



EMERALD

Deliverable D2.6 ML Model Certification – v1

Editor(s):	Ching-Yu Kao
Responsible Partner:	Fraunhofer Institute for Applied and Integrated Security (FhG AISEC)
Status-Version:	Final – v1.0
Date:	28.10.2024
Type:	OTHER
Distribution level:	PU

Project Number:	101120688
Project Title:	EMERALD

Title of Deliverable:	D2.6 – ML model certification – v1
Due Date of Delivery to the EC	31.10.2024

Workpackage responsible for the Deliverable:	WP2 – Methodology for knowledge extraction
Editor(s):	Ching-Yu Kao (FhG)
Contributor(s):	--
Reviewer(s):	Marinella Petrocchi CNR Cristina Martínez, Juncal Alonso (TECNALIA)
Approved by:	All Partners
Recommended/mandatory readers:	WP1, WP2, WP3, WP4, and WP5

Abstract:	This deliverable presents components for evidence extraction from machine learning models that can be integrated with the certification graph. It is the result of work performed in Task 2.4. This document is a first/interim version, the final version on source evidence extractors will be reported in D2.7
Keyword List:	Knowledge extraction, machine learning, deep learning, robustness, security, technical evidence
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0 DEED https://creativecommons.org/licenses/by-sa/4.0/)
Disclaimer	Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. The European Union cannot be held responsible for them.

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	03.09.2024	First draft version, outline	Ching-Yu Kao (FHG AISEC)
v0.2	07.10.2024	Added contents to AI-SEC	Ching-Yu Kao (FHG AISEC)
v0.3	18.10.2024	Finalization	Ching-Yu Kao (FHG AISEC)
v0.4	26.10.2024	Internal review	Marinella Petrocchi (CNR)
v0.5	28.10.2024	Modification after QA review	Ching-Yu Kao (FHG AISEC)
v0.6	28.10.2024	Final Review	Cristina Martínez/ Juncal Alonso (TECNALIA)
v0.7	29.10.2024	Modifications after final review	Ching-Yu Kao (FHG AISEC)
v1.0	31.10.2024	Submitted to the European Commission	Cristina Martínez/ Juncal Alonso (TECNALIA)

Table of contents

Terms and abbreviations.....	5
Executive Summary	6
1 Introduction.....	7
1.1 About this deliverable	7
1.2 Document structure	7
2 ML model evidence extractors in the EMERALD architecture	8
3 AI-SEC.....	9
3.1 Functional description.....	9
3.2 Technical description	10
3.2.1 Prototype architecture.....	10
3.2.2 Technical specifications	11
3.3 Delivery and usage	12
3.3.1 Package information	12
3.3.2 Installation	12
3.3.3 Instructions for use	13
3.3.4 Example for Running the Tool.....	14
3.3.5 Licensing information	16
3.3.6 Download	16
3.4 Limitations and future work.....	16
4 Conclusions.....	18
5 References.....	19

List of tables

TABLE 1. REQUIREMENT AI-SEC.01 - EXTRACTION OF SECURITY FEATURES FROM ML MODELS	9
TABLE 2. OVERVIEW AND DESCRIPTION OF PACKAGE STRUCTURE FOR THE AI-SEC.....	12
TABLE 3. SETUP FOR THE ML MODEL USING MNIST DATASET	15
TABLE 4. SETUP FOR THE ML MODEL USING CIFAR10 DATASET	15
TABLE 5. RESULTS ON MNIST AND CIFAR10 USING CLEVER SCORE, SHAPR SCORE, DATA POISONING AND LIME.....	15

List of figures

FIGURE 1. EMERALD COMPONENT OVERVIEW DIAGRAM [6]. THE RED RECTANGLE HIGHLIGHTS THE ML MODEL EVIDENCE EXTRACTION COMPONENTS, WHICH ARE DESCRIBED IN THIS DELIVERABLE.	8
FIGURE 2. AI-SEC ARCHITECTURE. TO EXTRACT FEATURES FROM ML MODELS, WE NEED TO EVALUATE THE POISONING LEVEL (ATTACK COMPONENT), ROBUSTNESS (DATA PROCESSOR1), PRIVACY LEVEL (DATA PROCESSOR1) AND EXPLANATIONS (DATA PROCESSOR2).....	10

