certain object parts within a specific class category, while it does not activate for images belonging to different classes. Importantly, this approach does not require any data annotations for object components. As a result, CNNs designed for explainability are capable of retaining more relevant information in their high-layer filters compared to those trained through traditional methods.

Regularization methods are commonly applied to enhance the predictive capabilities of AI models, and they can also contribute to explainability. An example of this is tree regularization[42], which aims to increase the explainability DNNs. The core idea behind this technique is to promote the development of a model whose decision-making process can be closely approximated by a small decision tree, enabling human interpretation of its predictions. This is achieved by incorporating a regularization term into the model's

training loss function. Employing this approach may result in models that retain their predictive accuracy while becoming more explainable.

It is worth to mention the methods using Teacher-Student architecture[43], [44]. It uses autoencoders to explain the important regions in a classified image. The model had two networks, one for encoding input image representations, and one for reconstructing an image with the same input image size. The model uses the reconstructed image to visualize important parts of the classified image by using a binary threshold.

2.3.2.2. Posthoc

Post-hoc explainability refers to techniques and methods used to explain the behaviour and decisions of a model after it has been trained. It involves the development of a second model, usually as a surrogate of the original model in order to provide users with explanations. These surrogate models are categorized into local or global types. A global surrogate model leverages the entire training dataset for its operation, while a local surrogate model focuses on approximation near a specific test input.

The Locally Interpretable Model-Agnostic Explainer (LIME)[45] functions as an intermediary model. It operates by creating synthetic data points derived from the original input, upon which it trains a straightforward machine learning model such as a decision tree or linear regression. This model is designed to replicate the outcomes of a more complex DL model. The predictions made by this simpler, more interpretable model are then utilized to mimic on how the original, "black-box" model arrives at its conclusions. Anchors[46] is a variant of LIME that identifies a decision rule to explain the individual predictions made by any black-box classification model. For generating local explanations of predictions, this method employs perturbations. The explanations are presented as simple IF-THEN rules, known as anchors, which differ from the surrogate models used by LIME.

Changing a pixel in the input image can lead to an increase (indicating a positive gradient) or a decrease (indicating a negative gradient) in the class's predicted probability. The magnitude of the pixel's alteration's impact is directly proportional to the absolute value of this gradient. This principle forms the basis of what are known as gradient-only methods. Several techniques are worth considering. Class Activation Map (CAM)[47] are saliency based visual explanations which provides a localization map using the weights of the global average pooling (GAP) layer added to standard CNNs. Gradient-weighted CAM (Grad-CAM)[48] provides for models within the CNN family, enabling explanations without requiring changes to the architecture or retraining. HiResCAM[49] is an improvement of Grad-CAM to highlight only the locations the model used to make each prediction. Guided Backpropagation[50] is a technique that visualizes the gradient with respect to images when backpropagating through the rectified linear unit (ReLU) activation function. Score-CAM[51] also incorporates gradient information, but perturbate the image by the scaled activations and measure how the output drops. Augmented Score-CAM[52] utilizes the image augmentation method employed in training convolutional neural networks. By using the input image, it produces a set of augmented images and generates a class activation map for each one. The final activation map is obtained by combining these augmented activation maps. Spatial Sensitive GRAD-CAM (SSGrad-CAM)[53] modifies the heatmap generated from Grad-CAM with space maps computed by normalizing the magnitude of gradients. It was applied to Single Shot Multibox Detector (SSD[54]) to incorporate spatial sensitivity and focus on the importance of both features and space. Occlusion sensitivity[55] describes importance of different regions in an input image for a given prediction. The basic idea is to systematically occlude different parts of the input and observe the impact on the model's prediction.

DeepLIFT (Deep Learning Important FeaTures)[56], is a method to decompose the output prediction of a neural network on a specific input. It achieves this by backpropagating the contributions of all neurons in the network to every feature of the input. DeepLIFT compares the activation of each neuron to its reference activation and assigns contribution scores according to the difference. These scores can be computed efficiently in a single backward pass.

Integrated Gradients[57], another method for attributing feature importance in the context of deep neural networks. This method measures the gradient of the model's prediction output with respect to each input feature, integrated along a straight path from a baseline input to the actual input. The baseline input is typically a neutral or zero input that represents the absence of all features. The path integral of the gradients effectively captures the significance of each feature in the transition from the baseline to the actual input.

Visualizations are widely used as techniques for explanation. A Partial Dependence Plot (PDP[58]) can interpret complex models by isolating and illustrating the impact of specific variables on predictions. PDPs show the effect of one or two features on the predicted outcome of a model, averaged over a dataset. This approach helps in understanding the relationship between the target response and the input features of interest, irrespective of the values of other features. Accumulated Local Effects (ALE)[59] is a method used to understand the effects of input features on the predictions of a model. Unlike global methods, which assess the overall importance of features, ALE focuses on how changes in feature values locally affect the prediction. This provides a more detailed view of the feature-prediction relationship by aggregating local gradients, thereby helping in identifying how features influence predictions across different regions of the feature space.

Example-based explanation methods provide interpretability by using specific instances from the data set to illustrate how a model makes its predictions. These methods make complex models more understandable by highlighting representative examples (prototype and criticisms) or showing minimal changes that alter predictions (counterfactuals), thereby offering intuitive insights into the model's decision-making process. The prototype and criticism method[60] identifies representative examples (prototypes) that encapsulate the main characteristics of a class and examples (criticisms) that highlight the model's limitations or areas where it might be less certain. Prototypes help users understand what the model considers a typical case, while criticisms point out exceptions or edge cases. A counterfactual explanation of an outcome or a situation takes the form “If X had not occurred, Y would not have occurred”[14]. Moreover, counterfactual explanations can be guided by prototypes[61].

Game theory methods, when applied to explainability, leverage concepts like Shapley values[62] to determine the importance of each feature in a predictive model's decision. This is done by assessing the impact of each feature on the model's output across all possible combinations of features. SHAP (SHapley Additive exPlanations)[63] provides a consistent and locally accurate method to attribute the impact of each feature, offering insights into how the model makes its decisions. The core idea behind SHAP is to explain the prediction of an instance by quantifying the contribution of each feature to the difference between the actual prediction and the baseline prediction (often the average prediction over the dataset). SHAP values ensure fairness (each feature's contribution is fairly allocated), additivity (the sum of the SHAP values equals the difference between the prediction and the baseline), and consistency (if a model changes so that a feature's contribution increases or stays the same, its SHAP value does not decrease). Gradient SHAP[64] is a variant of SHAP that specifically utilizes gradients to efficiently compute SHAP values for models where calculating exact Shapley values is computationally infeasible. By leveraging the gradient information available in differentiable models, Gradient SHAP efficiently approximates the contribution of each input feature to the model's prediction, offering a practical and scalable solution for explaining predictions of complex models like deep neural networks. It was applied as a pixel-wise feature attribution method to basis visualization of for YOLO (You Only Look Once[65]) object detection model of satellite images, in which texture is often complicated, and the target objects may be small.

Neural methods for explainability aim to help clarify the complex inner workings of neural networks, making their predictions more accessible and trustworthy. Layer-wise Relevance Propagation[66] (LRP) is a technique used to interpret the decisions of neural networks by backpropagating the prediction output back to the input layer, thereby attributing relevance scores to individual inputs. LRP generates heatmaps through finding the relevance of each neuron in the network (in relation to a layer-by-layer evaluation of

contribution of weighted inputs to other neurons) using a principle of conservation related to Kirchoff's electric circuit theory. Composite LRP[67] employs variations of the LRP algorithm tailored for specific layers within a network: certain adjustments are more effective for the later layers that are believed to represent components of objects, while other modifications are more suited to the initial or middle layers, which capture more basic features. Contrastive Relevance Propagation (CRP) [68] extends LRP to explain individually the bounding box and classification decisions of SSD. CRP for Localization Models (L-CRP)[69] is an extension of the CRP method, it was applied to YOLOv5 and YOLOv6 enabling local concept-based understanding of segmentation and object detection models. LRP-CRP method[70] builds upon the foundation of LRP by introducing a contrastive component. In addition to identifying the features that contribute to the prediction of a specific class, LRP-CRP also highlights features that argue against it, favouring alternative classes. This contrastive approach provides a more nuanced understanding of the model's decision process, showing not only why the model made a particular decision but also why it did not choose other options. It was applied to YOLOv5 to understanding the model's reasoning in a more detailed and contrastive way. Concept Relevance Propagation[71] builds upon LRP methods, extending them to not only identify important features but also to understand how these features relate to specific concepts or classes within the model's decision-making process. This approach helps in providing more interpretable explanations of model predictions by highlighting the relevance of input features in the context of the model's learned concepts.

Rules Extractors within the context of explainable methods refers to a class of techniques designed to extract simple, understandable rules from complex models to produce rough approximation of a predicted behaviour. DEXiRE[72] is a propositional rule extractor for binary classifier DL. It works by converting the activation values of a trained DL model into binary format (either 1s or 0s) to identify activation patterns, specifically focusing on the most active neuron in each hidden layer, from which it then extracts rules. Additionally, DEXiRE offers a version for multiclass classification tasks, which employs a decision tree to facilitate this process. ECLAIRE[73] is a polynomial-time decompositional algorithm able to scale to both large training sets and deep architectures. It analyses the network's structure and activations to generate a concise set of if-then rules that mimic the network's reasoning, making the model's decisions easier to understand and explain. RxTEN[74] relies on reverse engineering technique. It identifies the functionality of each input neuron by analysing the mistakes occurred at the removal of that neuron from the network. DeepRED[75] builds decision trees on the activations of the considered NN as the input and compose these trees, or rules derived from them, into more complex ones. Layer by layer it uses decision tree induction to generate a set of if-then rules, making the model's decisions understandable. It can be applied to multilayer perceptron's (MLPs[76]) of any depth. Two-step CNN rule extractor[77] uses two MLPs for extracting propositional rules from the feature extractor part of the discretized interpretable convolutional network, and the MLP head of the virtual discretized interpretable multi-layer perceptron. Deep Convolutional DNF Learner [78] utilizes binary neural networks to convert the inputs and outputs of neurons into binary form. This process is aimed at approximating the behaviour of a trained CNN to derive first-order logic rules.

It is possible to transfer information from a pre-trained deep neural network that has a high test accuracy to a simpler interpretable model or a very shallow network of low complexity and a priori low test accuracy. ProfWeight[79] uses a sophisticated DL model as a high-performing teacher which lessons can be used to teach the simple, interpretable, but generally low-performing student model. ModelSpeX[80] outlines a dynamic workflow for specifying deep neural network (DNN) models, engaging domain experts in the process. It incorporates explainable artificial intelligence (XAI) to analyse data and model relationships through visual analytics, utilizes XAI for extracting rules, and evaluates the model's performance in comparison to the DNN model. Additionally, it offers domain experts the ability to modify the specifications of the defined problem.

Understanding and modelling different forms of uncertainties in DL models for computer vision is essential for achieving more precise and dependable outcomes. Bayesian neural networks[81] offer a method to evaluate various uncertainties by replacing traditional network weights with probabilistic variables. It is

imperative to address both aleatoric and epistemic uncertainties to construct robust and reliable DL models in the realm of computer vision[82]. Aleatoric uncertainty is associated with the natural variability or noise present in the data, which remains unaffected even by an ideal model. On the other hand, epistemic uncertainty stems from the model's inadequacies in fully understanding the data's true distribution, highlighting the model's ignorance about certain facets of the problem at hand. Dropout, a technique employed to prevent overfitting in DNNs by intermittently excluding a fraction of the neurons during training, aids in fostering more generalized features within the network. This approach also serves as an approximation to Bayesian inference, enabling the model to gauge the uncertainty in its predictions by averaging the outcomes from several iterations of the DNN with dropout applied. This process improves the reliability of decision-making systems[83]. Studies have demonstrated a correlation between the measure of uncertainty and the accuracy of object detection[84]. Predictions that are marginally incorrect exhibit greater uncertainty compared to those that are more accurate, aligning with expectations from a reliable uncertainty estimation method. This insight suggests that uncertainty estimation in bounding box regression could enhance non-maximum suppression techniques by preferring boxes with lower variance in bounding box regression. Furthermore, the presence of aleatoric uncertainty has been linked to object occlusion, making it a useful indicator of inherent ambiguities in the data.

2.3.2.3. Antehoc

While intrinsic explainability techniques involve directly engineering DL model designs to enable them to implicitly learn concept-based explanations, and post-hoc explainability works with already trained models, there are hybrid approaches that modify pre-trained models by incorporating explanatory subcomponents and thus make them explainable, known as antehoc explainability.

Explainable models can modify the CNNs architecture to improve their interpretability. This modification can replace some CNN parts like layers and loss functions or add new components to the CNN network like attention layers, autoencoders, and deconvolutional layers.

Various types of attention mechanisms were incorporated into CNNs architecture. Global-and-local attention (GALA)[85] approach involves integrating attention layers into CNNs to generate attention activity maps. DomainNet[86] transforms pre-trained CNN and apply two attention levels for extracting object parts and features. Residual Attention[87] stacked attention modules and integrate with CNNs to generate attention-aware features. Unlike previous attention approaches, Loss-based attention[88] doesn't add attention layers to CNN. It removes max-pooling layer in CNN and added loss-based attention to identifying which parts of the image explain the CNN decision. D-Attn[89] uses text reviews to learn the features of users and items and predict their ratings. It adds attention layer before convolutional layer to learn local/global attentions for user reviews. ALL-CNN[50] replaces max-pooling layer with convolutional layer and increased stride to reduce dimensionality.

A different approach is to integrate CNNs architecture with other machine learning models. The Explainer model[90] enhanced the interpretability of pre-trained CNNs by integrating autoencoders, which decomposed and then reconstructed feature maps to highlight distinct object parts using interpretable filters. This approach resulted in more understandable feature maps and reduced localization instability compared to other CNNs, though this came with a trade-off in lower classification accuracy. The XCNN[91] model also utilized autoencoders within CNNs to identify regions of interest (ROI) in images. It comprised an autoencoder for creating interpretable heatmaps, which were then analysed by a CNN classifier. The effectiveness of XCNN's heatmaps was assessed both qualitatively, through class discrimination, and quantitatively, via object localization. Techniques such as LRP and Guided-Backpropagation highlighted the superior quality of these heatmaps. The Adaptive Deconvolutional (Adaptive DeConv) model[55] breaks down an image into its feature maps before reconstructing it. This approach incorporated the use of deconvolutional and max-pooling layers, and was later combined with a CNN classifier to identify objects. The reconstruction of images took place at either the intermediate or high levels of the CNN. Deep Fuzzy

Classifier (FCM)[92] utilizes fuzzy logic to categorize data points, incorporating a fuzzy classifier subsequent to the final convolutional layer. It employs fuzzy clustering alongside relevance feedback based Rocchio's algorithm on the feature map for the extraction of class representatives. The FCM model was capable of highlighting the importance of individual pixels in relation to the predicted class, offering saliency maps that were more comprehensible compared to those of conventional CNNs. One can also investigate image representations by inverting them using up-convolutional neural networks[93]. These networks implicitly learn natural image properties (priors) during training, allowing them to recover information like colour or brightness that might not be explicitly present in the original extracted features.

The "Network In Network" (NIN)[94] approach replaces traditional convolutional layers and linear filters with micro networks to enhance spatial invariance. The Class-Specific Gate (CSG)[95] method integrates a class-specific gate with filters in a CNN to allocate each filter to one or more specific classes. An alternative strategy involves altering the CNN's loss function to enhance interpretability. The Interpretable CNN[41] technique augments the feature map loss across all filters in the final convolutional layer, compelling each filter to capture unique object parts. Dynamic-K[96] deviates from the standard stochastic gradient descent, employing adaptive activation thresholding for CNN interpretation. The SAD/FAD[97] technique introduces spatial activation diversity (SAD) and feature activation diversity (FAD) loss functions to render the CNN more distinctive. The Forward-Backward Interaction[98] (FBI) method introduces an activation loss function for forward-backward interaction as a regularization strategy to refine CNN interpretability and mandates a triple-pass training to identify critical regions. The And-Or Graph (AOG)[99] framework employs a graphical model with And-Or graphs to semantically reorganize convolutional layer representations, effectively demystifying the "black box" by incorporating four additional layers: semantic part, part template, latent pattern, and CNN unit. The ProtoPNet[100] model utilizes a prototypical part network to dissect images for prototypical parts prior to final classification, introducing a prototype layer between the convolutional and fully connected layers. Lastly, ProtoPShare[101] aims to decrease the quantity of prototypes produced by ProtoPNet through a merge-pruning technique that facilitates prototype sharing across classes, entailing two phases: initial CNN training and prototype pruning.

Some methods focus on analysing existing models from the outside. DeepFool[102], for example, is an external technique that probes the model's decision boundaries to identify weaknesses. It generates adversarial examples to fool DL models. It employs a method of iteratively linearizing the classifier to create minimal disturbances that effectively alter classification outcomes. This approach enables a dependable evaluation of the classifiers' robustness. In contrast, ViT-NeT[103] is an interpretable architecture specifically designed for fine-grained image classification. It incorporates a neural tree decoder that mimics human-like decision-making, offering insights into the model's reasoning process.

2.4. XAI candidates for use within SAFEXPLAIN

This section discusses initial set of XAI algorithms as the candidates for usage within SAFEXPLAIN as a proof of concept. The algorithms have been through the first evaluation by the WP3 team and will gradually be enriched to the end of the project to provide complete guidelines and examples. The realizations of these algorithms into libraries are described in Deliverable D3.2.

2.4.1. Data explainers

2.4.1.1. Prototype selection

Prototype selections belong to a class of algorithms to extract a subset of representative instances (namely prototypes) from a dataset. For example, Figure 2 shows representative patches describing snowy road condition in an image dataset of different sizes (using kMean clustering algorithm and L2 distance). For image datasets, such extracted patches capture representative local patterns of the dataset, depending on the chosen patch size, distance metric, and clustering algorithms.

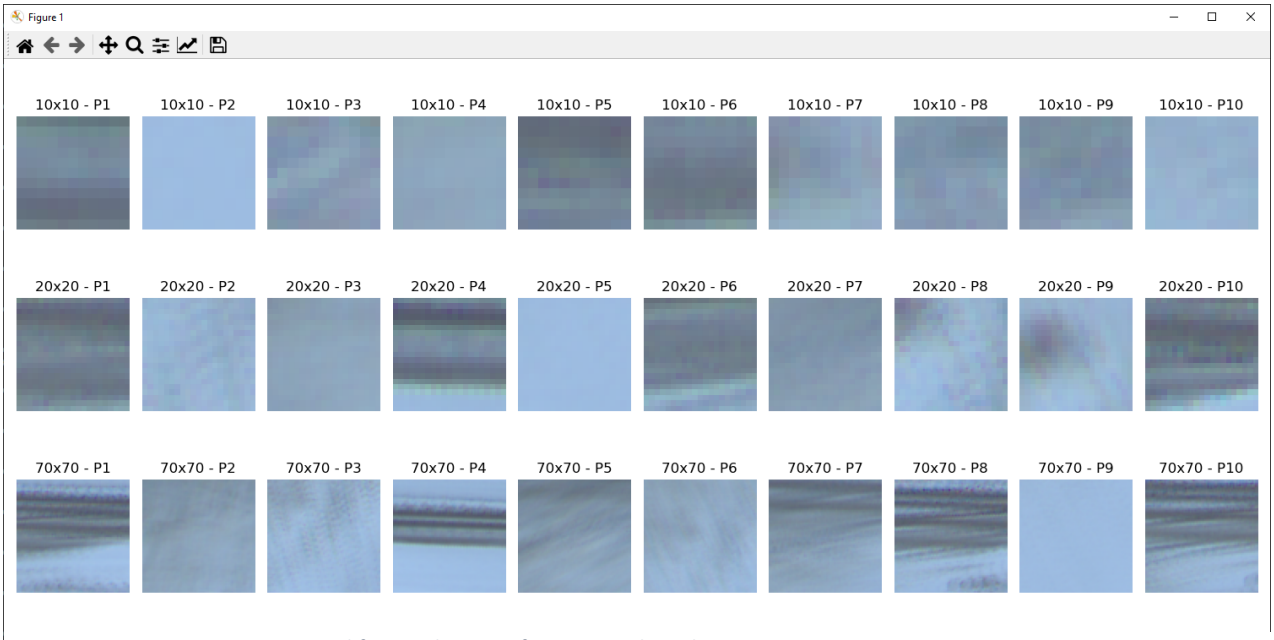

Figure 2: Data prototypes extracted from a dataset of snowy road conditions

ProtoDash[29] generates prototypes and criticisms to provide explanations. Prototypes illustrate the overall behavior of the dataset, whereas criticisms highlight instances that the prototypes do not adequately explain.

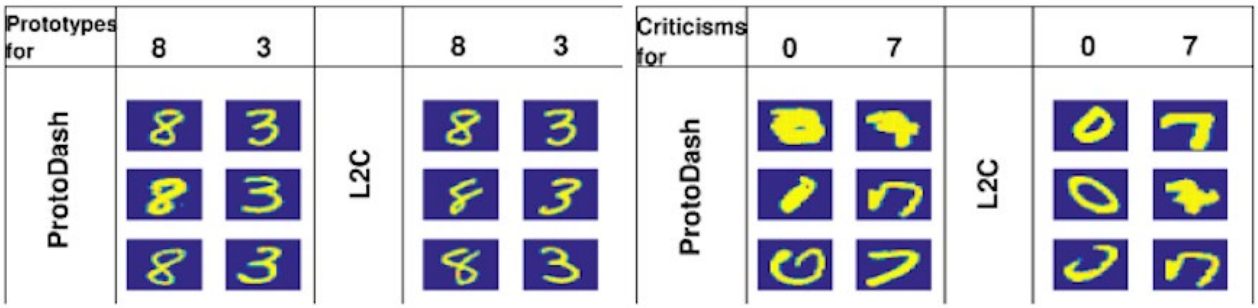

Figure 3: Extracted prototypes and critisms by ProtoDash (courtesy [29])

ProtoDash identifies a variety of prototypes (crititisms) that each offer a unique perspective on the dataset, ensuring a comprehensive understanding of its entirety. After pinpointing the first prototype (crititism), the algorithm proceeds to find a subsequent one. In this search, it aims to find a sample that exhibits similar while simultaneously uncovering novel patterns, ensuring this new prototype (crititism) differs from its predecessor. ProtoDash is designed to pinpoint examples within the training dataset that most accurately reflect the distribution of a given test input. Therefore, this algorithm offers a straightforward way to grasp the fundamental features influencing model's predictions.

Other prototype-based methods such as those described in Section 2.4.2.1 can also be used to extract representative prototypes from a dataset, from a model attention perspective.

