

Project Acronym: STAR
Grant Agreement number: 956573 (H2020-ICT-2020-1 – Research and Innovation Action)
Project Full Title: Safe and Trusted Human Centric Artificial Intelligence in Future Manufacturing Lines
Project Coordinator: INTRASOFT International



Funded by the Horizon 2020
Framework Programme of the
European Union

DELIVERABLE

D3.3 – Cyber-Defence Mechanisms against Poisoning and Evasion Attacks - Initial Version

Dissemination level	PU -Public
Type of Document	Report
Contractual date of delivery	31/03/2022
Deliverable Leader	UBI
Status - version, date	Final - V1.0, 26/04/2022
WP / Task responsible	WP3 / T3.2, T3.3
Keywords:	Poisoning attacks, Evasion attacks, Adversarial AI, Adversarial training

This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 956573. It is the property of the STAR consortium and shall not be distributed or reproduced without the formal approval of the STAR Management Committee. The content of this report reflects only the authors' view. The European Commission is not responsible for any use that may be made of the information it contains.

Executive Summary

This deliverable summarizes the work performed in the context of T3.2 and T3.3 for the design, development and evaluation of methodologies for detecting poisoning and evasion attacks. In this context, the deliverable provides the initial version of the AI Cyber Defence module as a main function of the AI security and data protection layer which aims to boost the safety, reliability and transparency of the functionalities of the upper operational layers of STAR.

Thus, this deliverable performed an analysis of state-of-the-art, focusing on Adversarial AI tools, libraries and methodologies in order to form a solid basis for the sections that follow. The deliverable discusses the positioning of the module in the STAR AI security and data protection layer, as well as the internal architecture of the module along with the defined interactions that provide an overview of its mode of operation. In addition, the integration of the AI Cyber Defence module with the Runtime Monitoring System, the STAR Blockchain infrastructure and the XAI libraries of STAR is highlighted.

Apart from the description of the system per se, a core contribution of the deliverable is the conduction of several experiments on the applicability of diverse AI attack and defence strategies, utilizing data stemming from the STAR pilot environments. The deliverable explores the challenges of designing an AI defence mechanism and sheds light on the applicability of the adversarial training approach.

In addition, the deliverable reports on a prototype, called FSD, as an outcome of the research actions performed so far in the context of T3.2 and T3.3. FSD paves the way also for the long-term research vision of UBITECH for the development of the AI Cyber Defence module, having an eye to the 2nd, and final, release of the module, which will be reported in D3.4.

Deliverable Leader:	Dimitris Papamartzivanos (UBI)
Contributors:	Thanassis Giannetsos, Dimitris Karras, Theodora Anastasiou, Sofia Karagiorgou, Entos Veliou (UBITECH) George Makridis, Ioannis Makridis, Dimosthenis Kyriazis, Spyros Theodoropoulos (UPRC) Nikos Kefalakis, Angela Maria Despotopoulou, John Soldatos (INTRA)
Reviewers:	Jože Rožanec (JSI) Mihail Fontul (IBER)
Approved by:	Ipektsidis Babis (INTRA)

Document History			
Version	Date	Contributor(s)	Description
V0.1	15/02/2022	UBI	Table of contents
V0.2	25/02/2022	UBI	Background and state of the art
V0.3	10/03/2022	UBI	Cyber defence module architecture
V0.4	25/03/2022	INTRA, UPRC	Contributions of partners in Section 3.4
V0.5	27/03/2022	UBI	Experimental setup
V0.7	01/04/2022	UBI	Section 5, towards enhanced data trustworthiness
V0.9	13/04/2022	UBI	Final Refinements
V0.95	18/04/2022	JSI, IBER	Quality Review
V1.0	26/04/2022	UBI, INTRA	Final version and submission

Table of Contents

EXECUTIVE SUMMARY	2
TABLE OF CONTENTS.....	4
TABLE OF FIGURES.....	6
LIST OF TABLES.....	7
DEFINITIONS, ACRONYMS AND ABBREVIATIONS	8
1 INTRODUCTION.....	9
1.1 SCOPE AND PURPOSE	9
1.2 RELATION TO OTHER WPs AND DELIVERABLES	9
1.3 METHODOLOGY.....	10
1.4 DELIVERABLE STRUCTURE	11
2 BACKGROUND AND STATE OF THE ART	12
2.1 AI TOOLS AND LIBRARIES USED IN THE EVALUATION METHODOLOGY	12
2.1.1 <i>Adversarial Robustness Toolbox (ART)</i>	12
2.1.2 <i>Cleverhans</i>	13
2.2 RESEARCH ENDEAVOURS IN THE FIELD	15
2.3 ADVERSARIAL AI IN MANUFACTURING	16
2.4 ATLAS MITRE.....	18
3 STAR AI CYBER DEFENCE MODULE	22
3.1 POSITIONING IN THE STAR ARCHITECTURE	22
3.2 AI CYBER DEFENCE MODULE INTERNAL ARCHITECTURE & INTERACTIONS.....	23
3.2.1 <i>Data Input Sources</i>	23
3.2.2 <i>Data pre-processing and normalisation</i>	24
3.2.3 <i>Adversarial Tactics</i>	25
3.2.4 <i>Configuration Manager</i>	25
3.2.5 <i>Discriminator</i>	25
3.2.6 <i>Alert Data Output</i>	26
3.3 FUNCTIONAL REQUIREMENTS FOR AI CYBER DEFENCE MODULE	27
3.4 INTERACTIONS WITH OTHER STAR TOOLS	30
3.4.1 <i>Runtime monitoring system</i>	30
3.4.2 <i>STAR Blockchain network</i>	31
3.4.3 <i>Explainable AI (XAI)</i>	31
4 EXPERIMENTAL SETUP	36
4.1 TESTBED SETUP	36
4.2 DATASETS.....	36
4.2.1 <i>Decocap dataset</i>	37
4.2.2 <i>Shavershell dataset</i>	38
4.3 PRE-PROCESSING	39
4.4 BASE MODEL TRAINING AND EVALUATION	39
4.5 CREATING ADVERSARIAL EXAMPLES USING ATTACKS FROM ADVERSARIAL ROBUSTNESS TOOLBOX	40
4.6 DEFENCE STRATEGIES.....	42
4.6.1 <i>Adversarial Training</i>	42
4.6.2 <i>ART Defence strategies</i>	42
4.7 EVALUATION RESULTS.....	43
4.7.1 <i>Binary classification approach</i>	43
4.7.2 <i>Multi-classed classification approach</i>	45