Terms and abbreviations

AI	Artificial Intelligence
AI-SEC	AI Security Evidence Collector
AMOE	Assessment and Management of Organisational Evidence
API	Application Programming Interface
BSI	Bundesamt für Sicherheit in der Informationstechnik
BSI C4	Artificial Intelligence Cloud Services Compliance Criteria Catalogue
CertGraph	Certification Graph
CIFAR	Canadian Institute For Advanced Research
Codyze	Static Code Analyzer from FHG
CSA or EU CSA	EU Cybersecurity Act
CSP	Cloud Service Provider
CSV	Comma-Separated Values
CLEVER	Cross Lipschitz Extreme Value for nEtnetwork Robustness
DoA	Description of Action
EC	European Commission
eknows	Platform for Software Analysis from SCCH
GA	Grant Agreement to the project
KPI	Key Performance Indicator
MEDINA	Predecessor project of EMERALD
MIT	Massachusetts Institute of Technology
MNIST	Modified National Institute of Standards and Technology database
LIME	Local Interpretable Model-agnostic Explanations
MEDINA	Predecessor project of EMERALD
PNG	Portable Network Graphics
SW	Software
SHAPr	SHapley Additive exPlanations
TOM	Technical and Organisational Measure
TRL	Technology Readiness Level
WP	Work Package

Executive Summary

This deliverable presents the initial design, architecture, and implementation state of the machine learning (ML) model evidence extractors of WP2, we call it *AI-SEC*. They contribute to the key result KR1-EXTRACT of EMERALD, a framework to continuously extract knowledge from well-trained ML models and prepare suitable evidence based on them.

EMERALD follows a knowledge graph-based approach to provide a unified view of the cloud service under certification at different layers of the service, ranging from the infrastructure layer (e.g., virtual resources), to the business layer (e.g., policies and procedures), to the implementation layer (e.g., source code files) and data layer (e.g., increasingly used AI models) in cloud applications.

The ML model evidence extractors, developed in Task 2.4 and described in this deliverable, aim at identifying critical security-related features, such as adversarial robustness, privacy, security and explainable AI. Other related deliverables in WP2, all due at project Month 12 (October 2024), provide functional and technical details on further evidence extractors from different sources, i.e., D2.2 [1] on source code evidence extraction, D2.4 [2] on evidence extraction from policy documents in Task 2.3 and D2.8 [3] on runtime data extraction in Task 2.5. All these details contributed to D2.1 [4] on the overall information model of the certification graph in Task 2.1.

The main part of this deliverable provides functional and technical descriptions of the evidence extractor *AI-SEC*, including its purpose and scope, the (current and planned) coverage of the EMERALD requirements, and the components' internal architecture. These descriptions are complemented by information on delivery and usage, as well as on limitations and future work. Finally, the document concludes with a short summary.

The ML model evidence extractors described in this deliverable contribute to KR1-EXTRACT by providing next-generation evidence gathering tools and techniques based on a knowledge graph approach. The presented extractors currently have the initial prototypes implemented and ready to be (to some degree) integrated with other components of the EMERALD architecture.

Based on the work described in this deliverable, the ML model evidence extractors will be further extended and integrated into the EMERALD framework. This is the first iteration of the deliverable coming from Task 2.4. The second and final version of this deliverable (D2.7 [5]) with the updated extractors will be delivered in project Month 24 (October 2025). Evidence will be prepared according to the integrated, graph-based model of semantically linked and combined evidence.

1 Introduction

EMERALD aims to provide a next generation set of evidence gathering tools and techniques based on a knowledge graph approach. KR1-EXTRACT supports an improved and unified tool-supported approach to continuously extract knowledge from different layers of a cloud service, e.g., infrastructure, platform, runtime information, policy documents, software, and AI models.

The objective of WP2 is to establish a unified view of the cloud service under certification by extracting and enriching knowledge of the different layers of the service and providing suitable evidence for security metrics. A graph-based model, called the certification graph (*CertGraph*), serves as a common structure that is filled by all evidence extraction tools.

1.1 About this deliverable

This deliverable focuses on the design, implementation, and initial evaluation of the tools and techniques that form the backbone of EMERALD's evidence-gathering framework. The deliverable emphasizes the role of *AI-SEC* in creating evidence from the ML models.