2.4.1.2. Data profiling

Data profiling is the process to generate summaries of data. The following approaches are usually adopted:

- Assessment of data quality:

- Structure discovery:
  - Assessment of format of the data, consistency across the dataset, and other compliant requirements (resulted from PhDM).
  - Data completeness/balance: Identify distribution specification of dataset over a specific data dimension or lower dimensional subspaces. Examples include histograms, interactions...
  - Descriptive statistics such as min, max, standard deviation...
  - Data rule validation: Defining the expected data quality rules and validate the dataset versus the specified rules to ensure or measure compliance violation of data accuracy, consistency, integrity, and validity. Examples include Great Expectations[1]
- Content discovery:
  - Analyse systemic issues.
  - Data pattern: Identify patterns or prototyping instances representative of the dataset.
  - Domain analysis
- Relationship discovery:
  - Finding relationship between different dimensions of the dataset or between different subsets of data points
  - Timeseries data reconciliation by timestamp
  - Sensor fusion data: For example, data registration from lidar, radar, camera
  - Data redundancy: e.g. datasets captured from different sources (weather forecast, vehicle perception sensors, infrastructure sensors, synthetic data)

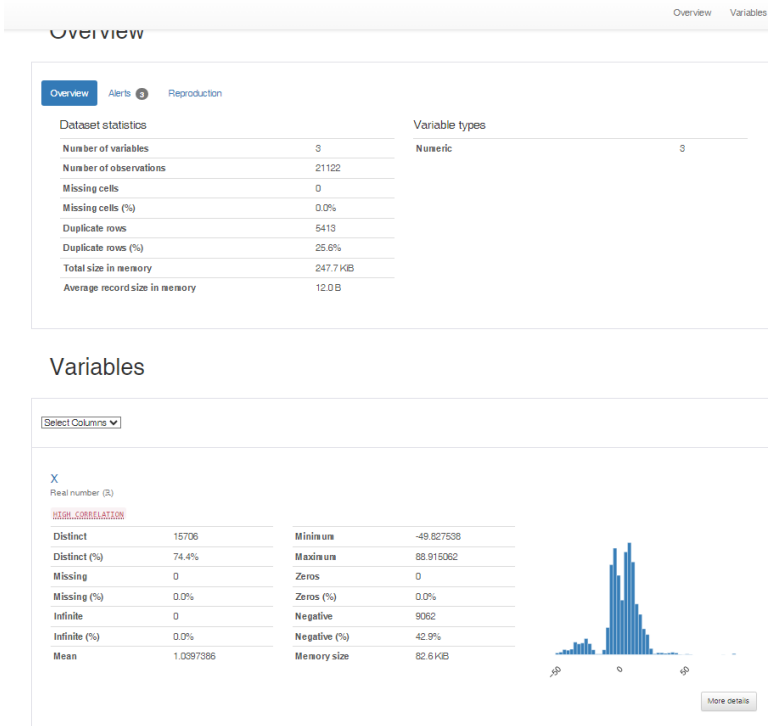

Figure 4: WADS Point cloud data profiling report using yData

[1] https://github.com/great-expectations/great_expectations

Several data profiling tools are available, however yData profiling (formerly Pandas profiling) has been selected within this first round of experiments.

Figure 4 illustrates a part of data summary report generated by yData profiling (formerly Pandas profiling) for a subset of WADS lidar point cloud dataset[104].

2.4.1.3. Variational Autoencoder

Variational Autoencoder [31] is a class of probabilistic generative models, consisting of a encoder network performing feature extraction and a decoder network that reconstruct from the extracted feature into the original dataspace. The model is optimized with two objectives: reconstruction quality (Euclidean distances) and data distribution quality (Kullback-Leibler divergence[105])

2.4.1.4. Data plots

2.4.1.4.1. Feature distributions

Different types of graphs can be used to visualize the distributions of annotations or input features of the dataset. Usually, one or two features are selected for the plots to ensure the human understandability.

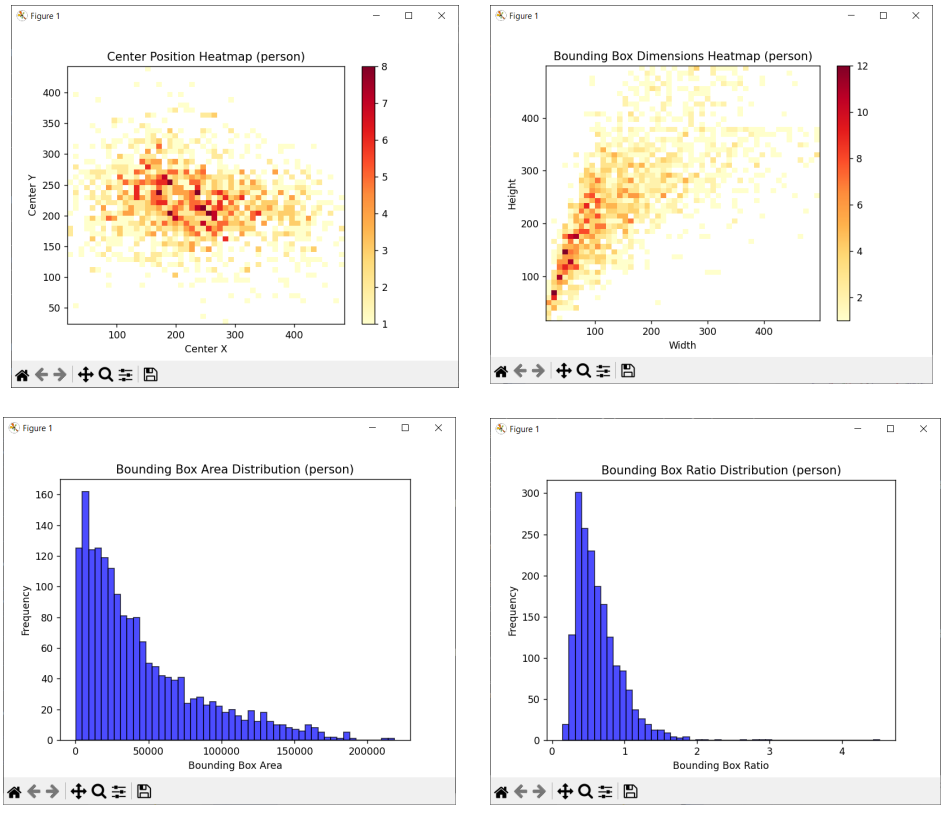

Figure 5: Visualizations of data annotation distributions (subset of PASCAL_VOC test dataset)

2.4.1.4.2. Feature interactions

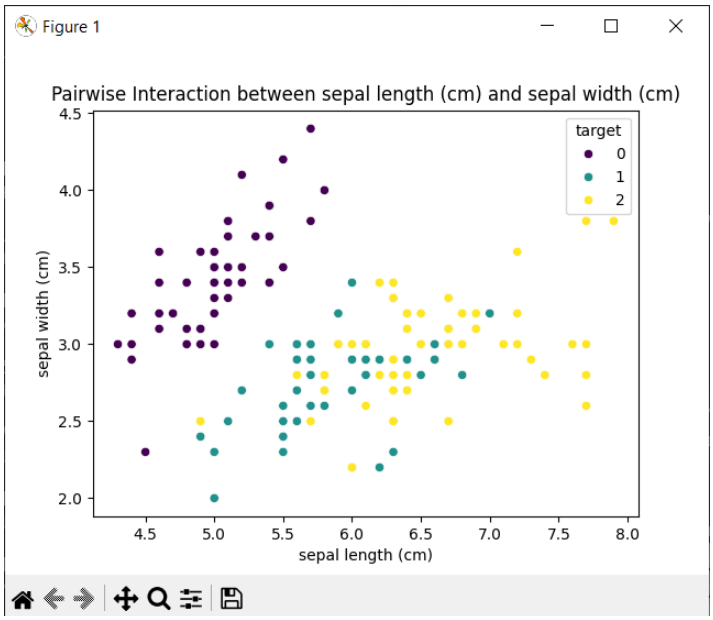

Figure 6: Feature interactions between sepal length and sepal width (IRIS dataset)

Feature interaction plots visualize the interaction between a pair of selected feature dimensions. For example, scatter plot can be used to describe the interaction between two features (dimensions) of the dataset (Figure 6).

2.4.1.4.3. t-SNE plot

t-SNE[20] proposed a variation of Stochastic Neighbor Embedding (SNE) to crowd points together in a two or three dimensional map. The cost function used by t-distributed SNE (t-SNE) differs from the one used by SNE in: (i) symmetrized version of the SNE cost function with simpler gradients, and (ii) Student-t

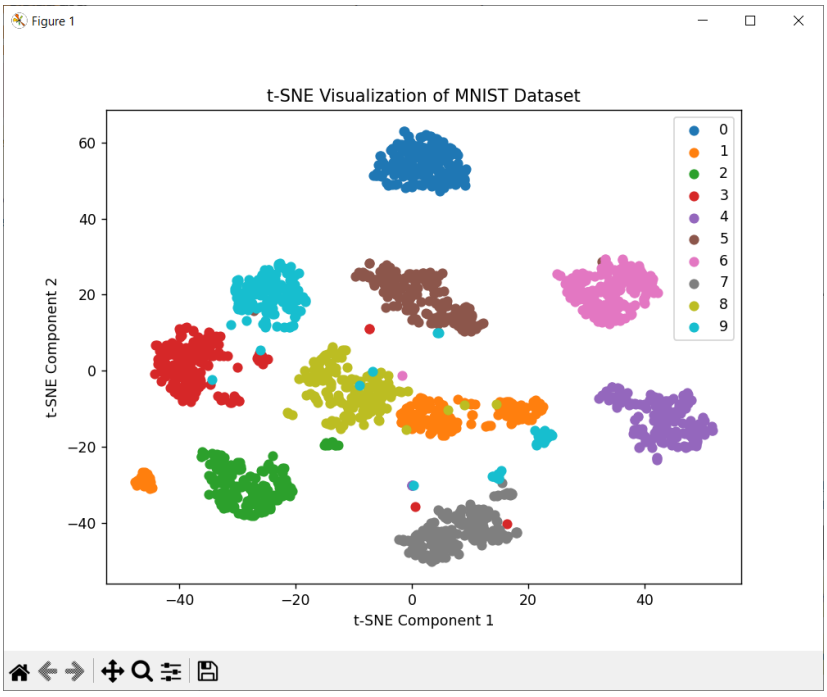

Figure 7: t-SNE 2D plot of MNIST digit image dataset, color dots representing different digits

distribution rather than a Gaussian to compute the similarity between two points in the low-dimensional space.

The plot can visualize high-dimensional data in a lower dimensional plot with the datapoints grouped by the clusters (by SNE) to increase understandability of a dataset.

2.4.2. Model explainers

2.4.2.1. Prototype/Criticism

ProtoPNet[106] is a DL model designed to provide explanations for its decisions, especially in distinguishing one object from another similar one (for example, differentiating a pigeon from a partridge) in real-time. It leverages convolutional layers to identify and extract specific features from the input image, which represent prototypical parts. These features are then matched against those from training images, resulting in an activation map that highlights the similarities. A distinctive aspect of this approach is the creation of "heat maps." These visualizations pinpoint the parts of an object (such as a bird's belly, wings, beak, etc.) that were crucial for its classification, essentially illustrating the model's thought process in distinguishing between object species.

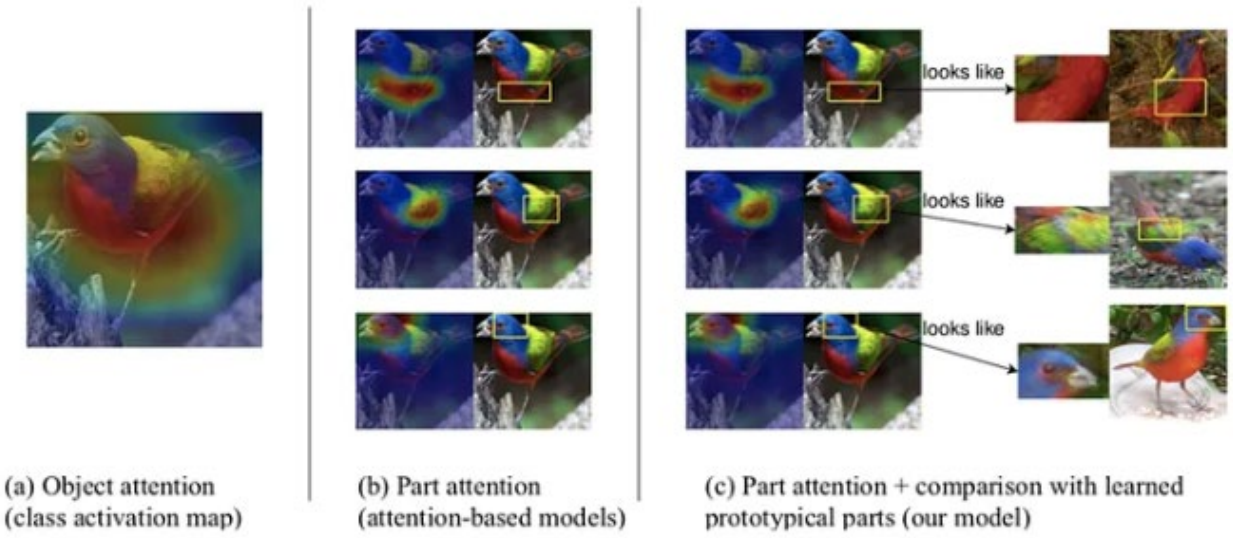

Figure 8: ProtoPNet example (courtesy [106])

In contrast to post-hoc approaches that generate interpretations after the model has been trained, ProtoPNet integrates a case-based reasoning mechanism that offers explanations concurrently with the classification process. Unlike attention-based models that merely highlight the critical features of a test image, ProtoPNet goes a step further by not only identifying these features but also illustrating how the test image compares with prototypical parts. Moreover, ProtoPNet employs a dedicated neural network architecture designed specifically for extracting features and learning prototypes through an end-to-end training approach. Additionally, ProtoPNet simplifies the visualization of prototypes without the need for a decoder and enhances its accuracy through the integration of multiple ProtoPNet models, thereby increasing the number of prototypes available for each class.

ProtoTree[107] similar to ProtoPNet, is a DL model designed for fine-grained image recognition. It incorporates prototypes within an understandable decision tree structure, allowing for comprehensive visualization of the entire model. Each binary tree node contains a trainable prototype part. Whether an

image features this prototype determines the path taken through a node. Consequently, decision-making mirrors human logic: If a bird possesses a red throat and a long beak, it is identified as a hummingbird.

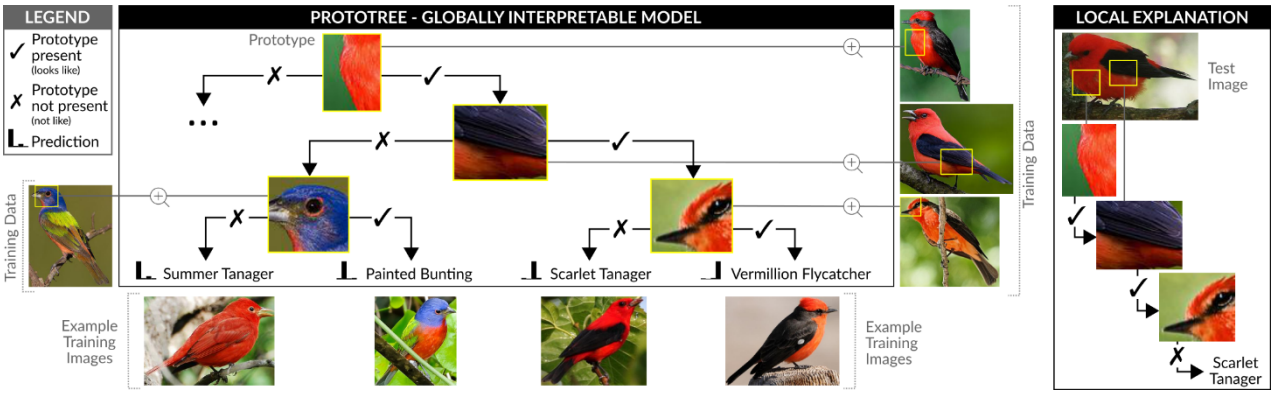

Figure 9: ProtoTree example (courtesy [107])

2.4.2.2. Feature importance

2.4.2.2.1. Accumulated Local Effects (ALE)

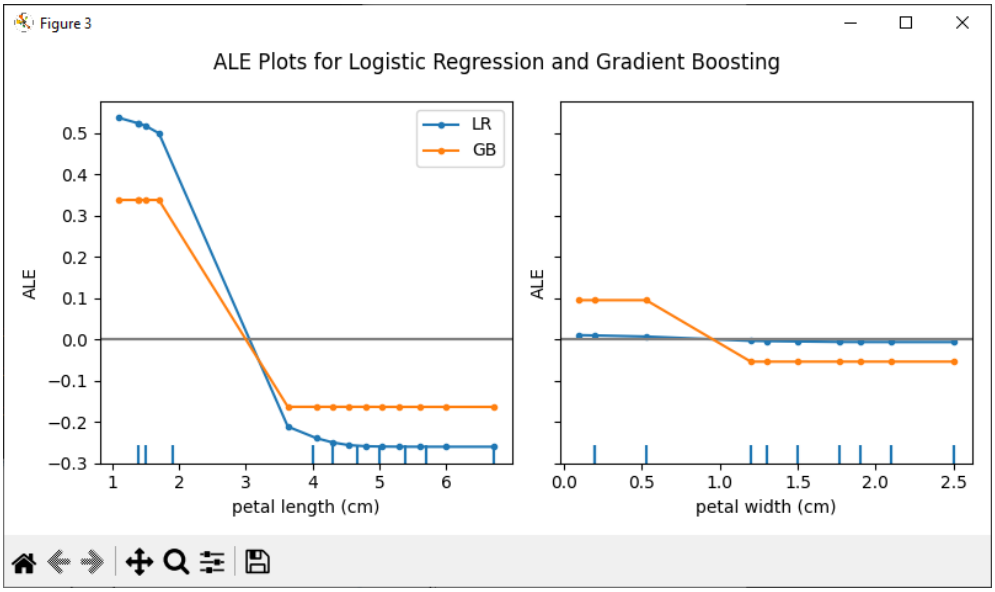

Figure 10: ALE plots for 2 trained models (Logistic Regression and Gradient Boosting) on IRIS dataset, selected input features are petal length and petal width

ALE[59] is a feature importance technique, it quantifies how the average prediction changes locally as a specific feature varies, providing insights into the non-linear relationships between features and model predictions.

Figure 10 illustrates the ALE plots for 2 different simple models (logistic regression and gradient boosting).

2.4.2.2.2. Partial Dependence, Individual Conditional Expectation Plots

Partial dependence plots show the dependence between the output prediction scores (or a dataset annotation) and a selected set of input features of interest, marginalizing over the values of all other features. To increase human understandability, we should only select small number of input features for each plot (typically 1 or 2 features of interest).

Individual Expectation ICE[108] plot, similarly, also visualize the dependence of the output prediction scores on a selected feature of interest. However, unlike PDP that only plots the average effect, ICE shows the dependence separately per data sample.

Combination of these two plots will provide the audience with information on how the output prediction scores of a DL component are dependent on specific dimensions of the dataset. Figure 11 shows an example

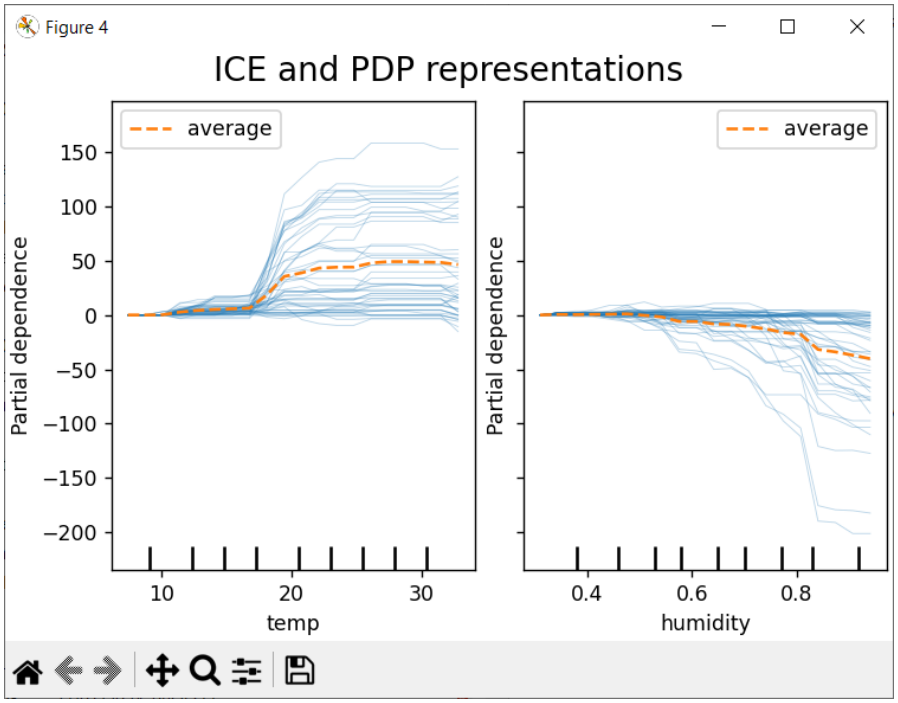

Figure 11: PDP (Orange) and ICE (blue) plots of the dependence of bike rental number predictions on temperature and humidity

of the two plots[109] on a trained MLP network on “bike sharing demand” dataset to predict the number of rental bikes. The plots show dependences of the prediction on selected features temperature and humidity.

The dependences on input features can be visualized for either (i) model output predictions or (ii)directly with the dataset annotation ground truth.

2.4.2.2.3. Permutation Feature Importance

Permutation Feature Importance (PFI) [110] determines the global feature importance by measuring the effects of permuting features on the correlation between the feature and the target and consequently on the model statistical performance

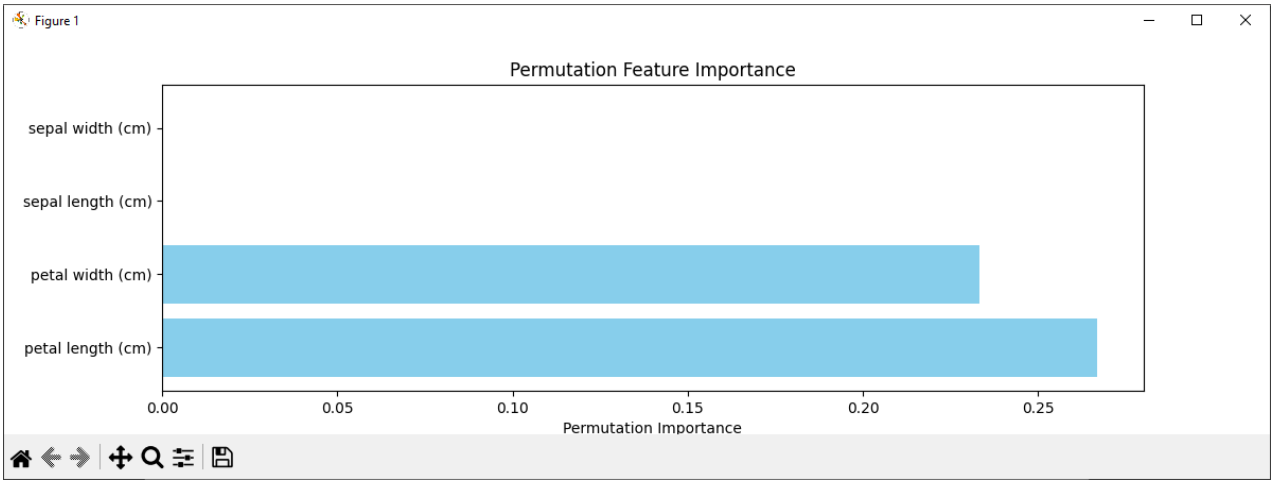

Figure 12: Permutation Feature Importance plot, Random Forest model, IRIS dataset

2.4.2.2.4. LIME

LIME (Local Interpretable Model-Agnostic Explanations[45]) is an explainability technique which locally approximates the AI model with a linear surrogate model. In this way, the complex decision boundary of the model can be simplified, and the features of the input can provide insights about their contribution to the final output of the model (e.g., how much age or gender contribute to a classification model for identifying a certain class of people).

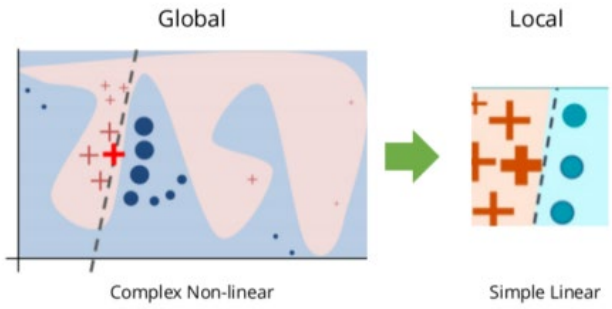

Figure 13: LIME provides local explanations as simple linear classification approximates.

LIME is local and model-agnostic; it is designed to be applied to classification models (binary or multi-class) but can adapt to a variety of other tasks, such as detection, segmentation, etc. It can work on different types of input, including numeric data, text, and images.

The LimeImageExplainer class works with images, where features are patches on the input image; it can generate a heatmap with patches relevance based on their contribution to the model output. While being model agnostic and providing useful results, the main downsides are that the choice of the patches/features is arbitrary, and that it is not fully deterministic and thus perfectly reproducible.

