Safe and Explainable
Critical Embedded Systems based on AI

# D3.4 Final DL components, libraries and the DL interface

Version 1.0

## Documentation Information

| | |
|---|---|
| **Contract Number** | 101069595 |
| **Project Website** | www.safexplain.eu |
| **Contractual Deadline** | 30.09.2025 |
| **Dissemination Level** | PU |
| **Nature** | OTHER |
| **Author** | BSC CNS |
| **Contributors** | Axel Brando (BSC), Roger Pujol (BSC), Jokin Labaien (IKR), Robert Lowe (RISE), Gabriele Giordana (Aiko), Maria Ulan (RISE), Ana Adell (IKR) |
| **Reviewer** | Thanh Hai Bui (RISE) |
| **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

# List of figures

# 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 M36) of SAFEXPLAIN's FUSA-aware dependable Deep Learning (DL) solutions within WP3. The document contains an overview on the code included in the deliverable.

The source code and libraries (for proprietary artifacts) for WP3 SW have been archived and uploaded to the B2DROP repository maintained by BSC, It can be accessed through the following link: https://b2drop.bsc.es/index.php/s/D9eteoH5fcYPaq2

# 1. Introduction

This document reports the final results of tasks "*T3.4* - Implementation of DL components" and "*T3.5 - DL ibraries*"  within the SAFEXPLAIN project.

It constitutes the culmination of the libraries involved in 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 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.

# 2. Realization and libraries

## 2.1. 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 (https://github.com/RI-SE/EXPLib), ensuring widespread accessibility and facilitating community engagement and contributions.

### 2.1.1. Library structure

The EXPLib library structure (Table 1) 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 1: EXPLib structure*

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

### 2.1.2. Guidelines and examples

#### *2.1.2.1. Overview*

Practitioners can use the library with the example usages provided via different Jupyter notebook scripts available and grouped into AI-FSM phases (Table 2).

- **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.
    - Epistemic uncertainty aware model.
    - Model explainers applied for MVP model and YOLO model
- **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)
- **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

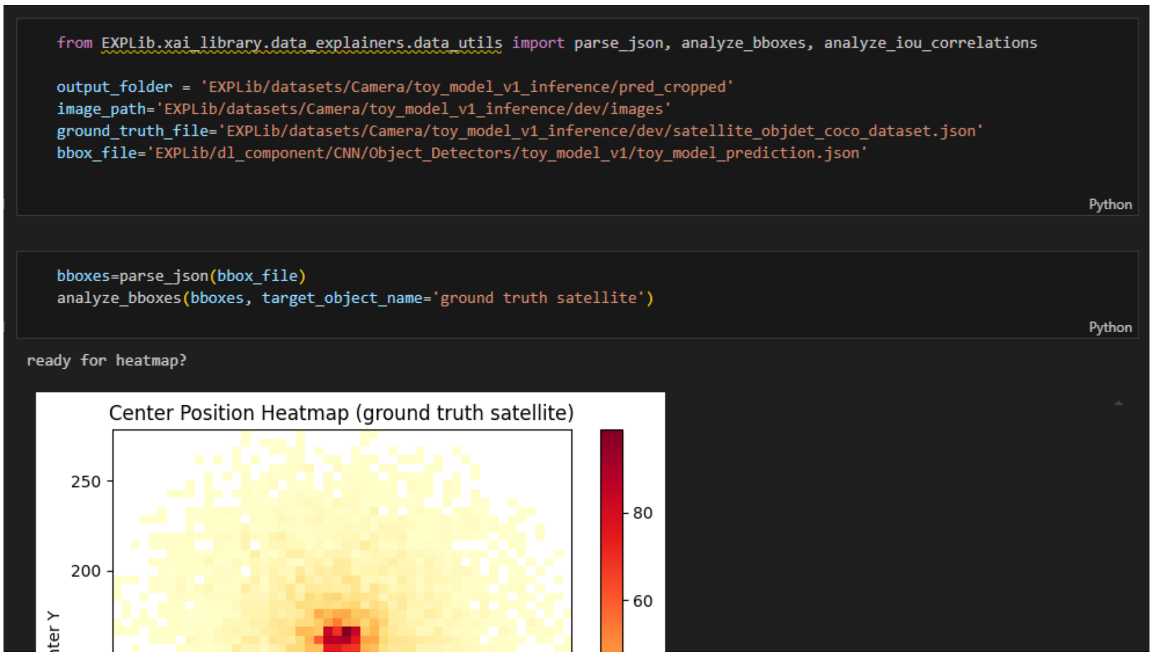
*Table 2: Structure of aifsm_phases sublibrary*

```
├── aifsm_phases
```

```
|       ├── data_management
|       ├── inference_management
|       └── operational_xai
|           ├── OOD
|           ├── ensemble_method
|           ├── surrogate_model
|           └── uncertainty_quantification
```