1.2 Document structure

The document is structured as follows.

In Section 3 we report on the design and implementation of *AI-SEC* ML model extractor. For the ML model extractor, functional and technical descriptions are provided, including their purpose and scope, the (current and planned) coverage of the EMERALD requirements, the components' internal architecture, their subcomponents, and details about the programming language, libraries, etc. used. These descriptions are complemented by information on delivery and usage, including package information, installation instructions, user manual, licensing and download information, as well as limitations and future work.

2 ML model evidence extractors in the EMERALD architecture

This section describes how the ML model evidence extractors interact with (selected) EMERALD components on a conceptual level. Figure 1 shows the EMERALD high-level architecture as a component diagram, as described in D1.1 [6]. In EMERALD, a component is defined as “any part of the EMERALD ecosystem that has a specific functionality and can be considered a separate entity with respect to other components” (see D1.3 [7]).

The components for collecting evidence about technical and organisational measures, i.e., *AMOE*, *eknows*, *AI-SEC*, *Clouditor-Discovery*, and *Codyze*, are represented at the bottom part of Figure 1. The ML model evidence extractor *AI-SEC*, which obtains technical evidence from the analysis of the ML model of cloud applications, is highlighted using a thick frame. *AI-SEC* is a newly developed component in EMERALD and analyses AI models for several key evidence regarding robustness against adversarial attacks, explainability, and fairness.

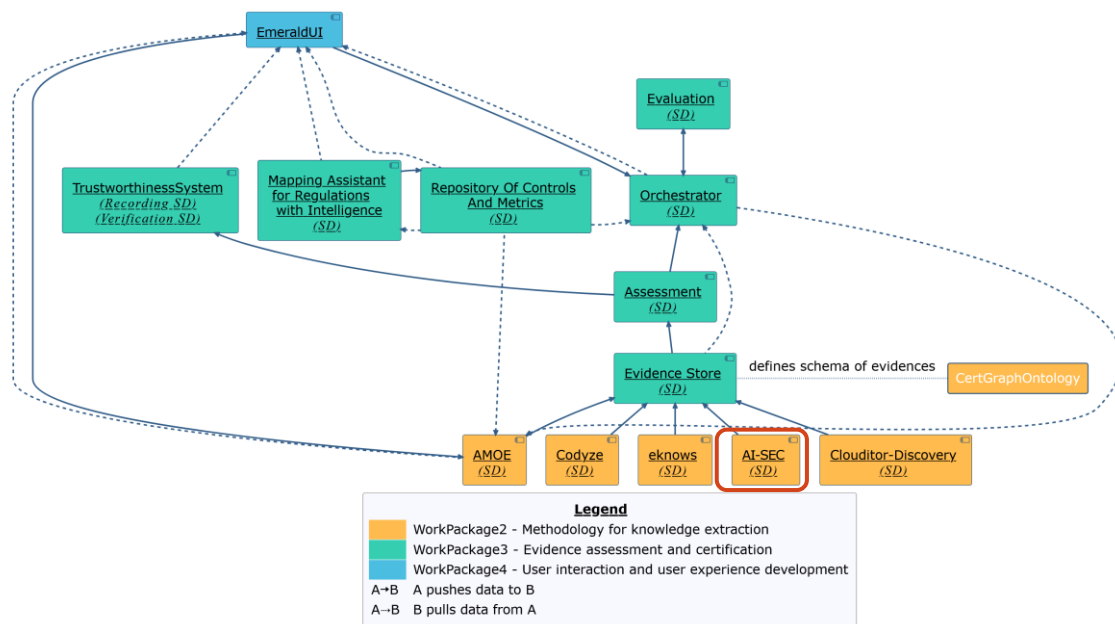


Figure 1. EMERALD component overview diagram [6]. The red rectangle highlights the ML model evidence extraction components, which are described in this deliverable.

3 AI-SEC

In alignment with the requirements defined in Section 6.2, "Security & Robustness Objective," of the BSI Criteria Catalogue C5, *AI-SEC* is designed to meet these critical security criteria to ensure comprehensive protection and compliance. Our solution addresses four key aspects: privacy, adversary resistance, explainability, and data leakage prevention. These elements are fundamental in establishing a robust and secure system capable of withstanding potential threats while maintaining transparency and data integrity.