4.7.3	<i>Discriminator</i>	47
4.7.4	<i>Summary</i>	49
5	TOWARDS ENHANCED DATA TRUSTWORTHINESS FOR AI SYSTEMS	50
5.1	INTRODUCTION.....	50
5.2	CONCEPTUAL ARCHITECTURE	51
5.2.1	<i>Training phase</i>	51
5.2.2	<i>Testing phase</i>	52
5.3	EXPERIMENTAL SETUP.....	53
5.4	EVALUATION	55
5.4.1	<i>Strategy 1: Impact of Adversaries in Pre-training</i>	55
5.4.2	<i>Strategy 2: Impact of Adversaries in Post-training</i>	56
5.5	SUMMARY	57
6	DISCUSSION AND CONCLUSION	58
	REFERENCES	59

Table of Figures

FIGURE 1 STAR AI SECURITY AND DATA PROTECTION LAYER ARCHITECTURE (WP3)23

FIGURE 2 AI CYBER DEFENCE MODULE ARCHITECTURE23

FIGURE 3 AI CYBER DEFENCE ALERT DATAMODEL26

FIGURE 4 AI CYBER DEFENCE ALERT OUTPUT EXAMPLE27

FIGURE 5 DATA PARTITIONING IN BATCHES FOR LOCAL EXPLANATION TESTING33

FIGURE 6 PSEUDO-ALGORITHM OF THE XAI METHOD OF STAR34

FIGURE 7 SHAP RESULTS ON IMAGE DATASET35

FIGURE 8 EXPERIMENTAL TESTBED FLOW36

FIGURE 9 DECOCAP DATASET SAMPLE37

FIGURE 10 DECOCAP IMAGE SAMPLES AFTER APPLYING HOG METHOD38

FIGURE 11 SHAVERSHELL DATASET IMAGE SAMPLES38

FIGURE 12 OTSU'S THRESHOLDING TECHNIQUE RESULT ON SHAVERSHELL IMAGE SAMPLE39

FIGURE 13 EXAMPLE OF ADVERSARIAL EXAMPLES FOR DECOCAP DATASET41

FIGURE 14 EXAMPLE OF ADVERSARIAL EXAMPLES FOR SHAVERSHELL DATASET41

FIGURE 15 APPLICATION OF DEFENCE METHODS ON THE DECOCAP DATASET42

FIGURE 16 APPLICATION OF THE DEFENCE METHODS ON THE SHAVERSHELL DATASET.....43

FIGURE 17 COLLECTIVE RESULTS USING ADVERSARIAL TRAINING FOR THE SHAVERSHELL DATASET IN BINARY CLASSIFICATION MODE44

FIGURE 18 COLLECTIVE RESULTS USING ADVERSARIAL TRAINING FOR THE DECOCAP DATASET IN BINARY CLASSIFICATION MODE44

FIGURE 19 COLLECTIVE RESULTS USING ART DEFENCE STRATEGIES FOR THE SHAVERSHELL DATASET IN BINARY CLASSIFICATION MODE45

FIGURE 20 COLLECTIVE RESULTS USING ART DEFENCE STRATEGIES FOR THE DECOCAP DATASET IN BINARY CLASSIFICATION MODE45

FIGURE 21 COLLECTIVE RESULTS USING ADVERSARIAL TRAINING STRATEGIES FOR THE SHAVERSHELL DATASET IN MULTI-CLASSED CLASSIFICATION MODE.....46

FIGURE 22 COLLECTIVE RESULTS USING ADVERSARIAL TRAINING STRATEGIES FOR THE DECOCAP DATASET IN MULTI-CLASSED CLASSIFICATION MODE46

FIGURE 23 COLLECTIVE RESULTS USING ART DEFENCES STRATEGIES FOR THE SHAVERSHELL DATASET IN MULTI-CLASSED CLASSIFICATION MODE47

FIGURE 24 COLLECTIVE RESULTS USING ART DEFENCES STRATEGIES FOR THE DECOCAP DATASET IN MULTI-CLASSED CLASSIFICATION MODE47

FIGURE 25 COLLECTIVE RESULTS USING THE DISCRIMINATOR APPROACH FOR THE DECOCAP DATASET IN MULTI-CLASSED CLASSIFICATION MODE48

FIGURE 26 COLLECTIVE RESULTS USING THE DISCRIMINATOR APPROACH FOR THE SHAVERSHELL DATASET IN MULTI-CLASSED CLASSIFICATION MODE49

FIGURE 27 TRAINING PHASE OF FSD52

FIGURE 28 TESTING PHASE OF FSD.....53

FIGURE 29 DISTRIBUTIONS OF LEGITIMATE SAMPLES (CLASS 0) VS. ADVERSARIAL SAMPLES (CLASS 1) FOR TWO FEATURES WITH $\mu = 0$ AND $\sigma = 1.0$; (A) ATTACK CASE I, AND (B) ATTACK CASE II.54

List of Tables

TABLE 1: MITRE ATLAS ATTACK PROGRESSION	19
TABLE 2 MEAN (μ) AND STANDARD DEVIATION (σ) OF BASE DATASET	53
TABLE 3 COMPARING RMSES ACHIEVED BY THE BEST TRAINED MLP MODEL, ONE LAYER LSTM, AND BEST TRAINED LSTMS.....	54
TABLE 4 AN OVERVIEW OF THE F-MEASURES OF FSD UNDER DIFFERENT ATTACK CASES III, IV AND V IN PRE-TRAINING	56
TABLE 5 AN OVERVIEW OF THE F-MEASURES OF FSD UNDER DIFFERENT ATTACK CASES III, IV AND V IN POST-TRAINING	57

Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
AI	Artificial intelligence
ART	Adversarial Robustness Toolbox
ATLAS	Adversarial Threat Landscape for Artificial-Intelligence Systems
CPS	Cyber-Physical Systems
DNN	Deep Neural Networks
FGSM	Fast Gradient Sign Method
IDS	Intrusion Detection System
JSMA	Jacobian-based Saliency Map Approach
MCS	Mobile Crowd Sensing
ML	Machine Learning
NLP	Neuro-Linguistic Programming
SoTA	State-of-the-art
WP	Work Package
XAI	Explainable Artificial Intelligence

1 Introduction

1.1 Scope and purpose

The aim of this deliverable is to document the research performed on defence strategies against data poisoning and evasion attacks, as the outcome of the actions performed in T3.2 and T3.3, respectively.

More specifically, the deliverable reports on the consortium member actions for the development of methods for detecting attacks that attempt to train the deep neural networks in ways that may compromise their correct operation, and on the developments of defence mechanisms against evasion attacks, i.e., attacks at the inference stage of a deep neural network, where malicious parties craft data that are incorrectly classified by the deep NN-based systems causing significant issues to their operation.

In this context, this deliverable analyses the state-of-the-art (SoTA) on adversarial AI tools and methods that can be used for the development of the AI Cyber Defence tool of STAR. Based on this, the deliverable documents the placement of the AI Cyber Defence tool in the overall architecture of STAR and elaborates how the detection services will be offered in STAR in order to protect AI-enabled manufacturing processes in the STAR pilots' environments.

As the AI Cyber defence tool will work in synergy with other components of STAR, this deliverable discusses the interactions that have been identified among the respective technology provider of STAR. More specifically, the AI Cyber Defence tool will combine data from the tracking and provenance mechanisms implemented in T3.1, along with explainable AI (XAI) (from WP4/T4.1) in order to identify potentially fraudulent data sources that will have an impact on the explainability of the learned models.

In the context of this deliverable, which is the initial version of the Cyber Defence mechanisms against poisoning and evasion attacks, we document the experimental results of an evaluation performed by using state-of-the-art adversarial techniques and defences. At this stage of the project, we aim to document the challenges towards offering the AI Cyber Defence service of STAR, by making a first attempt to apply the aforementioned techniques to the available data of the pilot sites. This evaluation will help us identify the advantages and disadvantages of the state-of-the-art approaches, so that to retrofit our research and come up with more robust adversarial detection methods by the end of the project.

Towards this goal, the deliverable concludes by documenting our long-term research plan for enhancing the data trustworthiness of AI systems.

1.2 Relation to other WPs and Deliverables

The deliverable is closely related to the following STAR deliverables:

D2.1 Requirements Analysis and State-of-Art Research: The specifications of the STAR AI Cyber Defence tool which are documented in this deliverable consider the functional requirements and specifications provided in deliverable D2.1. Those are analysed in section 3.3.

D2.2 Reference Scenarios and Use Cases for AI in Manufacturing: The reference scenarios of D2.2 are considered in order to guarantee that the development of the AI Cyber Defence tool would meet the expectation and the needs of the STAR pilots.

D2.4 Data Models and Data Collection-Initial version: The data models and the datasets that are available to the technical partners of STAR to develop the AI-enabled solution of STAR have been documented in D2.4. Thus, this deliverable exploits the dataset collected in D2.4 in order to feed the experiments on the evaluation of the adversarial and defence strategies.

D3.1 Decentralized Reliability for Industrial Data and Distributed Analytics-Initial version: This deliverable documents the blockchain-based service of STAR that aim to guarantee the trustworthiness of data and algorithms configuration. The AI cyber defence tool interfaces with the aforementioned service in order to ensure that the configurations of the AI cyber defence algorithms are set as expected.

D4.1 Library of XAI algorithms-Initial version: This deliverable documented the methodology used in WP4 for the development of the XAI libraries. These libraries will be used for the analysis of data, and they are destined to contribute to the processes of the AI Cyber Defence tool do the detection of poisoning attacks. The integration path between the aforementioned STAR components is given in Section 3.4.3.

D3.4 Cyber-Defence Mechanisms against Poisoning and Evasion Attacks-Final version: This deliverable will be the 2nd and final release of D3.3. It is expected to document the final version of the AI Cyber Defence tool and elaborate on the improvements made to the detection algorithms prior to the deployment to the pilots' environments.

D3.3 provides the initial version of the AI Cyber defence tool of STAR. This component is part of the overall WP3 architecture aiming to provide the Security and Data Governance infrastructure of STAR for AI Systems in Manufacturing, which will be documented in D3.5. Thus, D3.3 will be used as an input in D3.5, in order to define the placement and scope of the AI Cyber Defence tool in the WP3 architecture.

1.3 Methodology

UBITECH (UBI), as the leader of T3.2 and T3.3, is responsible for the delivery of the AI Cyber Defence tool of STAR, aiming to the detection of Poisoning and Evasion attacks against AI systems of STAR. Towards this goal, UBITECH have defined a roadmap consisting of the following steps:

- 1) "Know your Data": Acquire, analyse, and experiment with the data available in STAR by the pilot partners.
- 2) SoTA analysis and identification of the threats (poisoning and evasion attacks) to be examined in the context of STAR.
- 3) Use SoTA techniques to create attack-defence scenarios using well-established tools in the literature (e.g., Adversarial Robustness Toolbox)
- 4) Development of mechanisms for pinpointing the adversarial examples
- 5) Integration of the detection mechanism with STAR blockchain and XAI for a holistic approach.