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

#### 2.1.2.2.1. PhDM Data Management



```python
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')
```

```
ready for heatmap?
```

Figure 1: EXPLib - example notebook of Data Profiling within PhDM

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

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.

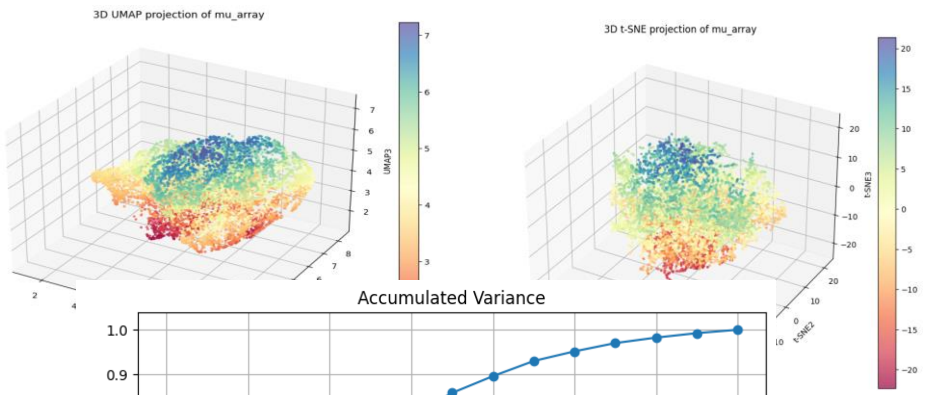


*Figure 2:* *omputed using*
*UMAP (lef*

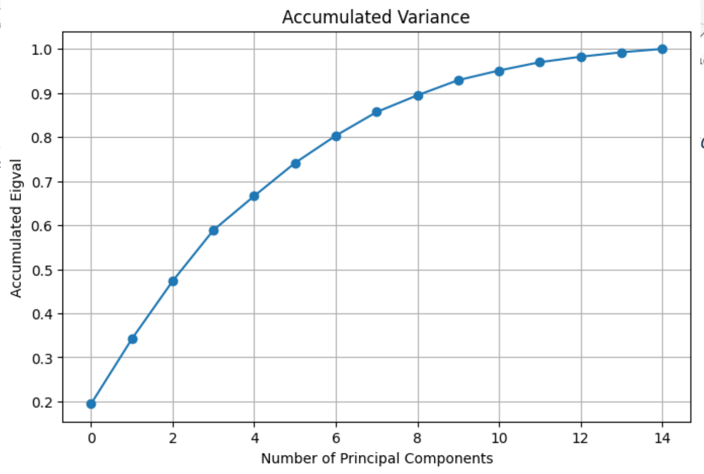


*Figure 3: PCA analysis of latent space (of VAE-based data descriptor). Accumulated eigenvalues of the first 15 eigenvectors*

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 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.