3.1 Functional description

Overall purpose. The prototype provides a comprehensive toolkit for evaluating and improving the security of machine learning models by focusing on adversarial robustness testing, privacy vulnerability assessment, data poisoning attacks, and model interpretability. It effectively assesses vulnerabilities using techniques such as CLEVER score calculation [8], SHAPr leakage analysis [9], backdoor data poisoning [10], and LIME-based explanations [11].

These features will be collected as evidence for the certification graph.

Context and scope. The toolkit assumes that users already have pre-trained models available, which can be directly utilized for evaluation. Additionally, it supports standard datasets for robustness and privacy assessments, while also allowing custom dataset imports for tailored evaluations.

Motivation. The toolkit aims to streamline the security evaluation of machine learning models by integrating multiple security assessments into a unified system.

Innovation. *AI-SEC* will focus on the following innovations:

- Integration of multiple security assessments, robustness, privacy, and interpretability into a single, unified system.
- Automation of calculations and result generation through command-line inputs

Requirements. The relevant requirements with their respective implementation state (partially / fully / not implemented) and a brief description of how they are / will be implemented are provided in Table 1.

Table 1. Requirement AI-SEC.01 - Extraction of security features from ML models

Field	Description
Requirement ID	AI-SEC.01
Short title	The extractor tool includes defined criteria
Description	The designed AI-SEC has the features based on BSI AIC4
Status	Work in Progress
Priority	Must
Component	AI-SEC
Source	Component, KPI
Type	Technical
Related KR	KR5_AIPOC
Related KPI	KPI 5.1
Validation acceptance criteria	Code review: Review code and check if analysis methods work for different ML models.
Progress	Partially implemented – 35%
Milestone	MS5: Components V2 (M24)

We have already started implementing this requirement. It is currently running locally, but some additional features are needed. More detailed information can be found in the Section 0.

3.2 Technical description

The following subsections describe the technical details of *AI-SEC* for EMERALD.

3.2.1 Prototype architecture

AI-SEC architecture refers to an initial design or framework of a system or software solution that is built to demonstrate its functionality and key components. It serves as a preliminary model to test and validate concepts, allowing developers to explore the system's structure and functionality before finalizing the full implementation. In a prototype architecture, the primary components are defined, and basic functionality is implemented, enabling early-stage evaluations and iterative improvements. It helps identify potential issues and make design adjustments based on testing and user feedback. It contains for main functions:

- **Data Processor:** Prepares datasets and labels for attacks and evaluations.
- **Attack:** Performs data poisoning attacks using the Hidden Trigger Backdoor Attack method [12].
- **Explainability:** Generates explanations of model predictions using LIME [11] .
- **Output:** Calculates robustness (CLEVER score [8]) and privacy vulnerability scores (SHAPr [9]).

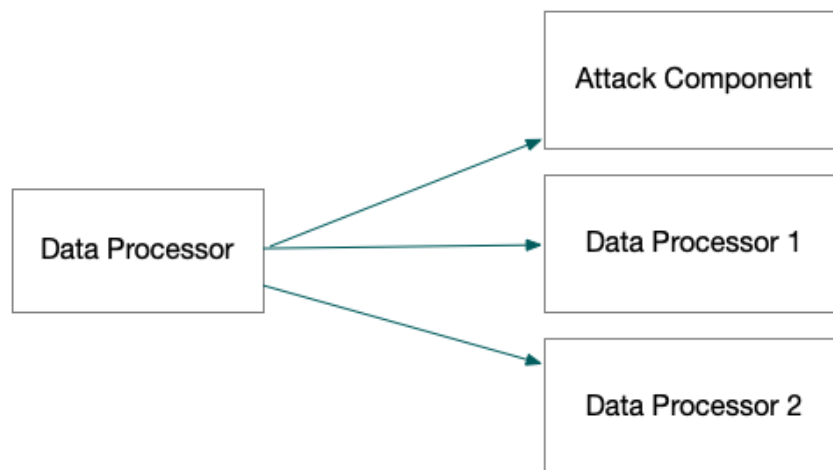


Figure 2. *AI-SEC* architecture. To extract features from ML models, we need to evaluate the poisoning level (Attack component), robustness (data processor1), privacy level (data processor1) and explanations (data processor2)

Based on this architecture, we will obtain the following results, which will serve as evidence for the *Certgraph*.