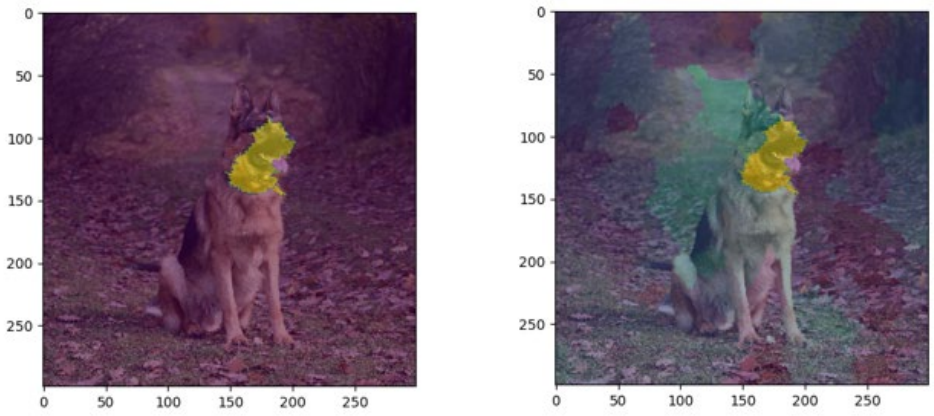

Figure 14: LIME explanation of a dog image. Image is segmented by superpixel regions. The yellow region represents the most influencing region of the model prediction (of "dog")

Its application is mainly for inspecting single edge cases and errors and could be employed during different phases such as development, verification, operation. It requires in any case some human inspection, since it provides no objective scoring but visual information on the input image.

2.4.2.2.5. SHAP

SHAP (SHapley Additive exPlanations[63]) is a renowned technique which employs the shapley values concept from game theory, which computes the features relevance to the model output by removing them and comparing the output with the original one. It is a local and model-agnostic technique; furthermore, some model-aware flavours are available. It can again work on tasks of classification (binary or multi-class) and can adapt to detection and visual-based tasks, with input ranging from data to text and images.

Since features in a model input can't be removed, they are masked using average or multiple values from a background dataset provided by the user. It is computationally exponential in the features number, so actual implementations approximate without exploring the whole graph of features removal possibilities; due to that, it is not fully deterministic. It must also be considered that SHAP computes only the relative importance of features, not an absolute one.

Like in LIME, SHAP (example illustrated in Figure 15) can work on images and compute the relevance of portions of the input. In this case, the patches masking must be provided by the user, which has control

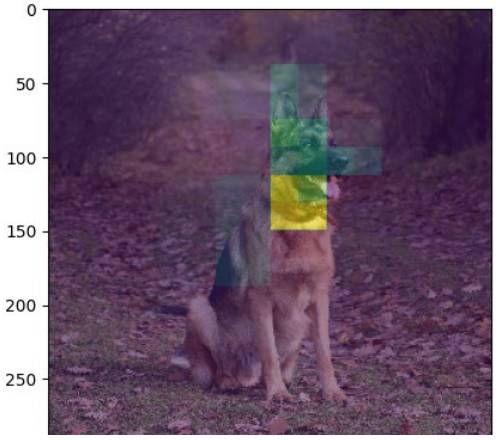

Figure 15: SHAP explanation of a dog image

over the shape and dimension. Again, like LIME, it can be used to analyse specific outputs of the model in different phases of the model lifecycle.

2.4.2.3. Layer wise relevance propagation (CRP/LRP)

Layer wise relevance propagation (LRP[66], [111]) is an explainable technique that works by propagating the relevance scores back through the layers of the DL network and assign importance values to each input features. In the case of visual based DL network, the features are pixels in input images that contribute to a specific prediction by the model. The techniques back trace layer by layer and redistribute the relevance scores from one layer to its precedent layer (Figure 16, from [107]). The relevance scores are conserved through layers and equal to the final layer's prediction score. Different variations of LRP introduce different approaches of score distribution, e.g. using "alpha-beta" rule considering both positive and negative contributions or the "z-beta" rule focusing on positive contributions.

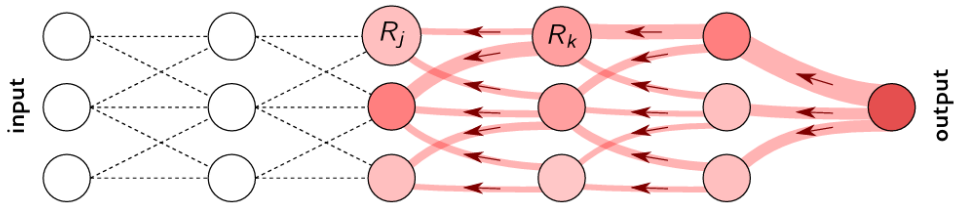

Figure 16: Illustration of the LRP procedure (courtesy of [107])

Concept Relevance Propagation (CRP[71]), introduced by Fraunhofer HHI group, further enhance the basic techniques of LRP by applying concept disentanglement to the last layer, thus instead of only propagating relevance score of the final prediction, the method also provide feature importance scores with regards to a known set of disentangled concepts.

2.4.2.4. Saliency maps

Saliency maps[112] or pixel attribution methods are DL explainability techniques that highlight which parts of an input image are most relevant for the model prediction. This kind of techniques, explain individual predictions by explaining how each individual input or feature, a pixel for example, is affecting the final prediction in a positive or a negative way.

In general, in a CNN architecture, the first convolutional layer's role is to learn lower-level spatial representation (features) such as edges and corners. The hierarchical structure allows convolutional layers to learn higher levels of abstraction and possibly features that can produce semantic meaning at a categorical level, such as classification and annotation of objects. Last convolutional layer's learned features proceed to the classification networks or dense layers in a CNN model[113].

Saliency maps are a general method used to visualize each pixel's unique quality independent from the predicted class. It works as a transformation where all data in the input space are re-represented using a lower number of dimensions. This transformation removes redundant and irrelevant features and segregates class relevant features from the background.

Many of the current methods based in saliency maps, are gradient-based methods. These methods, compute the gradient of the prediction with respect to the input features.

2.4.2.5. CAM

Class Activation Mapping[47] (CAM) is a technique used to generate visual explanations for deep neural networks (DNNs). It has been widely adopted to generate saliency maps to provide visual explanations for DNNs.

This method uses the notion of Global Average Pooling (GAP). The last stack of fully connected layers is substituted by a GAP layer which averages the activations of each feature maps and outputs the calculations as a vector. This vector's weighted average sum is introduced in the last softmax layer to highlight the most important parts of the predicted image. This is made by projecting back the weights of the output on the convolutional feature maps.

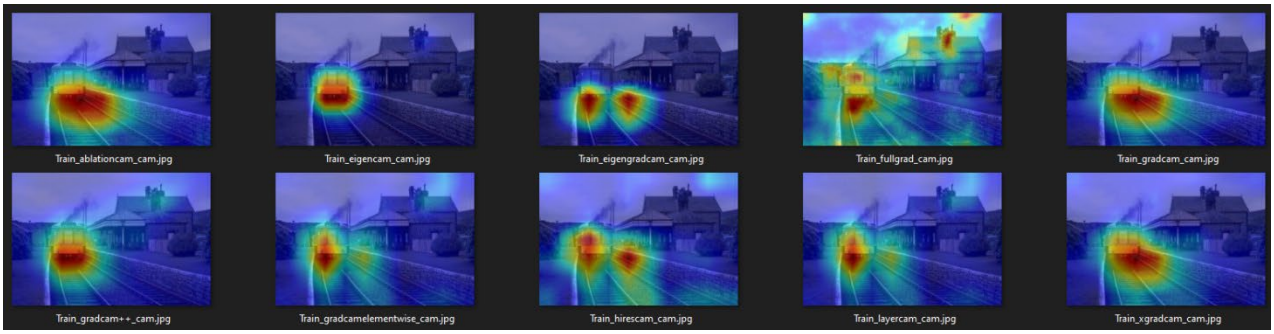

Figure 17: CAM maps overlaying on input image, VGG16 CNN classification

Gradient-weighted Class Activation Mapping (GradCAM[48]) is a more versatile version of CAM that can produce visual explanations for any arbitrary CNN, even if the network contains a stack of fully connected layers. It uses the gradients of any target concept flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept.

In this case, the gradient is not backpropagated until the image but to the last convolutional layer to produce a coarse localization map. It assigns each neuron a weight to choose the decision of interest.

As it has been mentioned earlier, the first convolutional layer of a CNN takes as input the images and outputs feature maps that encode learned features. The other convolutional layers do the same but take as input the feature maps of the previous convolutional layers. Grad-CAM analyses which regions are activated in the feature maps of the last convolutional layers.

In the first convolutional layer, only the raw values of each feature map could be visualized, averaged over the feature maps and overlay over the image. Since the feature maps encode information of all the classes and we want only information about the classified class, this approach would not be valid. Instead, each pixel of the feature map will be weighted with the gradient and after it, averaged over the different feature maps.

As a result, we will obtain a heatmap that highlights the regions that affect both positively or negatively to the prediction of the class. The heatmap will be sent through a ReLU function to set all negative values to zero because the positive values will be the only ones that contribute to the classification of the target class. After mapping to the original image and scaling the values the heatmap will be represented over it.

EigenCAM[113] uses Principal Components to provide visual explanations for Deep Convolutional Neural Networks. It is a novel approach that builds on CAM to cope with the increasing demand for interpretable, robust, and transparent models. EigenCAM computes and visualizes the principal components of the learned features/representations from the convolutional layers creating heatmaps that can be overlaid on top of the original image to highlight which parts of the image induced the greatest magnitude of activation from the convolutional layers. As convolutional layers grow more and more compact, they go from picking up smaller visual details to learning more global representations throughout the image, so the principal components extracted from one convolutional layer aren't necessarily like the ones extracted from a different layer. Typical visualizations might show heatmaps across several different layers or congregate them into a singular heatmap to highlight the overall focus of the image.

EigenCAM can also provide valuable information when the detection involves more than one object. In Figure 18, we can appreciate how EigenCAM, applied to a case of railways, can provide information about

where the model is focusing when detecting cars and a train. In the left figure, we can see the original image with predictions provided by the model. In the middle, we could see the saliency map produced by EigenCAM. In the third image, we see an image of the saliency map produced, but only for the detected bounding boxes.

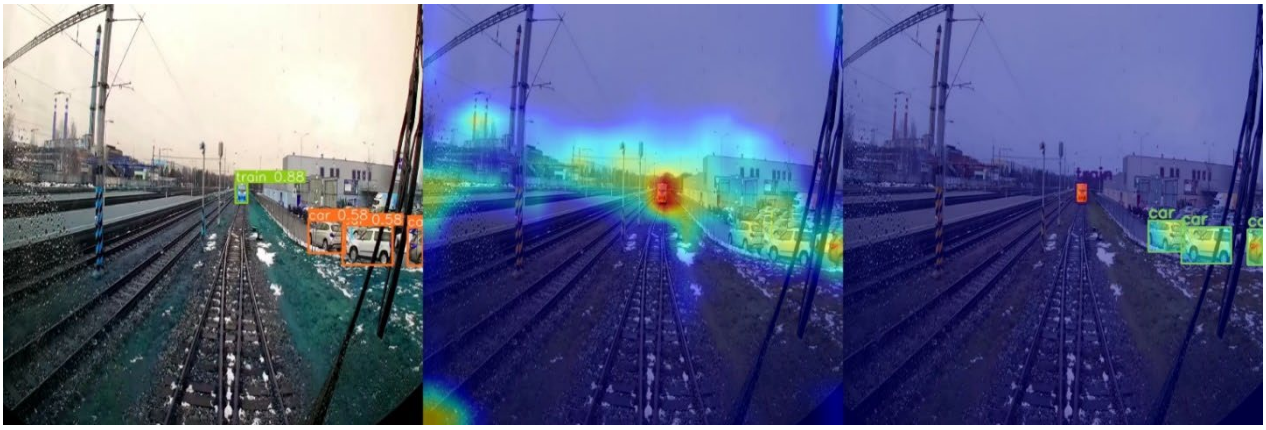

Figure 18: EigenCAM applied on YOLOV8 revealing where the model is focusing when detecting a train and some cars.

Anchors[46] is a local model posthoc explainer. It generates of simple rules to approximate the local behaviour of a DL model. It finds a set of conditions on the input features to guarantee the model prediction consistency.

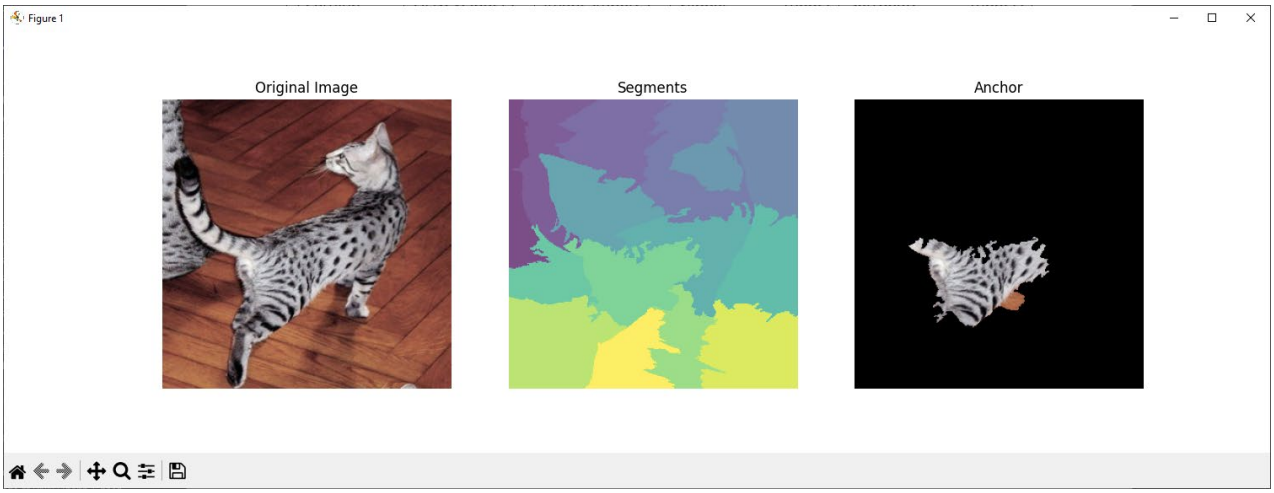

Figure 19: Anchors explainers applied for InceptionV3 model trained on ImageNet. From left to right: Original image, anchors superpixel segments (slic function) and segment that constitutes the anchor

2.4.2.6. Language-based Explainers

A surprising amount can be learned about the behavior of a deep network by understanding the individual neurons. Neuron-level descriptions are fine-grained descriptions that capture relational, and logical structure in learned features and make use of the full compositional vocabulary of humans. Language-based explainer approaches leverages recent advances in multimodal vision/language models which are gaining a lot of traction. Understanding the role of each neuron can help identify biases and vulnerabilities in the model that help make model-level decisions.

CLIP-Dissect[114] - An automatic neuron labeling method for vision networks leverages pretrained multimodal vision/language models. Unrestricted concepts without the need of any concept labeled data. It is training-free and model agnostic.

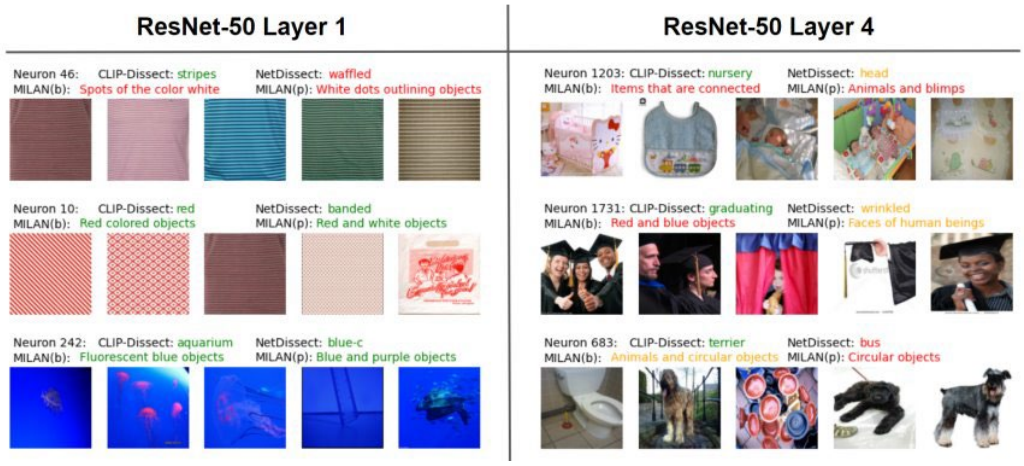

Figure 20: CLIP-Dissect and MILAN results on ResNet-50 explanations

MILAN[115] - A paradigm for labelling neurons with expressive, compositional, and open-ended annotations in the form of natural language descriptions. Given a neuron, MILAN generates a description by searching for a natural language string that maximizes pointwise mutual information with the image regions in which the neuron is active. Method involves representing each neuron via the set of input patches on which its activity exceeds a fixed threshold. Adopting an information-theoretic criterion for selecting descriptions: the final neuron description procedure optimizes pointwise mutual information between descriptions and exemplar sets.

Figure 21: CLIP-Dissect and MILAN results

2.4.3. Intrinsic explainable DL models

2.4.3.1. Interpretable models

Intrinsic explainability (a.k.a. interpretability) can be achieved by designing self-explanatory models which incorporate interpretability directly into the model structures. Traditional ML models can be considered as intrinsic interpretable models, examples include:

- **Linear models**: Linear regression, logistic regression, kernel ridge regression, general linear models (GLM), Exponential models (Poisson), quantile regression, Linear and Quadratic Discriminant Analysis (LDA, QDA), Support Vector Machine
- **Probabilistic models**: Naive Bayes, Gaussian Naive Bayes
- **Tree based models**: Decision Tree, Random Forest
- **Distance based Clustering**: k-Nearest Neighbours, Density-Based Spatial Clustering of Applications with Noise[116] (DBSCAN), Hierarchical clustering
- **Rule based models**: e.g. RuleFit[117], CN2[118] (Inductive Classifier 2), RIPPER[119] (Repeated Incremental Pruning to Produce Error Reduction), PART[120] (Partial Decision Trees)
- **Prototype based models**: Models that make predictions based on the similarity between the data instance (data point) and known clusters of data represented by prototypes. Examples of clustering algorithms include: k Nearest Neighbours, Centroid based distance, Learning Vector Quantization (LVQ), Self organizing maps (SOM)

2.4.3.2. Uncertainty aware DL models

To quantify the uncertainty of the DL model, several approaches that support to derive uncertainty-aware DL model from the original DL model are proposed. They are categorized into two main categories based on the uncertainty types (Reader consults more details in Section 3.2):

- **Epistemic uncertainties**: This type of uncertainties resulted from the lack of knowledge or the proper architecture/parameters of the model. It is thus connected to the uncertainty about the hypothetical true model of the problem.
    - Bayesian Neural Networks (BBN): This type of networks encodes and incorporate prior belief by implementing distribution over model parameters. They are computationally demanding and usually not appropriate to use for OM stage. Techniques include:
        - Bayesian CNN [121] where layers of the original CNN network can be modified
        - BBB[129] (Bayes by Backprop):Samples all the weights individually and then combines them with the inputs to compute a sample from the activations.
        - BBB_LRT (Bayes by Backprop w/ Local Reparameterization Trick): Combines Bayes by Backprop with local reparameterization trick from [130]. This trick makes it possible to directly sample from the distribution over activations.
    - Ensemble Methods: Using different models to provide a distribution of predictions (per model), then use the variance for uncertainty estimation.
    - Monte Carlo Dropout: involves utilizing the same DL model but running the network multiple times with random dropout to generate a distribution of predictions.
- **Aleatoric uncertainties**: This type of uncertainties is irreducible, i.e. the uncertainty remains even with larger datasets provided. This includes noise inherent in data observations, uncertainty in data/annotations or sensor errors. To estimate/quantify this type of uncertainty, the model output instead of providing scalar prediction, should provide the distributional estimations (provided the uncertainty cannot be reduced but can be modelled)
    - Heteroscedastic Regression Models: Assuming Gaussian distribution of output predictions, thus provide output as (Mean, variance). The network will be trained using NLL loss.

- Probabilistic Regression Models: Provide any types of output distributions, examples include density mixtures networks that provide enough mixture components to model the uncertainty nature of the problem.

Noted that the **Domain uncertainties** are addressed by the PhDM phase in AI-FSM and the employment of data anomaly detector in the OM stage.

2.4.4. Metaheuristic search

While not specifically related to XAI methods that originally focus on explanation of how DL model works, understanding of how DL performance metrics are dependent on different input parameters such as training hyperparameters, network optimization parameters, or ODD/scenario parameters is of important to support the development process of dependable DL components.

Within the domain of XAI, metaheuristic search represents a collection of algorithms or strategies aimed at addressing complex optimization problems that traditional, precise methods struggle to solve due to their scale, complexity, or absence of a deterministic algorithmic solution. This process involves estimating the conditional probability of a DL model's performance metric based on a set of input parameters through metaheuristic search.

Bayesian optimization[122] leverages a surrogate model, often a Gaussian Process, to act as a stand-in for the actual DL model. This surrogate model, built from past observations, efficiently approximates the relationship between input parameters and performance metrics, guiding the search for optimal configurations without requiring as many costly evaluations of the actual DL model. Sequential Model-Based Optimization for General Algorithm Configuration (SMAC) [123] is specifically designed for hyperparameter optimization, it offers distinct features tailored for algorithm configuration using random forests as a surrogate model. SMAC utilizes an aggressive racing mechanism, where it pits two configurations against each other to efficiently determine which performs better. In contrast to SMAC the tree-structured Parzen estimator (TPE) [124] uses tree-like structure to explore the hyperparameter space. It prioritizes exploring branches that are more likely to lead to better performance. TPE utilizes a non-parametric approach, relying on two separate distributions – one representing the "good" performing configurations and another for the "bad" ones. Spearmint[125] focuses choosing appropriate priors and inference procedures for the Gaussian Process surrogate model. This method performs slice sampling and employs various acquisition functions (e.g., Expected Improvement) to guide the search towards promising parameter configurations. FAst Bayesian Optimization on LArge data Sets (Fabolas) [126] addresses limitations of Bayesian optimization for large datasets by utilizing efficient Gaussian Process sampling and an exploitation-oriented strategy with early stopping. Fabolas combines adaptive selection and evaluation.

Simulated Annealing[127] inspired by metal cooling, finds the minimum of a function like error by iteratively exploring solutions. It starts with a random solution and a high "temperature," allowing exploration of both good and bad solutions. As the "temperature" gradually decreases, the algorithm becomes more selective, focusing on better solutions and eventually converging to the optimal one.

Bandit-based methods are hyperparameter optimization strategies derived from the multi-armed bandit problem. This problem illustrates a situation with several choices, like slot machines, where the potential rewards are not fully known. The objective is to maximize rewards over the long term by carefully selecting which options to explore and exploit over time. These methods are especially useful for achieving a balance between exploration and exploitation in metaheuristic search optimization, especially in scenarios with scarce information and numerous potential solutions. Successive Halving[128] finds the best configuration among many options. It starts by randomly evaluating all configurations with a fixed budget. Then, it repeatedly discards the worst half based on performance, focusing resources on the remaining options that look to be superior, until only one configuration remains. This approach balances exploring various options and exploiting the promising ones. Hyperband[129] is an algorithm that builds upon the Successive Halving

approach. It runs multiple Successive Halving processes with different budgets, allowing it to balance exploration, i.e., testing a wider range of configurations, with exploitation, i.e., focusing on the most promising configurations. While Successive Halving and Hyperband directly evaluate options and discard less promising ones, Gaussian Process Upper Confidence Bound (GP-UCB) [130] utilizes a different approach. It builds a Gaussian process of the objective function to predict potential outcomes and uncertainty. This allows it to prioritize exploration in areas with high potential value while considering unexplored regions, making it efficient with limited data, particularly valuable for expensive evaluations.

