D3.3 Final proofs-of-concept, arguments, and DL components and libraries

Version 1.0

Documentation Information

Contract Number	101069595
Project Website	www.safexplain.eu
Contractual Deadline	31.03.2025
Dissemination Level	PU
Nature	R
Author	RISE
Contributors	Thanh Hai Bui (RISE), Jokin Labaien (IKR), Roger Pujol (BSC), Robert Lowe (RISE), Axel Brando (BSC), Gabriele Giordana (Aiko), Maria Ulan (RISE)
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	Second draft
V0.3	Final draft sent to internal reviewer
V0.4	Internal reviewer feedbacks
V0.5	Final draft with revision
V0.9	Revised final draft confirmed by internal reviewer
V1.0	Final version

Table of Contents

1. Introduction 8
2. Background 9
2.1. AI trustworthiness related standards 9
2.2. SAFEXPLAIN AI Explainability concepts & categorization 10
2.2.1. SAFEXPLAIN XAI Concepts 10
2.2.2. SAFEXPLAIN XAI Categorization 13
3. Specification and Design of FUSA-aware DL solutions 19
3.1. System scope 19
3.2. Minimum Viable Product (MVP) 21
3.3. Uncertainty management strategy 22
3.3.1. Reduction of uncertainties within development cycles 23
3.3.2. Management of residual uncertainty within OM 25
3.4. XAI usages in AI-FSM Lifecycle 26
3.4.1. Safety goals & strategy 26
3.4.2. Robustness, Explainability 26
3.4.3. XAI usages in Data Management 30
3.4.4. XAI usages in Learning Management 34
3.4.5. XAI usages in Inference Management 39
3.5. XAI usages in Operation and Monitoring stage 40
3.5.1. Reference safety architecture 41
3.5.2. Reference safety components 42
3.6. Relevant metrics 55
3.6.1. Explainability 55
3.6.2. Traceability 57
3.6.3. Safety 58
3.6.4. Robustness 59
3.6.5. Failure rate 62
3.6.6. Easy to implement 62
4. Realization and libraries 62
4.1. EXPLib 62
4.1.1. Library structure 63
4.1.2. Guidelines and examples 63
4.1.3. Libraries of XAI tools 69
4.1.4. Models 71

4.1.5. Datasets 71
4.2. DLLib..... 71
4.2.1. Integration with ROS2 (Middleware)..... 72
4.2.2. Supervision Function & Decision Function 73
5. Discussions..... 73
References 75

List of figures

Figure 1: Explainable AI concept map 12
Figure 2: Sense - Think - Act 21
Figure 16: Example images from the MVP dataset 21
Figure 3: Alternative evolution of the scenario categories resulting from ISO21448[11] activities 22
Figure 4: Uncertainty types 23
Figure 5: FGSM attack on 4 sample images using different degrees of small perturbation (Eps) 29
Figure 6: Bounding box and class distribution for Railway use case. 32
Figure 7: Extracted prototypes of MVP dataset, patch size 100x100 33
Figure 8: Trained VAE model as data descriptor. The model describes its input image (left) as an reconstructed version (center) and corresponding latent vector. The reconstruction error map (right) highlights the reconstruction error area (anomalous area) 34
Figure 9: LIME Heatmap (30 SLIC segments) 36
Figure 10: MMD distance per MVP model's key conv layers (comparing two datasets) 36
Figure 11: SHAP heatmap with superpixels 37
Figure 12: MVP model EigenCAM 38
Figure 13: PhIM - Model performance (IoU metrics) dependency on input feature parameter (center position of object) 39
Figure 14: EigenCAM on railway use case. 40
Figure 15: Reference safety architecture for Operation and Monitoring stage 41
Figure 17: Modified SSDLite model to estimate aleatoric uncertainty 44
Figure 18: Aleatoric uncertainty aware model (modified SSDLite model) 45
Figure 19: Epistemic uncertainty aware model (modified SSDLite model) 46
Figure 20: Toymodel epistemic uncertainty estimates PDF functions 46
Figure 21: Anomalous region with VAE latent anomaly 48
Figure 22: Anomaly detector highlighting anomalous region from the latent space anomaly (PCA 15 dims). ufo dataset rgb_0134.png 48
Figure 23: Extracted model neuron activation patterns (right) and model gradient map (left), used as input for anomaly detector 49
Figure 24: Box plots of logit values over all 4 data examples displayed in Figure 5. 50
Figure 25: Extracted input features (keypoints and LBP pattern) to serve as input for MVP surrogate model 51
Figure 26: Statistical description of the set of keypoints 51
Figure 27: Surrogate model prediction 52
Figure 28: Groundtruth vs predicted bbox coordinates 53
Figure 29: Consolidated prediction by XGBoost 54
Figure 30: Importance of input parameters contributing to the final consolidated predictions of trained XGBoost model 54
Figure 31: Example output of decision function 55
Figure 32: EXPLib - example notebook of Data Profiling within PhDM 66
Figure 33: Latent distributions (of VAE descriptor) projected onto 3 dimension computed using UMAP (left) and t-SNE (right) methods. 66
Figure 34: PCA analysis of latent space (of VAE-based data descriptor). Accumulated eigenvalues of the first 15 eigenvectors 67
Figure 35: Experiments of using PCA reconstructed latent vector of data descriptor to identify anomalous area in the input image 67
Figure 36: Contrastive LRP saliency map generated for YOLOv8 (Railway UC) 68
Figure 37: LRP saliency map generated for YOLOv8 (Railway UC) 68

Figure 38: LIME heatmap computed for MVP model, using superpixel segmentation method Watershed as input for LIME 68
Figure 39: Model performance (IoU metric) vs input parameters: bounding box location and area 69
Figure 40: Example usage of Epistemic uncertainty aware MVP model 69

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
- MVP – Minimum Viable Product
- 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 report presents the final results (as of M30) of SAFEXPLAIN's FUSA-aware dependable Deep Learning (DL) solutions within WP3. This work aligns with the development of the FUSA for AI lifecycle (AI-FSM, D2.2) and reference architecture patterns (SPs, D2.1). We propose a systematic approach for cataloguing and categorizing Explainable AI (XAI) techniques, making them accessible to practitioners implementing FUSA-compliant solutions following SAFEXPLAIN's proposed approach. The document begins by refining the SAFEXPLAIN XAI concepts, incorporating latest standards, guidelines, best practices, and state-of-the-art XAI techniques. This establishes a strong foundation for dependable DL solutions that meet FUSA requirements.

1. Introduction

This document reports the final 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.

It constitutes the final update to deliverables D3.1 and D3.2, including practical and hands-on recommendations for the complete lifecycle of specifying DL components, from initial requirements gathering and architecture selection to the deployment of embedded Deep Neural Networks (DNNs).

The proposed strategy for achieving dependability in Deep Learning (DL) components designed for safety-critical applications is predicated on the comprehensive characterization and systematic mitigation of uncertainties inherent in AI systems (that are connected to hazardous situations). This approach is rigorously aligned and supports compliance with the AI-FSM development lifecycle (detailed in D2.1[1]) and implements the Safety Patterns deployment architecture (as described in D2.2[2]).

The report explores explainability, traceability, and robustness within the context of a Minimum Viable Product (MVP) and provides a practical handbook guiding the development of DL components for Critical Autonomous AI-based Systems (CAIS).

This document is organized as follows:

- **Section 2**: This section provides an overview of SAFEXPLAIN's adoption of XAI concepts, reviews the latest state of the art and updates on relevant standards, guidelines, and best practices. It also categorizes state-of-the-art XAI techniques to facilitate the usages throughout the lifecycle.
- **Section 3**: This section outlines a systematic approach to applying XAI techniques in the AI-FSM development lifecycle and safety architecture design for operational and monitoring (OM) stage. It provides guidance on how to specify dependable DL solutions.
- **Section 4**: This section describes two software libraries
    - EXPLib: Python library of XAI techniques and practices that support AI-FSM lifecycle compliance, and
    - DLLib: Python and low-level language library (optimized for embedded platform) for deploying the safety architecture and its components in OM stage.
- **Section 5**: This section provides some discussions and future recommendations

2. Background

2.1. AI trustworthiness related standards

SAFEXPLAIN aligns with the latest AI trustworthiness standards to ensure the safe and reliable deployment of deep learning (DL) components in safety-critical systems. AI-FSM lifecycle, Verification & Validation (V&V) strategy, and dependable DL system specification are informed by key standards and evolving regulatory requirements. This section details the relevant standards and guidelines, updated as of M30, that shape our approach to building trustworthy AI systems.

EU AI Act[3]: is a legal framework designed to ensure the safe, secure, and trustworthy development and deployment of AI systems across the EU and EEA, providing regulations and guidelines that prioritize the protection of fundamental rights and promote ethical considerations in AI development. The regulation categorizes AI systems into different risk levels, imposing strict requirements on high-risk AI systems, and prohibiting certain types of AI that manipulate or deceive individuals, exploit vulnerabilities, or infringe on privacy and human rights. SAFEXPLAIN focuses on the “High risk AI systems” category (described in Chapter III and relevant to Critical infrastructure Use case in Annex III).

ISO/IEC DTS 6254[4] *(Objectives and approaches for explainability and interpretability of ML models and AI systems)*: describes approaches for AI systems’ explainability, providing guidelines to various stakeholders (academia, industry, policy makers, end users) on achieving objectives throughout the AI system's lifecycle. The standard is a work-in-progress, with a draft submitted in Oct 2024.

IEEE 2894-2024[5] *(IEEE Guide for an Architectural Framework for Explainable Artificial Intelligence):* provides a technological blueprint for building transparent and trustworthy AI systems using explainable AI (XAI) methodologies, defining architectural frameworks and application guidelines for XAI.

P2976[6] *(Standard for XAI - eXplainable Artificial Intelligence - for Achieving Clarity and Interoperability of AI Systems Design)*: defines mandatory and optional requirements and constraints that need to be satisfied for an AI method, algorithm, application or system to be recognized as explainable. The working group had started, and no official release has been published yet.

ISO/IEC 8183:2023[7] *(Information technology — Artificial intelligence — Data life cycle framework):* provides guidelines of stages and actions for data processing throughout the AI system life cycle, from acquisition to decommissioning, and applies to all organizations using data in AI development and deployment.

ISO/IEC 42001:2023[8] *(Information technology — Artificial intelligence — Management system)*: specifies requirements for establishing and maintaining an Artificial Intelligence Management System (AIMS) within organizations. It provides guidance on responsible AI development and use, addressing ethical considerations, transparency, and risk management.

ISO/IEC TR 24030:2024[9] *(Information technology — Artificial intelligence (AI) — Use cases):* Providing a collection of AI use cases across various domains, it illustrates the applicability and potential of AI in different sectors.

ISO/IEC 5339:2024[10] *(Information technology — Artificial intelligence — Guidance for AI applications):* provides guidance on AI applications, emphasizing stakeholder engagement and the AI application life cycle.

It aims to enhance multi-stakeholder communication and acceptance by offering a framework that includes the make, use, and impact perspectives of AI systems.

ISO/IEC TR 5469:2024[11] *(AI Functional Safety and AI Systems)***:** provides guidance on integrating AI into safety-critical environments, defining six usage levels of AI in safety functions. SAFEXPLAIN applies this incremental approach to align AI's role with necessary supervisory mechanisms, ensuring robust safety management and explainability.

ISO 21448[12] *(SOTIF – Safety of the Intended Functionality)***:** complements ISO 26262 by addressing hazards due to functional insufficiencies rather than hardware/software failures. SAFEXPLAIN integrates SOTIF's scenario-based risk identification into its V&V strategy to mitigate unknown unsafe conditions, ensuring robust AI performance across edge cases.

ISO 26262[13] *(Functional Safety for Road Vehicles)***:** remains a core safety standard, but its deterministic assumptions challenge the integration of stochastic DL components. SAFEXPLAIN combines ISO 26262 with AI-specific safety measures (e.g., ISO/PAS 8800) to address ML-related failure modes, ensuring alignment with best practices in AI safety.

EASA ML Assurance Guidelines[14]**:** introduce a "W-shaped" AI lifecycle model, refining explainability, learning assurance, and risk assessment for ML systems in aviation. SAFEXPLAIN incorporates these principles to structure its AI-FSM lifecycle, ensuring transparency and regulatory readiness for critical applications.

ISO/PAS 8800:2024[15] *(Road Vehicles – AI Safety)*: published in December 2024, bridges ISO 26262 and SOTIF by proposing an AI-specific safety lifecycle. It introduces structured methods to handle AI functional insufficiencies, model failures, and post-deployment monitoring. SAFEXPLAIN adopts ISO/PAS 8800's trustworthiness principles (robustness, explainability, controllability) to ensure dependable AI-based functions.

UL 4600[16] **& ISO 61508**[17] provide additional safety frameworks. UL 4600, designed for fully autonomous systems, mandates a safety case approach for AI-driven decisions. ISO 61508, the foundational functional safety standard, underpins many AI-focused safety principles in emerging regulations. SAFEXPLAIN aligns with these to strengthen its AI risk assessment and lifecycle assurance.

ISO/IEC 22989:2022[18] *(AI Concepts and Terminology):* standardizes AI trustworthiness attributes, including robustness, reliability, explainability, and fairness. SAFEXPLAIN applies these definitions to maintain consistency in risk assessment, XAI implementation, and regulatory compliance across safety-critical AI components.

2.2. SAFEXPLAIN AI Explainability concepts & categorization

2.2.1. SAFEXPLAIN XAI Concepts

For completeness, this section reproduces relevant content from our previous deliverable, D3.1[19], with updates to reflect our evolving knowledge and insights.

Within SAFEXPLAIN, Explainable AI (XAI) concepts refer to methods, models, and algorithms that facilitate understanding of Deep Learning (DL) projects by stakeholders, including developers, data analysts, domain experts, and FUSA experts. We define explainability as the ability to generate human-understandable reasons for model predictions and internal workings, which is subjective and audience-dependent.

In our context, explainability involves providing a process and methods to map a black box DL component's behaviour to a knowledge level acceptable by humans (or machines, in deployment mode) as the target audience. This enables transparency and trustworthiness throughout the development lifecycle and deployment stage.

2.2.1.1. Explainability and Understandability

Explainable AI (XAI), Explainability and Understandability are closely related concepts, resulted from different actor perspectives in an interaction between human and AI:

- **Explainability:** refers to the ability of a DL model to provide insights into its inner working process via different types of explanations. Explainability is about providing transparency to the DL's reasoning and inner working via ability to generate relevant explanations.
- **Understandability**: in contrast, refers to the ability of humans (consumer of the explanations) to comprehend the provided explanations to be able to make informed decisions. Understandability is about the usefulness of the explanations to the users.

The level of understandability is dependent on the model complexity, which affects how easily humans can follow to understand the relationships among model structure and inputs/outputs. In other words, understandability is related to the number and type of associations that need to be made between model's components and its behaviours. Various metrics to quantify Understandability have been proposed, such as those mentioned in [20]. For example. 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.

2.2.1.2. Interpretability and Transparency

Interpretability and Transparency are a pair of related concepts to reveal insights into the internal workings of a DL system to make it more understandable and trustworthy.

- **Transparency** Refers to the ability to view or understand the inner workings of a DL system, including architecture, data, and algorithms. Transparency is about providing access to required information about the system's structure, parameters, and operations.
- **Interpretability** refers to the degree to which a DL model's decisions and behaviour can be understood by a human observer. Interpretability focuses on making sense of the information (provided via transparency), e.g. relationships between the input data, the model's parameters, and the output predictions. Interpretability can be characterized by different ways how humans gain their understandings from the explanations:
    - **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.

2.2.1.3. 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[21], 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.

Comprehensibility serves as a bridge between *Explainability and Understandability* and *Interpretability and Transparency* concepts. It shares the goal of making model knowledge accessible to humans with explainability, while also relating to how the model's internal representation can be made inherently interpretable. This concept is particularly important for deep learning models where transforming complex learned patterns into human-comprehensible representations is a significant challenge.

2.2.1.4. Concept mappings

Figure 1 illustrates the key XAI concepts used within SAFEXPLAIN, which are represented as part of the interactions between DL components and humans. A trustworthy interaction occurs when the DL component generates comprehensive and transparent explanations that provide insights into its internal working process, enabling humans to understand and trust the outcomes. From the human perspective, these explanations must be understandable, considering the subjective nature of human understanding, which can vary depending on the individual audience (e.g. via simulatability, decomposability and algorithmic transparency).

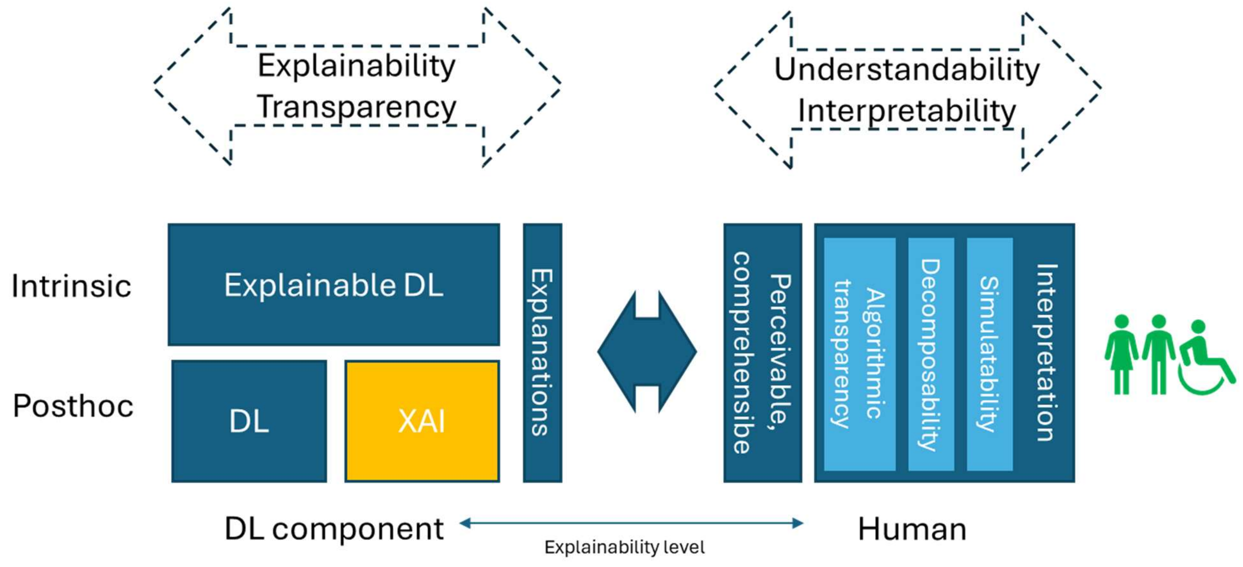


Figure 1: Explainable AI concept map

The SAFEXPLAIN framework leverages a range of Explainable AI (XAI) techniques addressing diverse audiences at different phases of its proposed processes. By providing a systematic approach, these concepts enable the selection of suitable XAI techniques that maximize and control trustworthiness throughout the entire process. In the subsequent section, we will discuss the categorization of XAI techniques, offering practitioners a comprehensive understanding of how to effectively apply these techniques to develop and deploy dependable DL components.

2.2.2. SAFEXPLAIN XAI Categorization

XAI methodologies are categorized within SAFEXPLAIN based on several dimensions to facilitate solution mappings (how and where each XAI method can be leveraged within the lifecycle).

These dimensions include target item types, which refers to whether the method applies to the dataset or the trained DL component. Another dimension is target audience of the explanations, including AI developers (focused on understanding model behaviour and design), data analysts (concerned with feature importance, data quality, and anomaly detection), domain experts (designed to help non-technical experts interpret AI decisions and verify predictions using their domain expertise), safety experts (engaged in risk assessment, reliability analysis, and V&V activities), and end users (who need interpretable models for direct interaction, such as vehicle drivers/operators). Explanations can be represented in different forms such as text/tabular data, graphs and images to accommodate diverse comprehension preferences.

Explanations can be scoped as either (i) local, focusing on individual predictions or data points, or (ii) global, addressing overall model behaviour. XAI methods can operate without user input or incorporate human-in-the-loop (HIL) approaches. Furthermore, techniques differ based on timing requirements, with some suitable for real-time deployment (as needed in the Operation and Monitoring stage) and others without such limitations.

XAI methods/approaches are also categorized by their scope, being either model/data-specific or model/data-agnostic. Furthermore, techniques for explaining DL models can be classified as intrinsic (interpretable by design) or post-hoc (applied to explain trained model predictions).

Transparency approach category refers to the interpretability nature that help human to understand the explanations as discussed earlier in Section 2.2.1: simulatability, decomposability and algorithmic transparency. Table 1 summarizes the categories and relevant values.

Table 1: Categories of Explainability methods

Category	**Values**
Target item	Dataset, trained DL component
Target audience	AI developer, data analyst, domain expert, safety expert, end user
Representation	Text/tabular, graph, image, numerical values
Scope	Local, global
Interaction	No interaction, interactive (human in the loop)
Time critical	Realtime, without time critical requirements
Scope of application	Model/data specific, model/data agnostic
Techniques to complement the DL models	Intrinsic, posthoc
Transparency approach	Decomposability, simulatability, algorithmic

XAI approaches 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.

2.2.2.1. Data explainability

Explainable AI (XAI) methods can be used to explain the data used in a DL development/deployment project. We refer to these methods as "data explainers" in this document.

Data explainers can be used to provide insights into datasets and/or datapoints within a dataset, categorized as below:

- **Dataset distribution insights**: Exploratory Data Analysis (EDA) is crucial for evaluating data quality, identifying skew, missing values, and potential feature explanations. Tools like yData profiling [22], SweetViz [23], and Google Facets [24] provide statistical summaries, while dimension reduction visualizations like t-SNE [25], and UMAP [26] reveal patterns, clusters, and correlations in high-dimensional data.
- **Data point insights**: Feature extraction techniques, including domain-specific and model-based approaches, can be used to identify significant features in a dataset. Domain-specific methods leverage expertise and insights from EDA, while model-based feature engineering employs mathematical models to analyse the inherent structure of a dataset. Examples include binary classifiers[27], clustering, and dictionary learning [28]. For image data, interpretable representations can be created using features such as DAISY[29], HOG[30], Haar[31], LBP[32], CenSurE[33], ORB[34], Gabor[35], SIFT[36], Shape index[37].
- **Proximity and Similarity analysis**: Methods that quantify the relationships between data points and datasets are essential for identifying if an input data is likely belonging to a known dataset.
  - *Statistical tests*: Kolmogorov-Smirnov and Chi-squared tests[38] compare distributional differences between datasets.
  - *Distance/similarity metrics*: Mahalanobis distance[39] measures the distance between a datapoint and a dataset. The Bhattacharyya distance[40] measures the similarity between two datasets modelled as probability distributions. Maximum Mean Discrepancy (MMD[41]) measures distance between datasets w.r.t. kernel induced Hilbert space.
  - *Kernel density estimation (KDE)*: non-parametric methods to estimate kernel density functions without assuming distribution, which can facilitate the distribution distance computations. These methods collectively enable practitioners to decide data integration strategies, perform dataset assessments or design data anomaly detectors.
- **Data summarization and representation**: Auto Encoder family of models, especially its statistical variants such as Variational Autoencoders (VAEs) [42] or Disentangled Inferred Prior Variational Autoencoder (DIPVAE) [43] can reconstruct data similar to the training set and provide a lower-dimensional representation (the latent space) of the dataset's underlying structure. The latent space in VAEs offers a compact summary of the dataset. Beyond generative approaches, deterministic dimensionality reduction techniques like Principal Component Analysis (PCA[44]), Linear Discriminant Analysis (LDA[45]), and Independent Component Analysis (ICA[46]) can reveal dataset structure and feature importance. These reduced representations can be visualized for intuitive understanding or statistically analysed to support the design of anomaly detectors in vector space. For example, applying these techniques to a VAE's latent space can identify high-level, abstract concept anomalies.
- **Data mining and profiling**: Data Readiness Levels (DRL[47]) proposed a systematic framework used to assess the quality and readiness of data for supporting data-driven algorithms. DRLs evaluate the usability, reliability, and completeness of data across different stages, providing organizations with a structured approach to identify gaps and improve data management practices. Standardized dataset documentation initiatives, such as Datasheets for Datasets [48], Dataset Nutrition Labels [49], and Data Declarations for Natural Language Processing [50], aim to bridge the communication gap between dataset creators and users by documenting essential information about datasets. Additionally, prototypes and criticisms methods such as ProtoDash [51] can be used for data

summarization and visualization, enabling the identification of prototypical examples (or criticisms) that characterize a dataset or its clusters.

2.2.2.2. Trained DL component explainability

Unlike the previous categorization of explainability methods in D3.1[19] into intrinsic, post-hoc, and ante-hoc—which organized techniques based on when they are applied in the model development lifecycle—in this section we proposed a new categorization characterized on the model functional purpose. The revised taxonomy groups techniques based on what aspect of the model they aim to explain. This approach provides practitioners with a more intuitive framework for selecting appropriate explainability methods based on their specific analytical goals in various steps of the AI-FSM lifecycle: to understand feature contributions, model structure, semantic concepts, or other aspects of a DL component.

Model explainability refers to the techniques to enhance explainability of DL models through various approaches that provide insights into how models make decisions, represent information, and process inputs to produce outputs.

DL model explainers can provide valuable insights into how models process information and make decisions. The following methods can be employed to achieve this:

- **Feature and Attribution Methods**: These techniques help understand the influence of different input features on the model's predictions and decisions. This includes:
    - Feature importance techniques, which quantify the contribution of each input feature to the prediction.
    - Visualization techniques, which create visual representations of feature attribution to facilitate understanding.
- **Model Structure Interpretation**: This involves analysing the architecture design of the DL model, including:
    - Architecture analysis: examining the organization and connections within the model.
    - Decision boundary mapping: identifying the boundaries that separate different decision regions.
- **Semantic Understanding Methods**: These methods assess whether the internal decision-making process of the DL model aligns with domain concepts. This includes:
    - Concept-based methods: evaluating the model's understanding of specific concepts and relationships.
    - Prototype methods: analysing the dominant concepts in the domain and if the DL model is using them for the predictions.
- **Interpretable Model Design**: This approach involves designing DL models with interpretability in mind, either by:
    - Design model architecture with white-box or gray-box approaches that provide explicit explanations for their decisions.
    - Incorporating hook checkpoints to export explanations when required.
- **Rule Extraction Methods**: These techniques describe the model's decision-making process in terms of rules, allowing domain experts to verify whether the rules align with their expertise. This includes:
    - Decision rule generation: extracting rules that approximate the model's decision-making process.
    - Architecture decomposition: breaking down the model into smaller functional components to understand its decision-making process.
- **Uncertainty and Reliability Analysis**: This involves quantifying the residual uncertainty of the model's predictions, which is essential for:
    - Mitigating uncertainty in AI-FSM phases.

- Controlling risk levels by ensuring that uncertainty does not lead to hazardous situations. This includes:
    - Uncertainty quantification: estimating the uncertainty associated with the model's predictions.
    - Model robustness evaluation: assessing the model's ability to withstand perturbations or changes in input data and understanding its limitations.
- **Enhanced Representation Methods**: These techniques design DL models with mechanisms that encourage them to derive predictions based on approved reasoning. This includes:
    - Attention mechanisms: focusing the model's attention on specific input features or regions.
    - Custom layers and structures for interpretability: incorporating specialized components that facilitate understanding of the model's decision-making process.