- 6) Feeding the detection mechanism with more pilots' data, make adaptations and improvements to the end models.

Until the moment of writing this deliverable and as documented in this deliverable, UBI has reached the 4th step of the described road map. More specifically, we have created a first version of the technical solution which will be used to detect the poisoning and evasion attacks.

During this period, UBI has been in close collaboration with the pilot leaders in order to be aligned with the datasets that they are going to provide and work towards meeting the expected goal of their pilots regarding the cyber defence of the AI systems. In addition, all the participants of WP3 have contributed to forming a consensus on the interaction and scope of all WP3 tools towards the provision of the Security and Data Governance infrastructure of STAR for AI Systems in Manufacturing.

1.4 Deliverable structure

The structure of the document is as follows:

- Section 2 offers an overview of the advancements in the adversarial AI domain. More specifically, the section documents state-of-the-art tools and libraries that can be used in the context of STAR, elaborates on research endeavours for the development of defence approaches against adversarial tactics and documents the actions of the cyber security society to analyse the emerging Adversarial Threat Landscape for Artificial-Intelligence Systems.
- Section 3 provides a comprehensive documentation of the AI Cyber defence tool, by discussing its placement in the STAR architecture, the provided services and data models used. In addition, this section elaborates on the integration path of the AI Cyber Defence tool with other tools of the STAR project.
- Section 4 provides the first documentation of the evaluation results of the experimental testbed that UBI created in order to evaluate different adversarial and defences methods utilising data stemming from the pilot sites.
- Section 5 elaborates on the long-term research plan of UBITECH based on which we aim to the definition of new AI methodologies for enhanced trustworthiness of data in order to solve the challenges identified as an aftermath of the evaluation performed in section 4.
- Section 6 concludes the deliverable, outlining the main outcomes of the work performed in the context of the task and providing a roadmap for the next deliverable of Tasks 3.2 and 3.3.

2 Background and State of the Art

This section offers an overview of the state-of-the-art of tools, methods, research endeavours and threat taxonomies that fall under the umbrella of the adversarial AI domain. The section begins with a reference to the adversarial ML tools, namely ART and Cleverhans that have been developed and brought together in a library of well-established attack and defence ML setups, discusses the advancements of recent work in the adversarial AI domain, and then, documents specific adversarial ML research actions in the manufacturing domain. The latter is aligned with Chapter 2 [REF-44] of the Book [REF-43], which have been published as a joined work of UBITECH's team members contributing to the development of the AI Cyber Defence module. Finally, as adversarial tactics are becoming a reality and affect deployed ML setups used in production, we document the latest status of the ATLAS MITRE threat taxonomy used specifically for the categorization of adversarial techniques.

2.1 AI tools and libraries used in the evaluation methodology

Next, we provide a brief overview of the AI tools and libraries used in the context of the evaluation methodology implemented within the STAR ecosystem in order to detect poisoning and evasion attacks, as part of the AI Cyber Defence module.

2.1.1 Adversarial Robustness Toolbox (ART)

One of the libraries employed to this end is the **Adversarial Robustness Toolbox (ART)**, which is a Python library intended for use in Machine Learning Security applications. Specifically, ART provides tools that enable the evaluation and verification of **Machine Learning (ML)** models that are used in order to detect evasion, poisoning, extraction, and inference attacks. It supports various popular commercially available ML frameworks, as well as various data types (images, audio, video, tables etc.) and types of machine learning tasks (classification, certification, generation, object detection).

ART provides various capabilities which can be used in the context of ML security mechanisms. One main capability of ART is the creation of **feature adversaries** that can be used in order to train ML attack detection models against various kinds of attacks. Specifically, it offers capabilities to create real-world adversarial patches that are able to fool **real-world object detection and classification models**. Using the generated attacks and adversaries, it is possible to assess the accuracy of the attack classification by using the employed ML model, as well as to train the model by generating appropriate training datasets.

Note that, with ART, it is possible to either use the built-in ART classifier in order to train a model in order to detect the desired kind of attacks, or to provide a pre-trained model to the ART classifier. With regards to the classifiers provided by ART, the most general and versatile classifier is referred to as **BlackBoxClassifier**, and only requires a single predict function definition without any additional assumptions or requirements.

One of the main goals of the STAR ecosystem, which will also be outlined in the present deliverable, is the usage of the AI Cyber Defence module in order to detect **poisoning and evasion attacks**. ART provides mechanisms that make it possible to defend against this kind of attacks. For example, in the case of poisoning attacks, it is possible to implement a

defence that consists of partitioning the data that enters the classifier into disjoint subsets, and afterwards training an ensemble model. It is also possible to implement the **Neural Cleanse** method, which is able to detect backdoor poisoning attacks. Specifically, this method consists of the following steps: (i) filtering out potentially poisonous input, (ii) unlearning the backdoor by retraining the model, and (iii) pruning the neural network of neurons associated with the backdoor.

Afterwards, it is possible to use the ART library in order to perform an **evaluation** of the Cyber Defence module and to compute the success rate of the classification of benign and malicious traffic. In order to evaluate ML models that aim to detect evasion attacks, ART utilizes an **Auto-Attack methodology**, which runs one or more attacks, either defaults or provided by the user, against a classification task. When executing an Auto-Attack process, ART optimizes the attack strength by only attacking correctly classified samples. First, it runs an untargeted version of each attack selected by the user, and afterwards it runs the targeted version of the attack against each possible target label.

ART also provides mechanisms in order to achieve the **certification and verification** of an ML attack detection method. For example, ART can leverage the **Randomized Smoothing method**, in order to achieve adversarial robustness for neural networks. Also, it is possible to utilize the **DeepZ framework** in order to compute certified robustness for neural networks, and to verify the adversarial robustness in decision tree ensemble classifiers (such as Gradient Boosted Decision Trees and Random Forests) using the **XGBoost, LightGBM** and **Scikit-learn frameworks**.