Metaheuristic search algorithms are also drawing inspiration from the nature to tackle complex problems[131]. Inspired by swarming birds, Particle Swarm Optimization (PSO[132]) finds the best solution (minimum or maximum) by iteratively exploring possibilities. Each solution, called a particle, moves based on its own best past position and the best position found by the entire swarm, balancing exploration, and convergence. Drawing inspiration from the way ants forage, Ant Colony Optimization[133] finds optimal solutions by mimicking their pathfinding behaviour. Virtual ants explore the search space, leaving a stronger "pheromone trail" on shorter, more efficient paths. As ants follow and update these trails, the algorithm converges towards the best solution, balancing exploration, and exploitation. Spotted Hyena Optimizer[134] inspired by hunting hyenas, tackles complex problems. Hyenas (representing solutions) move in the search space (like a savanna) based on successful hyenas and surrounding information, with some randomness. The "prey" (optimal solution) might move slightly to prevent early convergence. Other examples include CMA-ES[135] (Covariance matrix adaptation evolution strategy) and Differential Evolution[136].

3. Specification and Design of FUSA-aware DL solutions

Trustworthiness of AI systems[3] refers to characteristics which help relevant stakeholders (defined within the AI-FSM guideline) understand whether the AI system meets their expectations throughout different phases/steps of the AI-FSM development lifecycle and in the OM stage. These characteristics can help stakeholders to verify that:

- Development process under AI-FSM: Understand the boundaries of the AI systems where they can operate safely (with regards to the safety goals and scope)
    - FUSA-aware DL solutions have been properly designed and validated in conformance with state-of-the art rules and standards. This implies quality and robustness assurance.
    - FUSA-aware DL solutions are built for the benefits of the relevant stakeholders. This implies awareness of the workings of AI algorithms and an understanding of the overall functioning by stakeholders. It also implies qualification or certification assurance of AI development and operation in conformance with AI-FSM assurance process.
    - FUSA-aware DL solutions are provided with proper identification of responsible and accountable parties.
- Operation and Monitoring stage:
    - FUSA-aware DL solutions will not put the safety critical systems at risk by operating outside of their known environments as specified during the AI-FSM development lifecycle.
    - FUSA-aware DL solutions will provide the quantization of their irreducible uncertainties.
    - FUSA-aware DL solutions will not provide predictions using unknown (never seen before) interpretable logics that have been developed and verified during the development lifecycle.

Explainability is required at various phases of AI-FSM assurance process lifecycle. The SAFEXPLAIN FUSA-aware DL solutions will incorporate the explainability features into the design and architecture of DL

solutions throughout the entire AI-FSM lifecycle to support transparency, traceability and providing (where applicable) explanations in the form of evidence to support compliant argumentations.

The following principles are proposed:

- **Transparency**: The FUSA-aware DL solutions should be designed to maximize transparency of the decision-making process of systems relying on DL components. That includes the three key approaches: decomposition into irreducible (explainable) DL sub-components, simulatability, and algorithmic transparency of the decomposed sub-components.
- **Traceability/error backpropagation**: Explanations should be extracted from the DL specification items to support diagnosis and trace any specific decisions made by the DL component.
- **Safety awareness**: The FUSA-aware DL solutions should be aware of the operational context and safety goals (via the definition and measurement of metrics supporting safety requirement compliant)
- **Human in the loop**: Where applicable, an architecture that allow human interaction in the decision-making process provided explainability to facilitate the communication between AI and human.
- **Guided test automation**: The FUSA-aware DL solutions should suggest techniques to support V&V strategy.

The following subsections may repeat some of the concepts or definitions from AI-FSM (D2.1) or the safety architecture patterns (D2.2), with the intention to discuss and propose potential solutions to facilitate the compliance as set forth by these tightly connected guidelines.

3.1. Requirement Engineering for AI

Requirement engineering for AI software (RE4AI [137], [138], [139]) is a growing research field in recent years, providing new approaches but still facing challenges and limitations. Requirement engineering for traditional software is a well-established domain, which is now having trouble being applied to AI, due to its novel, data-driven approach, and its different development processes.

Requirement engineering is a crucial process for software. Since user needs are meant to drive the software design and development, RE encompasses processes in which they are collected, analysed, specified as requirements, and verified in the final product.

As described in RE4AI[13], different modelling languages are available for RE, and the most cited in studies regarding RE for the AI field are:

- GORE (Goal-Oriented Requirements Engineering), a framework posing the basis of requirements into the goals the user wants to accomplish; by refining broad goals into smaller, specific aims, and considering different models for achieving them, a set of specifications is negotiated and documented.
- UML (Unified Modelling Language), a framework in which goals, scenarios, users and jobs are captured and represented graphically, omitting at first the operational and implementation details and focusing on the general picture, on possibilities and threats of the early model design. On this view, the different components of the system are specified in detail.

These languages comprise different flavours and specific implementations that can suit different needs. Besides that, other modelling tools are available such as Signal Temporal Logic (STL), Traffic Sequence Charts (TSC), Conceptual Model, Causal Diagrams, Domain Specific Modelling (DSM) and more.

Many challenges are nevertheless still relevant in RE4AI, and in general in the general use of Artificial Intelligence, on which it is important to cast the stakeholder's attention:

- the **overconfidence in using AI**, applying it without the due experience to specify and develop adequately the system, and without a clear understanding of its limitations.

- the **definition of requirements**, which can be hard to specify or too vague, and sometimes are overlooked when the final, specific outcomes of DL software arise from training over data rather than from a clear set of specifications.
- the **data management**, which takes a pivotal role with respect to traditional software development. Datasets are essential for training, testing and validating DL software; data quality, variety, consistency need to be implemented as a routine process, and in particular data generation is a powerful tool which must be analysed and adequately used.
- the **black-box DL functioning**, as many processes and outputs can't be explained or reconstructed as in traditional software. A clear need for explainability techniques arises, together with new non-functional requirements such as transparency.
- the **trustworthiness and ethics-related issues**, such as abiding by the rules and regulations of the European General Data Protection Regulation (GDPR) or ensuring training datasets aren't affected by biases which could compromise the fairness of the algorithm.

Exactly in these fields the work in SAFEXPLAIN aims to step in and provide answers and guidelines, both in the AI-FSM compliant development process and in the practical implementation of AI-based software in Operation and Monitoring stage.

3.2. Uncertainty challenges of DL components

In the work of Brando et al. [134], the authors present a structured approach to dissecting the sources of uncertainty inherent in Deep Learning (DL) components. Their research is geared towards establishing a formalized process to pinpoint and categorize the origins of uncertainty, which is crucial for enhancing the reliability and effectiveness of DL systems.

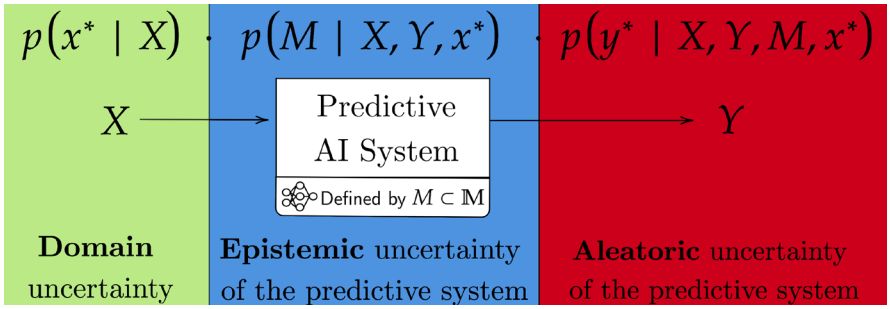

Figure 22: probabilistic disentanglement of the uncertainty sources for a supervised learning system (i.e. the predictive AI-based system). Note that only the last two types of uncertainty depend on the selected predictive system or model.

At the core of supervised learning, the process begins with a training dataset denoted as (X,Y), where 'X' represents the input data points and 'Y' signifies the corresponding annotations or labels. Both 'x' (individual data points within 'X') and 'y' (individual labels within 'Y') are treated as random variables, with 'p(X,Y)' indicating the joint distribution between them. The primary objective for a DL component, trained using this dataset, is to accurately learn the probabilistic relationship between 'y' and 'x'.

Brando et al. introduce a categorization of uncertainty sources into three distinct types:

- **Domain Uncertainty** [135]: This type relates to the uncertainty in modelling the input data distribution 'P(X)' concerning the specific problem at hand. It essentially reflects how well the sampled dataset represents the actual data encountered in real-world scenarios.
- **Epistemic Uncertainty**: This form of uncertainty arises from the limitations of the DL model itself, influenced by a finite set of hyperparameters. It encapsulates the gap between the actual model and the ideal model that could perfectly address the problem.

- **Aleatoric Uncertainty**: This pertains to the inherent variability in the output variable 'y', which can be attributed to factors such as occlusions, insufficient quality of data, or inadequate annotations that hinder clear predictions.

An illustration of these uncertainties is provided in Figure 22, further explaining how each type is addressed at different phases of the AI Framework for Safety Management (AI-FSM) and OM stage. Domain uncertainty is tackled before evaluating the supervised DL model (via *PhDM-Data Management* phase). Epistemic uncertainty is detected and managed via the engineering and V&V processes of the DL model (*PhLM – Learning Management* phase and *PhIM – Inference Management* phase). Finally, the aleatoric uncertainty, as it is irreducible, must be modelled and based on that design the decision system (uncertainty estimators in OM stage).

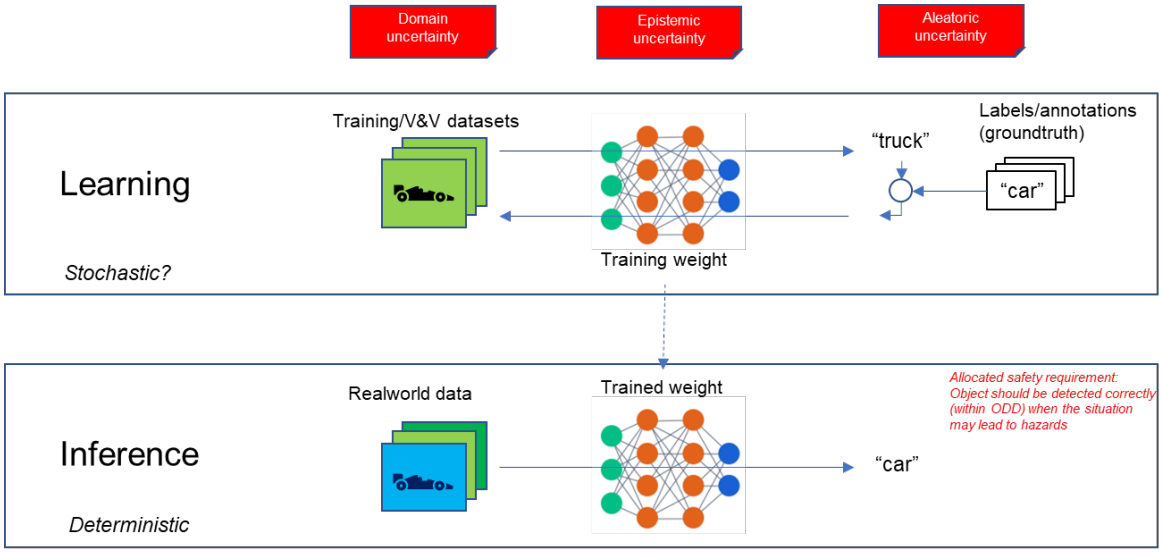

Figure 23: DL uncertainties

3.2.1. Data failures (Domain uncertainty)

Labelled datasets play a pivotal role in the training, validation, and testing phases of DL models. These datasets serve as the foundation upon which models learn to make predictions or decisions. However, the effectiveness of DL components is heavily contingent on the alignment between these datasets and the actual data encountered in real-world operational scenarios. Failures often emerge when there is a significant discrepancy between the training/validation/testing datasets and the operational environment. Such mismatches can lead to models that perform well in a test environment but falter in real-world applications.

To mitigate these risks and enhance the safety and reliability of DL models, it is crucial to adhere to specific requirements concerning the datasets used (as set forth by PhDM- Data Management. These requirements are delineated into primary categories: completeness, representativeness, balance, accuracy, volume and distinction between the three subsets (training/validation/test datasets).

Noted that since the datasets are captured/synthetized/prepared with some sampling strategies to reflect the real-world situation, i.e. they are lower dimensional projections of the realworld and are sensitive to the chosen sampling strategies. Within SAFEXPLAIN, the following iterative approach is proposed:

- Proceed data collection/preparation steps and assessment of the dataset compliance with the pre-knowledge of the area.
- Document the specification of these datasets as the baseline for the current dataset release.
- In later phases or in operation, different issues detected may result in better knowledge of the data requirements/specifications. This additional knowledge will be incorporated into the next iteration by revising the data requirements or specification.

3.2.2. Model failures (Epistemic uncertainty)

Model failures in the context of machine learning (ML) essentially question the model's ability to function as intended within its design parameters. These failures occur when an ML model, despite being trained on a representative dataset, does not perform accurately or reliably in its intended application domain. This discrepancy between expected and actual performance highlights the presence of epistemic uncertainty, which is rooted in the knowledge (or lack thereof) embedded within the model about its operational environment.

The quantification and understanding of epistemic uncertainty are crucial for diagnosing and addressing model failures. This type of uncertainty is primarily manifested in two ways:

- **Insufficient Performance**: This occurs when a model does not achieve the expected level of accuracy or efficacy in its predictions or decisions. The reasons for insufficient performance can be multifaceted, including:
  - **Insufficient Data**: Even if a dataset is representative, its size or diversity may not be adequate to capture the full complexity of the problem space, leading to a model that is undertrained or unable to generalize well.
  - **Network Design**: The architecture of the model itself might be unsuitable for the task at hand. This could be due to an overly simplistic design that fails to capture complex patterns in the data or an overly complex model that is prone to overfitting.
  - **Training Hyperparameters**: Incorrectly tuned hyperparameters can significantly impact a model's learning process and its eventual performance. This includes learning rates that are too high or too low, inappropriate regularization, or batch sizes that do not suit the dataset.
- **Insufficient Robustness**: This reflects the model's inability to maintain its performance across a range of scenarios, particularly in the face of new, unseen data or slight perturbations to the input. Insufficient robustness can be attributed to:
  - **Model Bias**: An inclination towards certain predictions due to biases in the training data or the model's architecture. This can lead to consistent errors in specific types of inputs or scenarios.
  - **Lack of Generalization**: The model's failure to apply learned patterns to new data. This often results from overfitting during training, where the model learns the noise in the training data instead of the underlying patterns.

3.2.3. Conditional occlusion failures (Aleatoric uncertainty)

Conditional occlusion failures highlight a critical aspect of aleatoric uncertainty, focusing on the inherent unpredictability and noise within the data that cannot be reduced or eliminated through model improvements or data cleaning. These failures are particularly challenging because they stem from the very nature of the data and its collection processes, affecting the model's ability to make accurate predictions. Key sources of conditional occlusion failures include:

- **Occlusions**: This occurs when part of the relevant data is obscured or hidden from view, making it difficult for the model to accurately interpret the available information. In visual processing tasks, for example, an object of interest may be partially covered by another object, leading to incomplete

or misleading data being fed into the model. Occlusions challenge the model's ability to accurately recognize patterns or objects, directly impacting its performance.

- **Sensor Limitations**: Every sensor used in data collection has its limitations and error margins, which introduce irreducible errors into the dataset. These limitations could be due to the sensor's resolution, sensitivity, or operational range, leading to data that is inherently noisy or imprecise. For instance, low-light conditions might reduce the quality of images captured by a camera, while environmental factors could affect the accuracy of temperature readings. Sensor limitations require models to be robust enough to interpret data within these constraints, acknowledging the uncertainty these errors introduce.
- **Subjective Data Annotation**: In tasks requiring human judgment for data labeling, such as annotating emotions in facial expressions or categorizing human behavior, there is a significant element of subjectivity. Different annotators may interpret the same data differently, leading to variability in the dataset that reflects human judgment rather than objective truth. This subjectivity adds a layer of complexity, as the model must navigate these variations to learn the underlying patterns.
- **Inherent Uncertainty in the Response Variable**: Some problems naturally involve a high degree of uncertainty in the outcome or response variable. This can be due to the complexity of the system being modeled, where multiple equally plausible outcomes exist for the same input. For example, in forecasting weather conditions or predicting stock market movements, the inherent uncertainty of the domain itself poses a significant challenge.

3.2.4. Failure modes

The endeavor to systematically catalogue and understand AI-related failure modes is an ongoing area of research, critical for advancing the field of artificial intelligence and machine learning. Recognizing and addressing these failure modes is essential for developing robust, reliable, and fair AI systems. Some of the typical failure modes encountered in DL-based applications include:

- **Fairness**: This failure mode pertains to the equitable performance of DL components across different demographic groups or classes. It addresses the risk of algorithmic bias where, for example, a facial recognition system might perform well for certain ethnic groups but poorly for others. Ensuring fairness involves implementing measures to detect and correct biases, promoting equality in AI outcomes across all classes.
- **Generalization**: A key challenge for DL models is the ability to generalize from training data to new, unseen data. Failure in generalization can lead to overconfidence, where a model makes incorrect predictions with high confidence. This often occurs when models are overfitted to the training data, lacking the flexibility to adapt to new information. Improving generalization requires techniques such as regularization, cross-validation, and ensuring a diverse training dataset.
- **Sensitivity to Adversarial Noises/Attacks**: DL models can be vulnerable to adversarial attacks, where small, intentionally designed disturbances in the input data lead to incorrect predictions. This sensitivity underscores the importance of designing models that are robust against such manipulations, incorporating adversarial training practices that anticipate and mitigate these vulnerabilities.
- **Timing Performance**: Ensuring a balanced trade-off between timing and functionality performance is crucial, especially in safety-critical applications where decisions must be made rapidly and accurately. Failures in timing performance can compromise both the safety and efficiency of AI systems. Optimizing algorithms for speed, without sacrificing accuracy, is vital for applications requiring real-time processing.
- **Consistency**: The integration of new knowledge with previously certified knowledge (e.g., through transfer learning) can present challenges in maintaining consistency. Models must be able to learn from new data without forgetting prior learning or compromising the integrity of established

knowledge bases. Techniques such as continual learning and memory replay are explored to address these concerns.

- **Interpretability/Explainability**: A fundamental challenge in DL is the ability to provide clear and understandable explanations for the decisions or predictions made by models. The lack of interpretability can hinder efforts to identify and correct errors, making it difficult to trust and validate AI systems. Developing models that are both accurate and explainable is a key focus, aiming to enhance the transparency and accountability of AI technologies.

3.2.5. DL related risk assessment aspects

In the development and deployment of DL-based systems, understanding and mitigating risks is crucial for ensuring their reliability and effectiveness. Based on insights from the literature [138], several key risk aspects are identified that require careful consideration:

- **Potential Impacts of Common Corruption Patterns**: DL models can be susceptible to performance degradation when exposed to various real-world corruption patterns. These patterns can include visual distortions due to harsh weather conditions, signal interference in audio data, or even anomalies in text data caused by typographical errors. Recognizing and preparing for these common corruption patterns is essential for developing models that maintain high performance levels even in challenging operational environments. Techniques such as data augmentation, robust training methods, and continuous monitoring can help in mitigating these risks.
- **Out Of Distribution (OOD) Awareness**: OOD awareness refers to the model's ability to recognize and appropriately handle input data that significantly differs from the training distribution. OOD data can lead to unpredictable model behavior and unreliable predictions. Implementing mechanisms for detecting and managing OOD inputs ensures that the model can either safely ignore such data or flag it for human review, thereby preventing potential errors or failures.
- **Uncertainty Estimation**: Accurately estimating uncertainty in predictions is a critical aspect of risk management in DL systems. Uncertainty estimation allows for a more nuanced interpretation of model outputs, distinguishing between high-confidence predictions and those where the model is essentially "guessing." Techniques such as Bayesian neural networks and ensemble methods can provide valuable insights into the confidence levels associated with predictions, guiding decision-making processes and highlighting areas where additional data or model refinement is needed.
- **Error Propagation Among Modules in the System Pipeline**: In complex DL systems consisting of multiple interconnected modules, errors can propagate from one module to another, amplifying the impact and potentially leading to systemic failures. It is vital to assess how errors in one part of the system can affect subsequent stages, implementing safeguards and feedback mechanisms to contain and correct errors as early as possible. This includes rigorous testing of individual modules, as well as end-to-end system evaluations to identify and mitigate error propagation pathways.

3.3. AI-FSM aware DL solutions

AI-FSM aware DL solutions (earlier referred to as FUSA-aware DL solutions) inherits the specificability and traceability from being complaint with and specified by all required artifacts within the AI-FSM lifecycle process. The traceability of the AI-FSM aware DL solution is enabled by the construction and instantiation of argument patterns (structured forms of reasoning) to support the overall system safety claims. Throughout the steps within AI-FSM phases, different explainability (and DL specific) techniques will be leveraged to support the steps in automating some subtasks or generating evidence to support the subclaims of the safety argumentation. This section will describe where and how different techniques shortlisted from existing technologies can systematically be mapped to the usages by AI-FSM phases and/or steps.

3.3.1. Focused scope

Deep Learning based systems can cover a wide range of tasks, for different kinds of applications in software and different types of input and output, typically consist of the following most common classes: Regression, Classification, Clustering, Anomaly detection, Dimensionality reduction, Generation.

Having such a variety of techniques and artifacts, a choice must be made to focus the effort and provide good solutions, at the same time applicable to different use cases and also specific enough to be effective. Within the scope and timeline of SAFEXPLAIN, the focus will be set mainly on the tasks of object detection and semantic segmentation, plus a more outstanding use case to add some variety. This choice is reasonable as a vast range of industrial applications, such as autonomous driving, navigation and monitoring, rely on processing images of the environment and identifying objects and classifying them.

The FUSA-aware DL solutions described in this section are categorized by their applications throughout the following key phases:

- **AI-FSM lifecycle phases**: The DL solutions (DL components) are described as a collection of modules and compliant supporting tools during the AI-FSM process (D2.1).
- **Operation and Monitoring stage**: Solutions and the key components to facilitate compliance with the reference safety architecture pattern in D2.2.

Figure 24 depicts the scenario categories as defined by SOTIF, characterized by the two key dimensions: Known/Unknown and Hazardous/Not hazardous. The FUSA-aware DL solution supports within AI-FSM can be mapped to the evolutions represented by the two arrows in the right part of the figure, i.e. expanding the “Known” subspace, as well as clarifying the boundary between the subspaces. During the OM stage, the FUSA-aware solution will ensure that the system operates within the “Known/Not hazardous” area, taken into account also the known irreducible uncertainties.

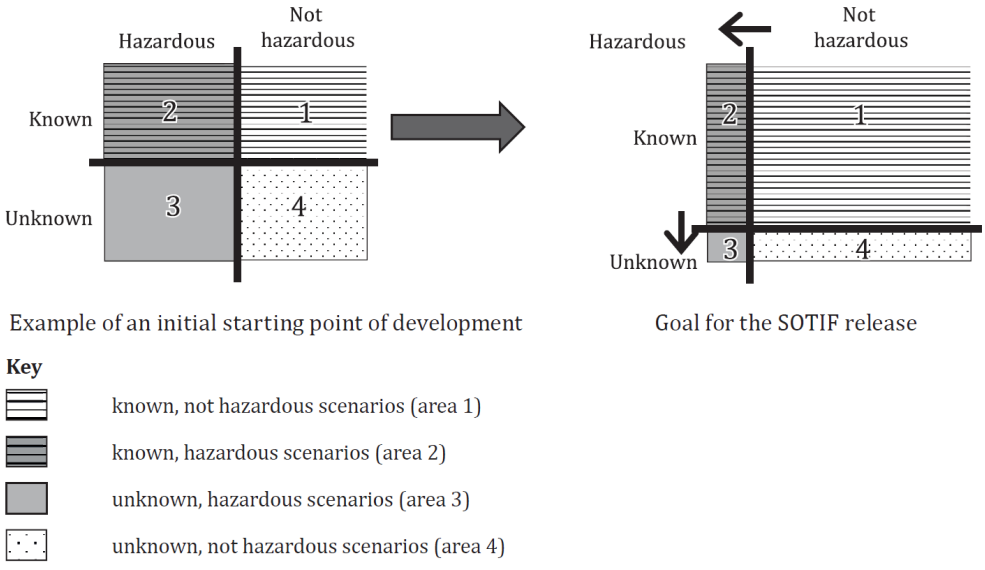

Figure 24: Alternative evolution of the scenario categories resulting from ISO21448 activities [REF]

3.3.1.1. Key focus during AI-FSM lifecycle

The application of XAI techniques within AI-FSM development lifecycle is mainly to provide the required transparency and evidence to support clarifying and enlarging "Known" area (Figure 24 -ISO 21448, SOTIF [140]). This also includes the need to increase the clarity of boundary between Known and Unknown areas.