More details will be provided in the following subsections.

2.2.2.2.1. Feature and attribution methods

Feature importance and attribution methods reveal how inputs contribute to model outputs, highlighting the most influential aspects of the decision-making process. By quantifying and visualizing feature importance, these techniques bridge the gap between raw inputs and model predictions, helping users understand "what" the model is focusing on.

These methods can be categorized as follows:

- **Activation based and/or Gradient based methods**: Utilize gradients of the model output with respect to input data to assign importance scores and propagate them back to individual features. Examples include: Saliency map[52], Integrated Gradient[53], Class Activation Map (CAM) [54], Grad-CAM[55], gradient based feature importance[56], DeepLIFT[57], SmoothGrad[58], Guided backpropagation[59], EigenCAM[60].
- **Perturbation based methods**: Assess feature importance by perturbing the input data and analysing the resulting changes in model predictions. Examples include: LIME[61], SHAP[62], Anchors[63], Feature occlusion[64], Randomized input sampling RISE [65], Counterfactual[66]
- **Representation based methods**: Provide insights into the model's internal workings by learning interpretable and disentangled representations. Examples include: TCAV[67], beta-VAE[68], Deformable ProtoPNet[69], Neural-symbolic learning[70], infoGAN[71],
- **Visual graph methods**: Visualize relationships between features and model outputs using graphs. Examples include: Partial Dependence Plot (PDP) [72], Accumulated Local Effects (ALE) [73], visualizing decision boundaries of classification model[74].
- **Feature space visualization and dimensionality reductions**: Techniques such as t-SNE [25] and UMAP [26] reduce the number of dimensions while preserving important relationships, allowing for visualization of data clusters and patterns in a 2D or 3D space. Methods that visualize higher-layer features [75] allow examination of what the model "sees" at intermediate stages of processing. Inceptionism[76] takes this further, leveraging the model's learned features to generate artistic and often surprising visualizations, offering insights into the model's internal representations and potential biases.

2.2.2.2.2. Model structure interpretation

Model structure interpretation methods analyse how different sub-components of a DL model interact and contribute to the overall functionality. These approaches explore how information flows through the internal components of a DL model. By examining the interactions between layers, neurons, and activation patterns, these methods provide insights into the model's internal "reasoning" process. Model structure interpretation helps clarify the black-box nature of deep networks by uncovering how the architecture transforms and processes information at each stage.

The methods can be categorized as follows:

- **Model explanation methods**: aim to understand the model's structure and decision-making process via tracing the influence scores through the network. Examples include: LRP[77] [78] [79], Deep Taylor decomposition[80], Spectral relevance analysis[81].
- **Feature and concept analysis methods**: focus on analysing and understanding the features and concepts learned by the model. Examples include: Concept Relevance Propagation[82], Decision boundary visualization (readers consult a review paper [83] for more example methods)
- **Representation and disentanglement methods**: aim to disentangle the extracted feature representations in a lower-dimensional feature space, often using techniques such as
    - Training strategies: Freezing certain latent dimensions during training to encourage different neurons to capture distinct concepts.
    - Dimension reduction and clustering of the latent vector space: Using traditional methods such as PCA[44], Isomap[84], LLE[85], NMF, and SVD[86] to reduce dimensionality and investigate clustered structures of the space.

2.2.2.2.3. Semantic understanding

Semantic understanding methods focus on connecting model features to higher-level, human interpretable concepts. These techniques aim to bridge the gap between the model's internal mathematical operations and human conceptual reasoning, making model behaviour more accessible to both technical and non-technical users.

These methods can be categorized as follows:

- **Example based methods**: Leveraging specific instances from the dataset as examples to illustrate how model derives its prediction, enhancing simulatability of the explanations. Examples include: Protypes and criticisms (ProtoDash[51], ProtoPNet[69], MMD-criticism[87]), counter factual explanations[66]
- **Concept based methods**: mapping the important features to specific concepts. Examples include: Concept Relevance Propagation [82], And-Or Graph (AOG) [88]

2.2.2.2.4. Interpretable model design

Interpretable model design approaches focus on creating models that are inherently easier to understand by design. By incorporating transparency directly into the architecture, these methods provide built-in explainability without sacrificing performance. Interpretable design represents a proactive approach to explainability that addresses the core challenge of balancing model complexity with human comprehension.

These methods can be categorized as follows, according to the three approaches of enhancing interpretability as described in Section 2.2.1.2:

- **Simulatability**: These simplified models allows humans to easily understand and can mentally simulate reasoning process. Examples include: Generalized Linear Models (GLM[89]), Generalized Additive Models (GAM[90]) Decision Trees (DT[91]), Decision sets[92], rule sets[93], [94], Random Forests (RF[95]).
- **Algorithmic transparency**: These models utilize structured mathematical functions where each parameter possesses a clear and readily interpretable meaning. This allows for direct understanding of "how" the model arrives at its predictions. Examples include: Tree regularization[96], ProfWeight[97], Teaching Explanations for Decisions (TED) [98], Self-Explaining Neural Networks [99]
- **Decomposability**: These approaches decompose DL models into smaller, understandable submodules. Examples include: ModelSpeX[100], which outlines a dynamic workflow for specifying deep neural network models, engaging domain experts and incorporating explainable AI to analyse

data and model relationships. ViT-NeT[101] is an interpretable architecture specifically designed for fine-grained image classification, incorporating a neural tree decoder that mimics human-like decision-making and offers insights into the model's reasoning process.

2.2.2.2.5. Rule extraction

These techniques transform the complex behaviour of neural networks into simpler, human-readable rule sets. By distilling a model's decision-making into logical statements, these methods provide clear explanations that can be easily verified and understood. Rule extraction offers a practical compromise between the high performance of deep learning and the clear interpretability of traditional rule-based systems.

Rule extraction methods can be categorized as follows:

- **Model decomposition/analysis**: These methods reverse engineer the trained network to identify and express its logic directly. Examples include: Anchors [63], RxTEN [102], DEXiRE [103] (for binary classifiers), and ECLAIRE [104].
- **Surrogate models**: These methods train a simpler, interpretable model to approximate the behaviour of the complex neural network. Examples include: DeepRED [105], TrePAN [106], RuleFit [107], Two-step CNN rule extractor [108]
- **Neuro-symbolic approaches**: These methods integrate neural networks with symbolic reasoning techniques, allowing for learning and inference with logical constraints. Examples include: Neural-symbolic learning [70], DL2 [109], DeepProbLog [110], Learning with logical constraints [111]
- **Using attention mechanisms and concept disentanglement**: These methods focus on identifying and utilizing meaningful concepts within the model's representations.
  - *Concept bottleneck models (CBM)*: Separates processing into predicting concepts from input, then predicting task labels from those concepts. Examples include: Concept bottleneck [112], Concept whitening[113]
  - *Posthoc Concept Extraction (CME)*: Extracts concepts from a pre-trained model's hidden representations. Examples: Now You See Me[114], ConceptSHAP[115]
  - *Disentanglement Learning*: Uses unsupervised or semi-supervised to encourage the DL model to learn disentangled latent factors, which are argued to represent human-perceived concepts. Examples include: Unsupervised Beta-VAE[68], Weakly-supervised[116]
  - *Visual Attention and Captioning*: Leveraging attention mechanisms to generate explanations, such as image captions, highlighting relevant visual features [117]

2.2.2.2.6. Uncertainty and reliability analysis

These approaches quantify how confident a model is in its predictions and under what conditions it might fail. By measuring different types of uncertainty, these methods help users understand the limitations of model outputs and identify potential edge cases. Uncertainty quantification is crucial for building trust in AI systems by providing honest assessments of prediction reliability in different contexts.

Bayesian neural networks [118] 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.

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[119].

Studies have demonstrated a correlation between the measure of uncertainty and the accuracy of object detection [120]. 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.

The presence of aleatoric uncertainty may also be also linked to partial object occlusion, making it a useful indicator of inherent ambiguities in the data.

2.2.2.2.7. Enhanced representation

These techniques modify model architectures to improve interpretability. By incorporating specialized components that highlight important information, these methods make model behaviour more transparent without redesigning the entire system. Enhanced representations offer a middle ground between post-hoc explanation and fully interpretable design by strategically improving model transparency where it matters most.

- **Attention mechanisms**: Attention mechanisms allow models to focus on the most relevant parts of the input data when making predictions. These techniques learn to assign different weights to different elements, highlighting those that contribute most to the task. Examples include:
  - Global-and-local attention (GALA) [121]
  - DomainNet [122]
  - Residual Attention [123]
  - Loss-based attention [124]
  - D-Attn [125]
- **Autoencoder-based**: Autoencoders are neural networks trained to reconstruct their input, forcing them to learn compressed, meaningful representations of the data. When used for interpretability, these representations (the latent space) can be analysed to understand which features are most important for encoding the input. Examples include:
  - Explainer model [126]
  - XCNN [127]
- **Other methods** to enhance representation also include:
  - Adaptive Deconvolutional (Adaptive DeConv) [128]
  - Deep Fuzzy Classifier (FCM) [129]
  - Network In Network (NIN) [130]

3. Specification and Design of FUSA-aware DL solutions

3.1. System scope

A DL component is considered dependable when its development process and deployment architecture adhere strictly to the guidelines (provided in SAFEXPLAIN project as AI-FSM lifecycle and Safety patterns for development and deployment stages respectively), with the system scope defined from the following perspectives:

- ***Operational scope***: Encompassing two main components that establish clear boundaries for a DL-based system's operation
  - *Operational Design Domain (ODD)*: specific conditions under which the DL-based system operates.

  - *Operational scenarios*: Intended use cases and interactions of the system and other actors within the ODD conditions. This provides boundaries on dynamic interactions.
- **Functional scope**: Intended functionalities that the DL component aims to provide within a defined ODD and operational scenarios
- **Integrated architecture scope**:
  - *System architecture*: Within this document, a typical Sense Think Act system architecture is in focus (Figure 2), where:
    - *Sense*: Responsible for perceiving the environment and gathering real-time data for system operations. The Sense component consists of various types of sensors (camera, lidar, radar, IMU…) capturing the surrounding situations and context. The Sense component also includes DL-based perception module, typically performs tasks such as object detection/tracking, semantic/instance/panoptic segmentation, estimate situations including environmental parameters and scenarios classification/parameters.
    - *Think*: Decision making component that process the detections from Sense component and make informed decisions in real-time. This component can either be rule-based, statistical based or DL based. In a safety critical system, this component shall take as its input the detections from Sense components together with other safety assessment measures (from supervision components) and safe operational boundaries to ensure that all actions adhere to the safety requirements with control level of operational risks that may lead to identified hazards.
    - *Act*: This component actuates the decisions made by Think component through actuators, HMI devices or other components that can interact with surrounding environment (e.g. external HMI or messaging). The component ensures that the actions taken are safe.
  - *Development/Deployment environments*
    - *Development environment*: the environment where the DL component is designed, trained and tested before deploying into real world environments. Development environment allows access to large datasets, powerful computing resources for training and testing, software frameworks/libraries, and interactions with different human stakeholders during the development lifecycle.
    - *Deployment (runtime) environment*: The environment under operational context where the DL component is operating in and interacting with real world under the predefined scope (ODD, operational scenarios). Deployment environment is usually limited in terms of computing power, network connectivity, storage and is subject to time critical requirements.
- **Safety aspects**[13], [17]:
  - *Safety goals*: specific objectives to ensure that the DL based system operates safely within the defined ODD and operational scenarios. The goals shall address all unreasonable risks or hazards associated with the system's interactions with environment, users or other systems.
  - *Safety strategies*: Strategies to be applied both in development environments (setting up safety requirements, define the V&V strategies and related test scenarios, applying explainability techniques to generate human understandable evidence) and deployment environment (which safety related components to be employed for the final safe operational architecture)

- Safety requirements: Detail specifications describing how the DL component shall be designed, developed and deployed to meet its safety goals and strategies. Depending on the domain specific risks and severity/critical levels, the safety requirements are required to ensure that system failures shall not lead to hazardous consequences. The DL component dependability relies on its ability to adhere to the requirements throughout the development and deployment lifecycles.

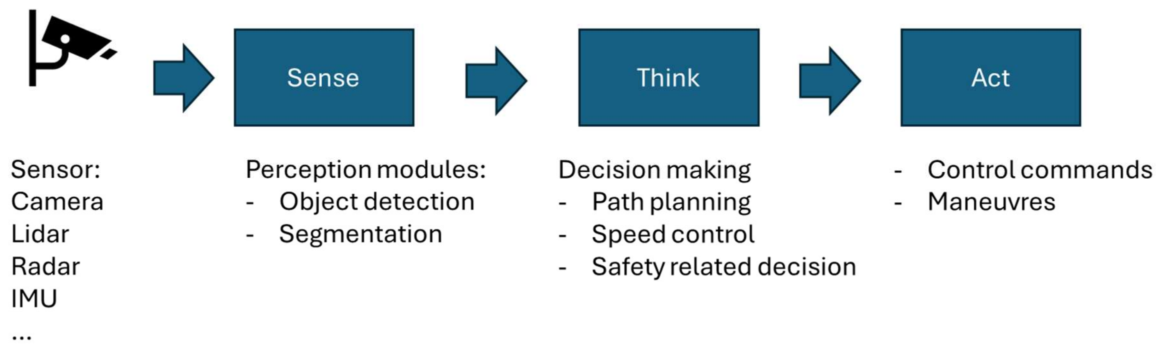


Figure 2: Sense - Think – Act architecture

3.2. Minimum Viable Product (MVP)

A Minimum Viable Product (MVP) prototype has been developed collaboratively across work packages and is used to illustrate how XAI methods can support different activities throughout this document. The MVP consists of the key components as follows:

- **MVP Datasets**: The dataset comprises 10,000 space images featuring diverse backgrounds and a target satellite. Each image is annotated with bounding boxes around the satellite object (Figure 3).
- **AI/ML constituent**: Two state-of-the-art object detection models were selected: (i) SSDLite [131]with MobileNet [132] V3 backbone and (ii) Faster R-CNN[133] also with MobileNet V3 backbone. Both models were trained, tested, and evaluated using the MVP datasets.
- **Safety components**: Anomaly detection, uncertainty-aware models, a safe surrogate model, and an ensemble model are included, with detailed descriptions provided in Section 3.5.2

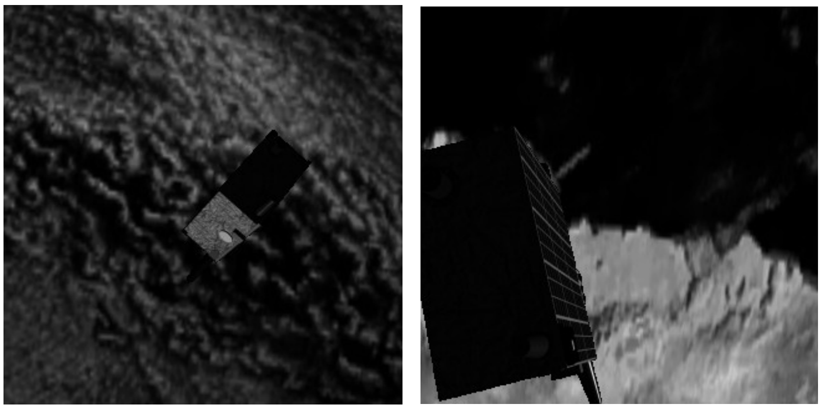
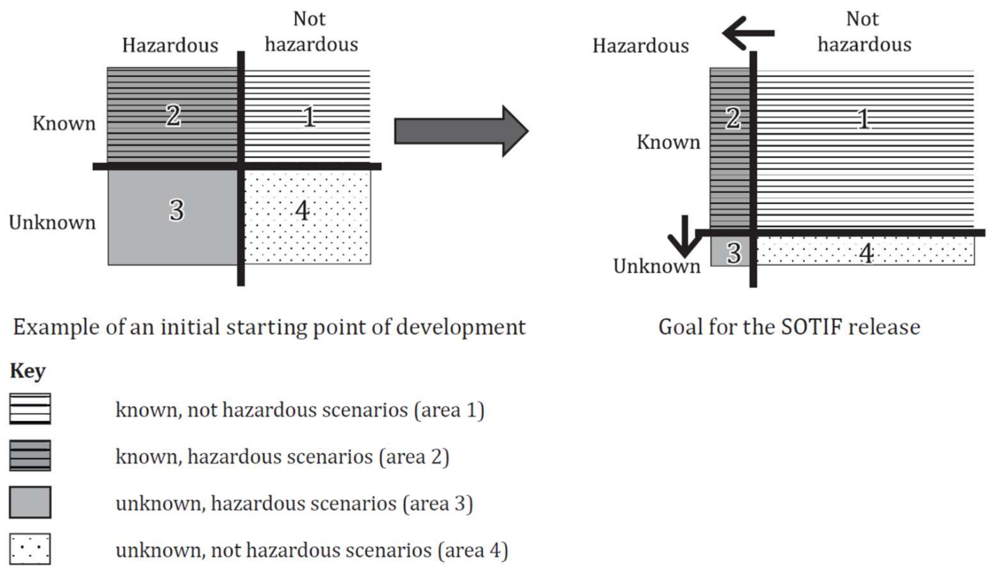


Figure 3: Example images from the MVP dataset

3.3. Uncertainty management strategy

A challenge in integrating DL into safety-critical autonomous systems is addressing the inherent uncertainty associated with its predictions. It is essential to adopt a structured and systematic approach that manages and mitigates this uncertainty to the greatest extent possible. The Safety of the Intended Functionality (SOTIF[12]) provides a guideline for achieving this goal. As illustrated in Figure 4, SOTIF defines scenario categories based on two dimensions: Known/Unknown and Hazardous/Not hazardous. This framework highlights the importance of understanding and managing uncertainty to ensure safe and reliable system operation.



Hazardous
Not hazardous
Known
Unknown
2
1
3
4
Example of an initial starting point of development
Hazardous
Not hazardous
Known
Unknown
2
1
3
4
Goal for the SOTIF release
Key
known, not hazardous scenarios (area 1)
known, hazardous scenarios (area 2)
unknown, hazardous scenarios (area 3)
unknown, not hazardous scenarios (area 4)

Figure 4: Alternative evolution of the scenario categories resulting from ISO21448[11] activities

The SAFEXPLAIN proposed FUSA-aware DL solution accommodates the SOTIF recommendations by actively expanding the "Known" subspace and refining the boundary between subspaces, as indicated by the evolutionary paths in Figure 4. Specifically, the solution corresponds to the two arrows on the right side of the figure, contributing to a safer and more reliable operation. During the Operation and Maintenance (OM) stage, the FUSA-aware DL safety architecture ensures that the system operates safely within the "Known/Not hazardous" region, thereby minimizing the risk of accidents or failures.

Effective management of uncertainty during both development and deployment phases can be achieved by formally identifying and separating the distinct sources of uncertainty. This disentanglement enables each type of uncertainty to be properly managed. Different algorithms can be employed at different steps to monitor and mitigate each uncertainty type in a targeted manner, ensuring that management strategies are tailored to meet system safety requirements.

In our previous work [134], we have proposed disentangled sources of uncertainty into three primary categories: domain uncertainty, model epistemic uncertainty, and aleatoric uncertainty. Figure 5 provides an overview of this disentanglement of uncertainty types, where the upper part of the figure corresponds to the development lifecycle (AI-FSM) and the lower part of the figure corresponds to the operation and monitoring stage (OM stage).

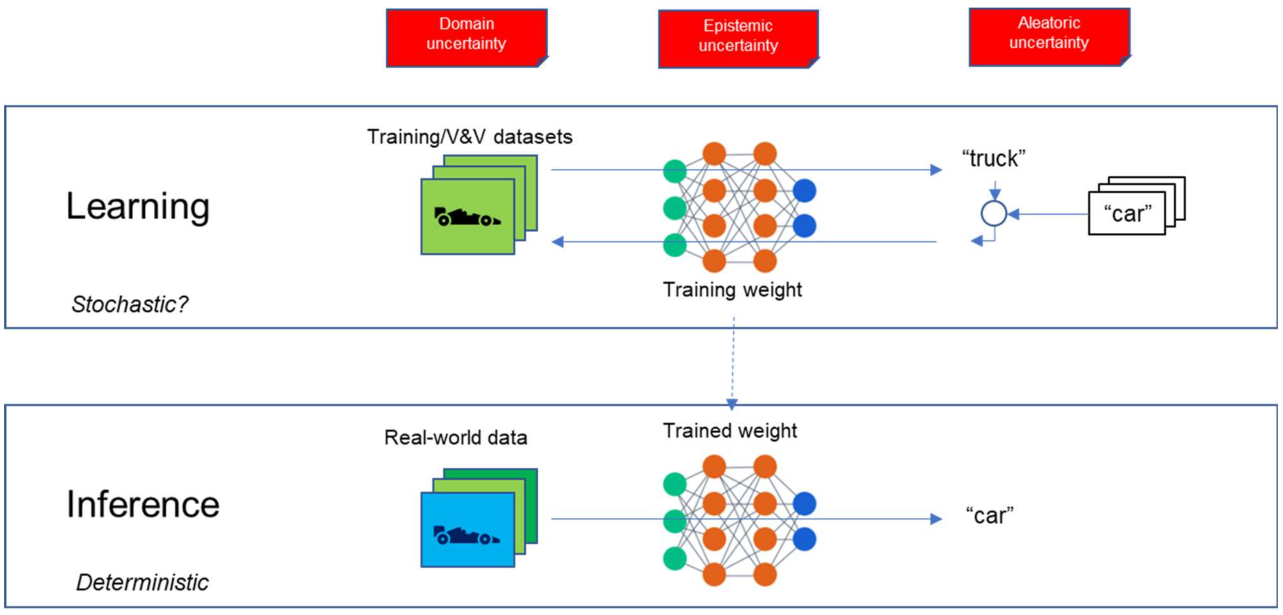


Figure 5: Uncertainty types

Each type of uncertainty will be addressed throughout the lifecycle as follows:

- **Uncertainty reduction in the Development Lifecycle (AI-FSM Phases)**: During this phase, we focus on identifying and minimizing reducible types of uncertainty. We also propose methods/approaches to estimate irreducible uncertainty and the residual components of reducible uncertainty. Additionally, we establish a clear understanding of the boundaries between the "Known" and "Unknown" areas, which serves as a baseline for the anomaly detectors to be used in OM stage. This ensures that the system operates within its safe boundaries, where all known uncertainties are properly managed and mitigated.
- **Uncertainty management in the Operation and Monitoring (OM Stage)**: In this stage, our architecture is equipped with safety components that guarantee the system operates strictly within its known scope (safe boundaries). The mitigation strategy for known uncertainty is continuously verified to ensure its effectiveness. Furthermore, we constantly estimate residual uncertainty and assess the associated risk of entering hazardous situations. If the derived risk exceeds an acceptable threshold, the decision-making module is alerted to execute mitigation actions, ensuring the system's safe operation and preventing potential hazards.

Detailed strategies with connections to the uncertainty types are discussed in the following subsections.

3.3.1. Reduction of uncertainties within development cycles

3.3.1.1. Reduction of domain uncertainty

AI-FSM data management (PhDM) phase is a specific phase designed to address mitigation of domain uncertainty.

XAI methods belonging to data explainer category can be used to support different steps within AI-FSM PhDM phase as follows:

- Identification of analytical and statistical gaps between the collected dataset and the real-world. This will be done with the supports of data explainers in PhDM phase.
    - Data value completeness analysis: Identify feature importance in missing values to determine which features are most critical to the problem.
    - Feature-wise completeness analysis: Extract important features from the data and evaluate missing features that are crucial to supporting the problem.
    - Reconstruction of missing data: Use data descriptors to describe the dataset and synthesize missing data samplings to close the data distribution gap.
    - Data balance analysis: Visualize feature distribution and data point distribution across selected data dimensions to identify potential biases or imbalances. Measure distribution distances between datasets and design approaches to close gaps, including strategies to mix different datasets (e.g., synthetic and collected datasets) to improve overall data quality in terms of distributional representativeness.
    - Data relevance analysis: Extract prototypes and/or criticisms to identify data points or features most representative of the dataset and problem space.
    - Data accuracy analysis: Analyse annotation distributions and identify potential mislabelled instances (e.g., anomalous labels). Use feature importance methods to explain how different data features influence annotations.
- Data collection planning
    - Uncertainty analysis: Use uncertainty quantification techniques (e.g., Bayesian neural networks) to identify high-uncertainty areas that may require more data to be collected.
    - Diversity analysis: Apply clustering methods to guide data balance across unsupervised clusters, ensuring that the collected data is diverse and representative.
- Data preparation
    - Data synthetic: Use GAN-based synthetic data to augment real-world data points in underrepresented areas.
    - Feature importance guided: Identify important features where data augmentation should be focused.
    - Counterfactual guided augmentation: Use counterfactual examples to guide data augmentation and improve the robustness of the dataset.
    - Noise modelling and augmentation: Identify realistic data noises and adversarial noises and use them to augment the dataset and address potential vulnerabilities.

3.3.1.2. Reduction of model epistemic uncertainty

The reduction of model epistemic uncertainty is done in the AI-FSM PhLM phase, provided the assumption that data domain uncertainty has been already mitigated in the PhDM phase. Various model explainers can be employed, targeting AI developer audience, including the below:

- Improving model architectures: Where applicable, design models with decomposition of concepts to ensure that all sub-concepts are accurately modelled and can be thoroughly tested. This approach enables the identification and mitigation of potential flaws in the model architecture.
- Enhancing model explainability: Utilize XAI techniques to increase the transparency of the model (intrinsic or posthoc designs).
- Design uncertainty-aware models: Leverage uncertainty quantification techniques to modify models, enabling them to provide not only predictions but also estimates of epistemic uncertainty.

- Analysing model global/local behaviour: Employ various model explainers to verify the learned feature space and assess the model's robustness to input changes, including both expected and unexpected variations.

3.3.1.3. Logging of aleatoric uncertainty

Aleatoric uncertainty, being irreducible, can be quantified and estimated using XAI methods within the AI-FSM PhLM phase. To achieve this, aleatoric uncertainty aware models will be designed and trained on the same datasets used for the primary model. This approach enables the estimation of uncertainty levels associated with the data quality and label quality.

Known uncertainties corresponding to the verified datasets are used as the baseline to support the OM stage.

3.3.2. Management of residual uncertainty within OM

3.3.2.1. Management of data/model epistemic uncertainty

Residual epistemic uncertainty from both data and model will be mitigated by XAI enabled supervision components to ensure the trustworthiness of the system, even in the presence of uncertainty. The key XAI components include:

- Out-of-Distribution (OOD) detectors: These detectors monitor the input data, model activation patterns, and output predictions to ensure they fall within established boundaries. These boundaries are defined based on the knowledge gained during the AI-FSM phases, as documented in corresponding artifacts.
- Certified surrogate model: This safe alternative model provides a fallback option when epistemic uncertainty levels exceed predefined acceptable limits. The certified surrogate model guarantees prediction confidence and handles situations where the primary model's uncertainty is too high.