### 2.1.2.2.2. 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.

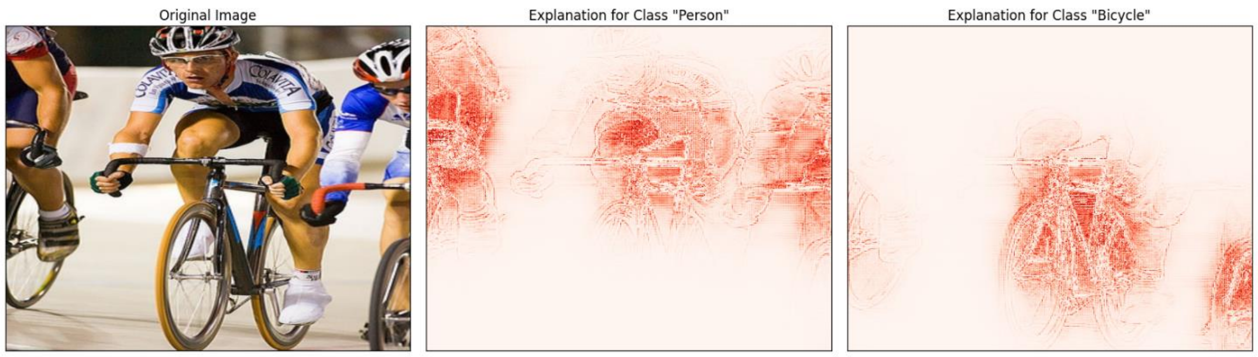

Original Image
Explanation for Class "Person"
Explanation for Class "Bicycle"

*Figure 6: LRP saliency map generated for YOLOv8 (Railway UC)*

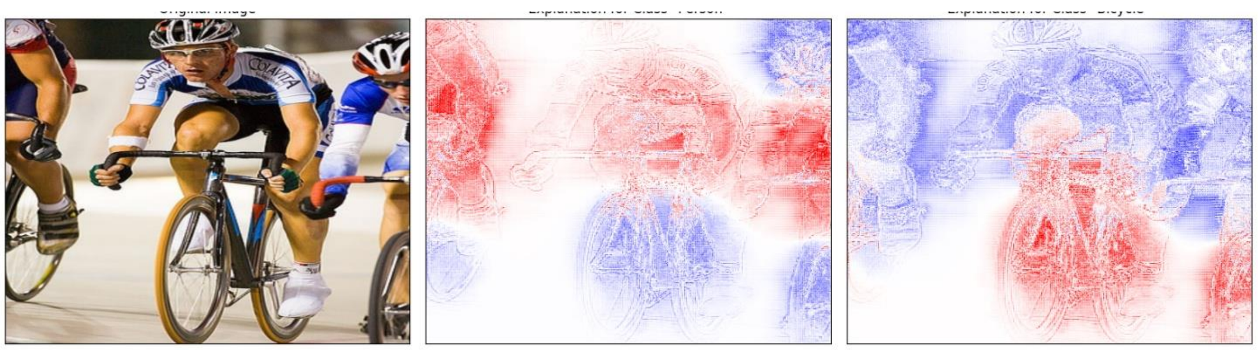


*Figure 5: Contrastive LRP saliency map generated for YOLOv8 (Railway UC)*

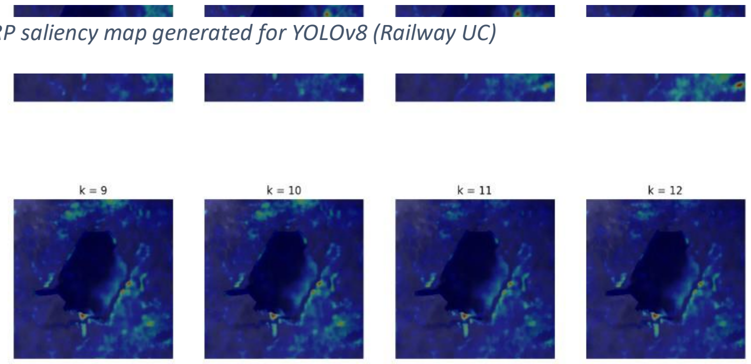


*Figure 4: Experiments of using PCA reconstructed latent vector of data descriptor to identify anomalous area in the input image*