More specifically, the key usages of XAI techniques are:

- Generating evidence and reports.
- Analyse/diagnose corner cases.
- Improvement and optimization.
- Baseline the safe operational conditions of DL component to support operation and monitoring.

3.3.1.2. Key focus during Operation and Monitoring stage

During the Operation and Monitoring, the key usages of XAI techniques are:

- Assessment if the DL model is operating within the "Known" area as specified and verified with AI-FSM, and thus its output predictions can be trusted for influencing safety-related actions.
- Estimate the DL model's uncertainties (irreducible uncertainties left during AI-FSM), to help controlling risks under acceptable levels.

Specifically, in connection to the reference architecture pattern (D2.2), the XAI usages will contribute to the *L1-Diagnostic and monitoring mechanisms* (L1DM), a component within *Supervision Components.*

L1DM will assess different aspects of the DL component within the safe system and provide alerts if there are risk of the DL component is operating or behaving outside of its "Known" boundary.

3.3.2. XAI usages in "Ph1-DL-related concept specification"

At this phase, no specific support from XAI techniques is required.

The specification of ODD and operational scenarios at this phase will set the baselines for XAI enabled quality assessments of datasets, fairness of the DL model and the setup Verification and Validation (V&V) scenarios in later phases.

3.3.3. XAI usages in "Ph2-DL requirements specification"

At this phase, no specific support from XAI techniques is required.

At this phase, the following requirements are relevant to guide the proposal of XAI usages in subsequent phases:

- Requirements of supervision components to be used within the OM stage, including model agnostic and model specific algorithms.
- Allocated safety requirements success criteria into the DL model and dataset related set of metric-based measures.

- Required level of explainability/transparency of the DL model: intrinsic/posthoc/antehoc.

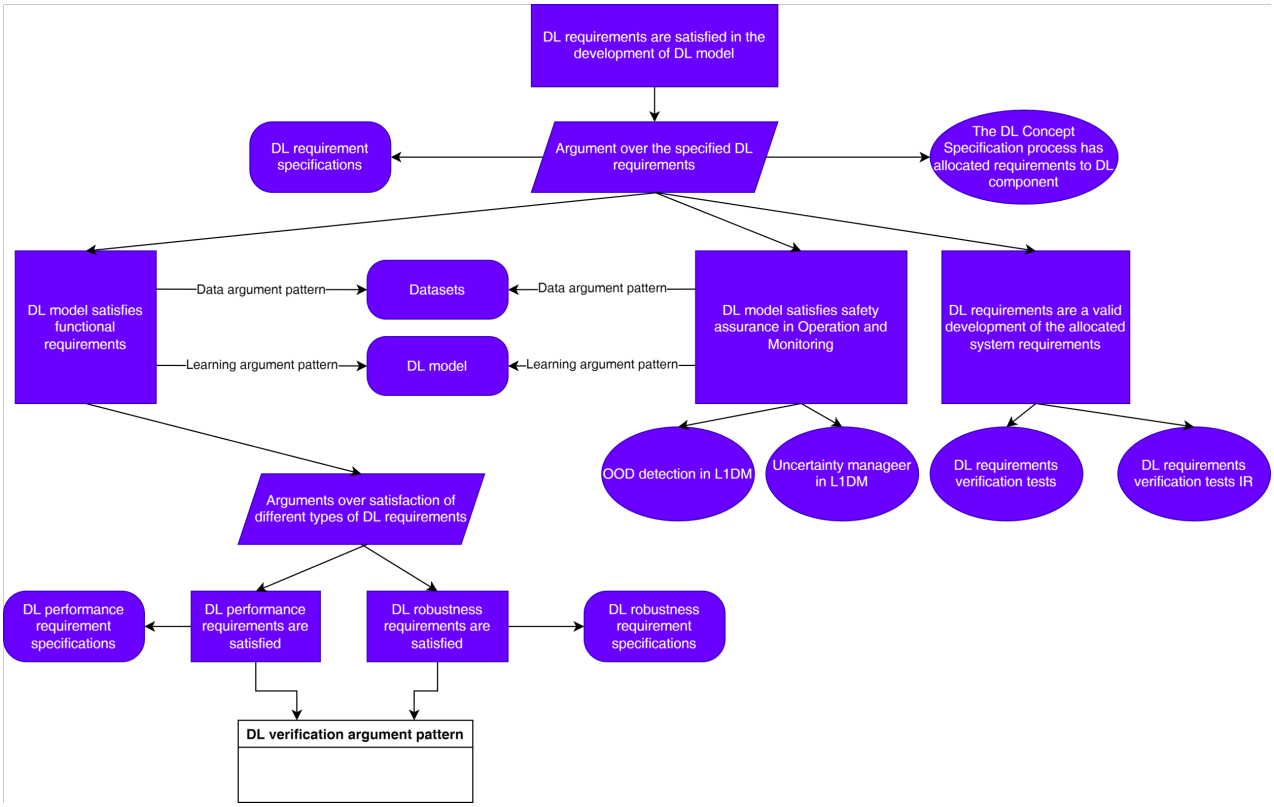

Figure 25: DL requirement argument pattern

Figure 25 shows DL requirement argument pattern, where the top claim is that the system (safety) requirements allocated to the DL component are satisfied by:

- DL component, subsequently allocated into data and model, which will be managed and facilitated with different XAI tools within the subsequent AI-FSM phases PhDM, PhLM and PhIM.
- Safety assurance in OM stage, subsequently allocated into the techniques relevant for L1DM component within the reference safety architecture pattern (D2.2)

The argument patterns are one of the key techniques to ensure the traceability of the FUSA-aware DL solution, where the XAI generated evidence can support different subclaims.

3.3.4. XAI usages in “PhDM Data management”

Using data explanation techniques is essential for PhDM phase, especially for the following usages:

- Assessment of the dataset against data requirement specification: Data explainer techniques should provide explanations and extract data quality metrics connected to a dataset desideratum as specified by the data requirements: completeness, representativeness, balance, volume, accuracy, and datasets distinction (train/validation/test)
    - Data collection: assessment of data volume, completeness and representativeness
    - Data preparation: assessment of data balance, accuracy and datasets split (if both three satisfy the same set of data requirements but are distinguishable).
- To provide/generate a detailed summary to a certification authority or external party to understand the datasets.
- To specify the boundaries between “known” and “unknown” w.r.t. data (input data/output annotations) to support V&V strategy and anomaly detectors in supervision components.

Figure 26 shows the Data argument pattern, where the top claim is that the three used datasets are sufficient. Data explainers (XAI techniques for data) will contribute to the required evidence for the data verification/validation reports.

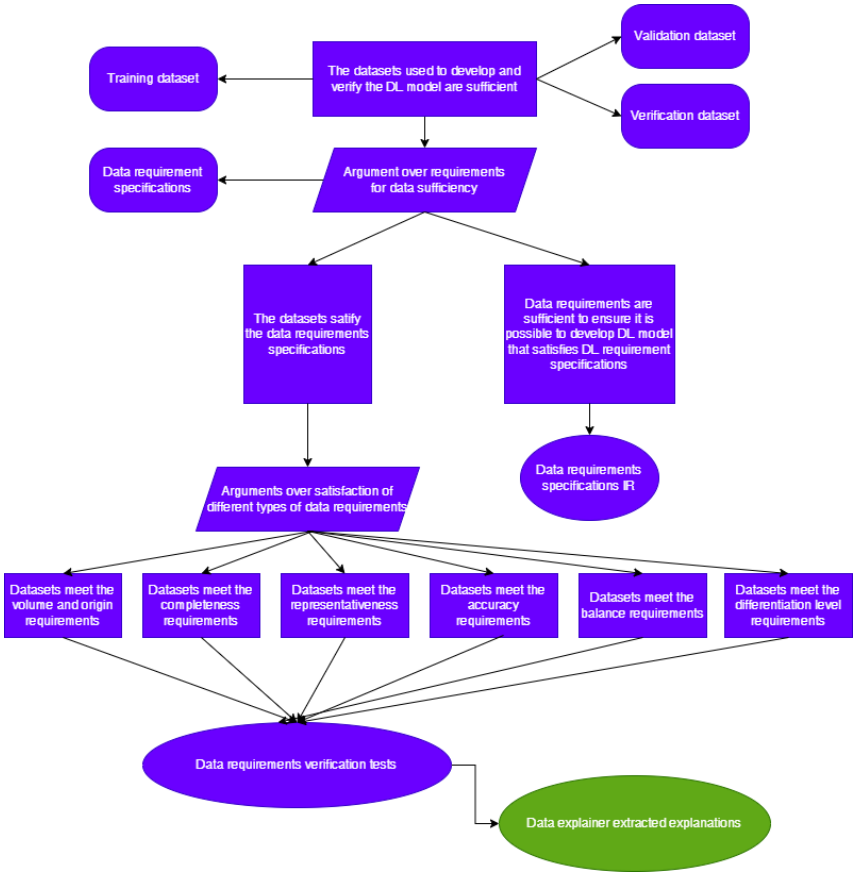

Figure 26: Data argument pattern

The following XAI techniques are applicable:

- **Dataset profiling**: Understanding the structure, content and quality of a dataset is essential before performing any analysis or modelling tasks. This involves examining various aspects of the data such as statistical summaries, data distributions, missing values, and patterns. Dataset profiling will be used to generate data summary report.
- **Data mining analyses**: Employing data mining techniques to learn meaningful patterns, trends, and relationships within the dataset, which can provide valuable insights for decision-making.
- **Data prototype extraction**: Identifying representative samples or prototypes from the dataset to capture essential characteristics and variability, to understand the data distribution.
- **Data distribution description**: The distributions of data are represented by distribution over a single dimension or a lower dimensional subspace of data, based on ODD/Operational scenarios parameters and distribution of annotations. The distribution descriptors will be used:
    - To assess and baseline the desiderata of the dataset (complete, balance, relevant and accurate). Data quality measures shall be logged to provide data traceability and support error backpropagation in case diagnosis.
    - To define the inlier/outlier boundaries, which will be used as thresholds and other parameters of Out of Distribution detection algorithms during Operation and Monitoring.
- **VAE based dataset descriptors**: Utilizing auto-encoder based methods to learn compact representations of the dataset, facilitating the extraction of meaningful features, and enhancing

interpretability. A trained Variational Auto Encoder (VAE[31]) can be used as an advanced dataset descriptor. By that we assume the trained weight will capture the general properties of any datapoint belonging to the dataset, and the latent vector will present the data point vector (defined by ODD and operational scenario specific parameters related to a data instance).

Table 2: Data Management steps and XAI mappings

Step	Supporting technique consideration
Data collection	– If sensor captured datasets and synthetic datasets are used: Data explainers to measure the datasets metrics. – Domain transfer (e.g. generative models such as AutoEncoder) to help convert and merge datasets.
Data preparation	– Used in connection with feature relevance techniques during the later phases to investigate corner cases (if the prediction relies on preparation quality) – Feature relevance of data preprocessing algorithms to provide traceability. – Reporting labelling statistics, consistency (aleatoric uncertainties) – Reporting occlusion level statistics – Reporting data metrics before and after data preparation
Data requirement verification	– Data explainer to export metrics and distribution of datasets per input dimensions (parameters of ODD, operational scenarios, annotations)
Data requirement specification	– Dimension reduction techniques such as VAE, PCA, prototypes... to describe data using a lower dimensional representative space

3.3.5. XAI usages in “PhLM Learning management”

Explainability is crucial in this phase as it provides insights into how the DL component makes decisions or predictions, bringing trust and ensuring alignment with safety requirements.

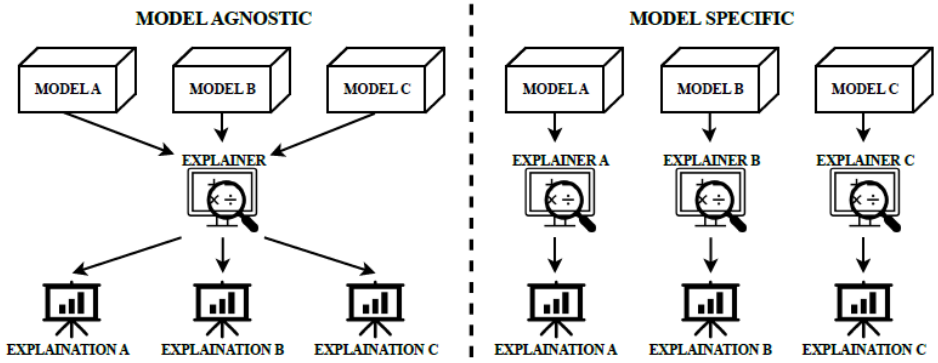

Figure 27: Model dependency.

At this phase, different explainability techniques that can be applied will be considered. The first consideration is the selection of DL model candidates with regards to the availability of explainability techniques. To this end, available methods can further be categorized by model dependency (Figure 27):

- *Model-specific methods*: These techniques are tailored to specific models, leveraging the internal structure of the model to generate detailed explanations. Examples include attention mechanisms.
- *Model-agnostic methods*: These techniques require only the input and output of a model, making them applicable across various models without needing access to internal structures. Examples include SHAP and LIME.

When designing this phase, the consideration of model-agnostic methods is not a primary concern, as these techniques can be applied universally across various models and architectures. However, they still offer valuable explainability insights and can be utilized effectively within this phase. Conversely, the focus on model-specific methods can potentially influence the training phase.

Given the nature of model-specific methods, which are tailored to particular models and leverage their internal structures for detailed explanations, incorporating them into the training phase might require additional considerations. For instance, the adoption of model-specific methods could impact the choice of model architecture or the preprocessing steps during training. For example, if attention mechanisms are chosen as a model-specific method, the training phase might need to prioritize architectures that are compatible with such mechanisms or require specific data preprocessing to facilitate their integration effectively.

Therefore, while model-agnostic methods offer versatility and can be seamlessly incorporated into the learning management phase, the consideration of model-specific methods may necessitate careful planning and adjustment within the training phase to ensure compatibility and optimize their effectiveness.

The following subsections will discuss specific usages as per relevant steps of the PhLM phase.

3.3.5.1. Learning requirement specifications

No application of XAI technique is required in this step.

3.3.5.2. Model design

At this step, the following XAI approaches are recommended to be adopted in the DL model design:

- **Decomposition** of problems into subproblems and related parts of the DL components in charge of processing. Examples include decomposition into conceptual blocks where each block is responsive to a specific concept.
- **Disentanglement** of the DL architecture to assign different subspace of the latent space (feature space) to different semantic labels that are related to safety requirements. Example includes e.g. weather-related dimensions, illumination-related dimensions, etc.
- **Uncertainty aware** design of the model (to be used later within L1DM component in OM stage) by applying stochastic process approaches to input, model's neurons, or output predictions.
- Design interpretable **surrogate model(s)** that can approximate the DL model global behaviour. The logics inside surrogate model(s) will be certified by the certifiers as acceptable model behaviour logics (up to the current level of knowledge). The surrogate model will also be a candidate for supervisory monitor algorithms in OM stage.
- Design how interpretability can be achieved from model internals (**intrinsic interpretability**).

- Metaheuristic search for optimal hyperparameters (Figure 28 shows different search algorithms that can be used to guide the training runs how to find a better set of training hyperparameters avoiding the exhaustive search).

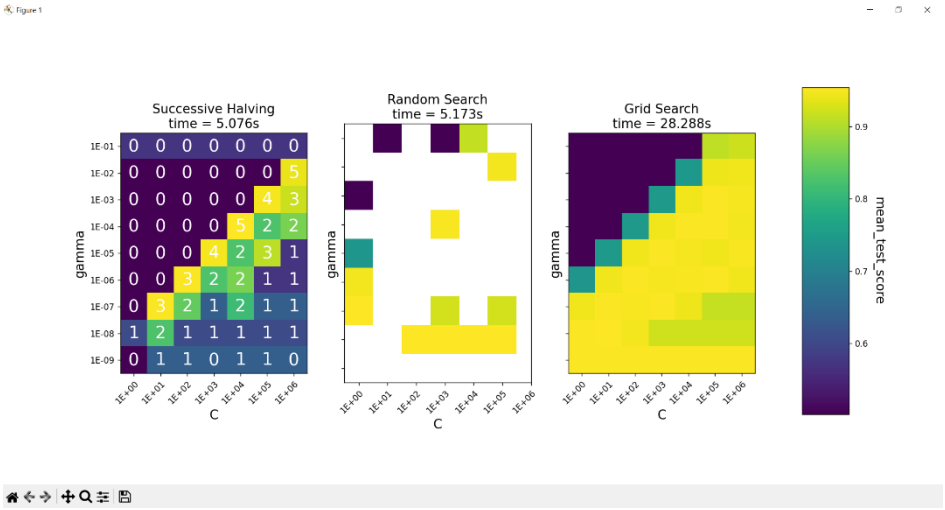

Figure 28: Heatmaps showing the test scores w.r.t. model hyperparameter settings (a simple SVC model with 2 hyperparameters gamma and C). Methods used from left to right: Successive Halving, Random Search and GridSearch in finding hyperparameters

3.3.5.3. Model training and Model Evaluation

The following XAI techniques are recommended within this step:

- Local/global model explainers:
  - Assessing feature importance to determine if test results are valid within neighboring subspaces of the input space.
  - Assessing model attentions with regards of input data (saliency maps and other attention-based techniques)
  - Prototype pattern extractions from data and model to ensure the required overlap.
  - Extraction of model behaviors, e.g. in form of neuron activation patterns, to use as the baseline for known behaviours. This will also serve to derive anomaly detector with regards to the neuron activation patterns, e.g. using similar techniques as data anomaly detector.
- Analysing the relationship between required performance metrics and input parameters to ensure clarity and understanding (example provided in Figure 29 shows different performance metrics as discrete functions of the input scenario parameters)
- XAI guided search-based methods: Employing search algorithms to enhance the effectiveness of the verification and validation process.

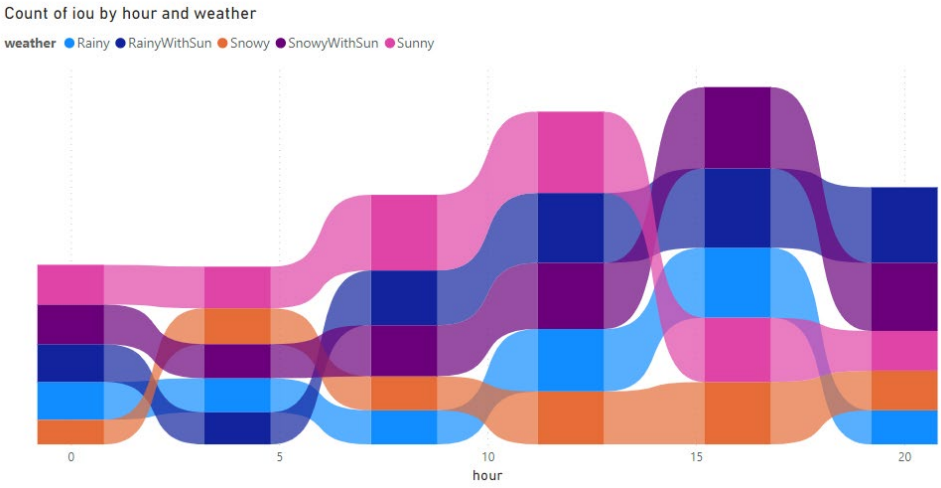

Figure 29: Model performance vs input parameters plot. The plot shows number of detected objects (measured by if IoU>0) by input parameters: time of day and weather conditions

3.3.5.4. Model verification

Local explanation methods such as input feature relevance, counterfactual can be used to provide information to support genetic-based metaheuristic search algorithms ([131], [141], [142], [143]). The algorithm is inspired by the natural selection process, using genetic operators such as selection, crossover and mutation to seek for potential new combination of test parameters that (assumed to) has better chance of achieving better DL performance.

The most common flow of Scenario based testing is described below:

- Creation of scenario database: At this step, statistical data explainers are needed to support analysis of the completeness and coverage of the scenarios and related verification dataset.
- Create test cases with initial setup of ODD and operational scenarios parameters.
- Collecting the test results (DL component's metric measures) connected to the test case input parameters.
- Analyse the test results to propose the next batch of scenarios and subsequently test cases. This step requires the XAI enabled intelligent test control algorithms to optimize the number of tests required.
- Identification of corner cases. This step requires different XAI techniques as proposed in PhLM and PhDM, together with the error backpropagation approach to diagnose the cases.

3.3.6. XAI usages in "PhIM Inference management"

Key items requiring XAI techniques include:

- Trade-off mechanism between DL performance, explainability and runtime
    - Pruning configuration: XAI techniques help determine which parameters in the DL model can be pruned without significantly impacting performance, thereby optimizing the model's efficiency, and reducing computational resources.
    - Feature subset selection (feature relevance/input dilation): XAI techniques aid in selecting relevant features or inputs for the DL model, ensuring that only the most informative features are used to improve model accuracy and efficiency.

- Pre-processing: e.g. resize, reshape the input data: XAI techniques support preprocessing tasks such as data resizing and reshaping, ensuring that input data is appropriately formatted and scaled for optimal model performance.
- Switch between redundant models designed with different complexity (e.g. number of layers): XAI techniques assist in evaluating and selecting between redundant models with varying complexities, helping to choose the most suitable model architecture for a given task while balancing performance and computational resources.
- Precision quantization: XAI techniques are employed to determine the optimal precision levels for model parameters.

- Support V&V:
  - Search based testing with support from XAI techniques (feature importance, surrogate models, counterfactual): Using XAI techniques such as feature importance analysis, surrogate models, and counterfactual reasoning to enhance search-based testing processes during verification and validation activities. This helps finding potential issues and improving the overall effectiveness of the testing process.
- Data explanation to analyse corner case: Employing XAI techniques for data explanation to delve into corner cases and edge scenarios, ensuring thorough analysis and understanding of the data. This helps in identifying potential anomalies or unexpected behaviour in the system.

3.4. XAI usages in Operation and Monitoring stage

XAI techniques are used in various supervisor algorithms to support L1-Diagnosis and Monitoring mechanism (L1DM), as described in Section 4. Noted that the terms supervisor refers to specific monitoring techniques designed for monitoring the DL model behaviour during OM stage, making sure that it will not operate outside of the “known” area that have been identified, specified and verified during AI-FSM lifecycle.

Table 3: Safety architecture pattern's supervision components and solution mappings

Artifact	Supporting technique consideration
L1DM	– Surrogate models: A simpler interpretable model that has been trained to approximate the DL components. Surrogate models that take its input from interim features (feature extract from a specific middle layer of the DL model) and generate output can also be considered. – Anomaly detection (Out of Distribution detection - OOD): Anomalies can be detected from the input data or any interim feature after several feature extraction layers of the network (Neuron activations) – Uncertainty based DL model: A modified version of DL component that can, besides the required output predictions, provide associated uncertainty quantifications. The uncertainties are of two types: epistemic and aleatoric. Candidate solutions include Bayesian network, training dropout, or assigning distributions as the model predictions (instead of single prediction). – Proxy model for ensemble of outputs (decision function)
Supervision function	– Data explainers to report safe boundary conditions (documented within AI-FSM)

More detailed descriptions of the techniques used in OM stage will be provided in Section 4.

3.5. Relevant metrics

As an important part of AI-FSM aware solution, different sets of metrics are needed throughout the lifecycle. This section describes the initial set of candidate metrics to measure different perspectives of the FUSA-aware DL solution.