3.3.2.2. Management of aleatoric uncertainty

The aleatoric uncertainty-aware model will be integrated into the OM safety architecture, providing insights to inform decision-making and ensure reliable system performance. The employment of this model enables various approaches, including:

- **Trade-off mechanisms**: When estimated uncertainty exceeds an acceptable threshold, the system can adapt by:
    - Running inference on multiple consecutive frames, increasing processing delay but reducing uncertainty to an acceptable level.
    - Dynamically adjusting the trade-off between prediction accuracy and latency, ensuring that the system operates within defined safety bounds.
- **Model fusion**: The decision function, based on an ensemble model, will utilize estimated uncertainty as input to:
    - Produce final consolidated predictions with higher certainty, considering the estimated uncertainty of individual model outputs.
    - Assign weights to each model's output based on its corresponding uncertainty estimate.
- **Defining safe limits for system operation**: The aleatoric uncertainty-aware model will help establish safe limits for the system, where:
    - Prediction variations within predefined boundaries can be tolerated and used by the decision-making component.
    - The system's operating range is defined, ensuring that predictions are reliable and trustworthy, even in the presence of uncertainty.

3.4. XAI usages in AI-FSM Lifecycle

The integration of explainable AI (XAI) techniques within the AI-FSM lifecycle ensures transparency, robustness, and compliance with safety standards throughout the development and deployment of deep learning solutions. XAI supports various phases of AI-FSM, from defining safety goals and enhancing model robustness to improving data management, learning processes, and inference monitoring.

The primary objective of ensuring DL model compliance within the AI-FSM framework is to systematically identify, mitigate, and document uncertainties, thereby enhancing the model's reliability, robustness, and overall trustworthiness. Specifically, this involves:

- **Reducing reducible uncertainties**: addressing domain uncertainty and model epistemic uncertainty through targeted efforts to improve knowledge and understanding in corresponding steps.
- **Characterizing residual uncertainties**: acknowledging and documenting the remaining uncertainties, including irreducible aleatoric uncertainties.

3.4.1. Safety goals & strategy

Ensuring safety throughout the AI-FSM lifecycle requires a structured approach based on safety argument patterns and rigorous verification and validation (V&V) activities. These mechanisms establish confidence in deep learning models by systematically assessing their performance, reliability, and compliance with predefined safety requirements. The core strategy is to reduce uncertainty-related risks by ensuring that all hazardous situations are identified, assessed, and documented, while continuously monitoring and reporting safe operational boundaries.

Safety argument patterns serve as a foundation for justifying the dependability of AI components. These patterns link system specifications, risk assessments, and supporting evidence from V&V activities to ensure that AI models meet functional safety standards. By integrating XAI techniques, these patterns gain traceability and interpretability, allowing stakeholders to understand and assess AI-driven decisions effectively.

V&V activities span across different lifecycle phases, from data validation and model training to inference monitoring and operational diagnostics. These activities focus on detecting and mitigating failure modes, biases, and edge cases, ensuring that AI models remain robust under real-world conditions. Techniques such as anomaly detection, uncertainty estimation, and neural coverage analysis contribute to a proactive risk management approach, reducing unexpected failures and improving fault tolerance.

A key element of the safety strategy is uncertainty management, which involves continuous monitoring of AI predictions to detect out-of-distribution (OOD) data, concept drift, and anomalous behaviours. XAI techniques play a crucial role here by providing interpretable insights into model confidence and decision boundaries, ensuring that AI systems operate within validated safe zones. The integration of supervisory monitors and decision functions further enhances reliability by dynamically assessing the trustworthiness of AI-generated outputs and enabling corrective actions when necessary.

Ultimately, embedding safety argumentation and explainability throughout the AI-FSM lifecycle ensures that deep learning models remain traceable, robust, and aligned with functional safety requirements, supporting their deployment in critical and high-risk environments.

3.4.2. Robustness, Explainability

Robustness in the AI-FSM lifecycle refers to the stability of model explanations when subjected to slight perturbations in input, ensuring that the model's output remains consistent. A robust AI system should provide stable predictions and interpretable explanations, even when exposed to noise, adversarial manipulation, or unseen environmental conditions within its ODD. Robustness is closely linked to explainability, as highly unstable explanations indicate potential vulnerabilities, making the system more susceptible to adversarial attacks or unsafe behaviours.

3.4.2.1. Adversarial attacks and defence mechanisms

Adversarial attacks exploit model vulnerabilities by introducing subtle perturbations that mislead AI predictions while avoiding being detected e.g. by maintaining human-perceptible similarity to valid inputs. These attacks can occur at different stages.

Adversarial Attack Types

- **Input based with or without model access (OM stage):** Manipulating input with crafted noises to influent output. These attacks introduce small but strategic changes to input data to cause incorrect predictions while keeping the changes imperceptible to humans. There are two types of such attacks:
  - **Black-box attacks**: Attackers do not have direct access to the model but probe its responses to generate adversarial inputs.
  - **White-box attacks**: Attackers have full knowledge of the model, allowing them to manipulate gradients to create adversarial perturbations (e.g., FGSM, PGD).
- **Model poisoning (AI-FSM DM):** add malicious behavior (to specific types of inputs) into the model by unauthorized access allowing attackers to manipulate training dataset.
  - Occur during training, where maliciously modified data is introduced into the dataset, leading to hidden biases or targeted misclassifications.
  - These attacks are particularly dangerous because they can compromise model reliability without visibly affecting training accuracy.
- **Manipulating the real world (OM):** adding strange objects, masking object, manipulating illuminations… These attacks alter the physical environment to mislead AI models without modifying digital inputs. Examples include:
  - **Adding strange objects** (e.g., adversarial stickers on traffic signs to fool autonomous vehicles).
  - **Masking critical features** (e.g., obscuring objects in a way that disrupts detection).
  - **Manipulating lighting conditions** (e.g., using shadows or reflections to distort object recognition).

Defense approach

To ensure robustness to an identified set of possible adversarial attacks, a combination of proactive (AI-FSM) and reactive (OM) defence mechanisms is required.

- Mitigation actions within AI-FSM lifecycle:
  - PhDM: adding anticipated adversarial data points to datasets. Incorporate **adversarial data augmentation**, adding perturbed examples to training datasets to improve generalization w.r.t. predefined types of adversarial attacks.
  - PhLM: adding crafted adversarial datapoints dynamically during training (adversarial training, Tradeoff balance robustness and accuracy)
    - Model design: Introducing adversarial loss functions, explicitly penalizing incorrect predictions caused by adversarial perturbations.
    - Implement adversarial training, dynamically injecting adversarial samples into training to improve model resilience.
    - Optimize the trade-off between robustness and accuracy, ensuring the model can handle adversarial examples without overfitting.
- Mitigation strategy in OM stage:

    - Uncertainty aware model: Use uncertainty aware models estimate prediction uncertainty, flagging unreliable outputs.
    - Adversarial attack detection as an additional supervisory monitor
        - Deploy **anomaly detection**[135] to identify **unexpected perturbations** in input data.
        - Cross-check model explanations over **consecutive frames or redundant AI models** to detect inconsistencies caused by adversarial attacks.

By embedding robustness and explainability throughout the AI-FSM lifecycle, AI systems become more resilient to adversarial threats and environmental variations. A robust model ensures that explanations remain stable even when inputs change slightly, while explainability helps detect and mitigate vulnerabilities before they compromise system safety. Integrating these principles enables AI-driven systems to operate reliably and transparently in safety-critical applications.

3.4.2.2. Robustness

Robustness is a multifaceted concept that encompasses not only performance but also generalization, resilience to adversarial threats, and explainability across all phases of the AI-FSM lifecycle. Achieving robustness requires a comprehensive approach that involves strategies at various levels.

Data management strategies

Robustness improvement can be done via data generalization. Within the context of SAFEXPLAIN, data generalization refers to improving the representativeness and relevancy of the datasets.

- Data augmentation: the followings can be considered:
    - Geometric transformations such as rotation, flipping, and scaling to ensure the model is robust against positional changes. These transformations are done to generate missing data points and achieve expected distribution (with respect to real world)
    - Colour and texture augmentation: to prevent over-reliance on low-level image statistics, which can be resulted from the data collection limitations but not necessarily reflect real world data.
    - Synthetic data generation to create additional data points that are underrepresented in real-world data samplings.
- Additionally, dataset profiling is performed to detect biases, ensuring that the model does not learn spurious correlations that could lead to unexpected failures in critical situations. Ensuring dataset balance, feature importance analysis, and diversity in training samples prevents overfitting and strengthens the model’s ability to handle novel situations.

Strategies for model design

A robust AI system must learn invariant features that remain stable across different conditions. If a model learns fragile correlations, it may perform well on training data but fail in deployment when faced with small variations. To enhance robustness at the architectural level, the following strategies can be adopted:

- **Feature disentanglement techniques**, ensuring that the model captures core semantic information rather than overfitting to specific patterns. For example, in object detection, the model should learn shape-based features rather than relying on background textures, which could mislead predictions.
- **Structural regularization methods**, such as dropout, batch normalization, and weight decay, which force the model to distribute its learned representations more evenly, reducing over-reliance on specific neurons or layers.
- **Intrinsic explainability mechanisms**, where models incorporate interpretable structures such as attention mechanisms, decision trees, or prototype-based learning to maintain explainability and stability.

By embedding these design choices into the model architecture, we can ensure that learned representations remain **consistent and interpretable**, even under **adversarial conditions**.

Strategies during model learning and inference

- **Model Learning**: balancing overall accuracy with distribution of accuracies across dataset space.
- **Inference**: preparing input data by removing noises and detecting attacks (e.g., unexpected input variations/distributions). Uncertainty aware model with adversarial noises
- **Verification and validation (V&V)**: testing with adversarial noises (both intended and unintended) to ensure robustness. As illustrated in Figure 6, an example of robustness testing against adversarial attacks is provided, where images are intentionally corrupted with varying levels of adversarial noise (measured by Epsilon, ε). Notably, at a noise level of ε = 0.002, the model begins to exhibit confusion in its detections, highlighting the importance of robustness testing in ensuring reliable performance under adverse conditions.

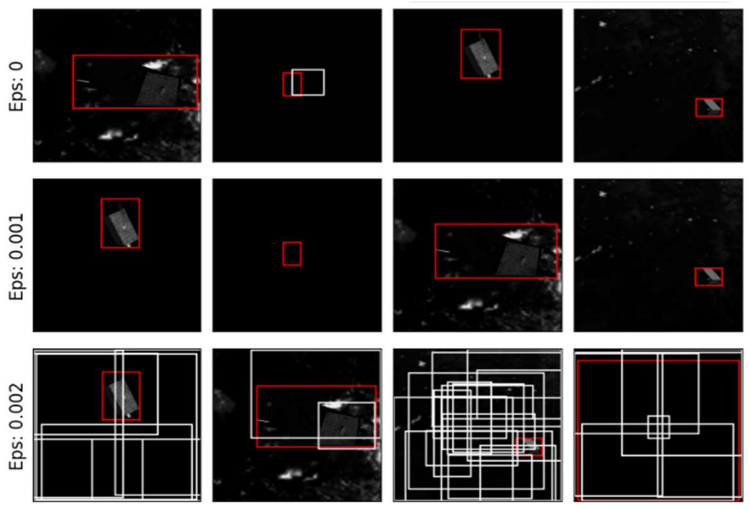


Figure 6: FGSM attack on 4 sample images using different degrees of small perturbation (Eps).

3.4.2.3. Explainability

To foster trust and confidence in generated artifacts, explainability is essential where applicable. This involves providing insights into the data and models used to produce these artifacts, ensuring that they are transparent, interpretable, and reliable.

Explainability shall be maintained for both data and model:

- **Data explainability and traceability**: Data should be accompanied by detailed explanations and statistical measures to assess its compliance with set requirements.
  - *Traceability*: The ability to track the origin, processing, and manipulation of data throughout its lifecycle.
  - *Statistical measures*: Providing summary statistics and other relevant metrics to characterize the data distribution, variability, and interactions between variables.
  - *Reproducibility*: Maintaining a record of data access, modifications, and confidentiality measures to ensure reproducibility and feature importance-based diagnoses.
- **Model explainability**: Model explainability is crucial for ensuring that the DL system behaves as intended within its scope and provides required safety related performance

- *Conceptual functionality decomposition*: Breaking down complex models into smaller, manageable components to verify that each sub-functionality behaves correctly according to design logic.
- *Supporting V&V strategies*: Using model explainers such as those of metaheuristic search or disentanglement of model's feature space w.r.t. model performance clusters.
- *Diagnosis*: Providing insights into how input data and/or feature affects model behaviour, enabling the identification of potential biases, errors, or areas for improvement.

3.4.3. XAI usages in Data Management

Effective data management in the AI-FSM lifecycle is critical for reducing domain uncertainties and defining the boundary between known and unknown data distributions. Explainable AI (XAI) techniques play a key role in ensuring data traceability, quality, and reliability, helping to establish a well-defined Operational Design Domain (ODD). This structured approach enables Out-of-Distribution (OOD) detection, ensuring that AI models remain robust against unseen or anomalous data.

3.4.3.1. Data requirements

To ensure that the dataset meets the Deep Learning (DL) system's functional and safety requirements, it must be traceable, reproducible, and free from data integrity issues. AI-FSM leverages XAI-based data profiling, validation, and integrity assessment tools to ensure that datasets align with the AI system's intended use.

- **Data Traceability:** Establishing a direct link between raw data and DL model requirements ensures transparency in dataset origins. Version control mechanisms track dataset modifications, preventing unintentional data shifts or inconsistencies during retraining. XAI-enhanced dataset lineage tracking enables explainability by maintaining a clear audit trail of data transformations.
- **Data Integrity:** XAI-based data validation rules ensure data quality and consistency by measuring the percentage of missing values and their impact on model learning, continuity of time-series data to identify unexpected gaps in sequences, and statistical consistency by verifying that feature distributions remain within expected ranges. Malicious data modifications, such as data poisoning, are detected using anomaly detection models and distribution consistency checks.
- **Raw Data and Annotation Requirements:** Bounding box consistency checks verify that object annotations are correctly labelled, avoiding misalignment issues that could impact model learning. Data balance analysis ensures that critical classes and features are well-represented in the dataset, preventing biases in model predictions.
- **Data Gaps and Compliant Tools:** Dataset completeness verification identifies missing data segments that may lead to performance degradation in critical scenarios. Compliant tools include explainability-driven dataset metrics and integrity checks, such as outlier detection for spotting inconsistencies and statistical summaries of dataset properties to detect imbalances and biases.

3.4.3.2. Data collection

The data collection phase establishes a baseline dataset by applying XAI-based analysis and reporting techniques to ensure that collected data is valid, representative, and free from systematic biases.

- **Baseline the Collected Datasets:** Ensuring dataset consistency involves verifying file format standardization to prevent errors in data loading, dataset volume and balance analysis through histograms and heatmaps to detect imbalances in class distributions, and dataset comparisons to track changes across different collection stages.
- **Generate Reports with Data Profiling:** XAI-enhanced data profiling tools, such as DataGradients[136] or SHAP explainers for the entire dataset as distribution, generate statistical graphs that describe feature distributions, label consistency, and dataset shifts over time.

- **Describe Multi-Dimensional Data:** Variational Autoencoders (VAEs) extract latent feature representations, providing a high-dimensional understanding of dataset structure. Clustering techniques, such as k-means and hierarchical clustering, reveal underlying data patterns, improving explainability.
- **Synthetic Data Quality Assessment:** Synthetic data augmentation is validated using Fréchet Inception Distance (FID) to measure similarity between synthetic and real data distributions, Kernel Inception Distance (KID) to assess feature-level alignment, and Structural Similarity Index (SSI) to evaluate perceptual consistency.

3.4.3.3. Data preparation

In the data preparation phase, XAI techniques ensure data cleanliness, accurate labelling, balanced distributions, and robust preprocessing.

- **Annotation Consistency and Accuracy:** Statistical plots reveal annotation inconsistencies, such as bounding box overlaps that could lead to misclassifications in object detection models. Class distribution analysis ensures sufficient representation across different ODD parameters, such as weather and time of day.
- **Data Augmentation for Robustness:** Augmentation techniques, including scaling, rotation, and flipping, improve generalization and robustness. XAI-based metrics measure class balance before and after augmentation to ensure that minority classes are sufficiently represented and analyse distribution stability after transformations to prevent data distortions that could affect model performance.
- **Data Cleaning and Labelling:** Noise characterization and removal use autoencoders to denoise corrupt images and restore missing information. Manual re-labelling recommendations help address datasets where automated correction is insufficient.
- **Data Preprocessing for Model Input Standardization:** Normalization, scaling, and feature selection improve dataset quality while maintaining explainability. XAI-based dimension reduction techniques, such as PCA, UMAP, and t-SNE, help measure distributions across different feature dimensions before and after preprocessing and identify the most important features while maintaining explainability.

3.4.3.4. Data verification

The data verification phase ensures that datasets align with the requirements for deep learning models, reducing domain uncertainties and improving explainability. By leveraging XAI tools, this phase validates whether collected data covers expected distributions, identifies representative samples, and establishes clear boundaries between known and unknown data.

3.4.3.4.1. Data Profiling

Data profiling involves evaluating data distributions and structure to confirm that the dataset meets predefined quality and completeness criteria. This step generates reports on dataset characteristics, ensuring that no significant biases or inconsistencies affect model training and inference. Key verification aspects include:

- **Class distributions**, which assess whether different object categories are adequately represented to prevent biases in model predictions.
- **Bounding box (Bbox) distribution analysis**, which checks the sizes and locations of detected objects to ensure consistency in training data, particularly in object detection tasks.
- **Image statistics**, including size, resolution, and colour profiles, which verify whether images in the dataset maintain a standard format, reducing preprocessing variability.

- **ODD and scenario-related parameter distributions**, which analyse contextual data such as time of day, illumination levels, weather conditions, target distances, viewing angles, and scene variations. Ensuring that these parameters are well-distributed helps the model generalize better across diverse conditions.
- **Timestamp-based data consistency**, which ensures that time-dependent data is correctly ordered, preventing temporal inconsistencies that could affect models reliant on sequential inputs.

Figure 7 illustrates class distribution and bounding box distribution as part of the data profiling activity for Railway UC.

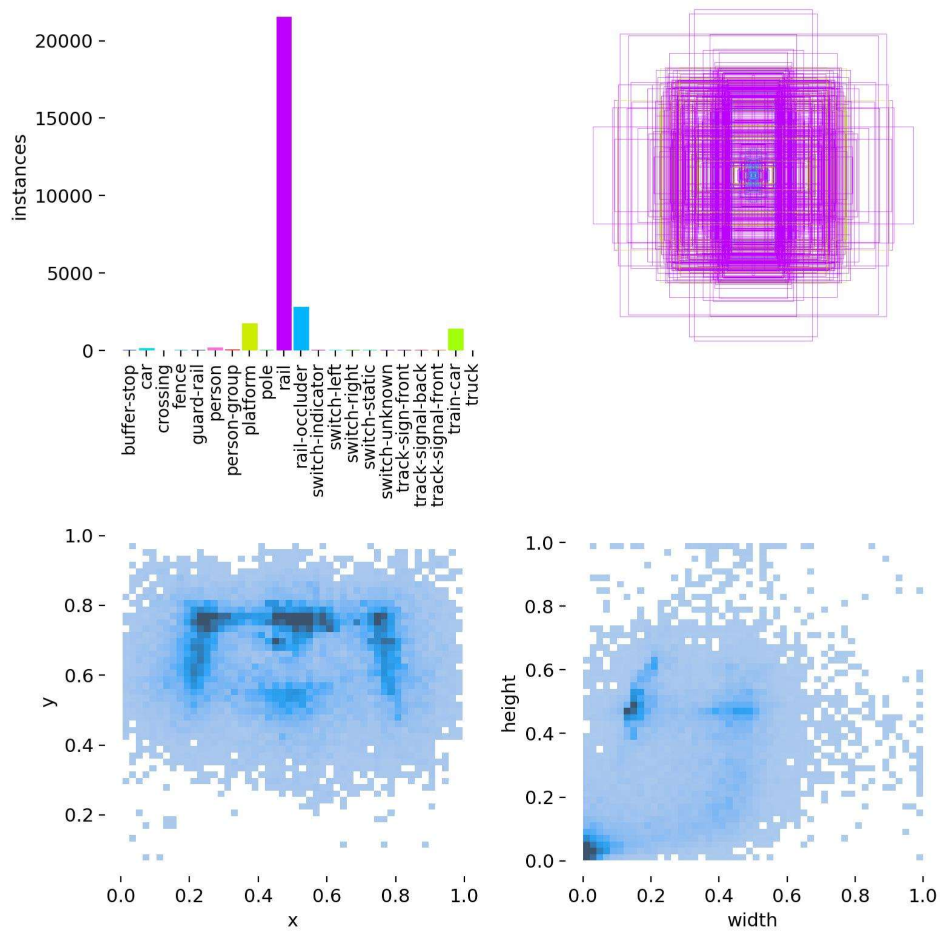


Figure 7: Bounding box and class distribution for Railway use case.

3.4.3.4.2. Data Prototyping

Data prototyping helps extract representative examples from the dataset, ensuring that key patterns, objects, or concepts are well captured and understandable. This process enables explainability by linking AI decision-making to specific dataset elements. Usages of data explainers for this purpose are:

- **Prototype extraction via clustering methods** identifies core representations of objects and concepts within the dataset. By grouping similar samples, clustering ensures that each category is well-defined, improving interpretability.
- **Prototype patching** examines whether smaller image regions still retain key characteristics necessary for classification. This method is particularly useful for assessing whether AI models rely on meaningful features rather than background artifacts or irrelevant details.

Prototyping enhances the explainability of model decisions by demonstrating how well the dataset covers real-world variations and whether the model's learned representations align with human-interpretable features.

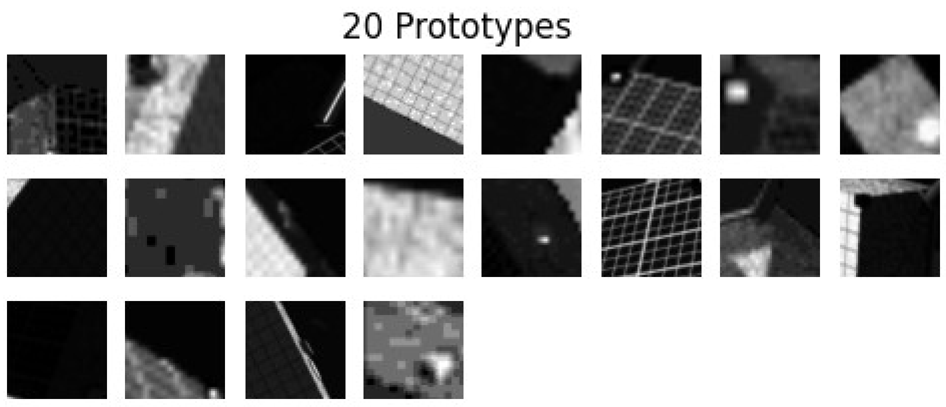


Figure 8: Extracted prototypes of MVP dataset, patch size 100x100

Figure 8 provides an example of extracted prototypes (typical small, squared patch of input images) that most likely representing the dominant image feature in the MVP dataset. The algorithm used is MMD-criticisms[87].

3.4.3.4.3. Data Descriptors

Data descriptors provide a structured way to summarize dataset characteristics, helping define the boundaries between known and unknown data distributions. These descriptors support the identification of OOD data, ensuring that AI models operate within reliable and well-understood domains. Applications of data explainers include:

- **VAE-based descriptors** use Variational Autoencoders (VAEs) to learn a compressed representation of dataset features. By encoding the dataset into a lower-dimensional latent space, VAEs help capture meaningful variations while filtering out noise, improving dataset interpretability.
- **Distribution descriptors** log feature distributions across different dataset dimensions. These descriptors may include operational design domain (ODD) parameters such as weather conditions, lighting variations, or target attributes, as well as annotations and feature-based representations extracted from VAEs or clustering methods.
- **Latent space analysis** applies techniques like t-SNE, UMAP, or PCA to visualize the structure of the dataset in lower-dimensional spaces. These methods help reveal whether different data categories are well-separated or if overlaps indicate possible misclassifications or biases in the dataset.
- **Boundary reporting** defines the limits between in-distribution and out-of-distribution data, identifying areas where the model might struggle to generalize. By mapping these boundaries, AI-FSM ensures that decision-making remains reliable and interpretable, reducing the risk of unpredictable behaviour in deployment.

Figure 9 illustrates the data descriptor, describing each data point (image) as a tuple {VAE weight, latent values}. The VAE model has been trained with MVP dataset.

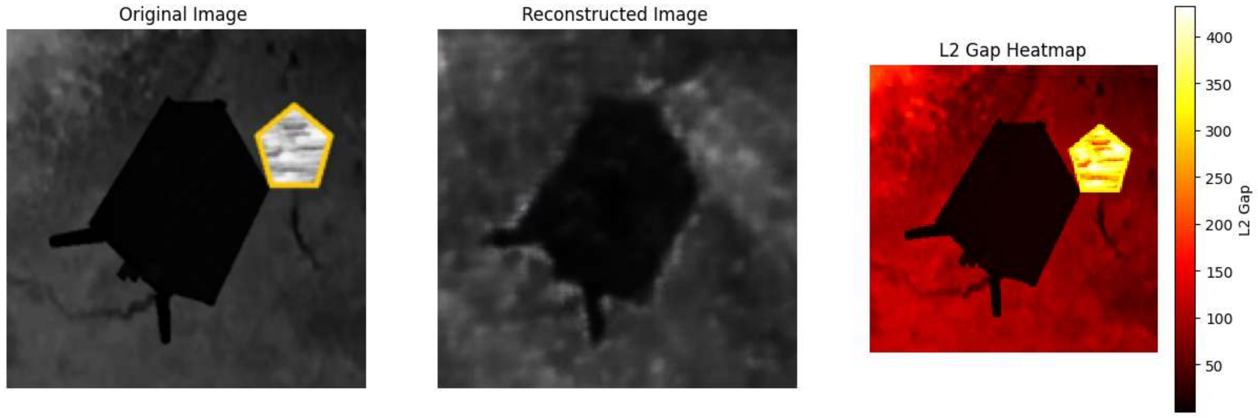


Figure 9: Trained VAE model as data descriptor. The model describes its input image (left) as an reconstructed version (center) and corresponding latent vector. The reconstruction error map (right) highlights the reconstruction error area (anomalous area)

3.4.4. XAI usages in Learning Management

Learning management in the AI-FSM lifecycle focuses on structuring models to mitigate the model epistemic uncertainty in a way that enhances interpretability, modular transparency, and verification of decision-making logic. Different verification and validation (V&V) activities should be planned to evaluate whether models follow structured, logical, and understandable reasoning processes.

Here we discuss how models shall be made explainable by design with reasoning logics shall be made transparent or (more transparent). Different V&V activities shall be planned to verify modular logics.

3.4.4.1. Model design

The selection of an appropriate design architecture for a deep learning (DL) model is a crucial first step in reducing epistemic uncertainty related to a specific problem. The chosen model structure serves as the foundation for defining the entire model space, where each trained model is represented by its unique set of parameters, effectively mapping to coordinates within this space. A well-designed model space should ideally encompass the optimal solution and possess a structure that enables efficient convergence towards this ideal solution during the training process.

Several critical questions must be addressed:

- Does the input data provide sufficient insights to enable the model of such architecture to solve the problem effectively?
- Are the dimensions of the model parameters in the model space comprehensive enough to capture and account for all important aspects of the problem adequately?
- Can the model space be decomposed into subspaces, corresponding to sub-problems, given the domain knowledge to ensure that the complex problem can be solve correctly via solving its subproblems?