The capabilities and methodologies mentioned above are organized into the ART architecture, which currently consists of six modules. These modules, along with the submodules contained in each, are as follows:

- **Attacks:** evasion, poisoning, extraction, inference
- **Defences:** detector, postprocessor, pre-processor, trainer, transformer
- **Estimators:** classification, object detection, poison mitigation, speech recognition
- **Evaluations:** security curve
- **Metrics:** CLEVER, verification decision trees, gradient check
- **Pre-processing:** expectation over transformation, audio

2.1.2 Cleverhans

Cleverhans is a software library that can be used in the context of the AI Cyber Defence module in order to detect and mitigate poisoning and evasion attacks. Specifically, the purpose of the Cleverhans library is twofold: (i) To provide **standardized implementations of adversarial training** and **reference implementations of adversarial example construction techniques**, and (ii) to provide **standardized benchmarks** for the performance of ML models in an adversarial setting. The standardization of both these aspects serves to provide an accurate benchmarking framework, since a benchmarking methodology that does not employ a standardized adversarial example construction may not provide accurate results, as a difference in the adversarial model can result in significantly different performance and accuracy of ML classification methods.

In the context of the Cleverhans software library, **adversarial examples refer to the inputs crafted by making slight modifications to legitimate inputs, in order to**

mislead machine learning models. This approach essentially aims to emulate evasion attacks, which have been mentioned in previous sections. These modifications are intended to be small in magnitude, so that a human observer would not have difficulty processing the resulting input, and may in fact not be able to perceive that anything has changed in the input dataset.

The Cleverhans library provides reference implementations of this kind of evasion attacks, which may be used in two ways:

- **To construct robust ML attack detection models by using adversarial training,** which requires the construction of adversarial examples during the training procedure.
- **To be used as standardized inputs when benchmarking an ML attack detection model,** so that the accuracy of attack detection can be evaluated and compared to other models, which also use the same reference inputs in order to produce the benchmark results.

Cleverhans can be used as a complementary tool to already existing libraries such as TensorFlow [REF-21] and Theano [REF-22], and ML libraries such as Keras [REF-23]. Cleverhans is a free, open-source software library, licensed under the MIT license, and developed in collaboration between OpenAI and Pennsylvania State University.

The two most important modules of the Cleverhans library are the following:

- ***attacks*:** This module contains example implementations of the algorithms, that construct the aforementioned adversarial attack models. These algorithms take a model and an input, and return the corresponding adversarial example.
- ***utils_tf*:** This module contains helper functions, that train and evaluate models using TensorFlow. The implementation of the adversarial training methodology is also included in this module, which constitutes a defence to make models more robust and resistant to adversarial examples.

The types of evasion attacks considered by the Cleverhans library are the following:

- **Fast Gradient Sign Method (FGSM) [REF-24]:** The motivation behind this type of attack is to select a training point, and to linearize the cost function used to train a model around that training point, so that the adversary is able to force the misclassification of that point. Note that a larger disturbance around the training point is more likely to cause the intended misclassification, but also makes it easier to detect by a human observer. The Cleverhans implementation of this method is parametrizable in order to control the magnitude of this disturbance.
- **Jacobian-based Saliency Map Approach (JSMA) [REF-25]:** This method computes a saliency score on the input, and causes disturbances to the features of the input that are more likely to move a sample from its intended source class to the class desired by the attacker. The Cleverhans implementation of this method is able to detect whether the target class has been achieved or not, as well as the number of input features that were modified to achieve misclassification.

Cleverhans also implements defence techniques, in order to achieve protection against this kind of attacks. These methodologies, in general, aim to **smoothen the model**, so that its sensitivity to small disturbances to the input are limited, thus making adversarial examples harder to craft. To this end, the **adversarial training** methodology has been implemented.

This method aims to inject **adversarial examples during training**, in order to improve the generalization of the ML model. In order to achieve this, Cleverhans provides functions, in order to include samples in the training set produced by the attack methodologies described above, while simultaneously limiting the impact on the correct classification of benign samples.

Another core functionality of Cleverhans is the ability to **report benchmark results**, in order to compare the accuracy of an ML attack detection model against other models that have been reported, by using the same input datasets. Also, a versioning system is provided, which aims to provide a basis for reproducible benchmarks, including a capability to introduce updated model versions, in the form of minor or major patches.

2.2 Research endeavours in the field

Several approaches and methodologies have been proposed in the literature in order to detect and defend against poisoning and evasion attacks. In particular, a major category of these approaches involves the use of Machine Learning (ML) and Deep Learning techniques, that employ a training dataset in order to identify potentially suspicious network traffic behaviours based on existing traffic patterns. An extensive overview of the threat taxonomy regarding adversarial attacks on machine learning, which refers to the malicious modification of the training (poisoning attacks) or test data (evasion attacks), is presented in [REF-26]. These threats may target various kinds of systems based on ML, such as **intrusion detection systems, spam filters, visual recognition**, and various other kinds of applications that employ neural networks or deep learning methodologies. An in-depth investigation of the possible attack routes is presented, aiming to provide a basis for constructing robust countermeasures.

Next, we provide a brief overview of the literature in this field regarding defence methodologies against this kind of attacks. The application of Deep Learning techniques and ML classifiers in **Mobile Crowd Sensing (MCS)** schemes, where the major challenges in identifying malicious traffic involve the large quantity of data and the collection of potentially harmful data through inputs and sensors, has been investigated in [REF-27]. Specifically, in this work, the presence of strong, colluding adversaries is addressed, while simultaneously aiming to efficiently manage the high influx of incoming user data. A prototype implementation is provided, along with a detailed performance evaluation under various attack scenarios, employing both synthetic and actual datasets.