3.5.1. Explainability KPI and metrics

As suggested and supported by [147], the following KPIs can be used to measure explainability:

- **Faithfulness**: quantifies to what extent explanations follow the predictive behaviour of the model, asserting that more important features affect model decisions more strongly
- **Robustness**: measures to what extent explanations are stable when subject to slight perturbations in the input, assuming that the model output approximately stayed the same
- **Localisation**: tests if the explainable evidence is centred around a region of interest, which may be defined around an object by a bounding box, a segmentation mask, or a cell within a grid
- **Complexity**: captures to what extent explanations are concise, i.e., that few features are used to explain a model prediction
- **Axiomatic**: measures if explanations fulfil certain axiomatic properties
- **Randomisation**: tests to what extent explanations deteriorate as the data labels or the model, e.g., its parameters are increasingly randomised.
- **Saliency maps evaluations:** Saliency map evaluation quantifies how well the generated saliency map aligns with where the model should have actually focused.
- **Uncertainty estimations:** Measures the confidence of the predictions given by the models.
- **Expert feedback for evaluating** explainability**.**

3.5.2. DL component metrics

3.5.2.1. Performance metrics

ISO/IEC TR24929[11] suggests the following example statistical metrics that can be used for DL model performance measures:

- Root means square error (RMSE) of the prediction errors.
- Max error: Absolute or relative metrics, either measure the absolute deviation of relative in comparison with the variation domain.
- Actual/predicted correlation: Measure the linear correlation between the ground truth values and the predictions. For multiclass classification types of DL model, confusion matrix can be used.

We started with the following set of basic performance metrics:

- Condition positive/negative (CP, CN): Number of positive/negative samples in the dataset respectively.
- Prediction positive/negative: Number of samples classified as positive/negative by the DL model respectively.
- True Positive (TP): instance belongs to the class and is predicted as belonging to the class.
- True Negative (TN): instance does not belong to the class and is predicted as not belonging to the class.
- False Positive (FP): instance does not belong to the class and is predicted as belonging to the class.
- False Negative (FN): instance belongs to the class and is predicted as not belonging to the class.
- Intersection over Union (IoU)

- Speed (per inference constraint): Inference running time per prediction with regards to a specific hardware/middleware. For the SAFEXPLAIN, this refers to the platform selected by WP4.

The set of secondary metrics can then be derived from the basic metrics:

- Precision: TP/(TP+FP): also known as positive predictive value or relevance, indicates the proportion of results correctly classified as positive in the total of results classified as positive
- Accuracy (TP+TN)/(CP+CN): indicates the proportion of all objects that are correctly classified (as positive or negative)
- True negative rate (TN/(CN): also known as specificity or selectivity indicates the proportion of objects correctly classified as negative in the total number of negative objects.
- Negative Predictive Rate (TN/(FN+TN)): also known as negative predictive value or separation ability indicates the proportion of results correctly classified as negative in the total number of results classified as negatives.
- Recall: TP/(TP+FN): also known as true positive rate, sensitivity, or probability of detection. Indicates the proportion of objects correctly classified as positive in the total number of positive objects.
- mAP: mean AP across object classes
- FPR: False Positive Rate FP/(FP+TN): also known as fall-out or probability of false alarm, indicates the proportion of objects falsely classified as positive that are negative. Thus, the probability of a false alarm is given.
- FNR False Negative Rate FN/(FN+TP): also known as miss rate, indicates the proportion of objects falsely classified as negative in the total number of positive objects.
- Negative likelihood relation (FNR/TNR): indicates the ratio of the false negative rate to the true negative rate.
- F1Score: 2 x Precision x Recall/(Precision + Recall): combines the true positive rate and the positive predictive value using the harmonic mean
- F-beta: Extension of F1 score, but also with weight parameter beta to emphasize either precision or recall in the formula. When beta=1, it reflects F1 score.

More advanced metrics:

- Precision recall curve: Precision/recall pairs of metrics are computed at different output thresholds.
- Receiver Operating Characteristic (ROC): plot of the true positive rate against the false positive rate at different settings of the hyperparameters (e.g. decision threshold)
- Lift: relative performance of a prediction system against another control group (usually randomly selected)
- Top K accuracy: measures how often the correct labels is among the top K predictions of a DL multi-class classification model.
- Matthews correlation coefficient (MCC): a measure on a set of classifications ((TPxFN)-(FPxFN))/SquaredRoot((TP+FP)x(TP+FN)x(TN+FP)x(TN+FN))
- Area under curve (AUC w.r.t ROC or AP): integral of the ROC curve which represents the performance of a model for every threshold of classification. The ROC curve shows the true positive rate relative to the false positive rate.
- Balanced accuracy[148]: average recall obtained on each class
- Confusion Matrix: Square matrix to measure performance of multi class classification DL model. It's cells contains correlations between instances categorized by DL model as class r and instances labelled as class r in the ground truth dataset.
- Cohen kappa: a measure of inter-annotator agreement
- (N)DCG: (Normalized) Discounted Cumulative Gain
- DET curve: Error rates for different probability thresholds

- Hinge loss: upper bound on the number of mistakes made by a classifier. In the general, multiclass case, the margin is computed by the Crammer-Singer method.
- Hamming loss: measure how often the labels are incorrectly categorized by a classifier.

For semantic segmentation, similar metrics can be used to account for pixel/ point level, resulting in:

- Pixel/Point accuracy: measure how often the pixels/points in the data instance (e.g. image or point cloud) are correctly classified by the DL semantic segmentation model.
- Dice coefficient: 2xTP/(2xTP+FP+FN): Measure the similarity between predicted and ground truth segmentation masks. It defines as twice of the overlapping area between these two masks divided by the sum of the two mask's areas.
- IoU: TP/(TP+FP+FN): Measures the ratio of intersection over union of the two areas defined by predicted and ground truth segmentation masks.

Example safety related performance metrics of an object detection:

- Intersection over Union (IoU): Similar to segmentation, this metric measures the IoU of predicted and ground truth bounding boxes.
- Distance of the first detected object:
- Recall sensitivity
- Precision confidence
- F1-score

3.5.2.2. Robustness metrics

Robustness is often used to describe the ultimate ability of a system to maintain its level of performance under any circumstances including external interference or harsh environmental conditions. Robustness encompasses resilience, reliability and potentially more attributes, as related to proper operation of a system as intended by its developers[10].

The following Robustness metrics are considered by measuring the performance metrics stability over the expected variations:

- Input data variation, perturbations
- Adversarial attacks
- Over temporal dimension
- Over valid ranges and distributions of ODD/scenario parameter dimensions
- Scenario-based performance loss
- Over different environments: Training, verification, validation, and inference

Measures related to bias/fairness[9] to consider within the scope of robustness:

- Confusion matrix
- Equalized oods: An algorithm's decisions are independent of a category A given the input Y. This implies that true positive rates (TPR) are equal across demographic categories and false positive rates (FPR) are equal across demographic categories.
- Equality of opportunity: an algorithm's decisions that $\hat{Y}=1$ are independent of a category A given the input Y=1. This implies equal True Positive Rates (TPR) across demographic categories.
- Demographic parity: equal prediction rates between categories
- Predictive equality: equal false positive rates (FPR) across demographic categories.

3.5.3. Dataset metrics

AI-FSM defines the data requirements with the following specifications: (i) Completeness, (ii) Representativeness, (iii) Balance, (iv) Volume, (v) Degree of differentiation between datasets. This section provides an initial set of metrics that can be used to support requirement specification and compliance assessment of the datasets.

3.5.3.1. Data completeness metrics

Data completeness considers the sufficiency of data dimension ranges with regards to the ODD, operational scenarios, including the examples reflecting the effects of identified system failure modes.

The data completeness is thus measured against:

- If the data samples include sufficient range of environmental factors (within ODD)
- If the data samples include sufficient range of objects (types, appearances, positions)
- If the data samples include sufficient range of object occlusion levels
- If the data samples include sufficient range of examples reflecting the effects of identified system failure modes.

Metrics:

- Missing value counts: Number of missing values/total number of values
- Feature dimension completeness: Number of missing values/total number of values, or intersection of feature ranges and expected ranges.
- Completeness Index: Number of fully observed samples/total number of samples.
- ODD/Scenarios completeness: Number of data samples/expected number of samples to represent all possible variations with a defined sampling strategy.
- Percentage of complete cases: number of scenario cases that have representative data samples in the dataset.
- Quantitative Projection Coverage[149]

3.5.3.2. Data representativeness metrics

Data relevancy considers the intersection between the dataset and the intended functionalities of the DL component(s) to support the system within the defined ODD and operational scenarios.

Relevancy is thus measured against:

- How well the scenes represent the ODD/scenarios in real-world.
- How well the sensor and data capture/synthesis settings represent the real-world settings.

Metrics:

- Total population: Total number of data samples (instances, data points) in the dataset.
- Prevalence: the proportion of a particular class in the total number of samples
- Feature distribution: Distribution of values within each data feature
- Number of annotation classes
- Data dimensionality: Number of data dimensions comparing with the expected dimensionality to fully describe ODD/Operational scenarios.
- Annotation completeness
- Annotation quality scores, inter-annotation agreement
- Entropy and diversity in features or annotations
- Availability of data samples to support generating test cases by defined V&V strategy.

3.5.3.3. Data balance metrics

Data balance considers comparable representations of data samples for each relevant class and feature. It can be measured by distribution metrics measured for any subspace or the entire dataset.

Examples include: DAC (Data Agreement Criterion[150]), DRC (Data Representativeness Criterion[151])

3.5.3.4. Data volume metrics

Candidates of data volume metrics are:

- Number of samples (observations)
- Number of features (input variables, dimensions)
- Data sample sizes
- Data sampling density: Such as image resolution, number of bins...
- Data sparsity: Measure the proportion of missing data in the dataset. For example a Lidar pointcloud that do not provide data measures for all possible voxels within its field of view
- Cardinality: Measure the quantified levels of data per specific dimensions (e.g. 8bit RGB)

3.5.3.5. Data accuracy metrics

This property considers how measurement (and measurement-like) issues can affect the way that samples reflect the intended operational domain. It covers aspects like sensor accuracy and labelling errors.

Metrics:

- Sensor accuracy
- Inter-annotator agreement

3.5.3.6. Dataset distances

Distance between a data point (data sample) and a dataset:

- Mahalanobis[152]
- Cosine similarity

Distance between two datasets:

- Bhattacharyya coefficient[153]
- Wasserstein metric (earth mover): Measure the distance between probability distributions on a given metric space. Its variants include sliced Wasserstein[154] and max-sliced Wasserstein[155] metrics.
- Kullback_Leibler (KL) divergence[156]: The KL divergence plays the role of L2 distance of projecting an distribution (a dataset) onto a convex set of another distribution.
- Jensen-Shannon (JS) divergence[157]: asymmetric metric the measure the relative entropy (closely related to KL divergence).
- Pearson Correlation Coefficient
- Hellinger distance

3.5.4. Supervisor performance metrics

Supervisors refers to techniques that address uncertainty challenges of DL-based system during the OM stage (more details are provided in Section 4.1).

The key objective of the supervisor is to be able to detect and suggest rejections of all outliers (cases that are not within the AI-FSM specification of the DL component) while accept all inliers (cases that are within

the specification). Therefore, the performance of a safety supervisor can be measured by the following basic metrics:

- False Positive: Supervisor fails to accept an inlier datapoint.
- False Negative: Supervisor fails to reject an outlier datapoint.

More complex performance metrics can be derived from the basic metrics as follows:

- AUROC: Area under ROC curve (Receiver Operating Characteristic). Trade-off between true and false positives
- AUPRC: Area under PR curve (precision-recall). Trade-off between true and false positives
- TPR05: true positive rate at 5% false positive rate
- P95: precision at 95% recall
- FNR95: false negative rate at 95% false positive rate. Measure over-confident that an outlier belongs to the inlier dataset.
- CBPL: coverage breakpoint at performance level. How restrictive the supervisor has to be to recover the original performance on the dataset including outlier samples, or if it is not achievable with the given supervisor.
- CBFAD: coverage breakpoint at full anomaly detection. If the supervisor can completely exclude outliers from the data, i.e., can we achieve full anomaly detection for some non-trivial (zero) value of coverage.

3.6. Structural coverage of DNN

Coverage refers to the extent to which a given verification activity has satisfied its objectives. Coverage measures can be applied to any verification activity, although they are most frequently applied to testing activities. Structural coverage aims at measuring the degree of confidence for syntactic correctness of the physical implementation that the test set achieves.

3.6.1. Structural coverage of traditional software

Structural coverage is the metric set used in software testing to measure the coverage of source codes of a software that have been executed during the test. The structural coverage analysis shall identify the coverage of different software elements including statements, branches, paths, and conditions.

3.6.2. Structural coverage of DNN

In contrast to the traditional software, a DL component is generally defined by its architecture and trained parameters, leading to differences in definition of structural items.

Typically, the performance metrics measured on a set of tests are used as the evaluation of the DL's performance. However, it is unclear if the selected test set cover all possible behaviours of the DL component (e.g. corner cases). Coverage metrics are thus proposed to measure the test strategy adequacy[158].

Similar to code coverage criteria, researches propose several coverage criteria for DNN [159], [160], [161], [162], observing the similarity between a neuron activation and a software statement.

Structural coverage metrics are used to measure the adequacy of DNN testing. The following coverage types are found in the literature:

- Block execution coverage: Neuron, block/layer of neurons.
    - Neuron Coverage[161] (NC) measures the coverages of activated neurons per layer, set of layers of the entire network (activation values greater than some threshold, provided a test dataset).

  - Neuron Boundary Coverage[160] (NBC): measures how many corner-case regions (w.r.t. both of the upper boundary and the lower boundary values) have been covered by the given test dataset
  - Strong Neuron Activation Coverage (SNAC): Percentage of neurons that are activated by at least one data point of the test dataset.
  - K-Multisection Neuron Coverage[160] (KMNC): measures how thoroughly the given set of test dataset covers the range [low, high] (where low, high denote lower and upper boundary of the output value respectively) of a neuron. The measure is taken in forms of a histogram of k-bin over the range.
  - Top-K Neuron Coverage[160] (TKNC): Measures how many neurons have once been the most active k neurons on each layer.
  - Top-K Neuron Patterns (TKNP): measures the number of different activation patterns for the most active k neurons on each layer.
  - Safety coverage: As described in [144], the input space is discretised with a set of hyper-rectangles, and then one test case is generated for each hyper-rectangle. This is referred to as “safety coverage”.
- Decision coverage: the metric is well known for white-box testing of traditional software, however because of the complexity of DL models, concept adaptation such as Neuron Path Coverage[163] has been proposed.
- Condition coverage: Coverage of input.
- Surprise Adequacy[159]: measures the relative novelty (surprise) of a given new input x with respect to the datasets used within development lifecycle.
  - Distance-based Surprise Adequacy (DSA): measures if x is closer to the target class or a different class.
  - Likelihood-based Surprise Adequacy (LSA): measures a normalized distance between the activation values of x and those of individual inputs.
- Modified condition decision coverage[164] (MC/DC): MC/DC was developed by NASA and is used in avionics software development guidance to ensure adequate testing of applications with the highest criticality. Sun et al.[165] proposed several variations of MC/DC, including: Sign-sign coverage (SS coverage), Value-sign coverage (VS coverage), Sign-value coverage (SV coverage), and Value-value coverage (VV coverage)

4. Improved DL component robustness and online monitorability

While the AI-FSM compliance is addressing systemic safety, alignment and robustness, the reference safety architecture pattern (D2.2) will help to secure the Operation and Monitoring stage (i.e. detect anomalies, monitoring predictions and discover unexpected model behaviours). Figure 30 provides a recap of safety architecture pattern provided by D2.2. This section focuses on different XAI and DL specific techniques that can be used to support the three components (marked 1,2,3 in the figure): L1 Diagnosis and Monitoring (L1DM), Decision function and partly supervision function.

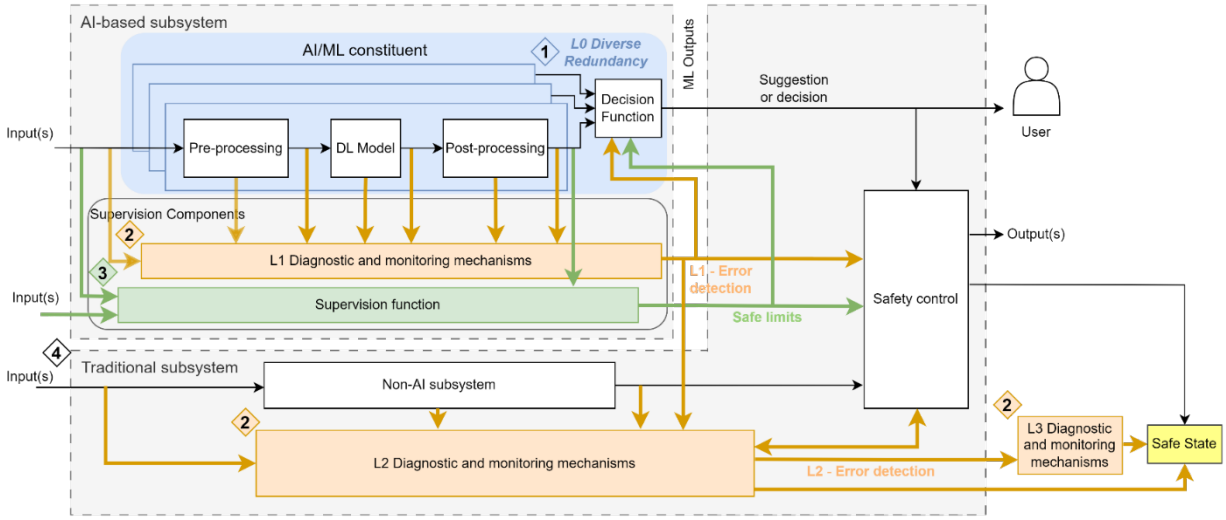

Figure 30: SAFEXPLAIN reference safety architecture pattern (D2.2)

The L1DM component consists of a number of supervisory monitors. For shorter terms, a supervisory monitor is also referred to as a supervisor or a monitor. This section discusses different techniques that can be used as a supervisor to monitor the DL component for ensuring the system's safety during OM stage.

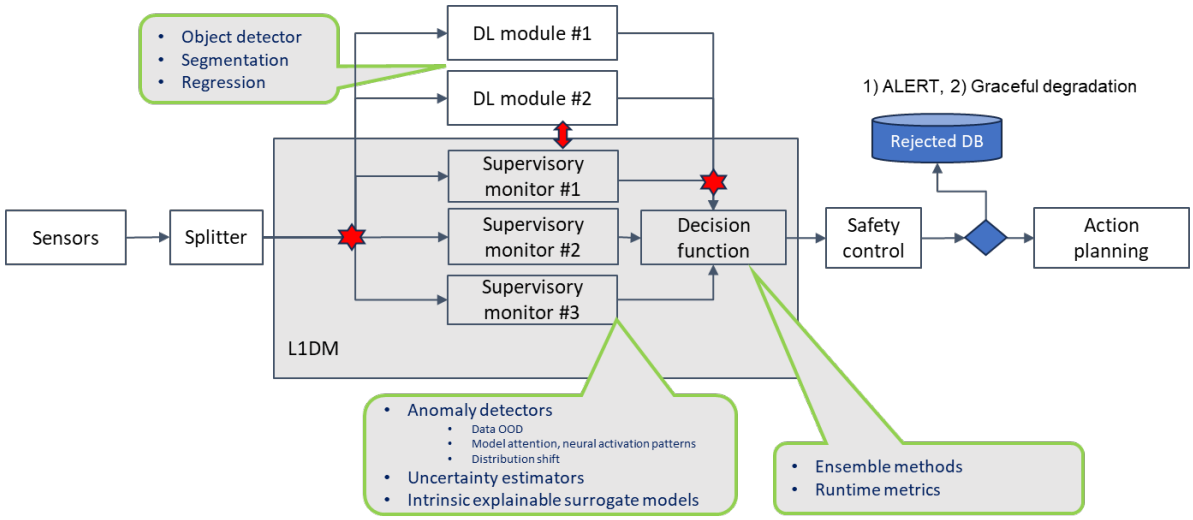

Figure 31: Multiple supervisors architecture to support L1DM

Figure 31 depicts a reference architecture where a selected set of supervisors are used to verify in real-time whether runtime input data, DL model behaviour and its output predictions are not outside of "known" area (where the baselines have been set within AI-FSM lifecycle phases).

The three checkpoints (marked as red stars in Figure 31) represent the three types of checks that the supervisors monitor:

- Input data check: During the PhDM phase, the distribution description(s) of datasets used for training/verification of the DL model have been specified/logged. These descriptions are used as the ground truth for the supervisor(s) to detect outlier data point.
- Model (local) behaviour check: Model local behaviours can be represented by the activation patterns found at a specific neural layer (or combination of layers). The activation pattern is again, comparing to the logged patterns (from the model responses to the training/validation datasets during PhLM/PhIM)

- Output check: Distribution shift or time consistency check

Another component within supervision component is *supervision function*, providing safe limits. This component is currently considered out of the work focus. However, the function can collect its inputs from the expected boundaries (together with uncertainty measures such as expected standard deviations) reported during AI-FSM.

A certified surrogate model as described for supervisory monitor, such as extracted rule-based model may also be useful for this function.

The decision function, itself can be engineered with some candidate algorithms proposed within this deliverable. While a subset of candidate algorithms are provided in D2.2, the proposed solutions within this section are focusing on using ML based approaches.

4.1. Supervisory monitor algorithms

This section complements Section 3.2.1 in D2.2, with focus on supervisory monitor algorithms that can be used within L1DM mechanisms, addressing risks related to insufficiency of DL model performance resulted from the DL uncertainty challenges.

4.1.1. Out of Distribution detection

Anomaly detection algorithms are used to detect out of distribution (OOD) anomalies in:

- Input data anomalies: Data descriptive analysis, OOD detection of input data against known data distribution/drift detection.
- Inner working of DL component: Interim feature anomalies (comparison with known expected activation patterns)
- Output data: Drift detection (over timeseries)

To address the domain uncertainties, dataset descriptors such as Variational AutoEncoder (VAE) descriptors can be used to reconstruct the input data from the extracted feature and compare with the captured data to derive reconstruction error (e.g. L2 errors) per prediction.

Noted that this approach requires logged specification of related reconstruction errors by using the same descriptors to specify the three datasets within PhDM.

A distance metric will then be used to measure the distance between the datapoint (the actual input data sample during the operation) and the “known” dataset specifications, in the form of distributions. The data distribution specification has been logged during the AI-FSM lifecycle as mentioned in Section 3.3.4

Metrics such as Mahalanobis distance can be used, with the assumption that the based distributions are some mixtures of normal distributions.

$$Anomaly\_score = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Where μ is the mean vector of the Gaussian Mixture Distribution (GMD), x is the actual datapoint, and Σ is the covariance matrix.

In the case of measuring distance between distributions (e.g. a sequences of input data forms a runtime distribution), other distance metrics between distributions can be used. Examples include Bhattacharyya or KL divergence.

Figure 32 illustrates an OOD techniques using a trained VAE as data descriptor. The VAE model has been trained with the same training/validation and test datasets as used for the development of DL model. The “*Anomaly Score*” reports the reconstruction errors. Noted that VAE is a special family of Auto Encoder

models, where the reconstruction quality is not optimized on single data sample but to better describe the entire dataset (data distribution).

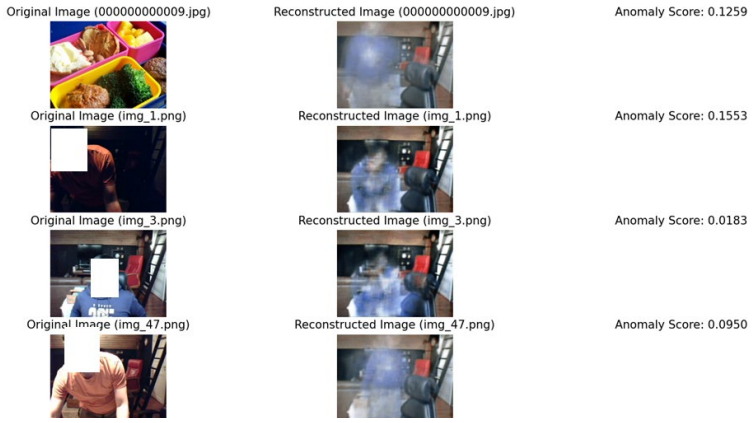

Figure 32: Variational AutoEncoder as Anomaly detection supervisor.

Similar techniques can also be applied for neuron activation patterns extracted from the DL model at a selected layer (or a weighted combination of layers). In this case, the descriptors will be trained on the corresponding set of relevant neuron activation patterns w.r.t. the datasets, and during OM stage, the actual activation pattern will be compared against the known "activation pattern distribution" as logged during the AI-FSM phase.