The modular design of deep learning models plays a key role in improving transparency and explainability. By decomposing models into functional blocks, it becomes easier to analyse and validate each component independently, ensuring that the system’s decision-making logic remains interpretable and traceable.

Transparency of internal working of selected model can be provided by some architectural modifications:

- **Explainable feature extractor** Specify key layer(s) of the model’s backbone and add functions to extract activation patterns and/or gradients (requiring backward pass).

- **Embedded attention mechanisms**: add additional block accounting for attentions such as CBAM[137]
- **Embedded uncertainty estimate mechanism**: add additional block accounting for uncertainty estimates and modify the loss function accordingly
- **Interpretable surrogate model**: train an interpretable surrogate model on the same dataset to predict the model predictions. This surrogate model can then be used to explain the main model.
- **Feature importance**: Design the model so that it best leverages the extracted dominant prototypes or data features as techniques provided in PhDM.
- **Disentanglement of backbone feature space**: Using architecture design and training techniques to introduce concept representation into the feature space (e.g. specific dimensions are connected to specific concepts)

3.4.4.2. Model training

During the Model training step in PhLM, XAI tools are leveraged to minimize model epistemic uncertainty from multiple angles, including:

- **Optimizer and Scheduler Selection**: ensuring that the chosen training optimizer and scheduler promote model robustness and resilience to varying training conditions
- **Convergence Monitoring**: verifying that the training process converges to a global optimum, rather than getting stuck in local minima.
- **Subobjective Alignment**: confirming that each subobjective (represented by individual loss components) is properly supported and balanced to achieve overall model objectives
- **Overfitting Prevention**: preventing model overfitting by carefully selecting the batch size and learning rate, and monitoring convergence stability through visualization of iteration logs to minimize fluctuations caused by incoming batches of data.
- **Reproducibility**: ensuring that the training process is reproducible, allowing for consistent results and facilitating model reliability and trustworthiness.

To ensure that the DL model behaves as expected and aligns with domain expertise, various XAI techniques can be employed in this step. These techniques can be categorized into several key areas:

Global model explainers can be employed to validate the overall behaviour of the DL model, ensuring it aligns with domain expertise and expectations. Domain experts verify that the model behaves as expected, providing confidence in its performance. Examples of global explainers include LIME (Figure 10), SHAP (Figure 12) or visualizing an interpretable model that surrogates the main model.

Explainability techniques enable AI developers to monitor the convergence process and ensure that the model is learning in the right direction. By analysing the explanations of intermediate results, developers can verify whether the model's performance is improving as it converges. This allows them to:

- **Track progress**: evaluate the model's improvement over time and identify potential issues

- **Identify biases**: detect any emerging biases or flaws in the model's decision-making process
- **Refine the model**: adjust the training process, hyperparameters, or architecture to optimize convergence and improve overall performance
- **Analysing model inner activation** and feature representation (if model uses or ignores important features).

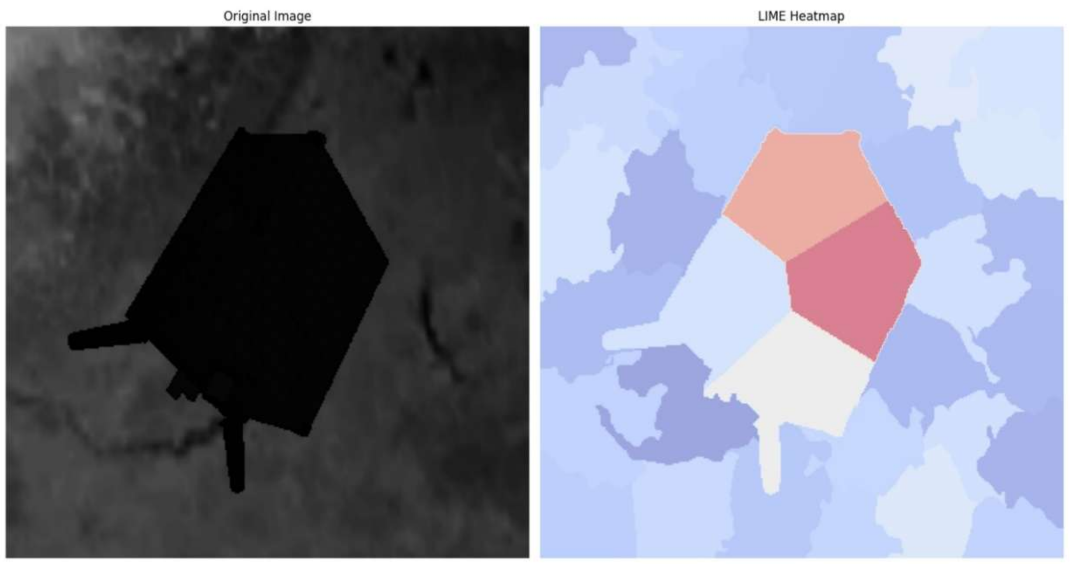


Figure 10: LIME Heatmap (30 SLIC segments)

To investigate how different parts of the model respond to distribution shifts in the input data, we can also leverage XAI tools. One such approach involves analysing the Mean Maximum Discrepancy (MMD) of activation patterns extracted at key layers of the DL model. This analysis provides insights into how the model's internal representations change when faced with different datasets. As illustrated in Figure 11, we computed these distances for key layers of the MVP Model using two distinct datasets: the original MVP dataset and a randomly selected dataset. By comparing the MMD values, we can gain a deeper

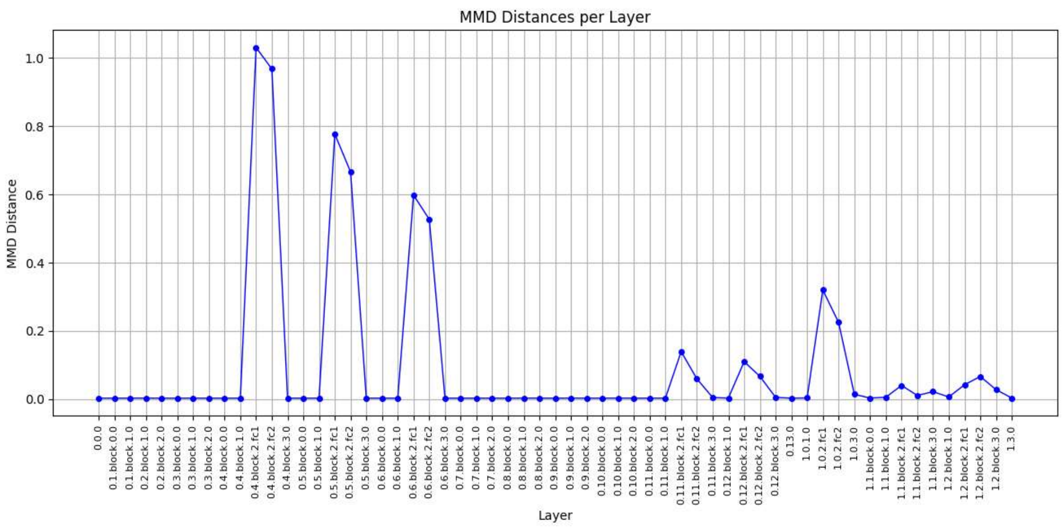


Figure 11: MMD distance per MVP model's key conv layers (comparing two datasets)

understanding of how the model's behaviour changes in response to variations in the input data distribution. A model that is truly robust to domain gaps between datasets should exhibit a decreasing Mean Maximum Discrepancy (MMD) distance as the input data propagates through its layers, ultimately converging to zero at the deeper layers.

3.4.4.3. Model evaluation

To determine which model architecture is best suited for a particular problem, it's essential to evaluate different architectural settings and measure their impact on model performance. This involves:

- **Layer setting evaluation**: assessing the effects of varying layer configurations, such as: Number of layers, Convolutional kernel size, Bottleneck size
- **Hyperparameter tuning**: optimizing hyperparameters to achieve optimal model performance

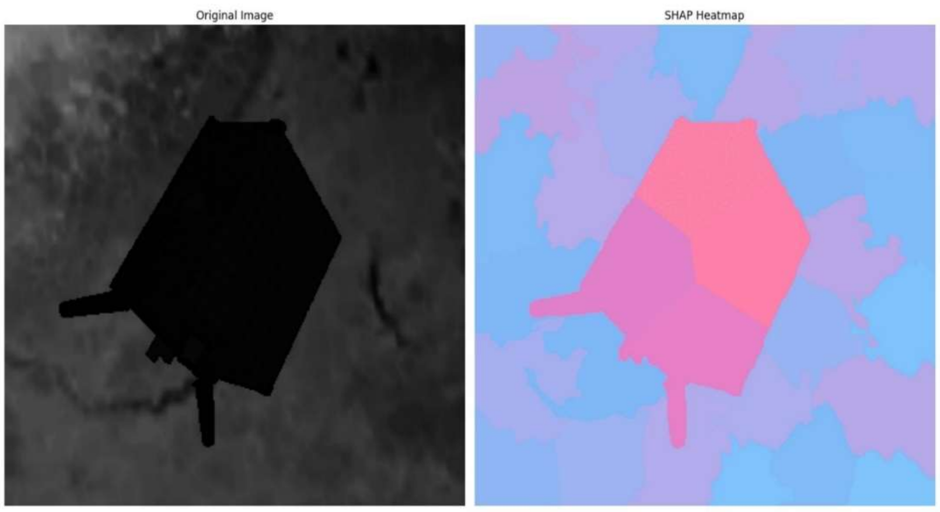


Figure 12: SHAP heatmap with superpixels

XAI tools can support this evaluation process by providing insights into the model's behaviour and decision-making processes. XAI techniques can be used in conjunction with hyperparameter selection to design models that provide the trade-off between:

- **Timing**: computational efficiency
- **Performance/Accuracy**: model accuracy and reliability
- **Explainability**: transparency and interpretability of model decisions

To gain a deeper understanding of the model's internal workings, XAI algorithms can be applied to analyse:

- **Feature importance**: measuring the contribution of different features or components (e.g., heads, skip connections) to the final prediction
- **Activation and gradient analysis**: visualizing activations and gradients at specific layers, for example
    - Last CNN layer
    - Different blocks of Feature Pyramid Network (FPN)
    - Region Proposal Network (RPN) heads
    - Classification heads

3.4.4.4. Model verification

In this step, the goal is to ensure that the model performs well and remains robust across the dataset space, considering variations in input parameters. This involves:

- **Performance evaluation**: assessing how the model's performance varies with different input parameters
- **Robustness assessment**: measuring the model's ability to withstand common adversarial noise and attacks

Explainable AI (XAI) algorithms can be applied to test and verification data to:

- **Evaluate trade-off relationships**: analysing the balance between
  - *Timing*: computational efficiency
  - *Performance*: model precision and reliability
  - *Uncertainty*: confidence in model predictions
  - *Explainability*: transparency and interpretability of model decisions
- **Measure performance degradation**: assessing how the model's performance deteriorates when faced with generalized datasets, including those with adversarial noise
- **Report correlation analysis**: identifying relationships between performance and dataset dimensions
- **Diagnose** if the **model's behaviours** are correct to the domain knowledge (e.g. analysing the attention heatmap as illustrated by Figure 13)

To test the model's robustness against adversarial noise, noise generators and attack simulators can be used to:

- **Generate adversarial examples**: creating input data that is designed to mislead or deceive the model
- **Simulate attacks**: testing the model's defences against various types of adversarial attacks

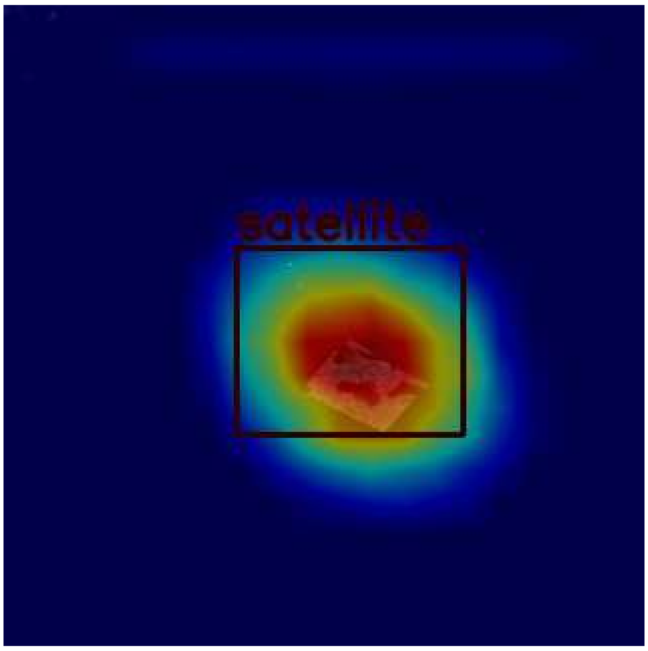


Figure 13: MVP model EigenCAM

XAI algorithms also support to generate report on expected behaviours, including:

- **Uncertainty levels**: predicting the level of uncertainty associated with different input parameters
- **Performance expectations**: estimating the model's performance for specific input parameters (see Figure 14)
- **In-distribution activation patterns**: analysing the model's internal workings and identifying typical activation patterns for in-distribution data

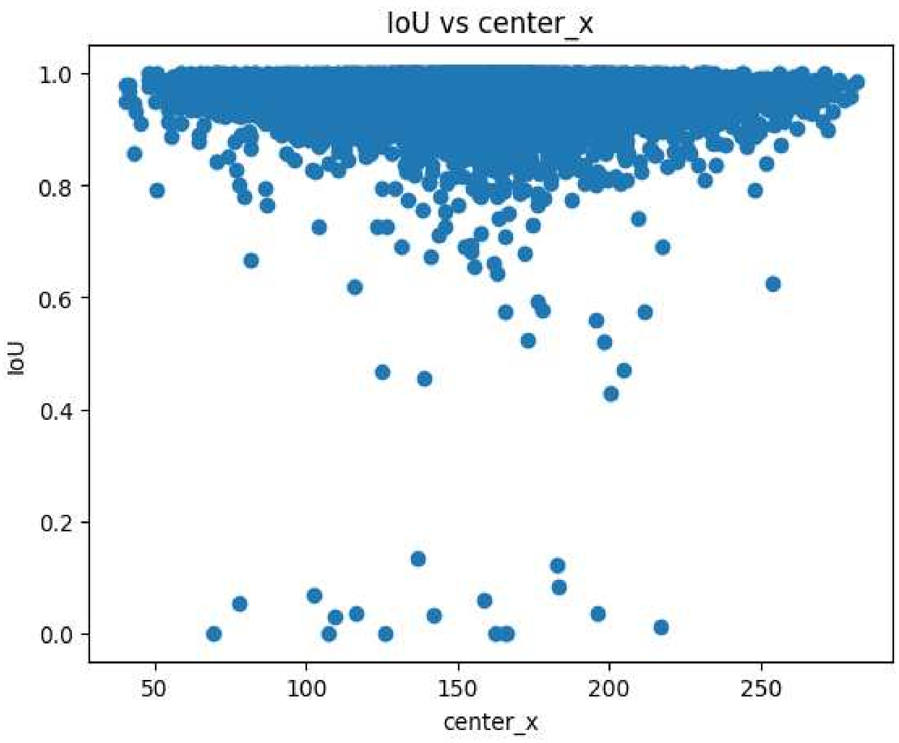


Figure 14: PhIM - Model performance (IoU metrics) dependency on input feature parameter (center position of object)

3.4.5. XAI usages in Inference Management

Inference management in AI-FSM ensures that deployed models operate within acceptable safe boundaries while maintaining consistency with their original training behaviour. Explainable AI (XAI) techniques play a key role in validating inference stability, monitoring deviations, and detecting Out-of-Distribution (OOD) inputs. By continuously assessing model predictions against expected behaviours, inference management ensures that decisions remain interpretable, reliable, and aligned with safety requirements.

A key requirement for inference management is to verify that the model performs as intended in real-world scenarios. XAI techniques help compare inference-time decisions with the original model's expected behaviour by:

- **Tracking feature importance shifts** between training and inference phases to ensure that the model relies on the same key features for decision-making.
- **Analysing activation patterns** in neural layers to detect any unexpected deviations that could indicate model drift or changes in learned representations.
- **Ensuring stability in explainability** metrics, such as attention maps and saliency visualizations, to confirm that inference-time reasoning remains consistent with training logic.

- **Comparing performance distributions** across different ODD conditions (e.g., different lighting conditions, object scales, viewing angles) to ensure inference robustness across varying environments.

Inference models must operate within predefined safe boundaries, ensuring that predictions remain trustworthy and interpretable. To achieve this, AI-FSM incorporates supervisory monitors, which act as a safety layer to cross-check predictions, detecting potential misclassifications or model failures before they impact decision-making:

- **Uncertainty estimation**, where models quantify their confidence levels in predictions, flagging high-uncertainty outputs for human review or fallback mechanisms.
- **Anomaly detection** for OOD inputs, which identifies when inference data significantly deviates from training data distributions. Techniques such as distance metrics in latent space, probabilistic modelling, and clustering-based detection help recognize unexpected scenarios.

To verify that the inference process remains transparent and trustworthy, AI-FSM applies various explainability techniques:

- **Feature attribution methods**, such as SHAP or LIME, analyse whether the model is using the correct input features for its predictions.
- **Prototype-based verification**, ensuring that the model's inferences align with known, representative samples rather than relying on spurious correlations.
- **Saliency maps**, which highlight the most important regions in input data that contribute to the model's decision-making. By visualizing which areas of an image, text, or structured data impact predictions, saliency maps provide insights into the reasoning process and help detect cases where models rely on unintended or misleading features. Figure 15 illustrates a saliency map generated for Railway UC.

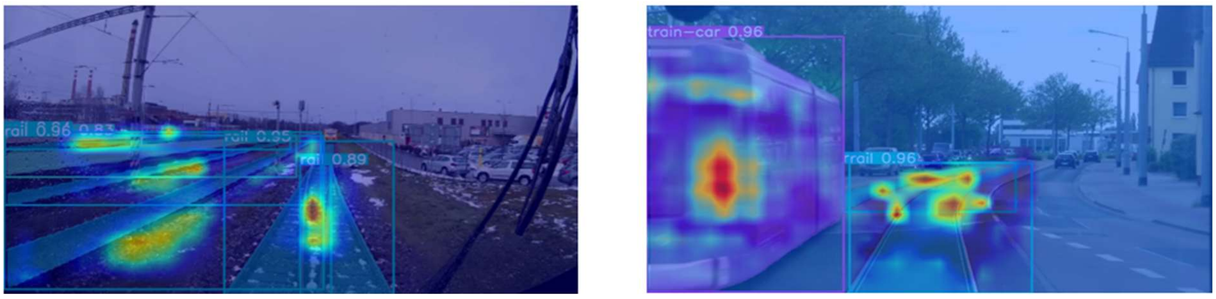


Figure 15: EigenCAM on railway use case.

3.5. XAI usages in Operation and Monitoring stage

Within the Operational and Monitoring (OM) stage, the residual uncertainties (irreducible uncertainties and residual reducible uncertainties) are managed by different safety components within the safety architecture. The components aim to identify and assess uncertainties, making sure that the system is operating safely as it has been designed for (upon the steps taken in AI-FSM lifecycle).

Table 2 provides a proposed mapping of XAI methods to the types of uncertainty encountered during the Operational Management (OM) stage. This mapping facilitates the integration of these XAI techniques as key safety components within the safety patterns detailed in subsequent sections.

Table 2: Uncertainty types and corresponding proposed supervisory monitors

Uncertainties	**Descriptions**	**Supervisory monitors**
Residual domain uncertainties	Out of Distribution: input data in runtime has not been seen (not likely belong to the datasets generated from AI-FSM PhDM)	• Anomaly detectors ○ Input data ○ Extracted feature ○ Output predictions
Residual model epistemic uncertainty	Low confidence of model predictions (or high uncertainty) due to non-representative training dataset or non-ideal model architecture	• Uncertainty-aware model • Surrogate model • Logical ruled constraints
Aleatoric uncertainty	Sensor noises, annotation uncertainties, occlusion Adversarial attacks	• Aleatoric uncertainty aware model • Adversarial detector, distribution shift detector

3.5.1. Reference safety architecture

The previous deliverable D3.1[19] has presented a high-level safety reference architecture for deployment in OM stage. This section provides a more detailed and updated final reference architecture where a minimum viable product (an MVP) version has been prototyped and implemented in the project's demo platform.

Figure 16 depicts the architectural diagram with the representative key components, to illustrate how the integration of these components constitutes a safe application of DL component in runtime environments.

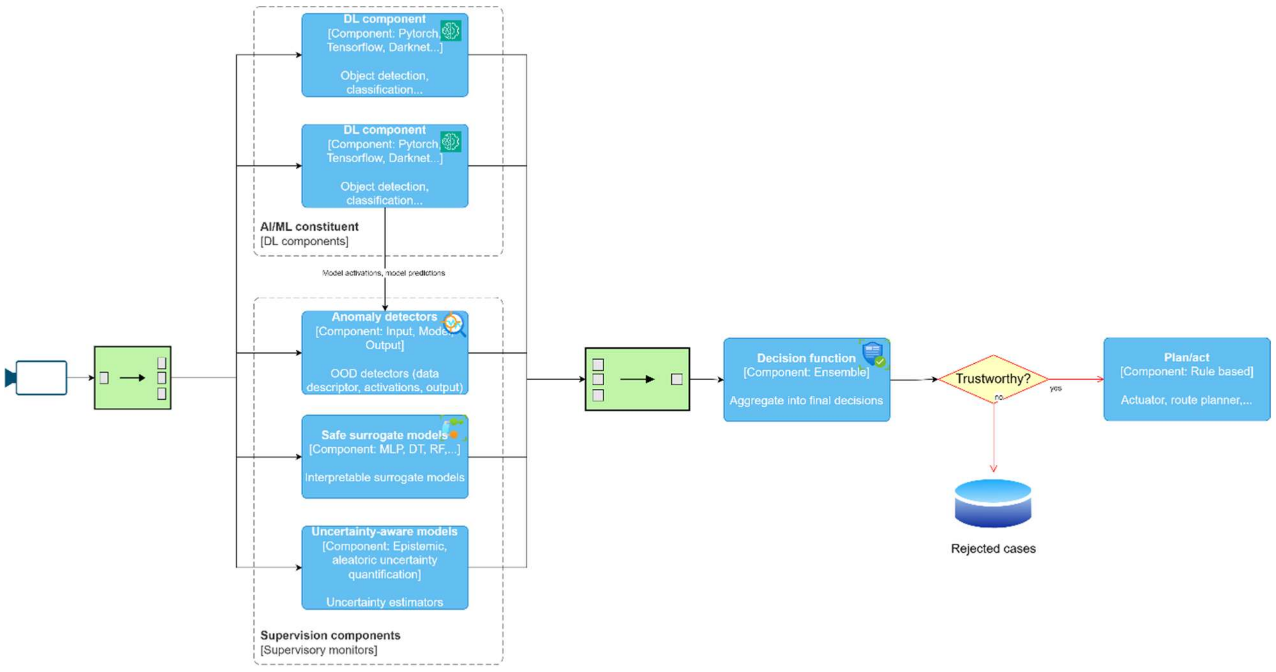

DL component
[Component: Pytorch, Tensorflow, Darknet...]
Object detection, classification...
DL component
[Component: Pytorch, Tensorflow, Darknet...]
Object detection, classification...
AI/ML constituent
[DL components]
Model activations, model predictions
Anomaly detectors
[Component: Input, Model, Output]
OOD detectors (data descriptor, activations, output)
Safe surrogate models
[Component: MLP, DT, RF,...]
Interpretable surrogate models
Uncertainty-aware models
[Component: Epistemic, aleatoric uncertainty quantification]
Uncertainty estimators
Supervision components
[Supervisory monitors]
Decision function
[Component: Ensemble]
Aggregate into final decisions
Trustworthy?
yes
no
Plan/act
[Component: Rule based]
Actuator, route planner,...
Rejected cases

Figure 16: Reference safety architecture for Operation and Monitoring stage

The reference architecture includes the following safety mechanisms:

- **AI/ML constituent**: A diverse set of redundant DL components, each designed to achieve the same objective, operate in parallel and generate related output predictions for a given input data point.
- **Supervision components**: We propose three types of supervision components, which practitioners can apply in various ways depending on their specific requirements. These component types include:
    - *Anomaly detectors*: The anomaly detectors are designed to identify deviations in the model's operating environment that may compromise its certified safe performance (within AI-FSM). These deviations can occur due to various factors, including:
        - Input anomalies: Unusual input data can cause the DL components to behave unexpectedly, often due to uncertainty about the data distribution (i.e., newly encountered data that differs from the training datasets). Examples of input anomalies include:
            - Data corruption (e.g., sensor noise, environmental interference, or adversarial attacks)
            - Missing or incomplete data, object occlusions
            - Changes in the sensor field of view or object distribution
        - Model internal anomalies: The model's internal behaviour may deviate from its expected and certified patterns, as logged during the AI-FSM PhLM phase. These anomalies can be:
            - Unusual neural activation patterns or feature extractions
            - Abnormal gradient computations (in reinforcement learning scenarios)
        - Output prediction anomalies: Unexpected predictions can occur in various aspects of the output, including:
            - Object type or classification
            - Distribution or probability estimates
            - Temporal consistency or coherence
            - Object size, shape, or position
    - *Safe surrogate models*: Interpretable surrogate models are trained to approximate the predictions of the main DL component. Their transparency allows domain and safety experts to verify and certify a safe, albeit less robust, alternative for critical operations.
    - *Uncertainty-aware models*: These models estimate the uncertainty of the main DL component for a given input/situation, enabling risk quantification for hazardous scenarios and informing safe decision-making.
- **Decision function**: This component aggregates predictions from diverse redundant DL components and supervisory monitors to generate a final output consisting of a consolidated prediction and a corresponding trustworthiness score.

3.5.2. Reference safety components

3.5.2.1. Diverse redundant AI/ML constituent

The AI/ML constituent consists of 2 DL components (SSDLite and Faster R-CNN, both are using MobileNetV3 as their CNN backbone to extract visual features) acting as object detectors, predicting the existence of the satellite object and its position within the input image frame (bounding box coordinates). The pre-processing steps are mainly converting input data into the required format (320x320 pixels), and the post processing step is formatting predictions into a triplet {object class, bounding box coordinates, detection score}.

3.5.2.2. Uncertainty-aware models

Although anticipated uncertainties have been mitigated throughout the various AI-FSM phases and steps, residual uncertainties may remain, resulting in need of real-time monitoring during operation to maintain an acceptable level of safety risk of encountering hazardous situations.

Uncertainty-aware models refer to a set of design approaches that aim to enhance the target DL component(s) with extra capability to quantify the uncertainties associated with in-distribution data, as well as potential out-of-distribution scenarios. These techniques provide real-time estimates of uncertainties alongside the model's predictions, thereby furnishing the decision-making module with the necessary additional information to make safe and informed decisions.