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

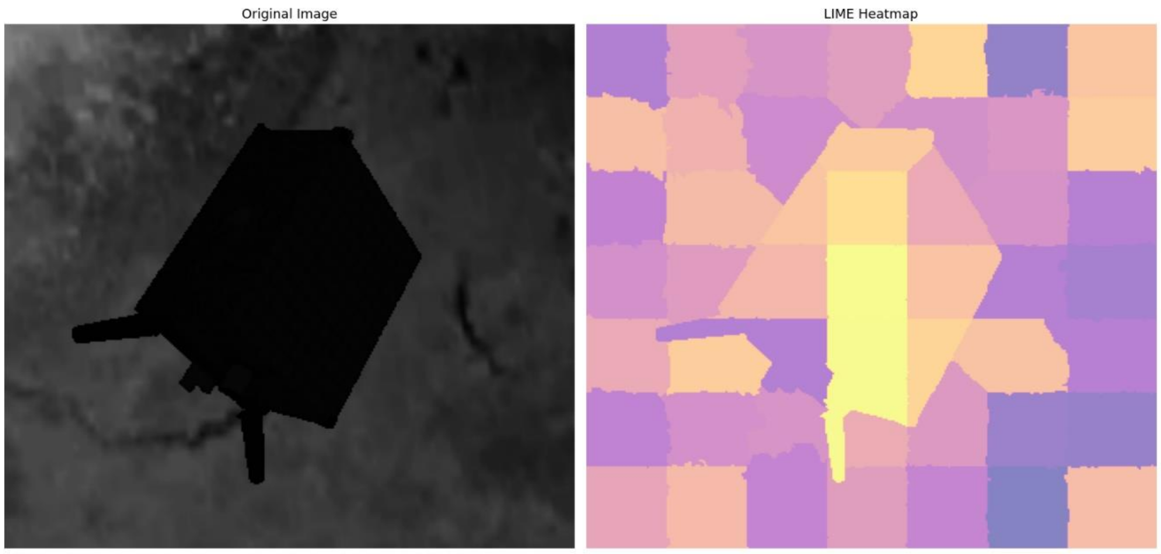


*Figure 7: LIME heatmap computed for MVP model, using superpixel segmentation method Watershed as input for LIME*

### 2.1.2.2.3. 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).

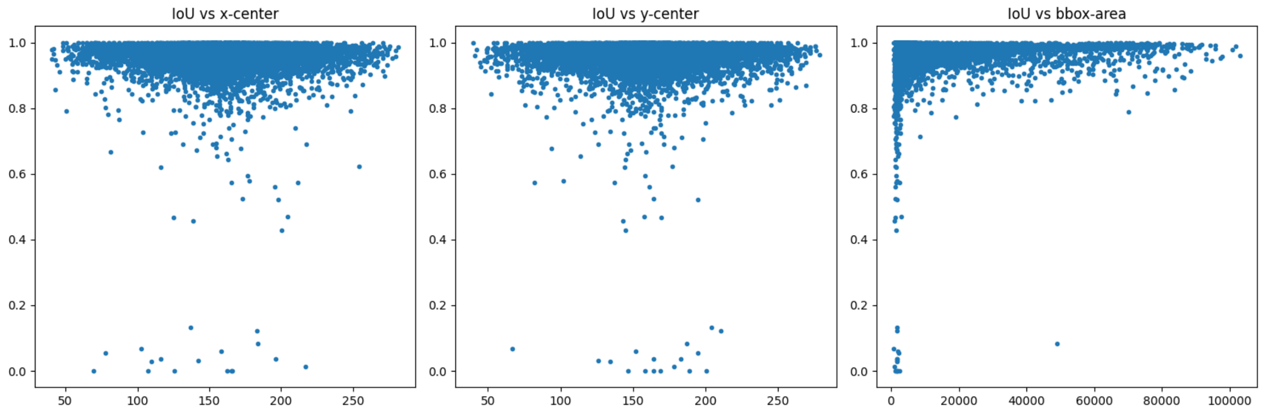
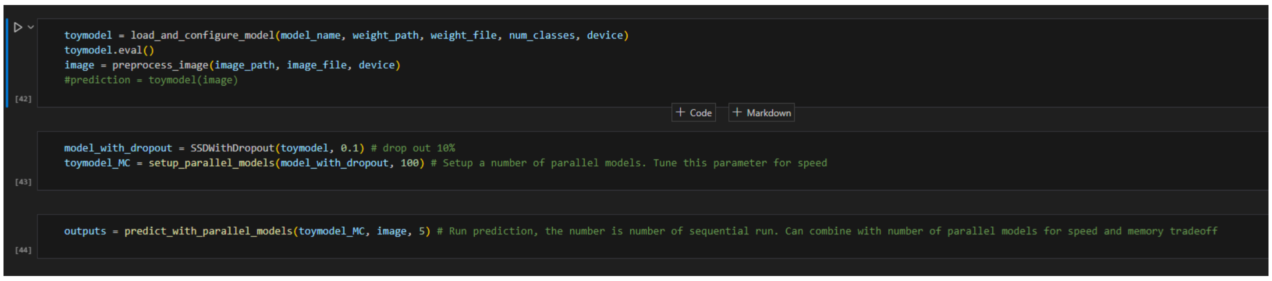