Other OOD techniques that can be considered include AEGMM[166], Prophet[167], Likelihood Ratios[168].

4.1.2. Uncertainty Estimators

Uncertainty analysis can help to (i) identify OOD cases and (ii) erroneous prediction of In Distribution cases. Identification of OOD cases will help the supervisory monitor to estimate the validity of predictions, while quantification of erroneous prediction of In Distribution cases can be used to decide if the known uncertainties (logged during AI-FSM) are acceptable for the final trustworthy decision.

Gawlikowski et al. [169] proposes 4 uncertainty estimator categories:

- Single deterministic methods: Provides uncertainty quantification prediction based on single forward pass of deterministic (inference) DL models, either by external or internal methods. Examples include:
  - External methods: Gradient Metrices, Additional network for uncertainties, distance to training data.
  - Internal methods: Prior networks, evidential NN, gradient penalties[170]
- Bayesian methods: Stochastic type of DL component (DL models with stochastic nature of prediction in inference, not within the defined scope of SAFEXPLAIN)
  - Variational inference
  - Sampling methods
  - Laplace approximation
- Ensemble methods: Combine predictions of different parallel DL components at operational. This type of techniques will be applied and discussed in section 0
  - Weight sharing
  - Reduce members.
  - Training strategies
- Test-time augmentation methods: Augment input data during testing to generate several predictions to evaluate the prediction uncertainty.

Reader can consult Abdar et al[171] for a more intensive review of uncertainty quantification techniques.

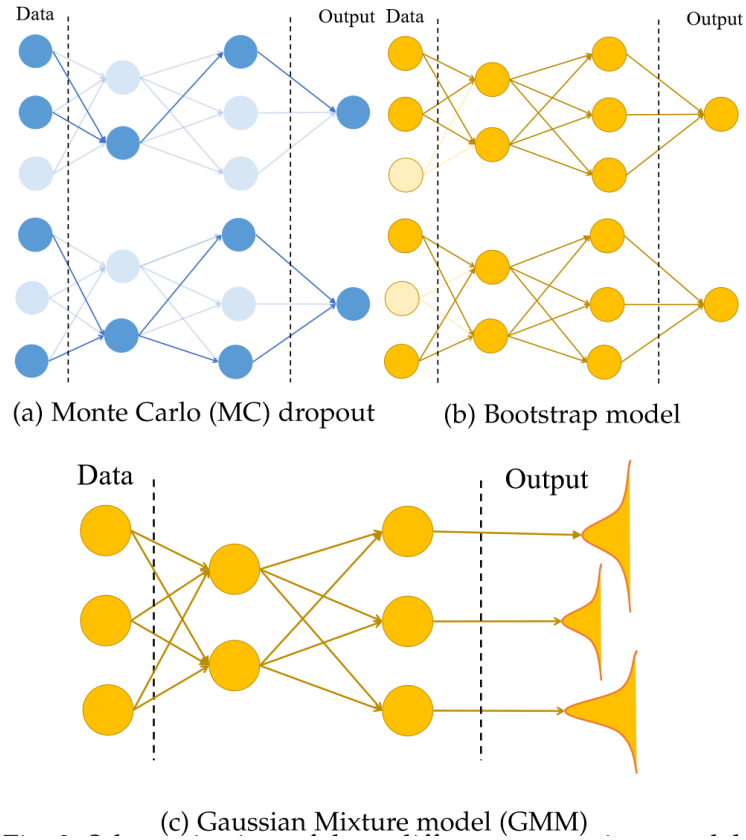

Figure 33: Schematic view of three different uncertainty models [161]

A schematic view of three different uncertainty models is illustrated in Figure 33(courtesy [171]).

From a different perspective, considering the three types of uncertainties as in Section 3.2, the relevant methods are described below:

- Domain uncertainties: Refer to the uncertainty level that the captured/prepared dataset represent the stated problem. This type of uncertainties is resulted by sensor properties, data capture environment, data sampling and data preprocessing algorithms.
- Epistemic uncertainties: Refer to the uncertainty level of the DL model to model the stated problem. Can be measured by:
  - Repeated measurements: In case of highly correlated timeseries data (such as video sequences), we can assume that consecutive datapoints are repeated measurements with noises and measure the DL's output uncertainty.
  - Input perturbation (or model weight perturbation): Inject noises to input and model parameters to measure the output uncertainty.
  - Modified DL model that can provide uncertainty estimates itself, e.g. [171] dropout as a Bayesian approximation [83], stochastic ensembles, Monte Carlo dropout, Bootstrap model, Gaussian Mixture model.
  - Model redundancy: Several models can be used to provide predictions from the same input.
- Aleatoric uncertainties: Refer to the irreducible uncertainty related to the stated problem.
  - Annotation uncertainty
  - Occlusion uncertainty

- Sensor performance degradation

Conformal prediction, also known as conformal inference [172], [173], [174], is a framework for generating statistically rigorous uncertainty sets or intervals for the predictions of the DL component. Unlike the methods described above, conformal prediction is model-agnostic and provides intervals (boundaries) that indicate uncertainty levels. This capability allows it to assist in controlling the DL-based system to operate within acceptable levels of risk.

Conformal Risk Control[175] combines the principles of conformal prediction and safety risk control (managing and mitigating hazards identified by safety critical systems, represented by allocated metrics to the DL component):

- Specify a bounded region around the model prediction by the uncertainty of the prediction.
- Associate risk (hazard) metric to different DL model predictions
- Establishes thresholds and rules considering both uncertainty and risk levels

Metrics: Uncertainty confidence indicator[176]

4.1.3. Surrogate models

Surrogate models (trained and tested together with the main DL components during AI-FSM) can be used to provide results in parallel. The interpretable logics behind the surrogate models shall be approved within the AI-FSM process. If the unexpected mismatch (e.g. if approximation errors are out of distributions) occurs between the main model prediction and the surrogate model outputs, it can be interpreted that the DL component may no longer use the certified logics for its detection in this specific case.

The rejected corner cases will be logged in to rejection database to be investigated in the next release development lifecycle.

The evolution of surrogate models (over DL release iterations) thus can be used to justify if and how the new release significantly updates its internal logics and thus may reduce V&V test efforts in subsequent iterations.

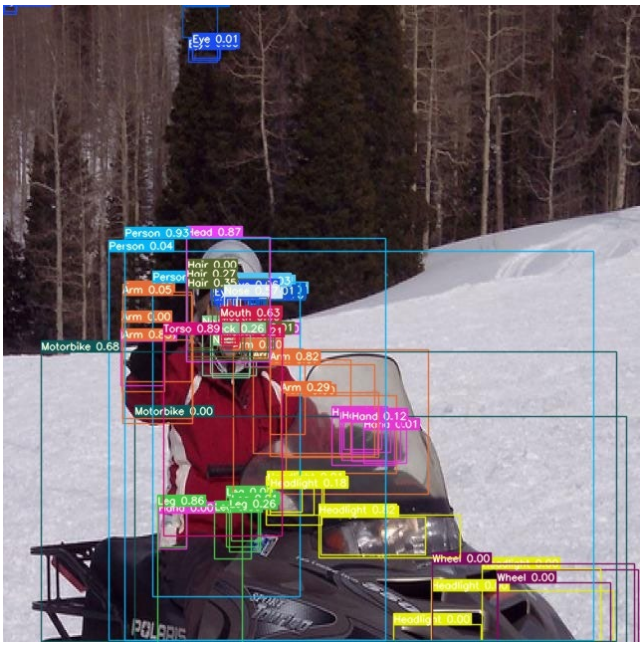

Figure 34: Detection of body parts and person by PASCAL-VOC YOLOv7

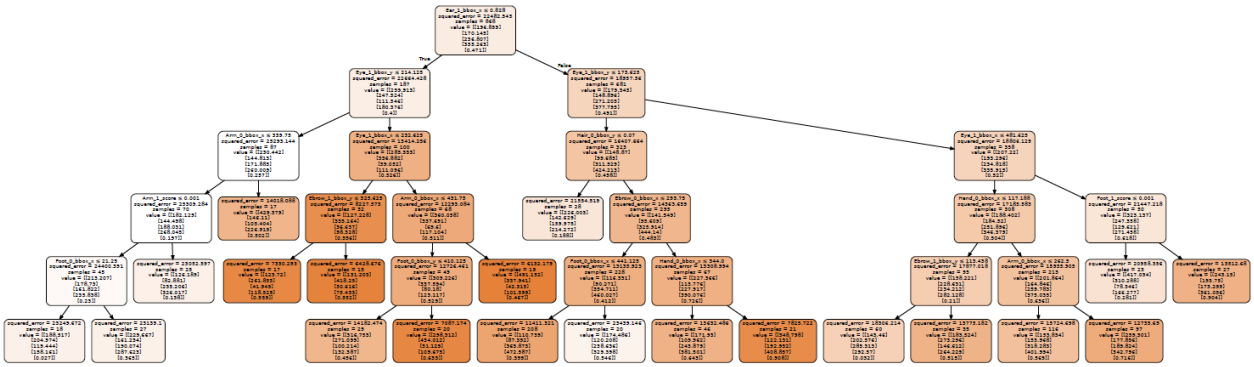

Figure 35: Decision Tree surrogate model on person detection w.r.t. bodypart detections (trained to approximate PASCAL-VOC YOLOv7

Figure 34 and Figure 35 illustrate an example of using Decision Tree as a surrogate model to approximate the target DL model (YOLOv7). The DT surrogate model provides predictions of detected person (bounding box, score) as DT based rules over the detected bodypart. Noted that in this specific example, the surrogate model does not work directly with the input data, but with the extracted prototypes from the input data, in the forms of detected body parts. The bodyparts in this example are provided by the same YOLO model, however the approach can be applied for any other extracted prototypes when other prototype extractors are selected. Other intrinsic models rather than DT can also be used.

4.2. Decision function

The decision function collects outputs from several parallel components, to aggregate into a final decision/suggestion to the safety control component and/or human operators. The collected outputs include:

- Prediction results from DL component(s) together with the confidence level.
- Anomaly scores, applied for: (i) input data, (ii) model activation at specific layer or a combination of layers, and (iii) output prediction.
- Approximated predictions from the surrogate models
- Uncertainties estimation: Uncertainties are provided by either or both.
    - Uncertainty-aware DL component (to provide uncertainty estimates besides the prediction).
    - Uncertainty propagation: Feature relevance algorithms in combination with data uncertainties.

The decision function can be selected from or combination of Dempster's rule (evidence theory) and/or ensemble methods:

- Dempster rules: combining evidence from different sources. In the case of SAFEXPLAIN OM, the evidence sources are output of several DL components or surrogate models.
- Model averaging: Weighted average of outputs from different DL and surrogate models.
- Model selection: Voting (Note: this technique is also mentioned in D2.2)
    - Hard voting: Majority vote
    - Soft voting: Weighted average of prediction scores (confidence/certainty). An example of soft voting is provided in Figure 36.
- Model combination/bagging: Surrogate model (decision tree, random forest, simple MLP, GLN...) that learns how to combine the inputs (using the datasets during the AI-FSM development cycle). The ensemble methods can also be used during the AI-FSM to derive redundant DL models that

focus the performance on different subset of the dataset, e.g. by negative correlation learning. Examples include:

- Random Forest: an ensemble of decision tree. The RF will be trained during the AI-FSM lifecycle to model how one should ensemble the outputs from DL models and supervisors in the inlier conditions.
- Stacking: Stacked generalization[177] consists in stacking the output of individual estimator and use a classifier to compute the final prediction. Stacking allows to use the strength of each individual estimator by using their output as input of a final estimator.

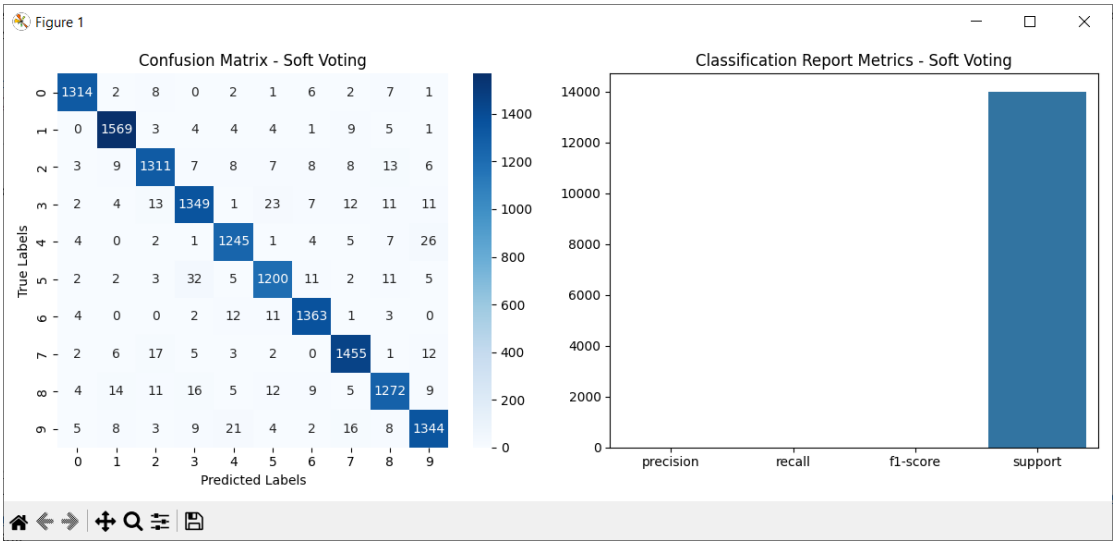

Figure 36: Use soft voting as ensemble method to combine predictions from 3 DL models (Logistic regression, SVC, DT), dataset used MNIST

4.3. Potential supports for supervision function

Supervision function is a subcomponent within supervision components applying elements of control theory to minimally bound AI operations (D2.2).

While the proposed techniques described within this section are not focusing on providing solutions for supervision, potential usages for supervision function, especially for supporting control barrier functions (CBF):

- Uncertainty aware DL models: The uncertainty aware DL models provide CBF with uncertainty estimates in the forms of a statistical distribution, that can be used as the inputs for CBF to adjust safety margins accordingly. Examples include deducing speeds or increasing distance to target vehicles.
- Anomaly detections: The anomaly scores provided by OOD detectors can be used for CBF to decide if backup solution(s) shall be triggered.

4.4. Integration with SAFEXPLAIN deployment platform

An example of a DL component and VAE based supervisor has been added to DLLib following the reference architecture with initial adaptation for later porting to ROS2 architecture.

References

[1] SAFEXPLAIN, "D2.1: SAFEXPLAIN Safety Lifecycle Considerations." Deliverable of the HEU SAFEXPLAIN project, Grant Agreement No. 101069595, 2024.

[2] SAFEXPLAIN, "D2.2: SAFEXPLAIN DL safety architectural patterns and platform." Deliverable of the HEU SAFEXPLAIN project, Grant Agreement No. 101069595, 2024.

[3] ISO/IEC 22989:2022, "Information technology — Artificial intelligence — Artificial intelligence concepts and terminology." 2022. Accessed: Mar. 01, 2024. [Online]. Available: https://www.iso.org/standard/74296.html

[4] ISO/IEC 23053:2022, "Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)." 2022. Accessed: Mar. 01, 2024. [Online]. Available: https://www.iso.org/standard/74438.html

[5] ISO/IEC TR 5469:2024, "Artificial intelligence — Functional safety and AI systems." 2024. Accessed: Mar. 01, 2024. [Online]. Available: https://www.iso.org/standard/81283.html

[6] ISO 21448:2022, "Road vehicles — Safety of the intended functionality." 2022. Accessed: Mar. 01, 2024. [Online]. Available: https://www.iso.org/standard/77490.html

[7] ISO 26262:2018, "Road vehicles — Functional safety." 2018. Accessed: Mar. 01, 2024. [Online]. Available: https://www.iso.org/standard/68383.html

[8] "IEEE Standard for Transparency of Autonomous Systems," *IEEE Std 7001-2021*, pp. 1–54, Mar. 2022, doi: 10.1109/IEEESTD.2022.9726144.

[9] ISO/IEC TR 24027:2021, "Information technology — Artificial intelligence (AI) — Bias in AI systems and AI aided decision making." 2021. Accessed: Mar. 01, 2024. [Online]. Available: https://www.iso.org/standard/77607.html

[10] ISO/IEC TR 24028:2020, "Information technology — Artificial intelligence — Overview of trustworthiness in artificial intelligence." 2020. Accessed: Mar. 01, 2024. [Online]. Available: https://www.iso.org/standard/77608.html

[11] ISO/IEC 24029:2023, "Artificial intelligence (AI) — Assessment of the robustness of neural networks." 2023. Accessed: Mar. 01, 2024. [Online]. Available: https://www.iso.org/standard/79804.html

[12] P. J. Phillips *et al.*, "Four principles of explainable artificial intelligence," National Institute of Standards and Technology (U.S.), Gaithersburg, MD, NIST IR 8312, Sep. 2021. doi: 10.6028/NIST.IR.8312.

[13] Board of Governors Federal Reserve System, "SR 11-7 on guidance on Model Risk Management." Accessed: Jan. 03, 2024. [Online]. Available: https://www.federalreserve.gov/supervisionreg/srletters/sr1107.htm

[14] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR," *SSRN Journal*, 2017, doi: 10.2139/ssrn.3063289.

[15] A. Gosain and J. Singh, "Investigating structural metrics for understandability prediction of data warehouse multidimensional schemas using machine learning techniques," *Innov. Syst. Softw. Eng.*, vol. 14, no. 1, pp. 59–80, Mar. 2018, doi: 10.1007/s11334-017-0308-z.

[16] R. S. Michalski, "A Theory and Methodology of Inductive Learning," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 83–134. doi: 10.1007/978-3-662-12405-5_4.

[17] F. Clemente, G. M. Ribeiro, A. Quemy, M. S. Santos, R. C. Pereira, and A. Barros, "ydata-profiling: Accelerating data-centric AI with high-quality data," *Neurocomputing*, vol. 554, p. 126585, Oct. 2023, doi: 10.1016/j.neucom.2023.126585.

[18] F. Bertrand, "sweetviz: A pandas-based library to visualize and compare datasets." Accessed: Feb. 26, 2024. [OS Independent]. Available: https://github.com/fbdesignpro/sweetviz

[19] J. Wexler, "Facets: An open source visualization tool for machine learning training data," *Google Open Source Blog*, 2017.

[20] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[21] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: Uniform Manifold Approximation and Projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018, doi: 10.21105/joss.00861.

[22] "Data Readiness – ianmarsh.org." Accessed: Feb. 26, 2024. [Online]. Available: https://ianmarsh.org/data-readiness/

[23] T. Shi, B. Yu, E. E. Clothiaux, and A. J. Braverman, "Daytime Arctic Cloud Detection Based on Multi-Angle Satellite Data with Case Studies," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 584–593, 2008.

[24] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Definitions, methods, and applications in interpretable machine learning," *Proceedings of the National Academy of Sciences*, vol. 116, no. 44, pp. 22071–22080, Oct. 2019, doi: 10.1073/pnas.1900654116.

[25] T. Gebru *et al.*, "Datasheets for Datasets." arXiv, Dec. 01, 2021. doi: 10.48550/arXiv.1803.09010.

[26] S. Holland, A. Hosny, S. Newman, J. Joseph, and K. Chmielinski, "The Dataset Nutrition Label: A Framework To Drive Higher Data Quality Standards." arXiv, May 09, 2018. doi: 10.48550/arXiv.1805.03677.

[27] E. M. Bender and B. Friedman, "Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 587–604, 2018, doi: 10.1162/tacl_a_00041.

[28] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, vol. 5, no. 4, pp. 2403–2424, Dec. 2011, doi: 10.1214/11-AOAS495.

[29] K. S. Gurumoorthy, A. Dhurandhar, G. Cecchi, and C. Aggarwal, "Efficient Data Representation by Selecting Prototypes with Importance Weights." arXiv, Aug. 12, 2019. doi: 10.48550/arXiv.1707.01212.

[30] A. Dhurandhar *et al.*, "Explanations based on the missing: towards contrastive explanations with pertinent negatives," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, in NIPS'18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 590–601.

[31] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114 [cs, stat]*, May 2014, Accessed: Dec. 06, 2021. [Online]. Available: http://arxiv.org/abs/1312.6114

[32] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational Inference of Disentangled Latent Concepts from Unlabeled Observations," presented at the International Conference on Learning Representations, Mar. 2023. Accessed: Mar. 23, 2023. [Online]. Available: https://openreview.net/forum?id=H1kG7GZAW

[33] B. Ustun and C. Rudin, "Supersparse linear integer models for optimized medical scoring systems," *Machine Learning*, vol. 102, no. 3, pp. 349–391, Mar. 2016, doi: 10.1007/s10994-015-5528-6.

[34] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker, "Accurate intelligible models with pairwise interactions," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, in KDD '13. New York, NY, USA: Association for Computing Machinery, Aug. 2013, pp. 623–631. doi: 10.1145/2487575.2487579.

[35] G. P. J. Schmitz, C. Aldrich, and F. S. Gouws, "ANN-DT: an algorithm for extraction of decision trees from artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1392–1401, Nov. 1999, doi: 10.1109/72.809084.

[36] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable Decision Sets: A Joint Framework for Description and Prediction," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1675–1684. doi: 10.1145/2939672.2939874.

[37] J. Jung, C. Concannon, R. Shroff, S. Goel, and D. G. Goldstein, "Simple Rules for Complex Decisions." Rochester, NY, Feb. 16, 2017. doi: 10.2139/ssrn.2919024.

[38] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowl Inf Syst*, vol. 14, no. 1, pp. 1–37, Jan. 2008, doi: 10.1007/s10115-007-0114-2.

[39] D. Alvarez-Melis and T. S. Jaakkola, "Towards robust interpretability with self-explaining neural networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, in NIPS'18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 7786–7795.

[40] M. Hind *et al.*, "TED: Teaching AI to Explain its Decisions," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, in AIES '19. New York, NY, USA: Association for Computing Machinery, Jan. 2019, pp. 123–129. doi: 10.1145/3306618.3314273.

[41] Q. Zhang, Y. N. Wu, and S.-C. Zhu, "Interpretable Convolutional Neural Networks." arXiv, Feb. 14, 2018. doi: 10.48550/arXiv.1710.00935.

[42] M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez, "Beyond sparsity: tree regularization of deep models for interpretability," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, in AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, Feb. 2018, pp. 1670–1678.

[43] D. Shah, V. Trivedi, V. Sheth, A. Shah, and U. Chauhan, "ResTS: Residual Deep interpretable architecture for plant disease detection," *Information Processing in Agriculture*, vol. 9, no. 2, pp. 212–223, Jun. 2022, doi: 10.1016/j.inpa.2021.06.001.

[44] M. Brahimi, S. Mahmoudi, K. Boukhalfa, and A. Moussaoui, "Deep interpretable architecture for plant diseases classification," arXiv.org. Accessed: Mar. 03, 2024. [Online]. Available: https://arxiv.org/abs/1905.13523v2

[45] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," *arXiv:1602.04938 [cs, stat]*, Aug. 2016, Accessed: Jul. 12, 2021. [Online]. Available: http://arxiv.org/abs/1602.04938

[46] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-Precision Model-Agnostic Explanations," *AAAI*, vol. 32, no. 1, Apr. 2018, doi: 10.1609/aaai.v32i1.11491.

[47] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization." arXiv, Dec. 13, 2015. Accessed: Mar. 23, 2023. [Online]. Available: http://arxiv.org/abs/1512.04150

[48] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 618–626. doi: 10.1109/ICCV.2017.74.

[49] R. L. Draelos and L. Carin, "Use HiResCAM instead of Grad-CAM for faithful explanations of convolutional neural networks." arXiv, Nov. 20, 2021. doi: 10.48550/arXiv.2011.08891.

[50] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for Simplicity: The All Convolutional Net," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6806

[51] H. Wang *et al.*, "Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, USA: IEEE, Jun. 2020, pp. 111–119. doi: 10.1109/CVPRW50498.2020.00020.