Typical methods that can be used to provide uncertainty estimates include:

- **Model epistemic uncertainty estimate**:
    - *Ensemble methods*: Using multiple models with different settings (training data subsets, model architectures, training hyperparameters). These can be combined with bootstrapping techniques, which create bootstrapped sub-datasets by resampling the original dataset.
    - *Monte Carlo dropout*: Applying random dropout to neuron activations creates a distribution over the model weights, capturing epistemic uncertainty.
    - *Bayesian Neural Networks (BNN)*: Adding a prior distribution over the model weights and updating it using Bayes' theorem. Variational inference can be used to approximate the posterior distribution over the model weights.
- **Aleatoric uncertainty estimate**: This type of uncertainty represents the irreducible uncertainties that are inherent to the stated problem itself, arising from the natural variability and noise present in the labelled ground truth data. A modification is applied to add an additional prediction head to estimate the parameters characterizing the output distribution, such as the mean and standard deviation, assuming a specific distribution (e.g. Gaussian or Gaussian mixtures)

For the MVP, two uncertainty-aware models have been developed as prototypes providing qualifications for these two types of uncertainties: Aleatoric uncertainty and Model epistemic uncertainty.

The following subsections will provide more details of the implementation details.

3.5.2.2.1. Aleatoric uncertainty aware model

The standard SSDLite model from the Torchvision model zoo has been modified to incorporate a custom prediction head, providing the estimated aleatoric uncertainty of the predictions.

The following assumptions have been made:

- **Safety related performance metrics**: The accuracy of predicted bounding box coordinates (e.g., object position and orientation) serves as a key performance metric related to safety.
- **Aleatoric uncertainty quantification definition**: Aleatoric uncertainties are quantified as the expected variation in predicted coordinates around their predicted values. This provides a measure of the uncertainty associated with each prediction.
- **Input dependent**: The uncertainties are dependent on the characteristics of the input image and are computed individually for each of the four bounding box coordinates (x1, y1, x2, y2).
- **Normal distribution**: The regression head, optimized using the L1 norm, predicts bounding box coordinates that form a normal distribution around the ground truth values. This assumption enables the estimation of aleatoric uncertainties using statistical methods.

Key modifications are as follows:

- **Uncertainty estimation head**: An additional head has been introduced, which inherits the architecture of the regression head. This uncertainty estimation head takes the backbone features corresponding to proposed anchor bounding boxes as input and generates uncertainty estimates for each anchor box as its output. Figure 17 illustrates the modified architecture of SSDLite model for this purpose.
- **Modified Loss Function**: The original loss function has been augmented with a Gaussian Negative Log Likelihood (NLL) term. This modification encourages the model to not only accurately predict bounding box coordinates but also to provide reliable uncertainty estimates that reflect the true distribution characteristics of the predictions. Readers consult [138] for a comprehensive overview of probabilistic models and concepts.

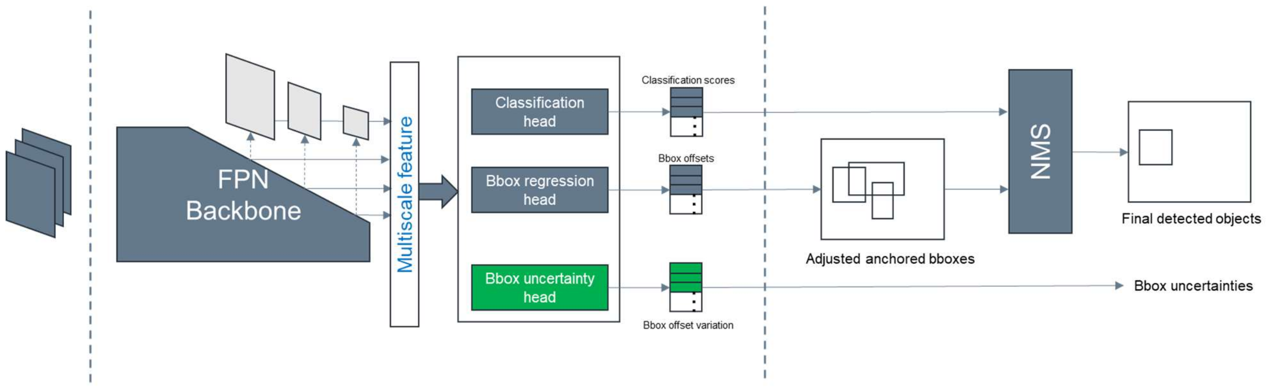


Figure 17: Modified SSDLite model to estimate aleatoric uncertainty

The modified model (namely Aleatoric uncertainty aware model) had been trained with the MVP dataset and will provide uncertainties together with the predictions.

The modified loss function is an weighted sum of classification loss (entropy loss), bounding box regression loss (L1 loss) and uncertainty loss (NLL), computed as follows:

$$L = \frac{1}{N}(\lambda_{bbox}L_{bbox} + \lambda_{cls}L_{cls} + \lambda_{unc}L_{unc}) \qquad (1)$$

Where NLL loss for uncertainty is provided as:

$$L_{unc} = \sum_{i=1}^{N}\left(\frac{(\widehat{x_i} - x_i)^2}{\sigma_i^2} + \log(\sigma_i^2)\right) \qquad (2)$$

Figure 18 visualizes the prediction results of the aleatoric uncertainty aware model, showing two bounding boxes corresponding to the lower and upper bounds at 95% confidence interval (CI95).

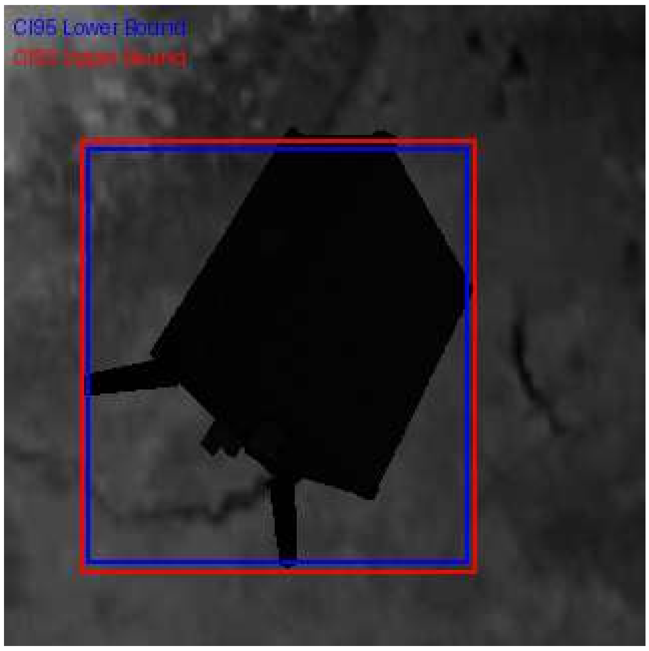


Figure 18: Aleatoric uncertainty aware model (modified SSDLite model)

3.5.2.2.2. Epistemic uncertainty aware model

We employed the Monte Carlo (MC) dropout technique to develop an epistemic uncertainty-aware model. This approach simulates multiple models by randomly dropping out units during inference and utilizes the variability in the resulting predictions as an estimate of the model's uncertainty.

The MC dropout is implemented as follows:

- **Modify model inference with dropout**: We ran the model inference multiple times while keeping the dropout layers active. During each iteration, a random subset of neurons was dropped out, and the output prediction was recorded.
- **Generating a distribution of predictions**: By repeating the model inference with dropout, we generated a distribution of output predictions for the detected objects' bounding boxes.

- **Estimating epistemic uncertainty**: We computed the uncertainty estimates of the generated distribution. The uncertainty can be quantified using metrics such as Mean and Standard Deviation or Entropy. For this specific prototype, the standard deviation and derived CI95 lower/upper bounds of bounding boxes are computed.

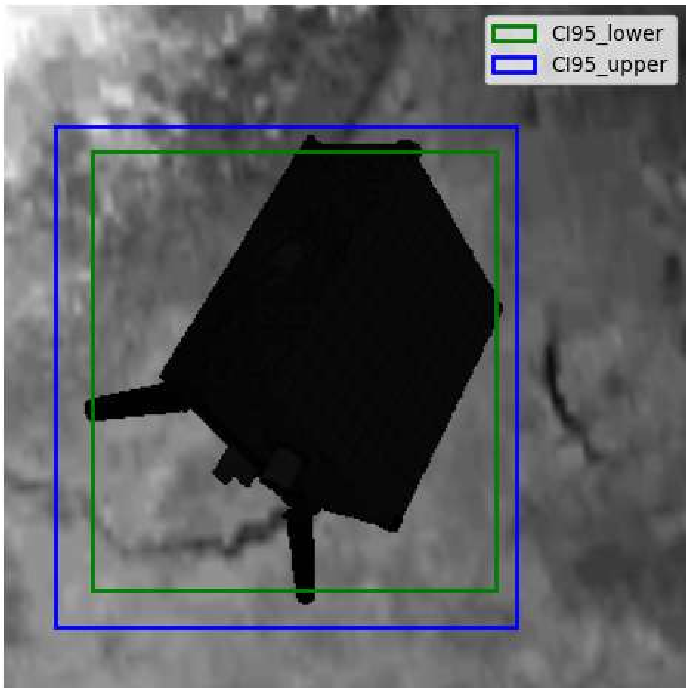


Figure 19: Epistemic uncertainty aware model (modified SSDLite model)

Figure 19 illustrates the output of the epistemic uncertainty aware model, displaying two bounding boxes that represent the lower and upper bounds of the 95% confidence interval (CI95). The true object bounding box is likely to locate in between these two bounds with 95% confidence.

The probability density functions (PDF) associated with the prediction distributions for each of the bounding box coordinates (x1, y1, x2, y2) are shown in Figure 20.

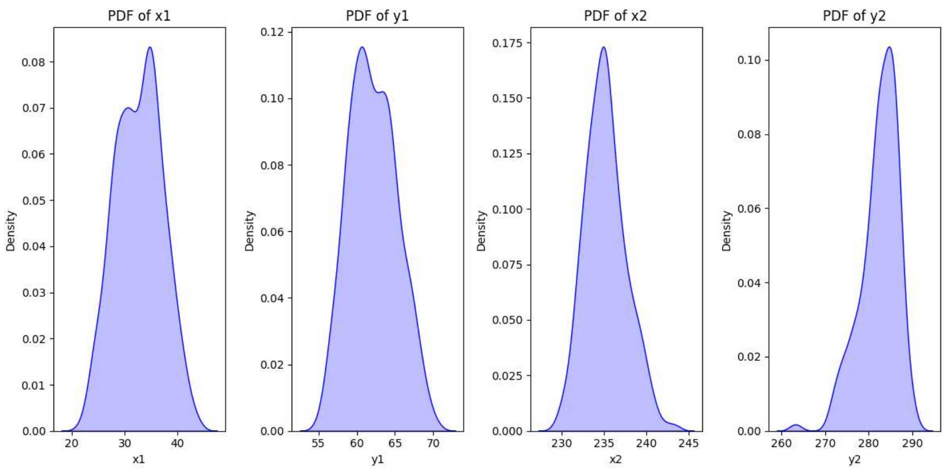


Figure 20: Toymodel epistemic uncertainty estimates PDF functions

3.5.2.3. Out Of Distribution Detector

Out-of-distribution (OOD) detectors or anomaly detectors are used identify data points that do not belong to the distribution of the datasets managed by AI-FSM Data Management. These detectors are required to ensure that the model safely operate within its certified safe boundaries and avoid risk getting into hazardous situations resulted from the epistemic type of uncertainties including domain uncertainties and part of model epistemic uncertainty.

Techniques that can be used as candidates for OOD detectors include:

- **Statistical methods**
    - *Distance based methods*: Compute distance between the input data point in runtime to the known/certified datasets. Examples include Mahalanobis distance or traditional statistical tests.
    - *Clustering methods*: These methods provide unsupervised clusters of data (e.g. KNN, LOF, SVM), thus can be used to detect anomalous input data as if it does not belong to any of the known clusters.
    - *Gaussian Mixture Models*: GMM can be used to model the distribution of the known dataset, and thus a data point that has low probability under this model can be considered anomalous.
    - *Kernel Density Estimation*: Non-parametric approach that estimates the underlying density of the known dataset, and thus data points with low density values are likely anomalous.
- **ML based methods**
    - *Auto Encoder or Variational Auto Encoder (VAE)*: These models can be trained to reconstruct a known dataset by first mapping the input data into a lower-dimensional representation (a latent space with a Gaussian or Euclidean structure) using an encoder network. The encoded data is then reconstructed back into the original input space using a decoder network. The reconstruction errors can be used as the anomaly scores, based on the assumption that the trained models learn to effectively represent the underlying structure of the valid data space. This allows the models to act as a descriptor of valid data points, where the anomaly scores are expected to follow a known distribution (often assumed to be Gaussian).
    - *GAN approaches*: GAN models typically consist of a generator and discriminator. The trained discriminator can be used as an anomaly detector, identifying data points that are unlikely to be generated by the generator.
    - *One-class classification models*: These models are trained to classify in distribution data points as belonging to a single class, thus can be used to identify data points that do not belong to this class and considered as anomalous data point. Examples of such classifiers include SVM and Isolation Forest.

Within the SAFEXPLAIN OM safety architecture, Out-of-Distribution (OOD) detectors are further categorized into three distinct groups based on the specific checkpoints at which they extract input data and evaluate anomalies. These groups are described as follows.

3.5.2.3.1. Input data anomaly detector

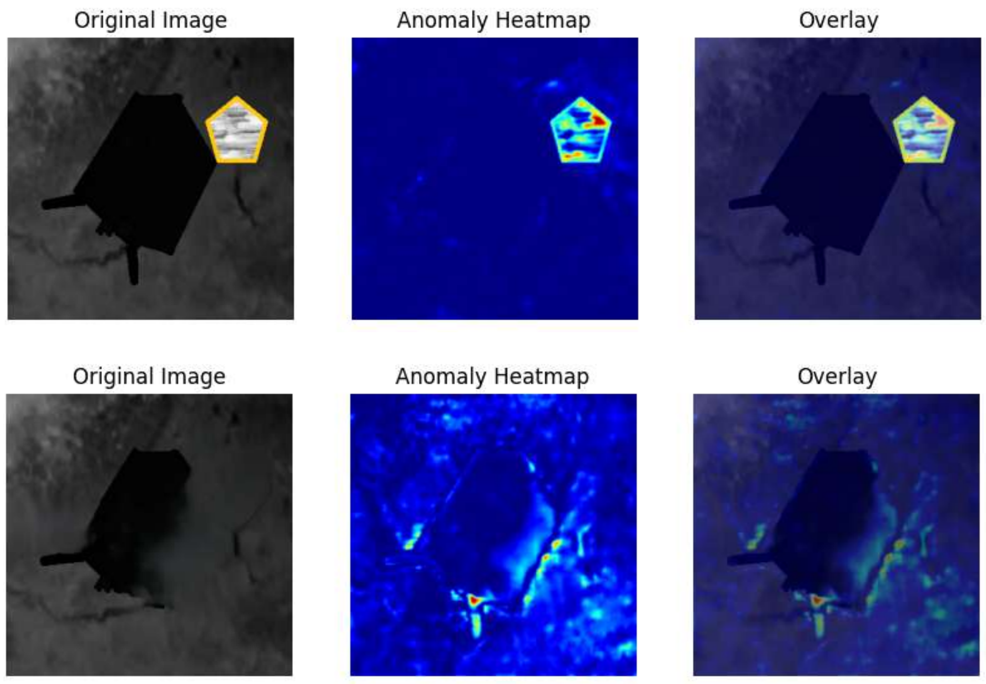


Figure 21: Anomalous region with VAE latent anomaly

To manage the domain uncertainty, input data anomaly detectors can be designed and deployed using different techniques:

- **Statistical methods**: Statistical techniques can be used to monitor various data/feature distributions and compare them against logged/expected distributions. This includes leveraging traditional statistical distance measures such as Mahalanobis distance or Kullback-Leibler (KL) divergence.
- **Data descriptor techniques**: Utilizing data descriptors (such as VAE based descriptor) to identify the anomalous input data based on reconstruction loss or out of distribution pattern in latent space (See figs)
- **Temporal consistency checks**: Methods to detect anomalies in temporal patterns such as Optical Flow[139]
- **Adversarial attack detectors**: Methods to detect adversarial attacks (adversarial noises added with intention to degrade the system performance)

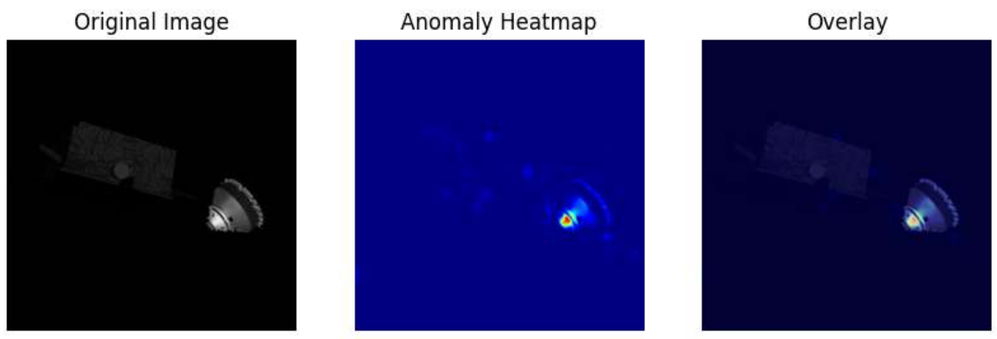


Figure 22: Anomaly detector highlighting anomalous region from the latent space anomaly (PCA 15 dims). ufo dataset rgb_0134.png

Figure 21 and Figure 22 illustrate how a trained VAE model can be used to detect anomalous data. In this example, the input data has been modified with an anomalous region (unknown concepts to the model). By analysing the latent space with PCA, the anomalies (PCA reconstruction errors) were projected back to the reconstruction space and can highlight the anomalous area.

3.5.2.3.2. Model anomaly detector

Model internal working anomalies can be considered as anomalies in the patterns of neuron activations. Similar techniques as input anomaly detectors can be applied here, considering the input data is now neuron activations at a selected key layer(s) of the model (recommended to use the key conv blocks). Key techniques are thus described as follows:

- **Internal feature anomaly detector**: Anomaly detectors working with input from the model extracted internal features, represented by the extracted activation patterns at the model's backbone. To this end, similar methods as of those presented in input anomaly detectors can be used including statistical methods and data descriptor-based methods.
- **CAM-based pattern anomaly detector**: Methods such as GradCAM/eigenCAM can be used to generate heatmaps representing how the DL component focus on different spatial regions of the input. Several metrics can be computed from the heatmaps for anomaly assessments:
    - Sum of heatmap values within object bounding boxes
    - Distance of different CAM heatmaps generated by different CAM methods
    - Other metrics such as heatmap variance or entropy

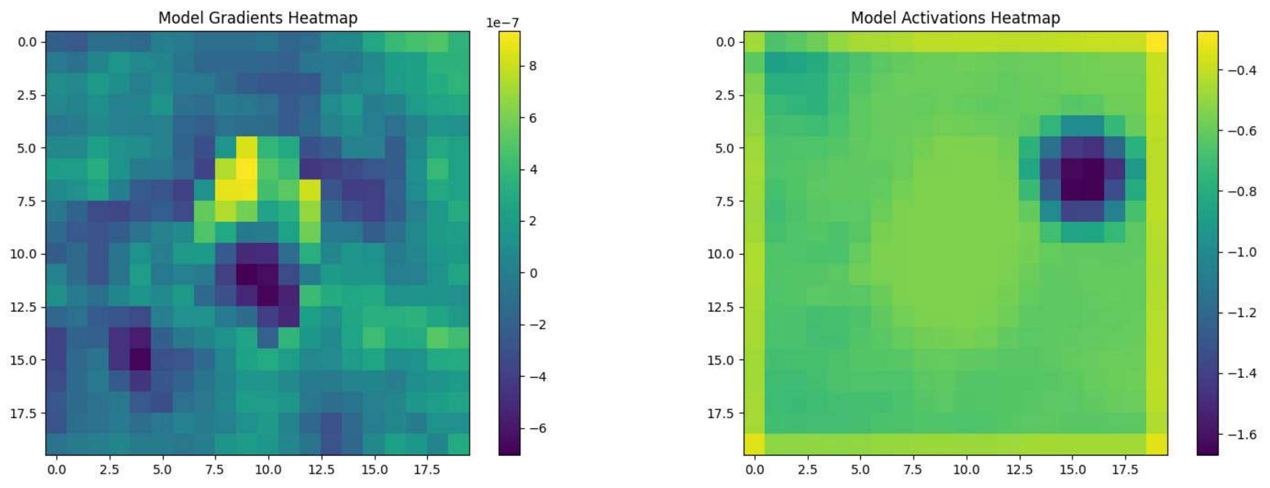


Figure 23: Extracted model neuron activation patterns (right) and model gradient map (left), used as input for anomaly detector

Figure 23 visualizes example inputs of model's activation patterns corresponding to a specific input image in the MVP dataset. The VAE-based anomaly detector has been trained using this as its new input types and thus can detect the anomalous neuron activations during runtime.

3.5.2.3.3. Output anomaly detection

The predictions of model themselves form a new data space, with expected characteristics that have been logged during AI-FSM lifecycle. These characteristics can be used as a baseline for output anomaly detection. Techniques can be used including:

- **Statistical methods**: Using predefined metrics as output data dimensions and analyse the anomalous patterns with that space. The dimensions are
    - Object size: width, height
    - Object distribution within a scene

- Object distribution over time, measured by IoU of the same detected object bounding boxes across consecutive frames
- Object shape: bounding box ratios width/height
- Object position within a scene

- **Distribution** of the model prediction confidence scores **over time**

3.5.2.3.4. Adversarial attack detector

Adversarial attacks are of intention, it tends to minimize the difference in altered input comparing to the original input, while maximizing the impact of failing the model detection.

The detector of such attacks thus can be designed, for example with the following approaches:

- Use the reconstructed data from the input anomaly detector as a new input to feed into the DL model and verify the differences of the model predictions comparing to incoming input
- Measure the anomalous neuron activations (anomalous values of neuron activations that have not been observed during AI-FSM lifecycle). Example of monitoring unusual pattern of logit values (as suggested by [140]) is provided in Figure 24.
- Train a classification model to classify adversarial data and original data.

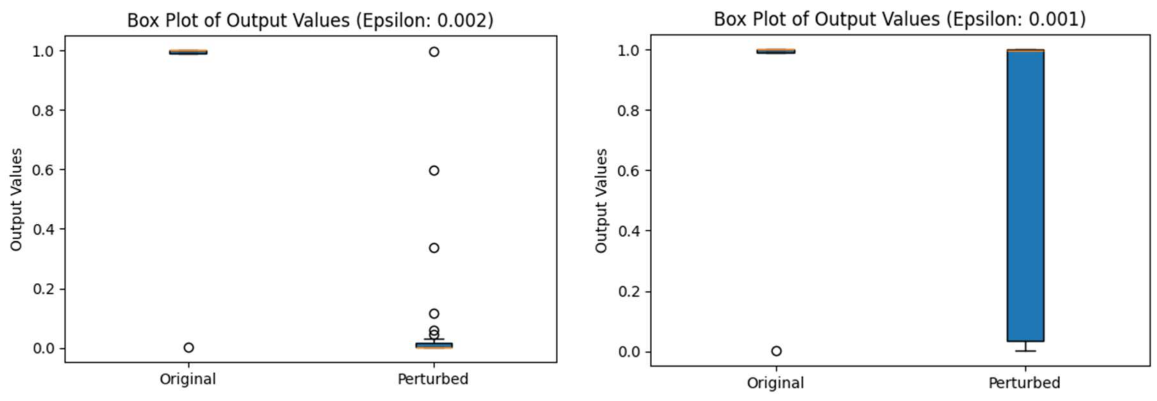


Figure 24: Box plots of logit values over all 4 data examples displayed in Figure 6.

3.5.2.4. Surrogate models

Safe surrogate models are simplified, interpretable models that learn to mimic the behaviour of the main DL models by being trained on the same input data (the MVP dataset) and the main model corresponding predictions. Although the surrogate model's predictions may not be as accurate, its internal logic is transparent and explainable, enabling safety experts to certify it within the AI-FSM development lifecycle. The predictions generated by the surrogate model are considered a safe rule-based, where the rules have been pre-certified by experts within the AI-FSM lifecycle.

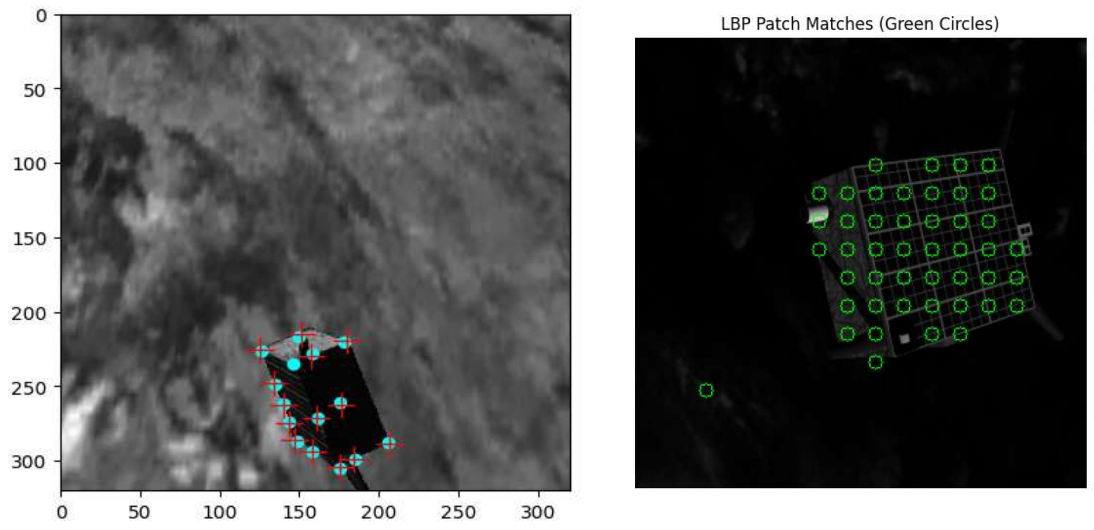


Figure 26: Extracted input features (keypoints and LBP pattern) to serve as input for MVP surrogate model

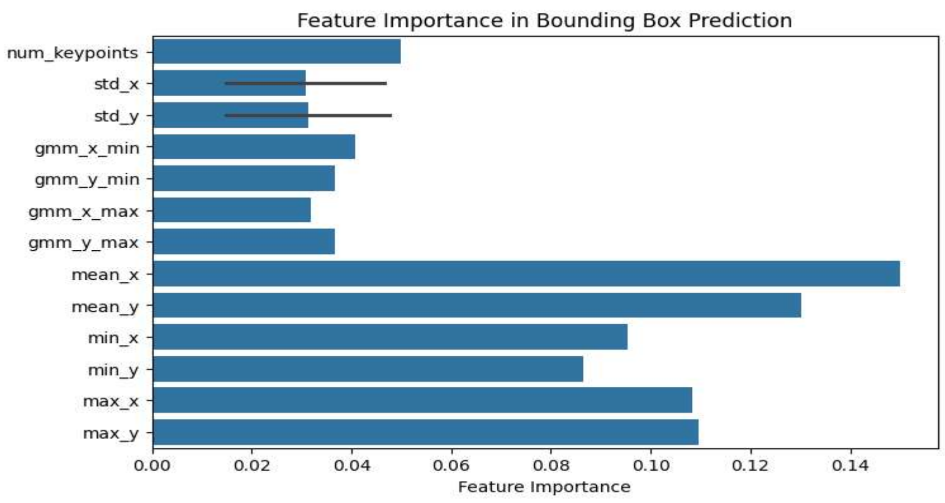


Figure 25: Statistical description of the set of key points and their importance on surrogate model prediction

When the main model's predictions diverge from those of the safe surrogate model, it indicates that the main model may be relying on unverified internal mechanisms to produce its outputs, compromising the required level of trustworthiness. During operation, if the discrepancy between the two models' predictions exceeds a predetermined acceptable limit, the main model's prediction will be rejected for use in safety-critical decisions. These rejected data points will then be considered in the next development cycle to potentially improve the system, and a new corresponding safe surrogate model will be derived, incorporating additional logical rules that can accommodate the new data or situations.