In [REF-28], the concept of **Explainable Artificial Intelligence (XAI)** is explored, in order to enhance data trustworthiness in Crowd-Sensing systems by expanding upon existing ML-based methods to distinguish between benign and malicious data. Specifically, the authors aim to increase the robustness and trustworthiness of such methods, by addressing the possible relation between the sensory input from devices, via the application and integration of XAI in their implementation and execution. By using such methods, it is possible to gain knowledge and insight into temporal data, that can typically be challenging in terms of representation, and improve on the effectiveness of ML-based attack detection methodologies.

A method for the **mitigation of poisoning attacks involving the use of provenance**, meaning the metadata paired with data records that detail the origins and changes to details

supporting the confidence or validity of data, was presented in [REF-29]. Specifically, the presented methodology employs contextual information regarding the origin and transformation of data sets in the training set to identify poisonous data, thus enabling ML applications to be re-trained during runtime by consuming data sources in potentially adversarial environments. Two variations of the proposed method are presented, one for partially trusted data sets and one for fully untrusted data sets.

Mitigation methodologies for evasion attacks have been proposed in the literature as well. For example, a combination of **adversarial example training** and **outlier detection** was used in order to defend against evasion attacks in [REF-30], and the possible attack space was reduced by nearly two orders of magnitude. Also, in [REF-31], a methodology was devised in order to enhance an ML classifier's resistance against adversarial samples, which leverages the capabilities of **autoencoders** in order to perform **dimensionality reductions**, as well as **input noising and denoising**, in order to reduce the effectiveness of evasion attacks.

A variety of other works have been published, that can assist the development of the ML-based intrusion detection methods required in the context of STAR. For example, in [REF-32], a **heterogeneous graph neural model** is presented, which is able to jointly consider heterogeneous structural graph information, as well as heterogeneous content information of each graph node. In terms of **spatio-temporal social** network analysis, complex network theory is used in order to analyse user mobility patterns in [REF-33], taking into consideration time properties rather than employing static models. Similarly, in [REF-34], temporal characteristics were taken into consideration in the application of ML methods in real-world scenarios.

2.3 Adversarial AI in manufacturing

It is of critical importance to ensure the operational security of manufacturing environments, where a malicious attacker may compromise assets of the system in a way that may cause material harm, or even harm to human workers. There have been several solutions proposed in the literature regarding security in smart manufacturing environments that artificial intelligence-based systems. These solutions aim to mitigate attacks, which target the training process and the inferencing mode of the leveraged ML models and may cause significant harm to the operation of the manufacturing supply chain ecosystem and the integrity of safety-critical operations of systems. Next, we present a state-of-the-art overview of the methodologies that have been proposed, in order to address this kind of threats and to secure ML models against intelligent aggressors.

In general, AI-based methods in smart manufacturing systems take some intelligence as input, which may be obtained by monitoring, external sources, and produce some updated intelligence as output. Sensors are a major source of intelligence in such systems, as they can provide information regarding light, humidity, or reclamation, and send a corresponding signal to a monitoring or controlling endpoint. Also, Cyber-Physical Systems (CPS) may be employed, such as reliable indoor positioning systems and activity recognition systems with motion capturing sensors. These sources can be used in order to control the operation of systems such as robotic architectures and conveyor belts.

In terms of cybersecurity, in recent years, AI-based cyber defence mechanisms have attracted considerable research interest. These mechanisms are decentralized and can more dynamically classify various attack vectors. Many efforts have been made and many algorithms have been developed, and the machine learning classification models for cyber defence have gotten more sophisticated and have improved dramatically in the last years. For example, anomaly detection that can be used to detect unusual behaviour on a network or system has been analysed in [REF-35]. Anomaly detection in image data has been addressed in [REF-36], performance monitoring and data acquisition (SCADA) was examined in [REF-37], and the issue of preventive equipment maintenance in smart manufacturing systems has been addressed in [REF-38]. In all the above methods, the motivation is to identify patterns of standard and expected practice that the algorithm has learned or indicated. Administrators will be notified if an activity deviates from the predetermined or accepted model of behaviour.

Compared to existing methods, anomaly detection has the benefit of being able to detect malicious activity. However, in this kind of system, a malicious party may attempt to bypass or manipulate the employed ML model, thus executing a stealthy attack without being detected by the classification model. As this is a relatively new kind of attacks, organizations lack the intellect to search an ML area for suspected adversarial ML related vulnerabilities. In this recently adopted definition, three kinds of attacks are considered: **poisoning**, **stealing**, and **evasion**. The overarching aim of these models is to minimize the classification's generalization error and potentially deceive the decision-making mechanism against desirable harmful calculation [REF-39].

In the previous sections, we mentioned the category of **poisoning attacks** that are considered in the context of the STAR ecosystem. In this kind of attacks, the adversary will attempt to contaminate the training data, by extracting and infusing an argument that reduces classification precision. This attack has the potential to totally alter the classification mechanism during the training phase, allowing the attacker to interpret the system's classification in whatever way he sees fit [REF-40]. The backdoor or Trojan attack, for example, is an especially sophisticated attack in this class, in which the attacker deliberately poisons the model by adding a backdoor key to ensure it performs well on normal training data and testing samples but misbehaves only when a backdoor key is used.

The other main category of attacks considered in STAR is **evasion attacks**, which refer to the effort of an adversary to cause an ML-based system to misclassify malicious activity as benign. When performing such attacks, the adversary aims to remain stealthy or imitate some favourable behaviour. In terms of network anomaly detection, an **intrusion detection system (IDS)** [REF-56][REF-57] can be avoided by interpreting the attack payload in such a manner that the target of the content can read it, but the IDS cannot, amounting to a misclassification. As a result, the perpetrator will damage the targeted device without being detected by the IDS. In evasion attacks, the attacker aims to reduce the performance of a classification process based on ML, which can be interpreted as an increase in false positives, in false negatives, or in both. To this end, the attacker may take one of the following approaches:

- **False positives:** The aim of an attacker is to make the targeted system falsely classify benign data as malicious. This would lead to the decision-making system missing crucial information.