[52] R. Ibrahim and M. O. Shafiq, "Augmented Score-CAM: High resolution visual interpretations for deep neural networks," *Knowledge-Based Systems*, vol. 252, p. 109287, 2022, doi: https://doi.org/10.1016/j.knosys.2022.109287.

[53] T. Yamauchi and M. Ishikawa, "Spatial Sensitive GRAD-CAM: Visual Explanations for Object Detection by Incorporating Spatial Sensitivity," in *2022 IEEE International Conference on Image Processing (ICIP)*, Oct. 2022, pp. 256–260. doi: 10.1109/ICIP46576.2022.9897350.

[54] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[55] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proceedings of the 2011 International Conference on Computer Vision*, in ICCV '11. USA: IEEE Computer Society, Nov. 2011, pp. 2018–2025. doi: 10.1109/ICCV.2011.6126474.

[56] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, in ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 3145–3153.

[57] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic Attribution for Deep Networks." arXiv, Jun. 12, 2017. doi: 10.48550/arXiv.1703.01365.

[58] J. H. Friedman, "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991, doi: 10.1214/aos/1176347963.

[59] D. W. Apley and J. Zhu, "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 82, no. 4, pp. 1059–1086, Jun. 2020, doi: 10.1111/rssb.12377.

[60] B. Kim, R. Khanna, and O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, in NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, pp. 2288–2296.

[61] A. Van Looveren and J. Klaise, "Interpretable Counterfactual Explanations Guided by Prototypes," arXiv.org. Accessed: Mar. 03, 2024. [Online]. Available: https://arxiv.org/abs/1907.02584v2

[62] L. S. Shapley, "17. A Value for n-Person Games," in *Contributions to the Theory of Games (AM-28), Volume II*, H. W. Kuhn and A. W. Tucker, Eds., Princeton: Princeton University Press, 1953, pp. 307–318. doi: doi:10.1515/9781400881970-018.

[63] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 4768–4777.

[64] H. Kawauchi and T. Fuse, "SHAP-Based Interpretable Object Detection Method for Satellite Imagery," *Remote Sensing*, vol. 14, no. 9, 2022, doi: 10.3390/rs14091970.

[65] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," presented at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[66] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," *PLOS ONE*, vol. 10, no. 7, p. e0130140, Jul. 2015, doi: 10.1371/journal.pone.0130140.

[67] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Muller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 1st ed. Springer Publishing Company, Incorporated, 2019.

[68] H. Tsunakawa, Y. Kameya, H. Lee, Y. Shinya, and N. Mitsumoto, "Contrastive Relevance Propagation for Interpreting Predictions by a Single-Shot Object Detector," in *2019 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2019, pp. 1–9. doi: 10.1109/IJCNN.2019.8851770.

[69] M. Dreyer, R. Achtibat, T. Wiegand, W. Samek, and S. Lapuschkin, "Revealing Hidden Context Bias in Segmentation and Object Detection through Concept-specific Explanations," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2023, pp. 3829–3839. doi: 10.1109/CVPRW59228.2023.00397.

[70] A. Karasmanoglou, M. Antonakakis, and M. Zervakis, "Heatmap-based Explanation of YOLOv5 Object Detection with Layer-wise Relevance Propagation," in *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*, Jun. 2022, pp. 1–6. doi: 10.1109/IST55454.2022.9827744.

[71] R. Achtibat *et al.*, "From Attribution Maps to Human-Understandable Explanations through Concept Relevance Propagation," *Nat Mach Intell*, vol. 5, no. 9, pp. 1006–1019, Sep. 2023, doi: 10.1038/S42256-023-00711-8.

[72] V. Contreras *et al.*, "A DEXiRE for Extracting Propositional Rules from Neural Networks via Binarization," *Electronics*, vol. 11, no. 24, 2022, doi: 10.3390/electronics11244171.

[73] M. E. Zarlenga, Z. Shams, and M. Jamnik, "Efficient Decompositional Rule Extraction for Deep Neural Networks." arXiv, Nov. 24, 2021. Accessed: Mar. 03, 2024. [Online]. Available: http://arxiv.org/abs/2111.12628

[74] M. G. Augasta and T. Kathirvalavakumar, “Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems,” *Neural Processing Letters*, vol. 35, no. 2, pp. 131–150, Apr. 2012, doi: 10.1007/s11063-011-9207-8.

[75] J. R. Zilke, E. Loza Mencía, and F. Janssen, “DeepRED – Rule Extraction from Deep Neural Networks,” in *Discovery Science*, T. Calders, M. Ceci, and D. Malerba, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 457–473. doi: 10.1007/978-3-319-46307-0_29.

[76] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[77] G. Bologna and S. Fossati, “A Two-Step Rule-Extraction Technique for a CNN,” *Electronics*, vol. 9, no. 6, 2020, doi: 10.3390/electronics9060990.

[78] S. Burkhardt, J. Brugger, N. Wagner, Z. Ahmadi, K. Kersting, and S. Kramer, “Rule Extraction from Binary Neural Networks with Convolutional Rules for Model Validation,” arXiv.org. Accessed: Mar. 03, 2024. [Online]. Available: https://arxiv.org/abs/2012.08459v1

[79] A. Dhurandhar, K. Shanmugam, R. Luss, and P. Olsen, “Improving Simple Models with Confidence Profiles.” arXiv, Nov. 19, 2018. doi: 10.48550/arXiv.1807.07506.

[80] U. Schlegel, E. Cakmak, and D. A. Keim, “ModelSpeX: Model Specification Using Explainable Artificial Intelligence Methods,” *Machine Learning Methods in Visualisation for Big Data*, p. 5 pages, 2020, doi: 10.2312/MLVIS.20201100.

[81] E. Goan and C. Fookes, “Bayesian Neural Networks: An Introduction and Survey,” vol. 2259, 2020, pp. 45–87. doi: 10.1007/978-3-030-42553-1_3.

[82] A. Kendall and Y. Gal, “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?,” *arXiv:1703.04977 [cs]*, Oct. 2017, Accessed: Apr. 25, 2022. [Online]. Available: http://arxiv.org/abs/1703.04977

[83] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, PMLR, Jun. 2016, pp. 1050–1059. Accessed: Apr. 30, 2022. [Online]. Available: https://proceedings.mlr.press/v48/gal16.html

[84] F. Kraus and K. Dietmayer, “Uncertainty Estimation in One-Stage Object Detection,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand: IEEE, Oct. 2019, pp. 53–60. doi: 10.1109/ITSC.2019.8917494.

[85] D. Linsley, D. Shiebler, S. Eberhardt, and T. Serre, “Learning what and where to attend.” arXiv, Jun. 11, 2019. Accessed: Mar. 03, 2024. [Online]. Available: http://arxiv.org/abs/1805.08819

[86] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Z. Zhang, “The application of two-level attention models in deep convolutional neural network for fine-grained image classification,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA: IEEE, Jun. 2015, pp. 842–850. doi: 10.1109/CVPR.2015.7298685.

[87] F. Wang *et al.*, “Residual Attention Network for Image Classification,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA: IEEE, Jul. 2017, pp. 6450–6458. doi: 10.1109/CVPR.2017.683.

[88] X. Shi *et al.*, “Loss-Based Attention for Interpreting Image-Level Prediction of Convolutional Neural Networks,” *IEEE Transactions on Image Processing*, vol. 30, pp. 1662–1675, 2021, doi: 10.1109/TIP.2020.3046875.

[89] S. Seo, J. Huang, H. Yang, and Y. Liu, “Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction,” in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, Como Italy: ACM, Aug. 2017, pp. 297–305. doi: 10.1145/3109859.3109890.

[90] Q. Zhang, Y. Yang, Y. Liu, Y. N. Wu, and S.-C. Zhu, “Unsupervised Learning of Neural Networks to Explain Neural Networks.” arXiv, May 18, 2018. doi: 10.48550/arXiv.1805.07468.

[91] A. Tavanaei, “Embedded Encoder-Decoder in Convolutional Networks Towards Explainable AI.” arXiv, Jun. 19, 2020. doi: 10.48550/arXiv.2007.06712.

[92] M. Yeganejou, S. Dick, and J. Miller, “Interpretable Deep Convolutional Fuzzy Classifier,” *Trans. Fuz Sys.*, vol. 28, no. 7, pp. 1407–1419, Jul. 2020, doi: 10.1109/TFUZZ.2019.2946520.

[93] A. Dosovitskiy and T. Brox, “Inverting Visual Representations with Convolutional Networks,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 4829–4837. doi: 10.1109/CVPR.2016.522.

[94] M. Lin, Q. Chen, and S. Yan, “Network In Network.” arXiv, Mar. 04, 2014. Accessed: Mar. 03, 2024. [Online]. Available: http://arxiv.org/abs/1312.4400

[95] H. Liang *et al.*, “Training Interpretable Convolutional Neural Networks by Differentiating Class-Specific Filters,” in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II*, Berlin, Heidelberg: Springer-Verlag, 2020, pp. 622–638. doi: 10.1007/978-3-030-58536-5_37.

[96] Y. Sun, S. Ravi, and V. Singh, “Adaptive Activation Thresholding: Dynamic Routing Type Behavior for Interpretability in Convolutional Neural Networks,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 4937–4946. doi: 10.1109/ICCV.2019.00504.

[97] B. Yin, L. Tran, H. Li, X. Shen, and X. Liu, “Towards Interpretable Face Recognition,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 9347–9356. doi: 10.1109/ICCV.2019.00944.

[98] C. H. Yoo, N. Kim, and J.-W. Kang, “Relevance Regularization of Convolutional Neural Network for Interpretable Classification,” 2019.

[99] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu, “Growing interpretable part graphs on ConvNets via multi-shot learning,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, in AAAI’17. San Francisco, California, USA: AAAI Press, Feb. 2017, pp. 2898–2906.

[100] J. Donnelly, A. J. Barnett, and C. Chen, “Deformable ProtoPNet: An Interpretable Image Classifier Using Deformable Prototypes,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 10255–10265. doi: 10.1109/CVPR52688.2022.01002.

[101] D. Rymarczyk, \Lukasz Struski, J. Tabor, and B. Zieliński, “ProtoPShare: Prototypical Parts Sharing for Similarity Discovery in Interpretable Image Classification,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, in KDD ’21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1420–1430. doi: 10.1145/3447548.3467245.

[102] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “DeepFool: a simple and accurate method to fool deep neural networks.” arXiv, Jul. 04, 2016. Accessed: Mar. 03, 2024. [Online]. Available: http://arxiv.org/abs/1511.04599

[103] S. Kim, J. Nam, and B. C. Ko, “ViT-NeT: Interpretable Vision Transformers with Neural Tree Decoder,” in *Proceedings of the 39th International Conference on Machine Learning*, PMLR, Jun. 2022, pp. 11162–11172. Accessed: Aug. 30, 2023. [Online]. Available: https://proceedings.mlr.press/v162/kim22g.html

[104] A. M. Kurup and J. P. Bos, “Winter adverse driving dataset for autonomy in inclement winter weather,” *Optical Engineering*, vol. 62, no. 3, p. 031207, 2023, doi: 10.1117/1.OE.62.3.031207.

[105] S. Kullback and R. A. Leibler, “On Information and Sufficiency,” *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951, doi: 10.1214/aoms/1177729694.

[106] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, “This Looks Like That: Deep Learning for Interpretable Image Recognition,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2019. Accessed: Feb. 15, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/adf7ee2dcf142b0e11888e72b43fcb75-Abstract.html

[107] M. Nauta, R. Van Bree, and C. Seifert, “Neural Prototype Trees for Interpretable Fine-grained Image Recognition,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 14928–14938. doi: 10.1109/CVPR46437.2021.01469.

[108] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, “Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation.” arXiv, Mar. 19, 2014. Accessed: Aug. 31, 2023. [Online]. Available: http://arxiv.org/abs/1309.6392

[109] C. Molnar, *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. 2019. Accessed: Mar. 02, 2024. [Online]. Available: https://christophm.github.io/interpretable-ml-book/

[110] A. Fisher, C. Rudin, and F. Dominici, "All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously.," *J Mach Learn Res*, vol. 20, p. 177, 2019.

[111] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-Wise Relevance Propagation: An Overview," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds., in Lecture Notes in Computer Science. , Cham: Springer International Publishing, 2019, pp. 193–209. doi: 10.1007/978-3-030-28954-6_10.

[112] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6034

[113] M. B. Muhammad and M. Yeasin, "Eigen-CAM: Class Activation Map using Principal Components," in *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020, pp. 1–7. doi: 10.1109/IJCNN48605.2020.9206626.

[114] T. Oikarinen and T.-W. Weng, "CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=iPWiwWHc1V

[115] E. Hernandez, S. Schwettmann, D. Bau, T. Bagashvili, A. Torralba, and J. Andreas, "Natural Language Descriptions of Deep Visual Features." arXiv, Apr. 18, 2022. doi: 10.48550/arXiv.2201.11114.

[116] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, in KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[117] J. H. Friedman and B. E. Popescu, "Predictive Learning via Rule Ensembles," *The Annals of Applied Statistics*, vol. 2, no. 3, pp. 916–954, 2008.

[118] P. Clark and T. Niblett, "The CN2 induction algorithm," *Mach Learn*, vol. 3, no. 4, pp. 261–283, Mar. 1989, doi: 10.1007/BF00116835.

[119] W. W. Cohen, "Fast Effective Rule Induction," in *Machine Learning Proceedings 1995*, A. Prieditis and S. Russell, Eds., San Francisco (CA): Morgan Kaufmann, 1995, pp. 115–123. doi: 10.1016/B978-1-55860-377-6.50023-2.

[120] H. Berger, D. Merkl, and M. Dittenbach, "Exploiting partial decision trees for feature subset selection in e-mail categorization," in *Proceedings of the 2006 ACM Symposium on Applied Computing*, in SAC '06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 1105–1109. doi: 10.1145/1141277.1141536.

[121] K. Shridhar, F. Laumann, and M. Liwicki, "A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference," *arXiv e-prints*. Jan. 01, 2019. doi: 10.48550/arXiv.1901.02731.

[122] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning." arXiv, Dec. 12, 2010. doi: 10.48550/arXiv.1012.2599.

[123] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *Learning and Intelligent Optimization*, C. A. C. Coello, Ed., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 507–523. doi: 10.1007/978-3-642-25566-3_40.

[124] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2011. [Online]. Available:

https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[125] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2012. Accessed: Feb. 27, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html

[126] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets." arXiv, Mar. 07, 2017. doi: 10.48550/arXiv.1605.07079.

[127] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983, doi: 10.1126/science.220.4598.671.

[128] K. G. Jamieson and A. Talwalkar, "Non-stochastic Best Arm Identification and Hyperparameter Optimization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, A. Gretton and C. C. Robert, Eds., in JMLR Workshop and Conference Proceedings, vol. 51. JMLR.org, 2016, pp. 240–248. [Online]. Available: http://proceedings.mlr.press/v51/jamieson16.html

[129] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: a novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6765–6816, Jan. 2017.

[130] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design".

[131] K. Sastry, D. Goldberg, and G. Kendall, "Genetic Algorithms," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E. K. Burke and G. Kendall, Eds., Boston, MA: Springer US, 2005, pp. 97–125. doi: 10.1007/0-387-28356-0_4.

[132] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Nov. 1995, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.

[133] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: 10.1109/MCI.2006.329691.

[134] G. Dhiman and V. Kumar, "Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications," *Advances in Engineering Software*, vol. 114, pp. 48–70, Dec. 2017, doi: 10.1016/j.advengsoft.2017.05.014.

[135] N. Hansen, "The CMA Evolution Strategy: A Comparing Review," in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds., in Studies in Fuzziness and Soft Computing. , Berlin, Heidelberg: Springer, 2006, pp. 75–102. doi: 10.1007/3-540-32494-1_4.

[136] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997, doi: 10.1023/A:1008202821328.

[137] K. Ahmad, M. Bano, M. Abdelrazek, C. Arora, and J. Grundy, "What's up with Requirements Engineering for Artificial Intelligence Systems?," in *2021 IEEE 29th International Requirements Engineering Conference (RE)*, Sep. 2021, pp. 1–12. doi: 10.1109/RE51729.2021.00008.

[138] K. M. Habibullah, G. Gay, and J. Horkoff, "Non-functional requirements for machine learning: understanding current use and challenges among practitioners," *Requirements Eng*, vol. 28, no. 2, pp. 283–316, Jun. 2023, doi: 10.1007/s00766-022-00395-3.

[139] K. Ahmad, M. Abdelrazek, C. Arora, M. Bano, and J. Grundy, "Requirements engineering for artificial intelligence systems: A systematic mapping study," *Information and Software Technology*, vol. 158, p. 107176, Jun. 2023, doi: 10.1016/j.infsof.2023.107176.

[140] International Organization for Standardization, "Road vehicles — Safety of the intended functionality," ISO 21448:2022. Accessed: Oct. 10, 2022. [Online]. Available: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/74/77490.html

[141] A. S. Fraser, “Simulation of Genetic Systems by Automatic Digital Computers II. Effects of Linkage on Rates of Advance Under Selection,” *Aust. Jnl. Of Bio. Sci.*, vol. 10, no. 4, pp. 492–500, 1957, doi: 10.1071/bi9570492.

[142] H. J. Bremermann, *The evolution of intelligence: The nervous system as a model of its environment*. University of Washington, Department of Mathematics, 1958.

[143] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[144] M. Wicker, X. Huang, and M. Kwiatkowska, “Feature-Guided Black-Box Safety Testing of Deep Neural Networks.” arXiv, Feb. 20, 2018. doi: 10.48550/arXiv.1710.07859.

[145] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, “Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks,” in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 97–117. doi: 10.1007/978-3-319-63387-9_5.

[146] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, “AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation,” in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 3–18. doi: 10.1109/SP.2018.00058.

[147] A. Hedström *et al.*, “Quantus: An Explainable AI Toolkit for Responsible Evaluation of Neural Network Explanations and Beyond,” 2023.

[148] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, “The Balanced Accuracy and Its Posterior Distribution,” in *2010 20th International Conference on Pattern Recognition*, Aug. 2010, pp. 3121–3124. doi: 10.1109/ICPR.2010.764.

[149] C.-H. Cheng, C.-H. Huang, and H. Yasuoka, “Quantitative Projection Coverage for Testing ML-enabled Autonomous Systems.” arXiv, May 11, 2018. doi: 10.48550/arXiv.1805.04333.

[150] N. Bousquet, “Diagnostics of prior-data agreement in applied Bayesian analysis,” *Journal of Applied Statistics*, vol. 35, no. 9, pp. 1011–1029, Sep. 2008, doi: 10.1080/02664760802192981.

[151] E. Schat, R. Van De Schoot, W. M. Kouw, D. Veen, and A. M. Mendrik, “The data representativeness criterion: Predicting the performance of supervised classification based on data set similarity,” *PLoS ONE*, vol. 15, no. 8, p. e0237009, Aug. 2020, doi: 10.1371/journal.pone.0237009.

[152] P. C. Mahalanobis, “On the generalized distance in statistics,” *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[153] A. Bhattacharyya, “On a Measure of Divergence between Two Multinomial Populations,” *Sankhyā: The Indian Journal of Statistics (1933-1960)*, vol. 7, no. 4, pp. 401–406, 1946.

[154] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau, and G. Rohde, “Generalized Sliced Wasserstein Distances,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2019. Accessed: Mar. 02, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/f0935e4cd5920aa6c7c996a5ee53a70f-Abstract.html

[155] I. Deshpande *et al.*, “Max-Sliced Wasserstein Distance and Its Use for GANs,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 10640–10648. doi: 10.1109/CVPR.2019.01090.

[156] I. Csiszar, “I-Divergence Geometry of Probability Distributions and Minimization Problems,” *The Annals of Probability*, vol. 3, no. 1, pp. 146–158, Feb. 1975, doi: 10.1214/aop/1176996454.

[157] F. Nielsen, “On the Jensen–Shannon Symmetrization of Distances Relying on Abstract Means,” *Entropy*, vol. 21, no. 5, Art. no. 5, May 2019, doi: 10.3390/e21050485.

[158] M. Usman, Y. Sun, D. Gopinath, R. Dange, L. Manolache, and C. S. Pasareanu, “An Overview of Structural Coverage Metrics for Testing Neural Networks.” arXiv, Aug. 05, 2022. doi: 10.48550/arXiv.2208.03407.

[159] J. Kim, R. Feldt, and S. Yoo, “Guiding deep learning system testing using surprise adequacy,” in *Proceedings of the 41st International Conference on Software Engineering*, in ICSE ’19. Montreal, Quebec, Canada: IEEE Press, May 2019, pp. 1039–1049. doi: 10.1109/ICSE.2019.00108.

[160] L. Ma *et al.*, "DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, in ASE '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 120–131. doi: 10.1145/3238147.3238202.

[161] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated Whitebox Testing of Deep Learning Systems," *Commun. ACM*, vol. 62, no. 11, pp. 137–145, Oct. 2019, doi: 10.1145/3361566.

[162] X. Xie *et al.*, "DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, in ISSTA 2019. New York, NY, USA: Association for Computing Machinery, 2019, pp. 146–157. doi: 10.1145/3293882.3330579.

[163] X. Xie *et al.*, "NPC: Neuron Path Coverage via Characterizing Decision Logic of Deep Neural Networks," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 3, Apr. 2022, doi: 10.1145/3490489.

[164] H. Kelly J., V. Dan S., C. John J., and R. Leanna K., "A Practical Tutorial on Modified Condition/Decision Coverage," NASA Langley Technical Report Server, 2001.

[165] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Structural Test Coverage Criteria for Deep Neural Networks," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, May 2019, pp. 320–321. doi: 10.1109/ICSE-Companion.2019.00134.

[166] B. Zong *et al.*, "Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection," presented at the International Conference on Learning Representations, Feb. 2018. Accessed: Mar. 18, 2024. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0-

[167] S. J. Taylor and B. Letham, "Forecasting at scale," PeerJ Preprints, preprint, Sep. 2017. doi: 10.7287/peerj.preprints.3190v2.

[168] J. Ren *et al.*, "Likelihood Ratios for Out-of-Distribution Detection." arXiv, Dec. 05, 2019. doi: 10.48550/arXiv.1906.02845.

[169] J. Gawlikowski *et al.*, "A survey of uncertainty in deep neural networks," *Artif Intell Rev*, vol. 56, no. 1, pp. 1513–1589, Oct. 2023, doi: 10.1007/s10462-023-10562-9.

[170] J. V. Amersfoort, L. Smith, Y. W. Teh, and Y. Gal, "Uncertainty Estimation Using a Single Deep Deterministic Neural Network," in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 2020, pp. 9690–9700. Accessed: Feb. 19, 2024. [Online]. Available: https://proceedings.mlr.press/v119/van-amersfoort20a.html

[171] M. Abdar *et al.*, "A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges," *Information Fusion*, vol. 76, pp. 243–297, Dec. 2021, doi: 10.1016/j.inffus.2021.05.008.

[172] V. Vovk, A. Gammerman, and G. Shafer, Eds., "Conformal prediction," in *Algorithmic Learning in a Random World*, Boston, MA: Springer US, 2005, pp. 17–51. doi: 10.1007/0-387-25061-1_2.

[173] H. Papadopoulos, K. Proedrou, V. Vovk, and A. Gammerman, "Inductive Confidence Machines for Regression," in *Proceedings of the 13th European Conference on Machine Learning*, in ECML '02. Berlin, Heidelberg: Springer-Verlag, Aug. 2002, pp. 345–356.

[174] J. Lei and L. Wasserman, "Distribution-free Prediction Bands for Non-parametric Regression," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 76, no. 1, pp. 71–96, Jul. 2013, doi: 10.1111/rssb.12021.

[175] A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, and T. Schuster, "Conformal Risk Control." arXiv, Apr. 29, 2023. doi: 10.48550/arXiv.2208.02814.

[176] M. Kläes, R. Adler, I. Sorokos, L. Joeckel, and J. Reich, "Handling Uncertainties of Data-Driven Models in Compliance with Safety Constraints for Autonomous Behaviour," presented at the 2021 17th European Dependable Computing Conference (EDCC), IEEE Computer Society, Sep. 2021, pp. 95–102. doi: 10.1109/EDCC53658.2021.00021.

[177] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992, doi: https://doi.org/10.1016/S0893-6080(05)80023-1.