This iterative approach enables the safe surrogate models to evolve alongside the main model throughout multiple development lifecycles while maintaining the required functional safety standards.

The MVP surrogate model has been developed as follows:

- **Compute traditional image features**: Traditional image features including key points and positions of prototype patches representing the LBP texture map (see Figure 26) were computed.
- **Fixed-Length Vector Representation**: Aggregated key point and matched prototype pattern features into a fixed-length vector composed of statistical descriptors. The specific statistical figures used to construct this vector, and their relative importance are illustrated in in Figure 25.
- **Surrogate Model**: A Random Forest (RF) model was trained to predict object bounding box coordinates from the input feature vectors. Feature importance is visualized in Figure 25.
- **Usage in OM**: Use the trained surrogate model to predict bounding box from input image in OM, with the first step is to prepare the feature vector. Figure 27 shows how the surrogate model detects the object bounding box from the computed features and also visualize extracted key points.

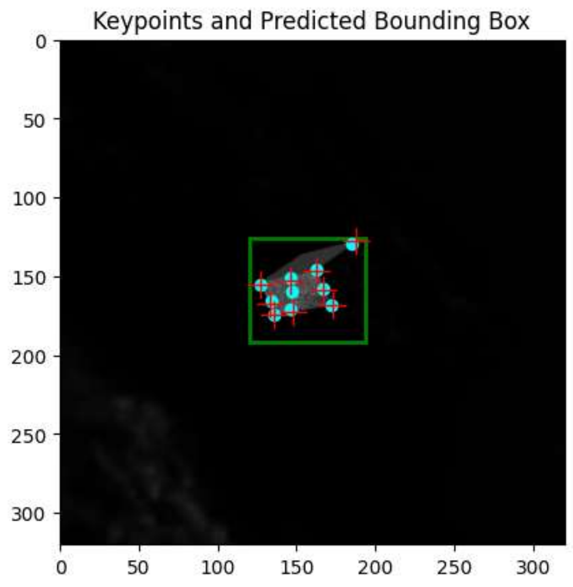


Figure 27: Surrogate model prediction

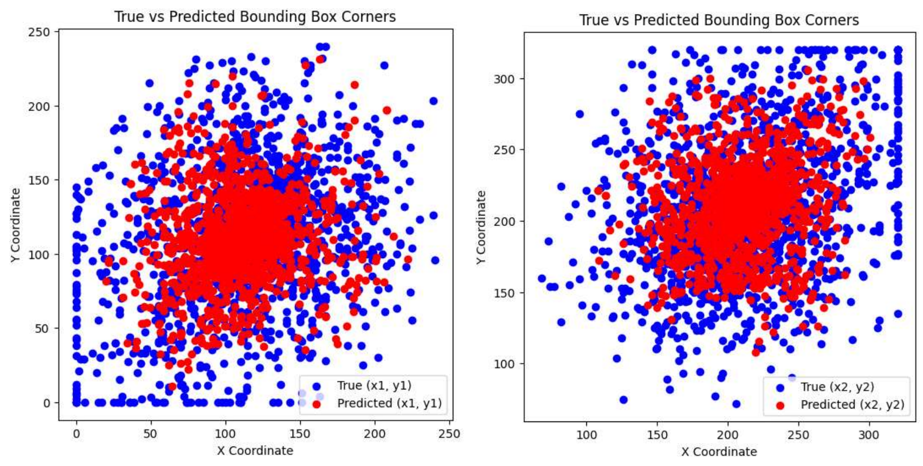


Figure 28: Groundtruth vs predicted bbox coordinates

Figure 28 illustrate the surrogate model accuracy (predictions vs ground truth)

3.5.2.5. Decision function

The decision function aggregates outputs from both diverse DL components and other components within the supervision component container, including anomaly detectors, surrogate models and uncertainty aware models. It generates a consolidated prediction, together with trustworthiness score and relevant explanations which are provided in the event of a rejection.

3.5.2.5.1. Combining anomaly scores into trustworthiness score using z-scores

To combine a set of anomaly scores (provided by anomaly detectors within the OM architecture), we assume that these scores follow normal distributions. We then use z-scores to normalize the anomaly scores of different types as the foundation for combination. The z-scores is calculated as follows:

$$z = \frac{x - \mu}{\sigma} \tag{3}$$

Where x is the value of an anomaly score of a specific type (input, model or output), μ and σ are the mean and standard deviation of such anomaly scores computed from the MVP dataset respectively.

Z-scores within the range [-r, r] indicate how likely a new anomaly score can be accepted as falling within a certain population coverage percentage of the corresponding anomaly scores computed for the known situations, i.e. MVP dataset. For example, a z-scores range of [-2,2] represents a 95% confidence level that the computed anomaly score for a specific input is within the known value ranges.

The input for z-scores for the MVP are derived from two main sources:

- **Anomaly detectors**: Input anomaly score, Model anomaly score, Output anomaly score
- **Predictions**: bounding box area and bounding box ratio

3.5.2.5.2. Combining detections from diverse DL models with interpretable model

Diverse redundant DL components will provide corresponding set of predictions (a set of objects bounding boxes as exemplified in the MVP). To combine these predictions into a single consolidated prediction, an interpretable model can be used. The interpretable model will be trained to predict the ground truth bounding boxes from the predictions and related confidence scores from a set of DL models. For MVP, we adopted supervised regression model XGBoost[141] for this purpose. XGBoost (eXtrem Gradient Boosting) is a scalable tree boosting framework, known for its speed and performance, thus is a suitable candidate for deployment component.

A XGBoost model has been trained taken input as a tuple of bounding boxes from the parallel models and corresponding scores to provide prediction as consolidated bounding box. Figure 29 visualizes the prediction results from the two MVP DL models (SSDLite and its backup model FasterRCNN) together with the consolidated prediction provided by the ensemble method.

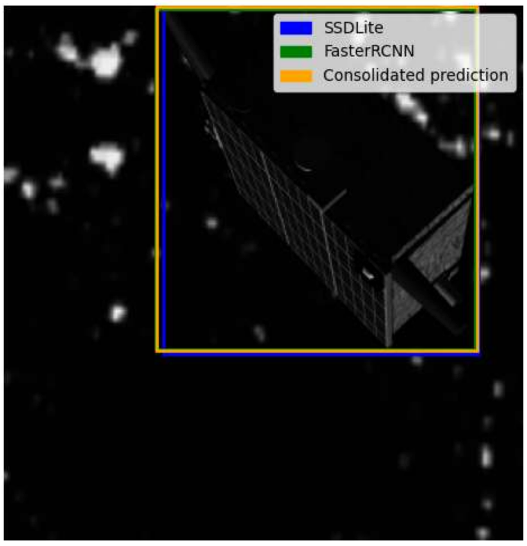


Figure 29: Consolidated prediction by XGBoost

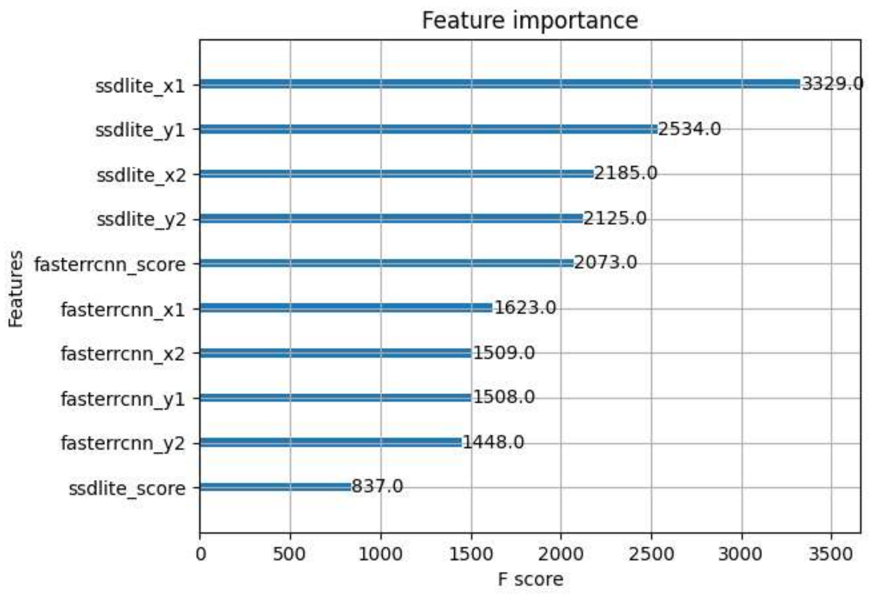
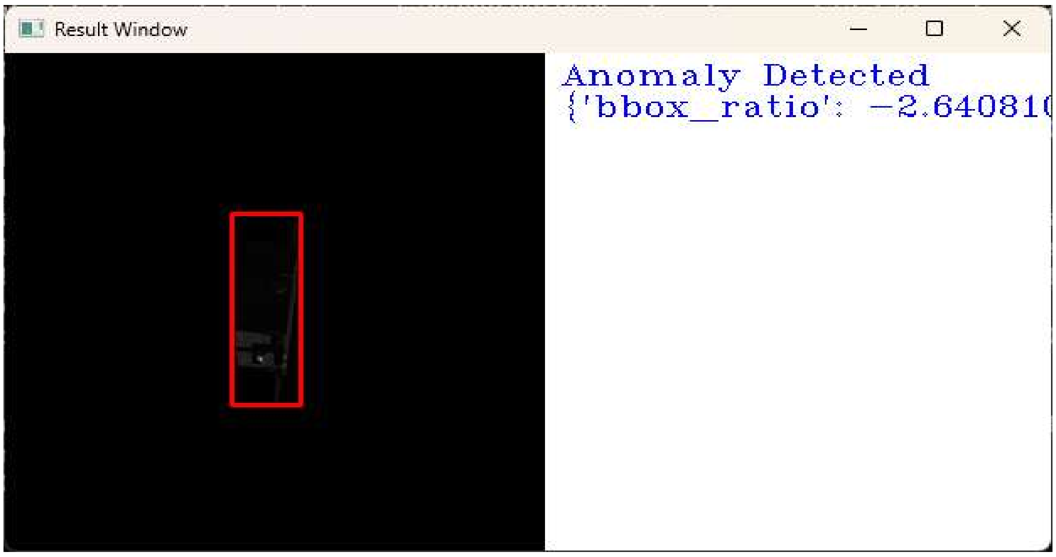


Figure 30: Importance of input parameters contributing to the final consolidated predictions of trained XGBoost model

Figure 30 illustrates feature importance of how different input parameters influence the ensemble model consolidated output.

Figure 31 illustrates an example of a detected anomaly, with the explanation provided by the ensemble with z-test results. The detected bounding box ratio is considered unseen.



```
Anomaly Detected
{'bbox_ratio': -2.64081
```

Figure 31: Example output of decision function

3.6. Relevant metrics

Different sets of metrics are employed at various stages of the development lifecycle to ensure multiple perspectives are considered. This section outlines the key sets of metrics used to evaluate the following aspects: explainability, traceability, safety, robustness, failure rate, and ease of implementation, as defined in the success criteria (WP1).

3.6.1. Explainability

Several metrics have been identified for evaluating explainability quality for machine learning including neural networks (and deep learning) models. Such metrics include: (i) *fidelity* - how well the explanation aligns with the underlying model (e.g.[142]), (ii) *stability* - whether similar inputs yield consistent explanations (e.g.[99]), (iii) *faithfulness* - how accurately the explanation reflects the true behaviour of the model, (iv) *monotonicity* - whether more of a certain feature lead to a stronger explanation, and (v) *complexity* - how easily the explanation can be understood, e.g. number of features and rules used in explanation. A subset of these approaches is discussed below and with reference to our own modelling.

Model fidelity metrics are important when evaluating surrogate interpretable models. Surrogate models, as simpler models that often have intrinsic explainability built in, e.g. for decision trees, enhance explainability of the black box deep learning model whose inner workings are required to be explained. In order for those explanations to be trusted, metrics for comparing surrogates to the original model are needed. Simple metrics have been provided by [143] for classification and regression problems. For classification, the metric is defined as follows:

$$F_{\text{classification}} = \frac{1}{n}\sum_{i=1}^{n} I\left(y_{\text{pred},i} = y_{\text{surrogate},i}\right) \tag{4}$$

which provides a measure for the number of matching classifications of the two models the evaluation of this metric being subject to domain experts (i.e. what is considered an acceptable degree of matching or not). $F_{\text{classification}}$ provides simply the proportion of instances (images in our case) that are correctly classified. For object detection problems it does not, however, take into account the number of false positives or false negatives being calculated. Furthermore, with respect to the MVP image dataset used to train our MVP model where only a single object of the same class is present, an adaptation of the above could concern the extent to which the object confidence score for the located object is correct for the surrogate with respect to the benchmark model. The simplest form of this would label the model or surrogate classification as '1' if the confidence is above a pre-specified threshold.

For regression problems, relevant, for example, when evaluating bounding box accuracy, the following equation ([143]) can be used for calculating the normalized relative error:

$$F_{\text{regression}} = 1 - \frac{1}{n}\sum_{i=1}^{n} \frac{|y_{\text{pred},i} - y_{\text{surrogate},i}|}{\max(|y_{\text{pred},i}|, |y_{\text{surrogate},i}|)} \tag{5}$$

where in this case *n* might be considered the number of bounding box coordinates for a given detected/located objected. The closer the score is to 1, the higher the fidelity. Alternatively, *n* can be considered the number of data samples, and an additional internal loop can be applied to compare coordinates of the bounding boxes.

Model stability concerns the extent to which **features** are consistently applied across different samples rather than just comparing performance. Metrics here can also be considered with respect to surrogate models, i.e. surrogate feature stability [99]. An example metric is provided here:

$$F_S = \frac{1}{k}\sum_{i=1}^{k} \frac{|F_{\text{orig}} \cap F_{\text{resampled},i}|}{|F_{\text{orig}} \cup F_{\text{resampled},i}|} \tag{6}$$

where F_{orig} represents features taken from the original model and $F_{\text{resampled}}$ represents features from the surrogate model. K could concern the number of features (local) or number of samples for a particular feature (global). A score close to 1 indicates higher stability for the surrogate model's feature selection.

Feature consistency between models, which could be surrogates or potentially quantized version of original more complex models (e.g. optimized inference model in PhIM) can also be assessed through feature-by-feature comparisons. A way to quantify differences in explanations (featural representations) can be to compare feature by feature differences in feature importance (e.g. Shap) /relevance (e.g. LRP) using simple distance-based metrics, e.g. Euclidean or through cosine distance ([144]).

$$\text{ShapGAP}(D, d) = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} d\big(S_{bb}(x_i), S_{wb}(x_i)\big) \tag{7}$$

The referred technique ShapGAP can in principle be applied not just to SHAP feature importance distances but to other feature importance or relevance algorithms potentially. Given that the models to be compared are architecturally the same, it may also be possible to calculate distances with respect to targeted layers of feature maps (where convolutional layers are used).

Naturally, what a feature consists of is important. Single pixels in an image are unreliable determinants of features. For input data as an image, SHAP practitioners utilise "super pixels" ([145]), which provide more semantically meaningful features with which to compute distances over. Explanatory consistency provides an important measure of model similarity and potential to generalise across new data instances that the model has not been trained on.

This approach can be used to compare, for example, how well the surrogate model provides consistent explanations with respect to the fuller DL model.

Model complexity can be measured by the Jensen-Shannon Divergence (JSD), which measures the extent to which the sum of feature distributions diverges from a uniform distribution (where all features, e.g. in the input, are equally important):

$$S_D(F) = \sum_{j=1}^{F} \frac{1}{2} D_{KL}(P_j \parallel M_j) + \frac{1}{2} D_{KL}(U \parallel M_j) \tag{8}$$

where P represents the normalized feature importance distribution, U is the uniform distribution, and M is the average of P and U, given by M = 1/2 (P + U). The JSD is symmetric, allowing it to capture both the deviation of P from U and U from P. This symmetry is achieved through the introduction of M, which enables the comparison of both directions of divergence. The S_D value ranges from 0 to 1, with lower values indicating that the feature distribution is closer to uniform, implying lower model complexity. The use of JSD provides a more nuanced measure of model complexity compared to simply comparing the KL divergence of feature distributions.

Above is a global based approach, **rank consistency**, based on SHAP, provides a global approach based on consistency of local feature importance ranking (to predictions). Features can be ordered by consistency across data instances as a means to ascertain reliable explanations of model predictions.

3.6.2. Traceability

Traceability is an important aspect to ensure the reliability and maintainability of complex systems. It can be measured from two distinct perspectives: process-based traceability and system-based traceability:

3.6.2.1. Process-based traceability

This refers to the ability to establish and maintain transparent relationships between various input/output artifacts throughout the AI-FSM development process. Effective traceability involves creating a comprehensive pair of connections between key artifacts, including:

- **Safety goals:** The overarching objectives that ensure the system's safe operation.
- **System safety requirements**: Allocated system requirements to achieve the safety goals within the defined scope (ODD, operational scenarios and intended functionalities)
- **Data requirements**: requirements allocated to datasets, including those properties as recommended by AI-FSM PhDM
- **Model requirements**: requirements allocated to DL model, including performance and robustness
- **Model designs**: Including arguments of why specific model architecture and parameters are chosen
- **V&V results (of Data and Model)**: The verification and validation activity outcomes for different phases of AI-FSM. This includes XAI generated evidence to support safety arguments.
- **Implementation codes**: The final DL software implementations, including development codes, deployment codes, and configurations

The traceability metric can then be defined as a measure of the percentage of artifacts that have established relationships with other relevant artifacts throughout the AI-FSM lifecycle.

Denote the metric as **Artifact Traceability Coverage** (ATC), we can provide the following computation:

$$ATC = \sum_{i \in \mathcal{A}} w_i \cdot \frac{|R_i|}{|P_i|} \tag{9}$$

Where $\mathcal{A}$ denotes the set of all artifact types, R_i is the set of established relationships for artifact type *i*, and P_i is the set of possible relationships for artifact type *i*. The weight w_i is defined for each artifact type upon its importance within AI-FSM process.

Specific traceability coverage metric can also be defined e.g. for Safety Argument Coverage: *SAC = (Number of safety arguments with sufficient evidence / Total number of safety arguments)*

Change Impact metric: To measure the impact of changes on the DL development project following AI-FSM, we define the following metrics:

- Change Impact Ratio (CIR): Number of affected artifacts / Total number of artifacts
- Change Impact Index (CII): (Number of affected steps / Total number of steps) × (Number of affected artifacts / Total number of artifacts)

3.6.2.2. System-based traceability

For the **system-based traceability**, the following metrics can be defined:

- **System Traceability Coverage Ratio**: Number of tracing steps that have access to supported XAI explanations/Total number of tracing steps needed for diagnosis.
- **Root Cause Identification Rate**: Number of errors with identified root causes/Total number of errors.

For instance, to diagnose a specific case of malfunctioning Object Detector model, the following steps may be necessary to trace from the misdetections back to the captured data:

- **Verify neuron activations**: Confirm that the activations at different layers of the neural network are as expected.
- **Identify influential input features**: Determine which specific input features or regions in the input image are driving the detection results, including any potential biases or anomalies.
- **Assess sensor/data quality**: Verify that the sensors have correctly captured the relevant features and/or regions that influences the detection, taking into account data confidence levels, noise, and other sources of uncertainty.

3.6.3. Safety

Safety related metrics are metrics that are derived from the safety goals and allocated into different requirement specifications at relevant AI-FSM phases:

- **Safety metrics related to data requirements**: Data quality metrics (as mentioned in D3.1) with focus on how well the datasets represent the ODD, operational scenarios, intended functionalities and safety goals. Metrics regarding availability of data to support V&V strategies are also of importance. Note that the V&V strategy employed by SAFEXPLAIN is scenario-based, which requires flexibility in selecting iterative sets of scenario parameters and corresponding data based on the results of metaheuristic searches.
- **Safety metrics related to model**:
  - *Failure detection rates (FDR)* can be evaluated with respect to the Operation and Monitoring architecture, i.e. percentage of safety-critical failures (in relation to model predictions that lead to safety-critical outcomes) with respect to all failures (false positive or false negatives.
  - *False alarm rates*: measure of unnecessary interventions/rejections of predictions not just as false positives but in relation to their safety criticality.
  - *Mitigation success ratio*: number of false predictions that are mitigated through appropriate action, e.g. cleaning data that is identified as being subject to adversarial attack.
  - *Adversarial attack detection*: See 3.5.4
  - *Diagnostics*: Percentage of correct attribution of cause to the identified rejected model prediction, e.g. does the low confidence in the model's prediction owe to the data being OOD or alternatively subject to an adversarial attack, which type of attack?
  - *Conformance to safety* standards: what is the % of compliance checks passed in relation standards such as SOTIF?

3.6.4. Robustness

We define robustness within the context of SAFEXPAIN as the ability to maintain the DL component expected performance (relevant to safety goals) across different areas within its scope.

The robustness measures are thus further broken down into:

- **Dataset robustness**: Deep Learning model performance variations with respect to the 3 sub-datasets (as defined in PhDM).
- **Robustness against noise/corrupted/perturbated input data** (expected levels of acceptable noise or percentage of corrupted data).
- **Robustness against adversarial noise**: Adversarial noise levels vs DL model performance degradation.
- **Robustness against Out of Distribution data (OOD)**: Metrics measure the performance of the Anomaly detectors being deployed within the OM stage.
- **Uncertainty robustness**: Distribution of estimated DL uncertainties across dimensions of the dataset.
- **Robustness against training dataset distribution changes**: Measure the performance differences of the DL component if new data points are added, or some data points are removed to/from the training dataset (examples include add new datapoints).
- **Training robustness**: Measure the convergence rate of training process w.r.t. training hyperparameters (within a predefined ranges).

Detailed metrics are discussed in the following subsections

3.6.4.1. Dataset robustness

In the PhDM phase of AI-FSM, datasets are split into training, validation, and verification datasets. Robustness against datasets reflects consistency of model achieved performance across these datasets, and can be measured by the following metrics:

- **Performance drops across datasets**: $P_{max} - P_{min}$. In most cases, the max performance is achieved with training dataset and the min performance is achieved with verification dataset.
- **Performance standard deviation**: $\sqrt{\frac{1}{3}\sum_i (P_i - \bar{P})^2}$
- **Robustness ratio**: $\frac{P_{validation}+P_{verification}}{P_{train}}$

Where P denotes average performance of the DL model w.r.t. a dataset

3.6.4.2. Robustness against noise

Model robustness against noise, corrupted or perturbated data measuring model's ability to maintain its expected performance when trained, evaluated or operated on data that subject to a certain level of inaccuracy resulting from random noises, missing values or corruptions. Examples include sensor noises, partial occlusions, environmental noises, sensor contamination, label noise.

Metrics to measure this type of robustness include:

- **Performance drop rate**: calculates the difference in model performance when operating on clean data versus data that has been subjected to common and expected types of corruptions.
- **Noise to signal ratio (NSR)**: By introducing different levels of noise into the input data, NSR measures the subsequent variation in model performance, thus highlights model sensitivity to noisy conditions.

- **Label noise robustness**: assesses the impact of noisy labelling on model performance by comparing the difference between training on clean data and data with intentionally introduced label noise, as exemplified in [146]
- **Alignment between confidence and performance**: evaluates how well the model's confidence scores align with its performance when faced with noisy data, ensuring that the model is not overconfident or underconfident in its predictions.
- **Uncertainty rate**: compares the model's uncertainty when processing noisy input data versus clean input data.
- **Performance over noise levels**: measures model performance across a spectrum of noise levels.

3.6.4.3. Adversarial robustness

Robustness against adversarial noise refers to evaluation of how well the model can maintain its performance under adversarial types of input data perturbations. Adversarial attacks generally focus on minimal perturbations that can mislead the model (degrade its performance to certain level).

Adversarial noise can be considered a special type of anomaly, which may or may not lead detectable OOD categorization depending on the sophistication of the attack. Thereby a means to promote defence against adversarial attacks is to provide a subset of training data that is perturbed according to a set of well-used/known attacks enabling a higher degree of model/feature robustness to such attacks in an adversarially trained model.

Metrics for robustness against adversarial attacks, when evaluated during development, i.e. in AI-FSM PhLM, typically take into account the proportion of failed attack attempts, relative to total numbers of data points perturbed by adversarial attacks (e.g. FGSM, PGD, BIM). This is compared against the proportion of clean data points that are correctly handled by the DL model with required level of performance (e.g. [147]). In this way relative decreases in model performance using adversarial data can be evaluated following training, i.e. in the PhIM phase of AI-FSM. Relative improvements to performance in PhIM, however, can be compared when evaluating a model trained using perturbed data and a model not trained with perturbed data on a dataset that contains data with adversarial perturbations.

Another metric of adversarial robustness can consider model performance as calculated above, but with respect to degree of perturbations. In Section 3.4.2.1 it was shown the effects on model performance fgsm with respect to different levels of perturbation (epsilon values). Robustness to adversarial attacks should thereby provide a metric that considers:

- Performance of (adversarially) trained model on new dataset that contains perturbed data with respect to performance of non-adversarially trained model on the same dataset.
- Average performance of the adversarially trained model with respect to different levels of perturbations.

For object detection models as in MVP, Bounding Box Perturbation Errors (BBPE) can be used, i.e. measuring the L2 distance between each bounding box on specific images before and after the adversarial perturbation. Again, examination of whether training the model on adversarial data can reduce such BBPE can provide a means for assessing robustness to adversarial attack of the trained model.

3.6.4.4. Robustness against Out of Distribution data

Handling Out of Distribution (OOD) data is done by the supervisory monitors in OM stage. The robustness against OOD data is thus related to the performance of the anomaly detectors and measured by basic metrics: false rejections (false negatives) and false acceptance (false positives). More complex metrics can be derived from these metrics as follows[148]:

- **Area Under the Receiver Operating Characteristics curve (AUROC)**: the area under the plot of true positive rate (TPR) against false positive rate (FPR) at different anomaly score threshold settings.

- **Area Under the PR Curve (AUPRC)**: Similar to AUROC but measures the area under Precision-Recall graph. This metric is better for imbalance data (valid data vs anomalous data)
- **True Positive Rate at 5% False Positive Rate (TPR05)**: provides a specific measure of the anomaly detector accuracy in rejection while limiting false acceptance.
- **Precision at 95% Recall (P95)**: measures the detector’s ability to accurately identify anomalies while maintaining a high recall rate.
- **False Negative Rate at 95% False Positive Rate (FNR95)**: assesses the detector's tendency to misclassify outliers as inliers, highlighting potential over-confidence in its predictions.
- **Coverage Breakpoint at Performance Level (CBPL)**: how restrictive the threshold must be to return to the original accuracy that was received based on test set.
- **Coverage Breakpoint at Full Anomaly Detection (CBFAD)**: 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.