1 Poisoning Resilience Score

To know if a model is poisoning resilience, we perform data poisoning attacks to evaluate model resilience against malicious examples. The “Hidden Trigger Backdoor Attack Sleeper Agent”¹ [10] is used for this. In this approach, the attacker inserts a *hidden trigger* (such as a specific pattern, object, or symbol) into some of the training data. The model learns to associate this trigger with a particular output. During normal use, the model behaves as expected, but when it encounters this hidden trigger, it produces the attacker’s desired outcome instead. The default class_source is 0 (source class), and class_target is 1 (target

¹ <https://arxiv.org/pdf/2106.08970>

class for misclassification). These values can be modified *in toolbox.py/class_source* and *toolbox.py/class_target*.

2 Adversarial Robustness

CLEVER Score Module: CLEVER Scores Calculation² [8] assesses the robustness of a model by calculating CLEVER scores. It provides an estimate of how resilient a model is to small perturbations in input data that could lead to incorrect or unexpected outputs.

A **higher CLEVER score** means the model is **more robust** or **resistant to adversarial attacks**. In simple terms, it would take a bigger or more noticeable "nudge" to the input data to trick the model into making a wrong decision. So, the higher the CLEVER score, the safer or more reliable the model is when facing small, potentially harmful changes in the data that are meant to confuse it. In contrast, a **lower CLEVER score** means the model is **more vulnerable** to these kinds of attacks, as even a small tweak to the input data could lead it to make mistakes.

The value range depends on the radius size, which is 0 - 5 by default and can be modified in the function *utils/compute_untargeted_clever()*.

3 Privacy Score

SHAPr Leakage Module: Evaluates model privacy using SHAPr leakage metrics³ [9]. The SHAPr Leakage score analyses the SHAP values to identify if private information (like unique patterns in sensitive data) could be reconstructed or inferred. If the SHAP values indicate that individual data points can significantly influence a model's prediction, there may be a risk of leakage. A higher final SHAPr score for a training sample means it is more vulnerable to privacy attacks. The values range from 0 to 1.

4 Explainability

This component leverages the LIME⁴ [11] (Local Interpretable Model-Agnostic Explanations) library to explain individual predictions made by the model. LIME works by generating perturbed versions of an input and observing how the model's predictions change, effectively creating a local approximation of the model's decision boundary around the input. This provides insight into which features in the inputs contribute most to a particular decision and helps the user understand the behaviour of the model.

3.2.2 Technical specifications

Programming Language:

- Python 3.7

Libraries:

- Torch⁵ (version 1.13.1) for model processing and evaluation.
- Torchvision⁶ (version 0.14.1) for image data handling.
- Lime⁷ (version 0.2.0.1) for model interpretability.

² <https://openreview.net/pdf?id=BkUHIMZ0b>

³ <https://arxiv.org/abs/2112.02230>

⁴ <https://github.com/marcotcr/lime>

⁵ <https://pytorch.org/>

⁶ <https://pytorch.org/vision/stable/index.html>

⁷ <https://github.com/marcotcr/lime>

Databases:

- Datasets are stored locally in compatible formats such as CSV for labels and PNG for images.

3.3 Delivery and usage

The following subsections detail the delivery and usage of *AI-SEC* for EMERALD. The provided information is currently work in progress and may change.

3.3.1 Package information

AI-SEC provides a comprehensive solution for evaluating model robustness, performing adversarial attacks, assessing privacy risks, and explaining model predictions using techniques like LIME. The package includes scripts, configurations, and resources needed for various deep learning model operations as depicted in Table 2.

Table 2. Overview and description of package structure for the *AI-SEC*

Folder	Description
art/	Imported external library
explained_imgs/	Stores result of explained images
models/	Stores the trained model files
models/my_model.pth	The model file, provided by the user (in .pth format)
trigger/	Folder of trigger images used for performing hidden trigger
backdoor_attacks/	Folder to upload image files uploaded by the user, used for LIME model explanation, etc.
toolbox.py	Main script file to execute various operations (e.g., CLEVER score calculation, privacy assessment, data poisoning)
model.py	Script file for neural network architecture
mydata.py	Script file for import custom datasets
config.yaml	Configuration file that specifies model paths, trigger paths, and other settings
mydata/	Folder to store custom data
mydata/labels.csv	CSV Label File
mydata/images/images_upload/class_0	Image folder of custom datasets
utils.py	Utility functions directory containing helper scripts
README.md	Instructions to build and use the extractor

3.3.2 Installation

In this section, we provide the steps and guidelines to help users set up and install *AI-SEC*. These instructions are crucial for ensuring that the component is correctly configured and operational on a user's system.