- **False negatives:** The attacker aims to cause the classification of malicious data as benign.
- **Both false positives and false negatives:** The attacker aims to reduce the overall confidence of the user in the decision-making process by letting malicious data go through and by filtering out benign data.
- **Clustering accuracy reduction:** Compared to classification, the accuracy of clustering is less straightforward to evaluate. In general, the reduction of accuracy is the overall aim of the attacker of a clustering algorithm.

It follows from all the above that an attack on ML systems, such as the poisoning and evasion attacks mentioned above, can have devastating effects when targeting a smart manufacturing system, as it is possible that such attacks can affect the integrity of the products outputted by production lines, causing significant increases in production costs due to the faulty products, or even harm to equipment or personnel due to reduced structural integrity.

The most recent offensive event against industrial infrastructure was the power grid attack in Ukraine in December 2015 [REF-41]. The attackers used a combination of cybersecurity techniques such as malware, denial of service, and phishing to take the entire electricity supply infrastructure to a point where it became difficult to repair, resulting in power failures across the country. These outages caused several blackouts, affecting 225,000 clients across Ukraine. In addition, major attacks have been launched against some of the more cutting-edge smart manufacturing systems, some of which are IoT based.

Note that, the examples of these cases provided in the present deliverable fall into the category of image data whose integrity can be checked by visual inspection. However, this kind of attacks can also be performed on other types of data, such as textual data (i.e., logs produced by the machinery and devices of a smart manufacturing system), Neuro-Linguistic Programming (NLP) systems, sound data, or analog waveforms.

2.4 ATLAS Mitre

This section briefly reports on the **MITRE ATLAS™ (Adversarial Threat Landscape for Artificial-Intelligence Systems)** [REF-42], which is a knowledge base of adversary tactics, techniques, and case studies for machine learning (ML) systems based on real-world observations, demonstrations from ML red teams and security groups, and the state of the possible from academic research.

ATLAS is modelled after the **MITRE ATT&CK® framework** and its tactics and techniques are complementary to those in ATT&CK. It enables researchers to navigate the landscape of threats to machine learning systems. ML is increasingly used across a variety of industries. There are a growing number of vulnerabilities in ML, and its use increases the attack surface of existing systems. Atlas was developed in order to raise awareness of these threats and present them in a way familiar to security researchers. Ultimately, the goal of ATLAS is to connect the research with the actual mitigations actions that the industry should be prepared to take, due to the attention that attacks on ML systems have started to attract. Therefore, a threat taxonomy has been created, in order to provide a reference point to cybersecurity actors regarding the types of attacks that exist, based on real-world examples.

In Table 1, the progression of tactics used in attacks according to the MITRE ATLAS taxonomy is presented. Note that the Table is organised in a sequential order, meaning that an attacker will generally follow the sequence of actions presented starting from the top of the table and moving towards the bottom. For each phase, we present a brief description, as well as the techniques that may be used by the malicious party in order to perform a successful attack.

Table 1: MITRE ATLAS Attack Progression

	Phase	Techniques	Description
1	Reconnaissance	Search for Victim's Publicly Available Research Materials, Search for Publicly Available Adversarial Vulnerability Analysis, Search Victim-Owned Websites, Search Application Repositories, Active Scanning	In this phase, the adversary aims to gather information that they can use to plan subsequent phases. This consists of techniques that actively or passively gather information, such as the details of the target organization's ML capabilities and research efforts . This information can be used by the adversary to obtain relevant ML artifacts, target ML capabilities used by the victim, or to lead to further Reconnaissance efforts.
2	Resource Development	Acquire Public ML Artifacts, Obtain Capabilities, Develop Adversarial ML Attack Capabilities, Acquire Infrastructure, Publish Poisoned Datasets, Poison Training Data, Establish Accounts	The adversary aims to establish resources that can be used to support their operations This phase may include creating, purchasing, compromising, or stealing resources that can be used to support targeting. These resources may include ML artifacts, infrastructure, accounts, or capabilities . The results of this phase can be used in subsequent phases, such as ML attack staging.
3	Initial Access	ML Supply Chain Compromise, Valid Accounts	The adversary attempts to gain access to the target system by using ML artifacts . The target system may be a network, mobile device, or edge device, such as a sensor platform . The ML capabilities used by the system could be local or may leverage cloud-based techniques. The result of this phase is the attack vectors that can be used to gain initial access to the system.
4	ML Model Access	ML Model Inference API Access, ML-Enabled Product or Service, Physical Environment Access, Full ML Model Access	The adversary attempts to gain some level of access to an ML model . This access may include gaining information, developing attacks, and obtaining a means to input data to the model. The level of access may range from the full knowledge of the internals of the model , to access to the physical environment where data is collected for use in the ML model. Varying levels of access may be used as part of the attack.
5	Execution	User Execution	The adversary aims to execute malicious code on the local or remote target system. This phase is usually paired with other phases to achieve broader goals, such as

			exploring a network or stealing data. For example, such an operation may include using a remote access tool to perform Remote System Discovery by running a PowerShell script.
6	Persistence	Poison Training Data, Backdoor ML Model	The adversary aims to maintain foothold of the system after initial access has been obtained. This may consist of techniques that attempt to keep access to systems across restarts, changed credentials , or other interruptions that may cut off the adversary's access. This phase includes leaving poisoned training data or backdoored ML models .
7	Defence Evasion	Evade ML Model	The adversary aims to avoid detection by security software. This consists of evasion techniques , whose goal is to bypass systems such as Deep Learning Detectors or Antimalware Neural Networks, so that the adversary can continue the attack undetected.
8	Discovery	Discover ML Model Ontology, Discover ML Model Family, Discover ML Artifacts	The adversary attempts to gain knowledge about the system and internal network by observing the environment, exploring what they can control, and what can be accessed by using their entry point, so that it can provide a benefit to their current objective. Native operating system tools may be used in this phase in order to gather information post-compromise.
9	Collection	ML Artifact Collection, Data from Information Repositories	After the target system has been compromised, the adversary aims to gather ML artifacts and other information relevant to their goal. Next, the adversary uses the collected data to exfiltrate the ML artifacts, or use the collected information to stage future operations. Targets of this phase can be software repositories, model repositories, container registries or object stores .
10	ML Attack Staging	Create Proxy ML Model, Backdoor ML Model, Verify Attack, Craft Adversarial Data	The adversary leverages the knowledge obtained in the previous phases in order to tailor the attack strategy to the specifics of the target system . Techniques used to prepare the attack may include training proxy models, poisoning the target model, or crafting adversarial data to feed the target model. Note that some of these techniques may be performed offline, making them difficult to mitigate.
11	Exfiltration	Exfiltration via ML Inference API, Exfiltration via Cyber Means	The adversary aims to steal target ML artifacts , by leveraging techniques that are able to target network data, and may include transferring this data over their command-and-control channel or an