*Figure 8: Model performance (IoU metric) vs input parameters: bounding box location and area*

### 2.1.2.2.4. 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).


```
[42]
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)

[43]
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

[44]
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 9: Example usage of Epistemic uncertainty aware MVP model*

More figures and examples can also be found in Section **Error! No s'ha trobat l'origen de la referència.**.

## 2.1.3. Libraries of XAI tools

The core functionalities, classes, and utilities within EXPLib are organized in the `xai_library` module (Table 3), 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 3: 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
         ├── anomaly_detector
         ├── ensemble_method
         ├── surrogate_model
         └── uncertainty_models
```

### 2.1.4. 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 4):

- **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 4: Structure of dl_component submodule*

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

### 2.1.5. 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.

## 2.2. DLLib

Building on 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, requiring optimizations for hardware architectures like NVIDIA GPUs and Arm-based CPUs. The corresponding source code, including DLLib, has been archived and uploaded to the B2DROP repository maintained by BSC, it can be accessed through the following link: https://b2drop.bsc.es/index.php/s/D9eteoH5fcYPaq2.

### 2.2.1. Library structure

DLLib is structured (Table 5) as follows:

- **SEMDRLIB:** Contains multiple redundant versions of image-based DL models.
- **toy_model_v1**: It's the baseline model, and the folder includes a standalone way of running the base model with most of the supervisor function features enabled.
- **uncertainty_models:** Contains multiple diverse redundant versions of image-based DL models and applies a number of user-selected transformations in input images to perform multiple diverse inferences intended to provide semantically identical, yet not bit-identical, results. Two main approaches are used the first one based on aleatoric uncertainty and the second one on parallel uncertainty.
- **DLETLIB:** Includes several features that provide some degree of explainability to the base AI models allowing to detect possible issues during their execution.
- **anomaly_detection:** 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:** 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.
- **surrogate_model:** A lightweight surrogate object detector that replaces a deep model with handcrafted cues. It extracts Harris keypoint stats and LBP texture matches, summarizes

them via GMM cluster features, and concatenates these descriptors. A trained RandomForest regressor then maps this feature vector to the image’s bounding box.

*Table 5: Structure of DLLib*

```
├── DLLib
│   ├── SEMDRLIB
│   │   ├── toy_model_v1
│   │   └── uncertainty_models
│   ├── DLETLIB
│   │   ├── anomaly_detector
│   │   ├── ensemble
│   │   └── surrogate_model
```

### 2.2.2. Integration with ROS2 (Middleware)

It is worth noting that we have successfully integrated all the VAE based anomaly detectors (input, model, output, Uncertainty Models, Surrogate Models, and Ensemble Methods) into the Middleware, customized to utilize ROS2 for efficient communication between various modules. 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.

All ROS2 implementations of DLLib, are included in the core Demo of the SAFEXPLAIN project.

# 3. Discussions

This document presents the final outcomes of T3.4 and T3.5 within the SAFEXPLAIN project, providing a comprehensive summary of the specification, design, and improvement of dependable deep learning (DL) components.

# 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.