1. **Install Required Libraries:** Run the following commands to install the necessary libraries:

```
pip install torch==1.13.1
pip install torchvision==0.14.1
pip install lime==0.2.0.1
```
2. **Modify the Configuration File (config.yaml):**

- Set the `model_path` to the location of your trained model checkpoint.
 - Set the `trigger_path` to the location of the trigger image used for data poisoning.
3. **Define Your Neural Network Architecture:**
 - Implement the `Net` class in `model.py` to define your custom neural network.
 4. **Store Model Files:**
 - Place your trained model files (e.g., `my_model.pth`) in the `models/` directory.
 5. **Prepare Images for LIME Explanations:**
 - Put the images you want to process into the `images_upload/class_0/` directory.
 6. **Prepare Trigger Image for Data Poisoning:**
 - Place the trigger image in the `trigger/` directory.
 7. **Setup for Custom Datasets:**
 - **CSV Label File (labels.csv):** This file should have two columns: the first for the image file name and the second for the corresponding label. Example format:
`image1.png, 0`
 - **Image Files:** Ensure all image files are stored in the image folder, with file names matching the entries in the CSV file.

By following these steps, you can properly set up your environment for model training, data processing, and analysis.

3.3.3 Instructions for use

This section provides instructions on how to use the tool for calculating robustness scores, assessing privacy, performing data poisoning, and explaining model predictions.

1. Calculate CLEVER Scores

To evaluate the robustness of your model using CLEVER scores, follow these steps:

```
$ python your_script.py -d <dataset> -t robustness -c <nb_classes>
```

- `<dataset>`: Specify the dataset you are using (e.g., "cifar10", "mnist", "mydata").
- `<nb_classes>`: Replace with the total number of classes in your dataset.

2. Assess Privacy (SHAPr Leakage)

To evaluate the privacy of the model using SHAPr leakage analysis, use the following command:

```
$ python your_script.py -d <dataset> -t privacy -c <nb_classes>
```

- `<dataset>`: Indicate your dataset.
- `<nb_classes>`: Specify the number of classes in the dataset.

3. Perform Data Poisoning

To generate poisoned data and evaluate the impact of a data poisoning attack, execute the following commands:

- To generate poisoned data and evaluate the attack:

```
$ python your_script.py -d <dataset> -t poison -c <nb_classes> -s
<patch_size> -test
```

- To generate poisoned data only:

```
$ python your_script.py -d <dataset> -t poison -c <nb_classes> -s
<patch_size>
```

- <dataset>: Specify your dataset.
- <nb_classes>: Replace with the number of classes in your dataset.
- <patch_size>: Define the size of the patch for the poisoned data.

4. Explain Model Predictions Using LIME

To generate explanations for model predictions with LIME, run the following command:

```
$ python your_script.py -d <dataset> -t explain -c <nb_classes> -ch
<num_channels>
```

- <dataset>: Specify your dataset.
- <nb_classes>: Replace with the number of classes in the dataset.
- <num_channels>: Set the number of channels in your input images (e.g., 1 for grayscale, 3 for RGB).

Example for using CIFAR-10 dataset⁸:

Here are examples of using the commands with the CIFAR-10 dataset:

1. Calculate CLEVER Scores:

```
$ python toolbox.py -d cifar10 -t robustness -c 10
```

2. Assess Privacy:

```
$ python toolbox.py -d cifar10 -t privacy -c 10
```

3. Perform Data Poisoning:

```
$ python toolbox.py -d cifar10 -t poison -c 10 -s 8 -test
```

4. Explain Model Predictions:

```
$ python toolbox.py -d cifar10 -t explain -c 10 -ch 3
```

Notes:

- Replace placeholders (<dataset>, <nb_classes>, <patch_size>, <num_channels>) with the actual values based on your data and model requirements.
- All commands should be executed in the terminal or command line interface where your Python environment is set up.