OOD detectors can be susceptible to adversarial attacks whereby the perturbed data may remain in distribution and therefore not be detected ([149],[150]). Robustness to such adversarial is discussed below in relation to metrics to assess such robustness.

We adopted an approach for using a trained VAE for anomaly detection in the OM stage (see Section 3.5). Where it is possible to compare model prediction performance on dataset distributions, i.e. the one on which the VAE model was originally trained, and that which is used either in the PhIM phase of AI-FSM or in deployment (Operation and Monitoring stage), standard techniques can be used to evaluate dimensions of the predictions (e.g. bounding box size or latent distributions). These techniques include KL divergence or Fréchet Inception Distance (FID) – see [151]. The robustness of the VAE to accurately predicting anomalous data can then be a function of the distance between the model performance on the training data and the model performance on the new, unseen data.

3.6.4.5. Uncertainty estimation robustness

For uncertainty aware model, the robustness shall be evaluated in terms of uncertainty estimate accuracy. This measures how well the estimation of uncertainty aligns with the model performance and confidence across the “known” datasets.

Metrics that can be used include:

- **Negative Log Likelihood (NLL)**: How well the estimated uncertainty distribution aligns with the ground truth.
- **Variation Ratio**: Measure disagreement across model predictions (for epistemic uncertainty aware model). If the distribution is assumed to be a Gaussian, standard deviation can be used.
- **Mutual information (MI[152])**: measuring information gain between the model parameters and the data, thus can distinguish between epistemic and aleatoric uncertainty types. High MI means the model is uncertainty because of lack of knowledge and can be improved by minimizing domain uncertainty.

3.6.4.6. Robustness against training dataset distribution changes

Measure the performance differences of the DL component if new data points are added, or some data points are removed to/from the training dataset*.* Data point influence metrics include:

- **Shapley value**: compute Shapley value (contribution) of added/removed datapoint to overall model performance
- **Data point loss**: measures the effect of new or removed data points on the model's loss function or gradients, indicating how much each point influences the model's behaviour.

- **Performance drop curve**: plots the model's performance as a function of the number of removed or added data points in the training dataset, revealing the sensitivity of the model to changes in the data distribution. Area under curve metric can be used here.

3.6.4.7. Training robustness

Different metrics can also be used to assess how well a neural network converges, generalizes and resists instability due to variations in hyper-parameters and initialization (weights) conditions. Training loss convergence can be measured in relation to how loss reduces over epochs (over a particular window of epochs) and can be evaluated in relation to hyperparameter changes – to evaluate how sensitive the model is to premature or late convergence based on small changes of the hyperparameters. Robust models must generalize well to the new data and apply various well-established techniques to limit the chances that the model is overfitting to the data it is trained on: comparing differences between validation and training loss periodically over epochs to see whether validation loss is increasing, applying regularization techniques, e.g. dropout and weight decay to reduce the prospect of model performance depending on artefactual features in the training dataset. Cross-validation, e.g. K-fold, can also be applied to assess variance in training performance and thereby sensitivity to particular hyperparameter settings or initialization of weights. High variability indicates that the trained model may not generalize well and thereby data augmentation techniques, or increased data diversity (e.g. targeting increased representation of particular features in the dataset) should then be applied in PhDM.

3.6.5. Failure rate

Traditional metrics can be used, however within SAFEXPLAIN the failure rates shall be measured for both Deep Learning components and corresponding supervisor monitors.

3.6.6. Easy to implement

Another necessary metrics concern measuring how easy it is to implement the Deep Learning component in order for it to be compliant with the AI-FSM lifecycle and engineering the required components for safety architecture patterns in Operation and Monitoring stage.

Some possible metrics include:

- **Time-to-Competence**: time spent in order to study the guidelines in the first place in order to be able to implement the deep learning component consistent with the requirements.
- **Learning curve time**: Time for an average AI engineer to learn but also adopt this guideline for engineering a dependable DL component as measured by periodic assessments; frequency of mistakes in implementation practices can also be evaluated.
- **Total resources used** to develop required safety components (supervisory monitors and other OM safety architecture's components): Including human effort/time, computing power required for implementation, computational time for processing the component given requirements' adherence.

4. Realization and libraries

4.1. EXPLib

EXPLib is an open-source Python library designed to facilitate the development and deployment of dependable DL components, leveraging Explainable Artificial Intelligence (XAI) techniques to ensure transparency, accountability, and reliability. The library provides a comprehensive framework for building, testing, and deploying DL models, with a focus on safety enabled by explainability.

By utilizing XAI tools and methods, EXPLib empowers users to gain valuable insights into the decision-making processes of their DL models, enabling the identification of potential safety-related risks and vulnerabilities. This facilitates the creation of more dependable and trustworthy DL components that compliant with the AI-FSM guidelines and safety architecture patterns. The library's modular and flexible architecture allows users to seamlessly integrate their own XAI methods and tools, as well as customize the library to suit their specific needs and requirements. Practitioners can focus on building high-quality DL models that not only achieve accuracy but also adhere to safety-critical practices and guidelines, provided by AI-FSM and Safety Patterns, ensuring the development of reliable and trustworthy AI systems.

Upon completion of the project, the EXPLib library will be made publicly available through open-source repositories such as GitHub or GitLab, ensuring widespread accessibility and facilitating community engagement and contributions.

4.1.1. Library structure

The EXPLib library structure (Table 3) is organized as follows:

- **UCs**: contains some specific files related to the 3 use cases of SAFEXPLAIN: Automotive, Railway and Space.
- **aifsm_phases**: holds example Jupyter notebooks showcasing how the library can be used to support AI-FSM phases and steps compliance (generating artifacts or assisting practitioners with relevant explanations).
- **configs**: stores configuration files for the library.
- **datasets**: provides datasets used in the library.
- **dl_component**: Typical DL components, especially those included in the Demo MVP model and the UCs
- **xai_library**: contains explainable AI (XAI) library resources to support both AI-FSM and OM compliance

Table 3: EXPLib structure

```
EXPLib/
├── UCs
├── aifsm_phases
├── configs
├── datasets
├── dl_component
└── xai_library
```

4.1.2. Guidelines and examples

4.1.2.1. Overview

Practitioners can use the library with the example usages provided via different Jupyter notebook scripts available and grouped into AI-FSM phases (Table 4).

- **data_management**: contains example notebook scripts demonstrating how to use the library to support various steps within the PhDM-Data Management phase.
    - *Data Profiling and Data mining*: Statistical reports and mining examples on dataset, both raw data and annotations statistics.
    - *Data prototypes*: Representative data prototypes extracted from the dataset, along with criticisms describing the dataset. Example representations include prototypical square patches extracted from an image dataset.
    - *Data descriptors*: Example implementations of VAE (Variational Autoencoder) based data descriptors to help describing the perception and conceptual features of the dataset. This descriptor will also be used later in OM stage as anomaly detector.
- **learning_management**: contains example notebook scripts demonstrating how to use the library to support various steps within the PhLM-Learning Management phase.
    - Intrinsic interpretable model design (integrated gradCAM)
    - Posthoc interpretable surrogate model
    - MVP uncertainty aware models: Aleatoric uncertainty aware model and MC dropout epistemic uncertainty model.
    - YOLOv8 epistemic uncertainty aware model.
    - Model explainers applied for MVP model and YOLO model: LIME, SHAP, CAM, LRP,…
- **inference_management**: contains example notebook scripts demonstrating how to use the library to support various steps within the PhIM-Inference Management phase.
    - Performance assessments: Plots of model performance wrt different input parameters
    - Scenario generators using metaheuristic searches
    - Uncertainty aware model
    - XAI for model (similar to PhLM): LIME, SHAP, CAM, LRP,…
    - Tradeoff mechanism.
- **operational_xai**: contains codes related to safety components for OM stage (to be transferred to DLLib for porting into the project NVIDIA ORIN platform).
    - OOD: MVP anomaly detectors (trained for detecting anomalies from input image, model's neuron activations patterns and model output). Anomaly detectors for UCs are not hosted within this library but are maintained by UC partners.
    - Uncertainty aware model: MC model providing epistemic uncertainty estimate and Aleatoric uncertainty aware model. The MC model is implemented with preloaded MC dropout models in a mix parallel/sequential architecture to support trade-off mechanism (memory, speed and model performance)
    - Surrogate models: Safe and interpretable surrogate models, train on the MVP dataset and the MVP model output
    - Ensemble model: Decision function using ensemble methods to aggregate outputs from different components and provide consolidated output of prediction and trustworthiness score

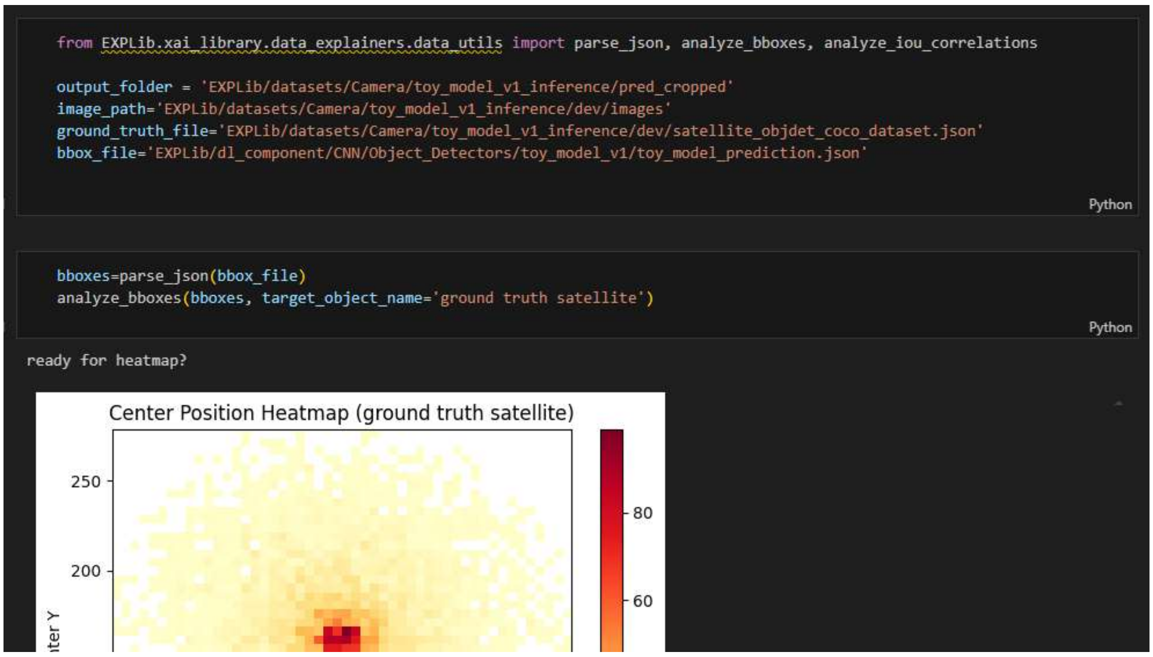
Table 4: Structure of aifsm_phases sublibrary

```
├── aifsm_phases
│   ├── data_management
│   │   ├── data_plot
│   │   └── data_profiling_logs
│   ├── inference_management
│   │   ├── performance_assessment
│     │     ├── scenario_generator
│     │     └── tradeoff
│     ├── learning_management
│     │     ├── model_election
│     │     │     ├── intrinsic_design
│     │     │     │     └── disentanglement
│     │     │     ├── posthoc_surrogate
│     │     │     └── uncertainty_aware
│     │     ├── model_training
│     │     └── model_verification
│     │           ├── model_output_files
│     │           └── supervisor_L1_output_files
│     └── operational_xai
│           ├── OOD
│           │     ├── likelihood_ratios
│           │     └── saved_vae_model
│           ├── ensemble_method
│           ├── queue_thread
│           ├── splitter
│           ├── surrogate_model
│           └── uncertainty_quantification
```

4.1.2.2. *Example scripts and resulting figures in AI-FSM phases*

4.1.2.2.1. PhDM Data Management

A screenshot of Data Profiling script is provided in Figure 32.



```
from EXPLib.xai_library.data_explainers.data_utils import parse_json, analyze_bboxes, analyze_iou_correlations

output_folder = 'EXPLib/datasets/Camera/toy_model_v1_inference/pred_cropped'
image_path='EXPLib/datasets/Camera/toy_model_v1_inference/dev/images'
ground_truth_file='EXPLib/datasets/Camera/toy_model_v1_inference/dev/satellite_objdet_coco_dataset.json'
bbox_file='EXPLib/dl_component/CNN/Object_Detectors/toy_model_v1/toy_model_prediction.json'
```

Python

```
bboxes=parse_json(bbox_file)
analyze_bboxes(bboxes, target_object_name='ground truth satellite')
```

Python

```
ready for heatmap?
```

Figure 32: EXPLib - example notebook of Data Profiling within PhDM

An example of latent space investigation is provided in Figure 33. The latent space is populated with datapoints from the MVP dataset using a VAE-based data descriptor. Initially, the latent space has a dimensionality of 256, which is then reduced to a three-dimensional representation using UMAP and t-SNE techniques to facilitate better human understanding.

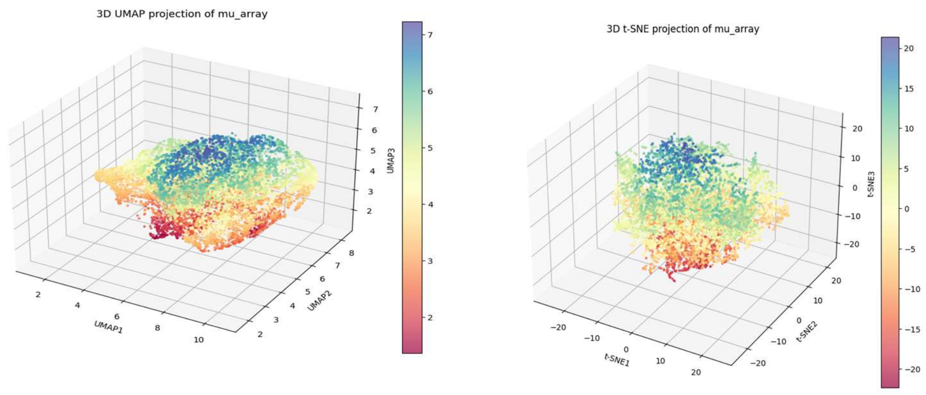


Figure 33: Latent distributions (of VAE descriptor) projected onto 3 dimensions computed using UMAP (left) and t-SNE (right) methods.

Figure 34 presents the results of a Principal Component Analysis (PCA) on the 256-dimensional latent space of the VAE-based data descriptor. Our analysis reveals that the first 15 eigenvectors can effectively capture the underlying structure of the high-dimensional latent space, allowing for accurate representation with a significantly reduced dimensionality. This insight led to the development of concept-based anomaly detection methods, which enable the identification of anomalous regions within an image (even in cases where the overall L2 reconstruction error is minimal) by leveraging the PCA error vectors and the VAE's deconvolutional layers for reconstruction.

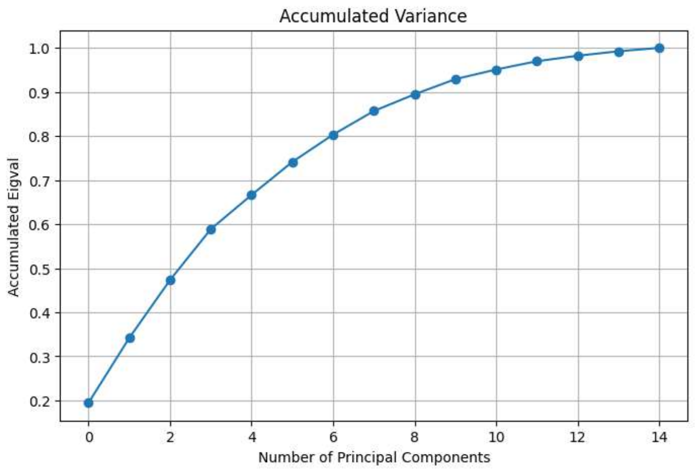


Figure 34: PCA analysis of latent space (of VAE-based data descriptor). Accumulated eigenvalues of the first 15 eigenvectors

Figure 35 demonstrates the performance of the "descriptor latent" anomaly detector using a varying number of eigenvectors (ranging from 1 to 12). Notably, even with as few as 2 eigenvectors, the accidentally removed antenna of the satellite is successfully identified as an anomalous region.

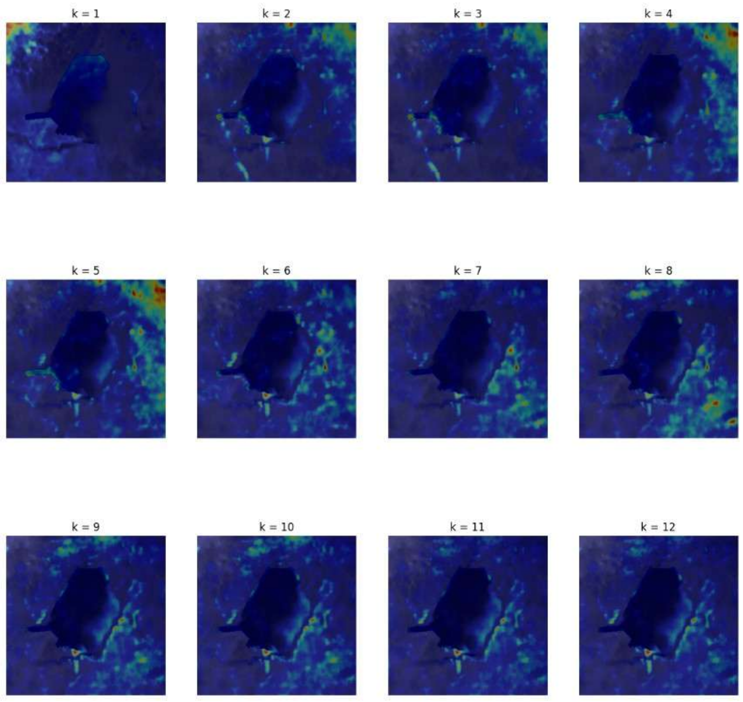


Figure 35: Experiments of using PCA reconstructed latent vector of data descriptor to identify anomalous area in the input image

4.1.2.2.2. PhLM Learning Management

Examples of using LRP and Contrastive LRP for generating attention heatmaps for different classes (Person and Bicycle) are shown in Figure 37 and Figure 36 respectively.

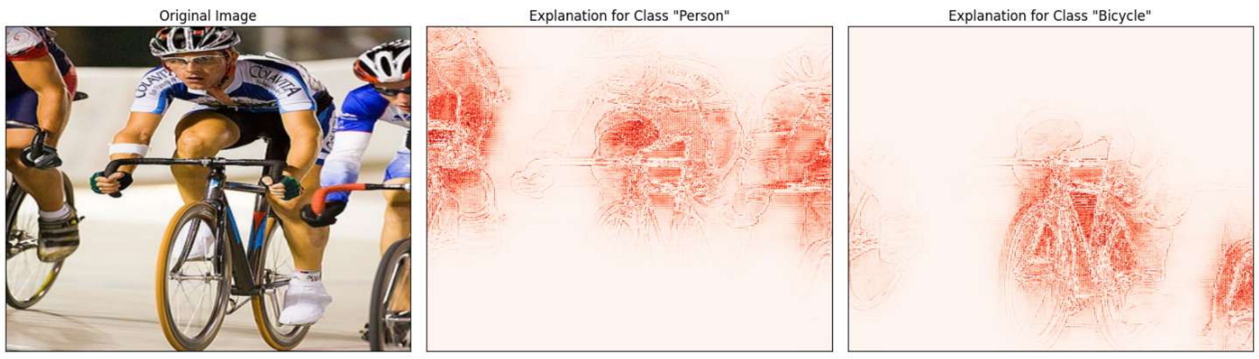

Original Image
Explanation for Class "Person"
Explanation for Class "Bicycle"

Figure 37: LRP saliency map generated for YOLOv8 (Railway UC)

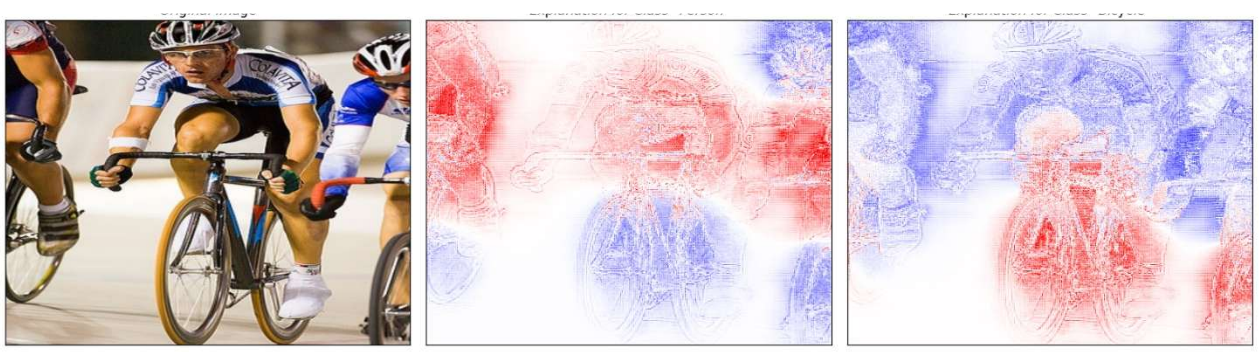


Figure 36: Contrastive LRP saliency map generated for YOLOv8 (Railway UC)

LIME Heatmap generated for MVP model is illustrated in Figure 38. The supported segmentation methods include SLIC and Watershed.

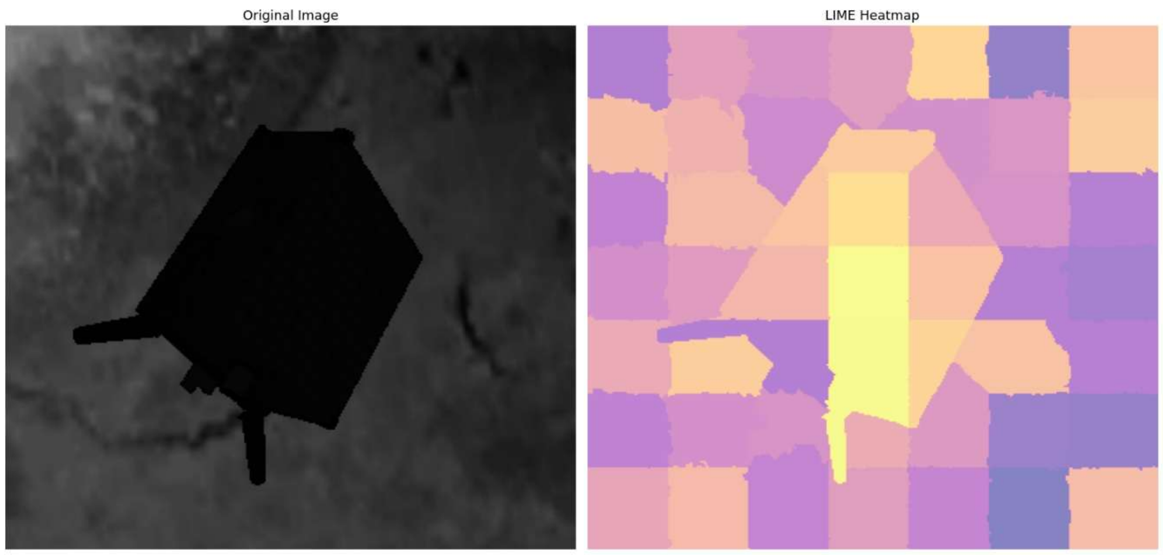


Figure 38: LIME heatmap computed for MVP model, using superpixel segmentation method Watershed as input for LIME

4.1.2.2.3. PhIM Inference Management

Figure 39 demonstrates how XAI tools in the EXPLib can be utilized to support performance assessment. The plots illustrate the relationship between safety-related performance metrics (IoU in this case) and input data parameters, specifically object location (represented by bounding box centre coordinates) and appearance size (translated into distance to the sensor and represented by area).

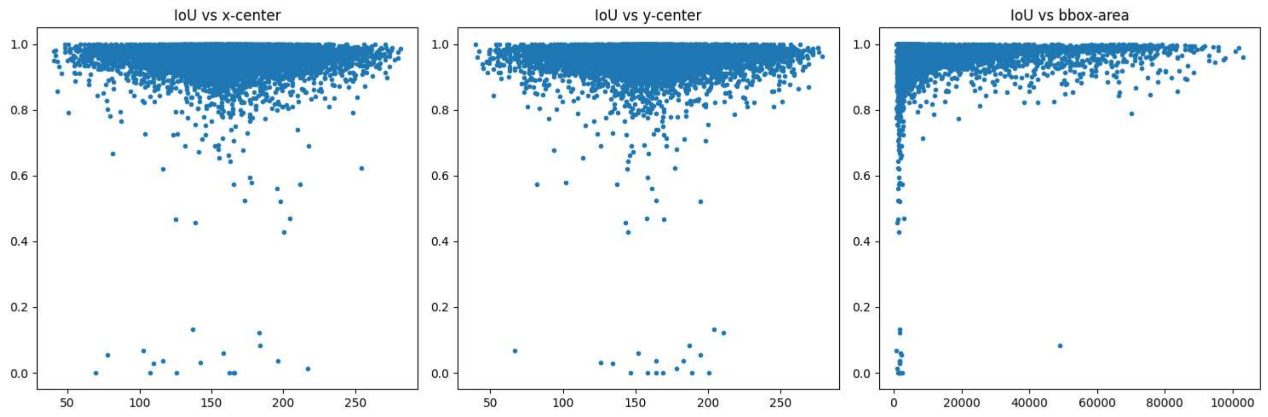
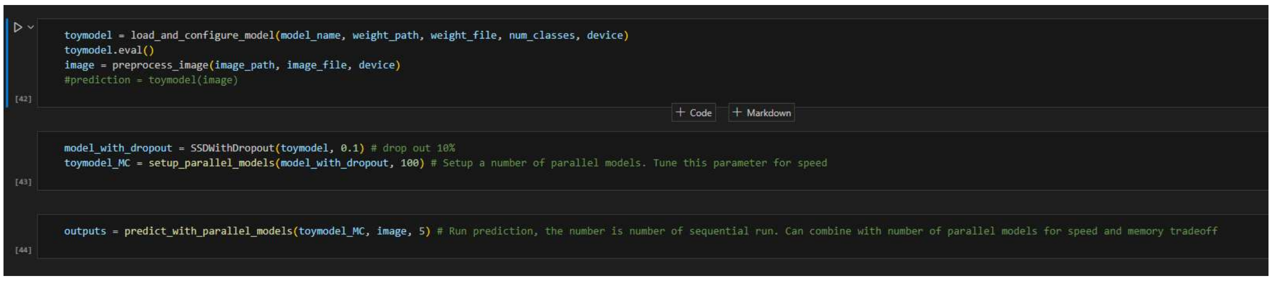


Figure 39: Model performance (IoU metric) vs input parameters: bounding box location and area

4.1.2.2.4. Operational XAI

Figure 40 presents a screenshot illustrating an example usage of the library, where practitioners can create an uncertainty-aware model for deployment to the OM stage. In this example, MC dropout models are pre-loaded into a mix structure, comprising 100 parallel runs of 5 sequential models, resulting in a total of 500 models in the ensemble. The configurable selection of number of parallel models and number of sequential models enables trade-off mechanisms between three key factors: (1) accuracy of uncertainty estimates (measured by the number of samples in the prediction distribution), (2) RAM usage (influenced by the number of parallel models), and (3) execution time (affected by the number of models in the sequential path).



```
toymodel = load_and_configure_model(model_name, weight_path, weight_file, num_classes, device)
toymodel.eval()
image = preprocess_image(image_path, image_file, device)
#prediction = toymodel(image)
```

```
model_with_dropout = SSDWithDropout(toymodel, 0.1) # drop out 10%
toymodel_MC = setup_parallel_models(model_with_dropout, 100) # Setup a number of parallel models. Tune this parameter for speed
```

```
outputs = predict_with_parallel_models(toymodel_MC, image, 5) # Run prediction, the number is number of sequential run. Can combine with number of parallel models for speed and memory tradeoff
```

Figure 40: Example usage of Epistemic uncertainty aware MVP model

More figures and examples can also be found in Section 3.5.