3.3.4 Example for Running the Tool

Data: The datasets used in this example are MNIST⁹ and CIFAR-10¹⁰. The MNIST dataset consists of 28×28 pixel grayscale images with 10 classes. The CIFAR-10 dataset comprises 32×32 pixel RGB colour images with 10 classes.

⁸ <https://www.cs.toronto.edu/~kriz/cifar.html>

⁹ <https://yann.lecun.com/exdb/mnist/>

¹⁰ <https://www.cs.toronto.edu/~kriz/cifar.html>

Model:

We trained the model using two datasets, namely the model for MNIST (see Table 3) and the model for CIFAR10 (see Table 4).

Table 3. Setup for the ML model using MNIST dataset

For MNIST	Layer	Description
1	Conv2D + ReLU	32 Filters (3×3), stride 1
2	Conv2D + ReLU	64 Filters (3×3), stride 1
3	Max Pooling	2×2 pooling
4	Flatten	Flatten the tensor
5	Dense (Fully Connected) + ReLU	128 Units
6	Dense (Fully Connected)	10 Units (Classes)

Table 4. Setup for the ML model using CIFAR10 dataset

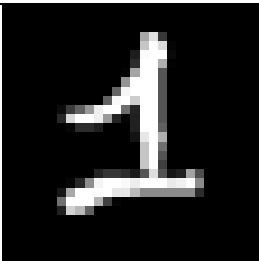

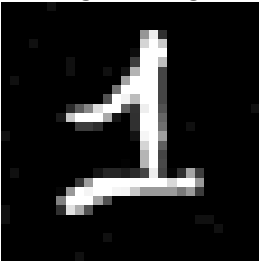

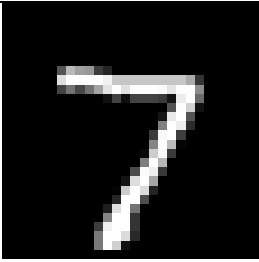
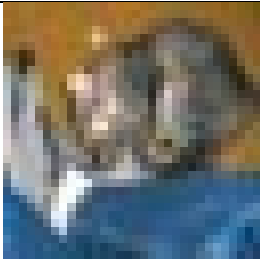
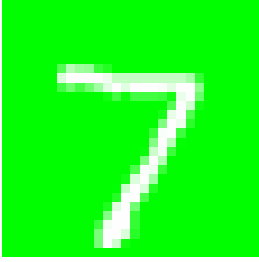
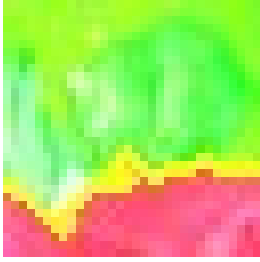
For CIFAR-10	Layer	Description
1	Conv2D + ReLU	6 Filters (5×5), stride 1, 3
2	Max Pooling	2×2 pooling
3	Conv2D + ReLU	16 Filters (5×5), stride 1
4	Max Pooling	2×2 pooling
5	Flatten	Flatten the tensor
6	Dense (Fully Connected) + ReLU	120 Units
7	Dense (Fully Connected) + ReLU	84 Units
8	Dense (Fully Connected)	10 Units (Classes)

Result:

Table 5 shows the results of our basic testing on the two models. The CLEVER scores for the two models are 3.76 and 2.81, respectively. It means that compared to CiFAR10 model, the MNIST model is more robust against adversarial examples. The SHAPr scores are 0.98 and 0.27, for the privacy leakage analysis, which indicates that the MNIST model is more privacy preserved. For data poisoning, we provide images with the necessary pixel modifications, and when collecting evidence, the amount of pixel change will be recorded as evidence. For LIME, we present the model's explanation results, and in the evidence collection, we provide binary results (0 indicates an unreasonable explanation, while 1 indicates a reasonable explanation).

Table 5. Results on MNIST and CIFAR10 using CLEVER score, SHAPr score, data poisoning and LIME.

	Result for the model (MNIST)	Result for the model (CIFAR-10)	Note
CLEVER Scores	3.76/5	2.81/5	Calculations were made using 50 samples from the dataset
SHAPr Score	0.98/1	0.27/1	The computation time is long, it takes 10000 iterations.

Data Poisoning	 original image	 original image	
	 poisoned image	 poisoned image	
LIME-based explanations	 original image	 original image	
	 model prediction	 model prediction	

3.3.5 Licensing information

Since LIME and SHAPr use permissive MIT licenses, we choose to license this tool under the MIT License.

3.3.6 Download

The currently implemented parts are stored on EMERALD's Gitlab¹¹.

3.4 Limitations and future work

The current tests using accessible models have shown reasonable results, demonstrating the *AI-SEC* potential. However, a significant limitation is that many cloud services do not grant direct access to their models, which poses a challenge for comprehensive evaluation. To address this, a potential solution is to use a proxy model as a substitute for the cloud-based model. While promising, further experimentation is required to evaluate the effectiveness of the proxy model in accurately reflecting the behaviour of the original cloud model. Additionally, optimization of

¹¹ <https://git.code.tecnalia.com/emerald/public/components/ai-sec>

the tool is necessary to enhance its efficiency and ensure it can run smoothly in various environments.

Future work will focus on refining the *AI-SEC* performance and exploring alternative methods for model evaluation in scenarios where direct access is restricted. These efforts will help improve the adaptability and robustness of *AI-SEC* across different use cases. Future activities will also cover the integration of the *AI-SEC* component in the EMERALD CaaS framework as evidence extractor tool and with the *EMERALD UI*. All these changes will be reported in the subsequent version of this deliverable, namely, D2.7 [5] in project month M24.

4 Conclusions

In this deliverable, as an initial output of Task 2.4, we presented the design, architecture, and current implementation status of the EMERALD model evidence extraction components. These components follow the holistic approach of the EMERALD framework and are aligned with the technical requirements gathered in WP1 (D1.3 [12]). The report outlines the relationship between the presented component, *AI-SEC* and other parts of the EMERALD framework, detailing the internal structure of the component, its subcomponents, and relevant information about its technical implementation.

The component introduced in the report, *AI-SEC*, supports evidence extraction for machine learning models. At the current stage of the project, this component, based on preliminary work, has a working prototype that can (partially) integrate with other EMERALD components and has been tested using accessible ML models. Future work will involve testing this tool with more complex models.

The subsequent and final iteration of this report (D2.7 [5]), which will provide updates on the progress of the component, is planned for project month 24.

5 References

- [1] EMERALD Consortium, “D2.2 Source Evidence Extractor–v1,” 2024.
- [2] EMERALD Consortium, “D2.4 AMOE – v1: Evidence extraction from policy documents that can be integrated with the certification graph,” 2024.
- [3] EMERALD Consortium, “D2.8 Runtime evidence extractor – v1: Evidence extraction from runtime data that can be integrated with the certification graph,” 2024.
- [4] EMERALD Consortium, “D2.1 Graph Ontology for Evidence Storage: Description of a uniform schema for storing and linking heterogenous data,” 2024.
- [5] EMERALD Consortium, “D2.7 ML model certification–v2”.
- [6] EMERALD Consortium, “D1.1 Data modelling and interaction mechanisms - v1,” 2024.
- [7] EMERALD Consortium, “EMERALD Glossary in D1.3- EMERALD solution architecture - v1,” 2024.
- [8] T.-W. Weng, H. Zhang, P.-Y. Chen, Y. Jinfeng, D. Su, Y. Gao, C.-J. Hsieh and L. Daniel, “Evaluating the robustness of neural networks: An extreme value theory approach,” *arXiv preprint*, 2018.
- [9] V. Duddu, S. Szyller and N. Asokan, “Shapr: An efficient and versatile membership privacy risk metric for machine learning,” *arXiv preprint*, 2021.
- [10] H. Souri, L. Fowl, R. Chellappa, M. Goldblum and T. Golstein, “Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch,” *Advances in Neural Information Processing Systems* 35, 2022.
- [11] M. T. Ribeiro, S. Singh and C. Guestrin, ““Why should I trust you?” Explaining the predictions of any classifier,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135-1144, 2016.
- [12] EMERALD Consortium, “D1.3 EMERALD solution architecture - v1,” 2024.
- [13] A. Saha, A. Subramanya and H. Pirsiavash, “Hidden trigger backdoor attacks,” in *Proceedings of the AAAI conference on artificial intelligence*, , vol. 34, no. 07, pp. 11957-11965, 2020.