4.1.3. Libraries of XAI tools

The core functionalities, classes, and utilities within EXPLib are organized in the `xai_library` module (Table 5), which is structured into three primary categories:

- **data_explainers**: This module provides XAI tools related to data and datasets, including:
    - Data descriptors
    - Prototypes and criticisms
    - Similarity and distance metrics

  - Data profiling
- **model_explainers**: This module offers XAI tools focused on model analysis, such as:
  - Saliency maps
  - Surrogate models
  - Various types of plots
  - Utilities for extracting neuron activations and computing gradients
  - Building uncertainty-aware models
- **supervisor**: This component provides XAI tools and training scripts for AI-based components intended for deployment as supervision components within the OM stage. Notably, these components are trained on the "Known" area, which is represented by the datasets and related results, including model activations, gradients, predictions, and other relevant outputs.
- **metric_extractors**: Several metric computation utilities are provided within this submodule, including e.g. model performance metrics (IoU, F1), structural coverage (NBC, NC…), similarity and distance metrics.

Table 5: Structure of xai_library

```
└── xai_library
    ├── data_explainers
    │   ├── data_descriptors
    │   └── prototypes
    ├── metric_extractors
    ├── model_explainers
    │   ├── DT
    │   ├── anchors
    │   ├── cam
    │   ├── extract_layer_activation
    │   ├── lrp
    │   ├── model_utils
    │   ├── plots
    │   ├── search_algorithms
    │   ├── shap
    │   ├── surrogate
    │   └── uncertainty_models
    └── supervisor
        ├── SMF
        ├── anomaly_detector
        │   ├── likelihood_ratios
        │   ├── vae
        ├── ensemble_method
        ├── surrogate_model
        └── uncertainty_models
```

4.1.4. Models

This module contains implementations of DL models utilized within the project. The current models included are categorized by type, as reflected in the following library structure (Table 6):

- **AEs**: Various Variational Autoencoder (VAE) models used within SAFEXPLAIN.
- **CNN**: Scripts and weights for visual-based Convolutional Neural Networks, including image classification and object detection models (SSDlite, FasterRCNN, and associated backbones).
- **MLP**: Placeholder – Multilayer Perceptrons have been integrated directly into CNN models.
- **RNN_LSTM**: Memory-based models (RNN, LSTM) constructed with simple Torch layers.
- **Transformers:** Standard BERT models.

Table 6: Structure of dl_component submodule

```
├── dl_component
|    ├── AEs
|    ├── CNN
|    |    ├── Image_Classifiers
|    |    └── Object_Detectors
|    ├── MLP
|    ├── RNN_LSTM
|    └── Transformers
```

4.1.5. Datasets

This subfolder contains datasets used to test and demonstrate the various XAI methods employed within the project. For publicly available datasets, we provide references to the data owners rather than hosting the data directly.

The two main datasets hosted within this library are the MVP dataset (together with annotations) and a supplementary dataset containing added UFO objects, which is used to evaluate different safety mechanisms.

4.2. DLLib

Building on the progress from EXPLib, we focus on the integration of DL software components for OM stage into DLLib. To support the complex models and large datasets previously discussed, high-performance computing is essential, driving the use of low-level C-based libraries such as cuBLAS and cuDNN (included in NVIDIA Jetpack), optimized for hardware architectures like NVIDIA GPUs and Arm-based CPUs.

With the recent, albeit slightly delayed, inputs from T3.4 now available, the development of DLLib includes most of the supervision techniques and is actively advancing to incorporate the remaining techniques as detailed in Table 7.

Since efforts have shifted towards working with ROS2 in the Middleware layer to optimize functionalities more effectively, as detailed in a later section, the current organization of DLLib has been aligned with the

Middleware's structure. The updated DLLib can be found in the Middleware Git repository under the branch "feat_sp2_wp3."

Table 7: Porting status of OM components

Component	Status	Notes
DL Models	In Progress	SSD model ready; FasterRCNN model ready; Uncertainty Model ongoing optimization
Input Data Management	Awaiting Review	Offline verification datasets prepared
VAEs	Completed	All three VAEs are implemented
Pre-processing	Completed	Optimization for NVIDIA platform ongoing
Post-processing	Planned	Development to start post initial deployment tests
Surrogate Model	In Progress	Recently received
Ensemble Method	In Progress	Ongoing update to include the new models.

This phase bridges theoretical frameworks and practical implementation by strategically selecting and designing DL models, data management protocols, and XAI methodologies tailored to the computational needs of the target hardware platforms. It involves adopting or re-implementing C and C++ libraries, including open-source alternatives, to optimize performance and efficiency while leveraging the AI4EU platform to promote reuse and minimize redundant development.

Specifically, we focus on implementing DL software on the SAFEXPLAIN prototyping platform (NVIDIA AGX Orin) with the low-level middleware PMULib, keeping as a cornerstone the performance optimizations tailored to case studies. This involves adapting black-box libraries, optimizing data preprocessing, and fine-tuning runtime metrics to enhance DL system capabilities while maintaining design integrity and achieving project goals. By strategically aligning these implementation efforts with the platform's hardware capabilities, we maximize computational efficiency and ensure robust, scalable solutions for real-world applications.

4.2.1. Integration with ROS2 (Middleware)

We have successfully integrated all the VAE based anomaly detectors (input, model, output) into the Middleware, customized to utilize ROS2 for efficient communication between various modules. Building on this foundation, we are now incorporating additional advanced features, including Uncertainty Models, Surrogate Models, and Ensemble Methods, into the ROS2-based Middleware layer. These integrations are aimed at enhancing the system's diagnostic and supervisory capabilities, which are critical for maintaining reliable and adaptive real-time performance.

The integration process involves adapting and extending functionalities from EXPLib, organizing them into reusable ROS2 nodes that encapsulate each feature. This modular design ensures seamless communication and interoperability through ROS2's publish-subscribe mechanisms, supporting scalability and maintainability. By structuring these modules as independent nodes, we enable flexible deployment and dynamic updates without interrupting system operations.

Leveraging ROS2's real-time communication capabilities, alongside its robust support for distributed systems, allows us to efficiently manage data flow between various components, including DL models, Supervisors, Decision Functions, and other system elements. This architectural design ensures low-latency, high-throughput communication, which is essential for real-time applications that demand rapid decision-making and adaptive behaviour.

Additionally, the Middleware is architected to support asynchronous processing and dynamic model updates, facilitating adaptive supervision and robust uncertainty management. This adaptability is particularly crucial for real-time environments where conditions can change rapidly. The iterative development approach enhances the system's reliability and responsiveness while maintaining modularity for future scalability. Ultimately, this design strategy ensures that the DL modules are seamlessly integrated with the overall control and diagnostic framework, paving the way for advanced, intelligent supervision in complex, distributed environments.

4.2.2. Supervision Function & Decision Function

DLLib incorporates three different Supervisor systems, detailed in Section 3.5, which in turn are aligned with the reference architecture pattern outlined in Deliverable D2.2[2].This implementation significantly enhances the reliability and efficiency of deployed DL components through the integration of advanced supervisory monitor components, including:

- **OOD Detector Algorithms:** Trained Variational Autoencoder (VAE) descriptors have been utilized to identify data points or patterns that deviate significantly from the model's training distribution. This component effectively flags potential outliers or novel scenarios that require additional scrutiny, ensuring the system's robustness against unexpected inputs. We have three different VAEs currently implemented:
- **Input VAE:** This component checks for outliers on the input image.
- **Output VAE:** This component checks for outliers on the output image of the main model.
- **Activation VAE:** This component checks for outliers during the inference of the main model by looking into some of the activations from key layers.
- **Ensemble Methods:** Predictions from multiple models are combined with the supervisors' outputs to validate their reliability and accuracy. This approach effectively leverages the strengths of various models, achieving superior performance and robustness compared to any single model alone.

By integrating these supervisor monitor components with low-level libraries, DLLib has achieved a deep and nuanced understanding of the deployed system's performance. This integration enables effective monitoring and management of DL components by leveraging detailed performance and error metrics to make informed decisions about model adjustments, parameter tuning, and resource allocation.

The comprehensive Supervision and Decision system has ensured that the deployed DL system remains efficient, accurate, and reliable. It dynamically adapts to evolving operational conditions, maintaining high standards of performance over time. This successful implementation demonstrates DLLib's commitment to delivering state-of-the-art solutions for advanced deep learning systems.

5. Discussions

This document presents the final outcomes of tasks T3.1, T3.2, and T3.3 within the SAFEXPLAIN project, providing a comprehensive summary of the specification, design, and improvement of dependable deep learning (DL) components. The report updates and finalizes the contents of D3.1 and D3.2, offering actionable recommendations for specifying dependable DL components in compliance with AI-FSM and recommended safety patterns. Furthermore, it provides in-depth discussions on explainability, traceability, and robustness exemplified for a Minimum Viable Product (MVP), serving as a handbook on developing DL components for safety critical applications.

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] "Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act)," *Official Journal of the European Union*, vol. L 1689, pp. 1–100, Jul. 2024.

[4] International Organization for Standardization, *Information technology — Artificial intelligence — Objectives and approaches for explainability and interpretability of machine learning (ML) models and artificial intelligence (AI) systems*, 2025. [Online]. Available: https://www.iso.org/standard/82148.html

[5] IEEE Computer Society, "IEEE Guide for an Architectural Framework for Explainable Artificial Intelligence." 2024. [Online]. Available: https://standards.ieee.org/standard/2894-2024.html

[6] IEEE Computational Intelligence Standards Committee, "Standard for Explainable Artificial Intelligence - for Achieving Clarity and Interoperability of AI Systems Design." 2021. [Online]. Available: https://standards.ieee.org/standard/P2976.html

[7] International Organization for Standardization, "Information technology — Artificial intelligence — Data life cycle framework." 2023. [Online]. Available: https://www.iso.org/standard/83002.html

[8] International Organization for Standardization, "Information technology — Artificial intelligence — Management system." 2023. [Online]. Available: https://www.iso.org/standard/81230.html

[9] International Organization for Standardization, "Information technology — Artificial intelligence (AI) — Use cases." 2024. [Online]. Available: https://www.iso.org/standard/84144.html

[10] International Organization for Standardization, "Information technology — Artificial intelligence — Guidance for AI applications." 2024. [Online]. Available: https://www.iso.org/standard/81120.html

[11] International Organization for Standardization, "Artificial intelligence — Functional safety and AI systems (ISO Standard No. 5469:2024)," 2024. [Online]. Available: https://www.iso.org/standard/81283.html

[12] International Organization for Standardization, *Road vehicles — Safety of the intended functionality (ISO Standard No. 21448:2022)*, 2022. [Online]. Available: https://www.iso.org/standard/77490.html

[13] International Organization for Standardization, *Road vehicles — Functional safety (ISO Standard No. 26262:2018)*, 2018. [Online]. Available: https://www.iso.org/publication/PUB200262.html

[14] M. Consortium, "EASA Research – Machine Learning Application Approval (MLEAP) Final Report," EASA, May 2024. [Online]. Available: https://www.easa.europa.eu/sites/default/files/dfu/mleap-d4-public-report-issue01.pdf

[15] International Organization for Standardization, *Road vehicles — Safety and artificial intelligence (ISO Standard No. 8800:2024)*, 2024. [Online]. Available: https://www.iso.org/standard/83303.html

[16] Underwriters Laboratories, *Standard for Evaluation of Autonomous Products (UL 4600)*, 2023. [Online]. Available: https://ulstandards.ul.com/standard/?id=4600

[17] International Electrotechnical Commission, *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (IEC 61508)*, 2010. [Online]. Available: https://webstore.iec.ch/publication/61508

[18] International Organization for Standardization, *Information technology — Artificial intelligence — Artificial intelligence concepts and terminology (ISO/IEC 22989:2022)*, 2022. [Online]. Available: https://www.iso.org/standard/74296.html

[19] SAFEXPLAIN, “D3.1: SAFEXPLAIN Specifiability, explainability, traceability, and robustness proof-of-concept and argumentation.” Deliverable of the HEU SAFEXPLAIN project, Grant Agreement No. 101069595, 2024.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] E. Tola, V. Lepetit, and P. Fua, “DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 815–830, May 2010, doi: 10.1109/TPAMI.2009.77.

[30] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, Jun. 2005, pp. 886–893 vol. 1. doi: 10.1109/CVPR.2005.177.

[31] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA: IEEE Comput. Soc, 2001, p. I-511-I–518. doi: 10.1109/CVPR.2001.990517.

[32] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, Jan. 1996, doi: 10.1016/0031-3203(95)00067-4.

[33] M. Agrawal, K. Konolige, and M. R. Blas, “CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching,” in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds., Berlin, Heidelberg: Springer, 2008, pp. 102–115. doi: 10.1007/978-3-540-88693-8_8.

[34] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2564–2571. doi: 10.1109/ICCV.2011.6126544.

[35] G. H. Granlund, “In search of a general picture processing operator,” *Computer Graphics and Image Processing*, vol. 8, no. 2, pp. 155–173, Oct. 1978, doi: 10.1016/0146-664X(78)90047-3.

[36] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Sep. 1999, pp. 1150–1157 vol.2. doi: 10.1109/ICCV.1999.790410.

[37] J. J. Koenderink and A. J. van Doorn, “Surface shape and curvature scales,” *Image and Vision Computing*, vol. 10, no. 8, pp. 557–564, Oct. 1992, doi: 10.1016/0262-8856(92)90076-F.

[38] I. M. Chakravarti, R. G. Laha, and J. Roy, *Handbook of Methods of Applied Statistics, Volume I*. John Wiley and Sons, 1967.

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

[40] A. Bhattacharyya, “On a measure of divergence between two statistical populations defined by their probability distributions,” *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99–109, 1943.

[41] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A Kernel Two-Sample Test,” in *Journal of Machine Learning Research*, 2012, pp. 723–773.

[42] 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

[43] 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

[44] K. Pearson, “LIII. On lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901, doi: 10.1080/14786440109462720.

[45] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936, doi: 10.1111/j.1469-1809.1936.tb02137.x.

[46] A. Hyvärinen and E. Oja, “Independent component analysis: algorithms and applications,” *Neural Networks*, vol. 13, no. 4–5, pp. 411–430, 2000, doi: 10.1016/S0893-6080(00)00026-5.

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

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

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

[50] 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.

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

[52] 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

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

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

[55] 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.

[56] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, “Not Just a Black Box: Learning Important Features Through Propagating Activation Differences,” *arXiv:1605.01713 [cs]*, Apr. 2017, Accessed: Jul. 12, 2021. [Online]. Available: http://arxiv.org/abs/1605.01713

[57] 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.

[58] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg, “SmoothGrad: removing noise by adding noise,” *CoRR*, vol. abs/1706.03825, 2017, [Online]. Available: http://arxiv.org/abs/1706.03825

[59] 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

[60] 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.

[61] 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

[62] 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.

[63] M. T. Ribeiro, S. Singh, and C. Guestrin, “Anchors: High-Precision Model-Agnostic Explanations,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Art. no. 1, Apr. 2018, doi: 10.1609/aaai.v32i1.11491.

[64] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 818–833. doi: 10.1007/978-3-319-10590-1_53.

[65] V. Petsiuk, A. Das, and K. Saenko, “RISE: Randomized Input Sampling for Explanation of Black-box Models,” in *British Machine Vision Conference (BMVC)*, 2018. [Online]. Available: http://bmvc2018.org/contents/papers/1064.pdf

[66] 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.

[67] B. Kim *et al.*, “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV),” in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, Jul. 2018, pp. 2668–2677. Accessed: Nov. 13, 2023. [Online]. Available: https://proceedings.mlr.press/v80/kim18d.html

[68] I. Higgins *et al.*, “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework,” in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl

[69] 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.

[70] “Neural-Symbolic Learning Systems,” in *Neural-Symbolic Cognitive Reasoning*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 35–54. doi: 10.1007/978-3-540-73246-4_4.

[71] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2016. Accessed: Sep. 21, 2021.

[Online]. Available: https://proceedings.neurips.cc/paper/2016/hash/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Abstract.html

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

[73] 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.

[74] M. A. Migut, M. Worring, and C. J. Veenman, "Visualizing multi-dimensional decision boundaries in 2D," *Data Mining and Knowledge Discovery*, vol. 29, no. 1, pp. 273–295, Jan. 2015, doi: 10.1007/s10618-013-0342-x.

[75] D. Erhan, Y. Bengio, A. Courville, P. Vincent, and P. O. Box, "Visualizing Higher-Layer Features of a Deep Network," 2009.

[76] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going Deeper into Neural Networks." 2015. [Online]. Available: https://research.google.com/blog/2015/06/inceptionism-going-deeper-into-neural-networks.html

[77] 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.

[78] 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)*, Budapest, Hungary: IEEE, Jul. 2019, pp. 1–9. doi: 10.1109/IJCNN.2019.8851770.

[79] 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.

[80] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, May 2017, doi: 10.1016/j.patcog.2016.11.008.

[81] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking Clever Hans predictors and assessing what machines really learn," *Nat Commun*, vol. 10, no. 1, p. 1096, Mar. 2019, doi: 10.1038/s41467-019-08987-4.

[82] 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.

[83] A. Nguyen, J. Yosinski, and J. Clune, "Understanding Neural Networks via Feature Visualization: A survey," Apr. 18, 2019, *arXiv*: arXiv:1904.08939. doi: 10.48550/arXiv.1904.08939.

[84] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000, doi: 10.1126/science.290.5500.2319.

[85] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000, doi: 10.1126/science.290.5500.2323.

[86] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[87] R. K. Been Kim and S. Koyejo, "Examples are not Enough, Learn to Criticize! Criticism for Interpretability," in *Advances in Neural Information Processing Systems*, 2016.

[88] 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.

[89] P. McCullagh and J. Nelder, *Generalized Linear Models*. London, UK: Chapman & Hall / CRC, 1989.

[90] 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.

[91] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, pp. 81–106, 1986.

[92] 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.

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

[94] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille, “A Bayesian Framework for Learning Rule Sets for Interpretable Classification,” *Journal of Machine Learning Research*, vol. 18, no. 70, pp. 1–37, 2017.

[95] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Aug. 1995, pp. 278–282 vol.1. doi: 10.1109/ICDAR.1995.598994.

[96] 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.

[97] A. Dhurandhar, K. Shanmugam, R. Luss, and P. A. Olsen, “Improving Simple Models with Confidence Profiles,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/972cda1e62b72640cb7ac702714a115f-Paper.pdf

[98] 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.

[99] 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.

[100] U. Schlegel, E. Cakmak, and D. A. Keim, “ModelSpeX: Model Specification Using Explainable Artificial Intelligence Methods,” in *Machine Learning Methods in Visualisation for Big Data*, D. Archambault, I. Nabney, and J. Peltonen, Eds., The Eurographics Association, 2020. doi: 10.2312/mlvis.20201100.

[101] 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*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., in Proceedings of Machine Learning Research, vol. 162. PMLR, Jul. 2022, pp. 11162–11172. [Online]. Available: https://proceedings.mlr.press/v162/kim22g.html

[102] 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.

[103] 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.

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

[105] 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.

[106] M. Craven and J. Shavlik, “Extracting Tree-Structured Representations of Trained Networks,” in *Advances in Neural Information Processing Systems*, MIT Press, 1995. Accessed: Mar. 24, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1995/hash/45f31d16b1058d586fc3be7207b58053-Abstract.html

[107] 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.

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

[109] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, and M. Vechev, “DL2: Training and Querying Neural Networks with Logic,” in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, May 2019, pp. 1931–1941. Accessed: Mar. 24, 2025. [Online]. Available: https://proceedings.mlr.press/v97/fischer19a.html

[110] R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. De Raedt, “Neural probabilistic logic programming in DeepProbLog,” *Artificial Intelligence*, vol. 298, p. 103504, Sep. 2021, doi: 10.1016/j.artint.2021.103504.

[111] E. Giunchiglia, M. C. Stoian, and T. Lukasiewicz, “Deep Learning with Logical Constraints,” in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, Vienna, Austria: International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 5478–5485. doi: 10.24963/ijcai.2022/767.

[112] P. W. Koh *et al.*, “Concept Bottleneck Models,” in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., in Proceedings of Machine Learning Research, vol. 119. PMLR, Jul. 2020, pp. 5338–5348. [Online]. Available: https://proceedings.mlr.press/v119/koh20a.html

[113] Z. Chen, Y. Bei, and C. Rudin, “Concept whitening for interpretable image recognition,” *Nature Machine Intelligence*, vol. 2, no. 12, pp. 772–782, Dec. 2020, doi: 10.1038/s42256-020-00265-z.

[114] D. Kazhdan, B. Dimanov, M. Jamnik, P. Liò, and A. Weller, “Now You See Me (CME): Concept-based Model Extraction,” Oct. 25, 2020, *arXiv*: arXiv:2010.13233. doi: 10.48550/arXiv.2010.13233.

[115] C.-K. Yeh, B. Kim, S. Arik, C.-L. Li, T. Pfister, and P. Ravikumar, “On Completeness-aware Concept-Based Explanations in Deep Neural Networks,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., Curran Associates, Inc., 2020, pp. 20554–20565. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/ecb287ff763c169694f682af52c1f309-Paper.pdf

[116] F. Locatello, B. Poole, G. Rätsch, B. Schölkopf, O. Bachem, and M. Tschannen, “Weakly-supervised disentanglement without compromises,” in *Proceedings of the 37th International Conference on Machine Learning*, in ICML’20. JMLR.org, 2020.

[117] K. Xu *et al.*, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention,” in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, Jun. 2015, pp. 2048–2057. Accessed: Mar. 24, 2025. [Online]. Available: https://proceedings.mlr.press/v37/xuc15.html

[118] 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.

[119] 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

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

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

[122] 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.

[123] 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.

[124] 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.

[125] 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.

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

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

[128] 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.

[129] 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.

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

[131] 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.

[132] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv:1801.04381 [cs]*, Jan. 2018, Accessed: May 10, 2019. [Online]. Available: http://arxiv.org/abs/1801.04381

[133] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jan. 06, 2016, *arXiv*: arXiv:1506.01497. doi: 10.48550/arXiv.1506.01497.

[134] A. Brando, I. Serra, E. Mezzetti, F. J. Cazorla, and J. Abella, "Standardizing the Probabilistic Sources of Uncertainty for the sake of Safety Deep Learning," in *Proceedings of the Workshop on Artificial Intelligence Safety 2023 (SafeAI 2023) co-located with the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023), Washington DC, USA, February 13-14, 2023*, G. Pedroza, X. Huang, X. C. Chen, A. Theodorou, J. Hernández-Orallo, M. Castillo-Effen, R. Mallah, and J. A. McDermid, Eds., in CEUR Workshop Proceedings, vol. 3381. CEUR-WS.org, 2023. [Online]. Available: https://ceur-ws.org/Vol-3381/11.pdf

[135] G. Vacanti and A. V. Looveren, "Adversarial Detection and Correction by Matching Prediction Distributions," Feb. 21, 2020, *arXiv*: arXiv:2002.09364. doi: 10.48550/arXiv.2002.09364.

[136] *Deci-AI/data-gradients*. (Mar. 11, 2025). Python. deci.ai. Accessed: Mar. 19, 2025. [Online]. Available: https://github.com/Deci-AI/data-gradients

[137] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," in *Computer Vision – ECCV 2018*, vol. 11211, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds.,

in Lecture Notes in Computer Science, vol. 11211. , Cham: Springer International Publishing, 2018, pp. 3–19. doi: 10.1007/978-3-030-01234-2_1.

[138] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[139] G. Farnebäck, “Two-Frame Motion Estimation Based on Polynomial Expansion,” in *Image Analysis*, vol. 2749, J. Bigun and T. Gustavsson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370. doi: 10.1007/3-540-45103-X_50.

[140] J. Aigrain and M. Detyniecki, “Detecting adversarial examples and other misclassifications in neural networks by introspection,” in *arXiv preprint arXiv:1905.09186.*, 2019.

[141] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” 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. 785–794. doi: 10.1145/2939672.2939785.

[142] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, “A Survey of Methods for Explaining Black Box Models,” *ACM Comput. Surv.*, vol. 51, no. 5, p. 93:1-93:42, Aug. 2018, doi: 10.1145/3236009.

[143] C. Munoz, K. da Costa, B. Modenesi, and A. Koshiyama, “Evaluating Explainability in Machine Learning Predictions Through Explainer-Agnostic Metrics,” in *arXiv preprint*, 2024.

[144] E. Mariotti, A. Sivaprasad, and J. M. A. Moral, “Beyond prediction similarity: ShapGAP for evaluating faithful surrogate models in XAI,” in *World conference on explainable artificial intelligence*, Cham: Springer Nature Switzerland, 2023, pp. 160–173.

[145] G. Yang, Q. Ye, and J. Xia, “Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond.,” *Information Fusion*, vol. 77, pp. 29–52, 2022.

[146] P. Chen, J. Ye, G. Chen, J. Zhao, and P. A. Heng, “Robustness of accuracy metric and its inspirations in learning with noisy labels,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 11451–11461.

[147] G. K. Nayak, I. Khatri, S. Randive, R. Rawal, and A. Chakraborty, “DAD++: Improved data-free test time adversarial defense.,” *Neurocomputing*, 2025.

[148] J. Henriksson, C. Berger, M. Borg, L. Tornberg, S. R. Sathyamoorthy, and C. Englund, “Performance Analysis of Out-of-Distribution Detection on Various Trained Neural Networks,” presented at the 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Greece, Aug. 2019.

[149] M. Azizmalayeri, A. Soltani Moakhar, A. Zarei, R. Zohrabi, M. Manzuri, and M. H. Rohban, “Your out-of-distribution detection method is not robust!,” *Advances in neural information processing systems*, vol. 35, pp. 4887–4901, 2022.

[150] B. Zhao *et al.*, “Ood-cv: A benchmark for robustness to out-of-distribution shifts of individual nuisances in natural images,” in *European conference on computer vision*, Cham: Springer Nature Switzerland, 2022.

[151] J. Linden, H. Forsberg, M. Daneshtalab, and I. Soderquist, “Evaluating the robustness of ml models to out-of-distribution data through similarity analysis,” in *European Conference on Advances in Databases and Information Systems*, Cham: Springer Nature Switzerland, 2023, pp. 348–359.

[152] L. Smith and Y. Gal, “Understanding Measures of Uncertainty for Adversarial Example Detection”.