<p>12</p>	<p>Impact</p>	<p>Evade ML Model, Denial of ML Service, Spamming ML System with Chaff Data, Erode ML Model Integrity, Cost Harvesting, ML Intellectual Property Theft</p>	<p>alternate channel. After all the previous phases have been performed, the adversary can achieve their objectives, which may include manipulating, interrupting, reducing user confidence, or destroying systems and data. The adversary may leverage techniques that disrupt business operations and compromise operational processes, such as destroying or tampering with data. It is also possible to make an impact on the target system in an undetectable way, by altering business operations to benefit the adversary's goals without disrupting the operations themselves.</p>
------------------	----------------------	--	--

3 STAR AI Cyber Defence Module

3.1 Positioning in the STAR architecture

The AI Cyber defence module aims to defend the STAR-enabled manufacturing platforms against poisoning and evasion attacks. Smart manufacturing ecosystems nowadays consist of several AI-powered components in order to improve their production practices. However, AI systems, as described in Section 2, are susceptible to attacks that target both the training (i.e. poisoning) and the operational (i.e. evasion) phases of Deep Neural Networks (DNNs). In this direction, the AI cyber defence component will boost the robustness of DNNs against adversarial inputs and attempts to contaminate the training datasets, and against active attacks that aim to evade the inference process of AI models.

The AI Cyber defence tool is positioned at the AI Security and Data protection layer of the STAR architecture and will work in synergy with the blockchain-based data provenance mechanisms, the Data management and Analytics engine, and the Explainable AI. The output of the AI Cyber Defence mechanism will be used as input to the Security policy manager to perform risk assessment and attack mitigation functionalities. Hence, the aim of the AI Cyber defence module coincides with the aim of the AI security and data protection layer which is to boost the safety, reliability and transparency of the functionalities of the upper operational layers of STAR.

Figure 1 illustrates the architecture of the STAR AI Security and Data protection layer, which is an outcome of the collaborative actions of all partners of WP3. The AI Cyber Defence tool is positioned in the middle, between the manufacturing plants (left) and the Security Policies manager and the Risk Assessment & Mitigation Engine (right). The purpose of the tool is the evaluation of the training and streaming data stemming from the datalakes and the deployed systems, respectively, so that to detect possible poisoning and evasion attacks. Upon the detection of an incident, Alerts are generated and pushed through the Data Bus to the Security Policies manager and the Risk Assessment & Mitigation Engine for further analysis and for informing the security administrator about the detected incidents.

The placement of the AI Cyber Defence module is highlighted in Figure 1. More details on the overall architecture of the STAR AI Security and Data protection layer will be given in D3.5 and D3.6.

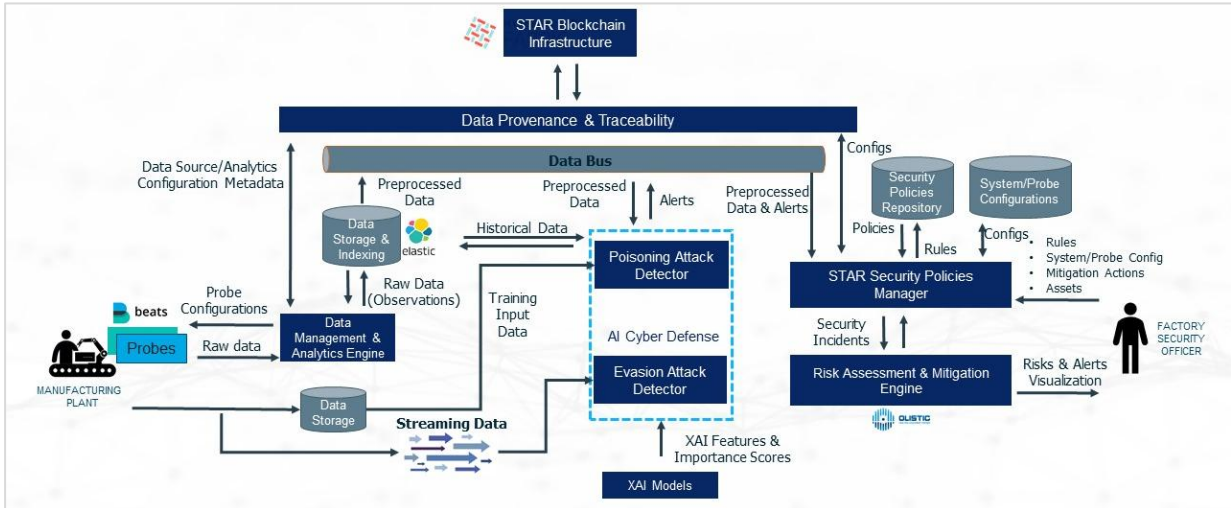


Figure 1 STAR AI Security and Data protection layer Architecture (WP3)

3.2 AI Cyber Defence Module internal architecture & interactions

Figure 2 illustrates the internal architecture of the AI Cyber Defence module along with the interactions that have been identified with other STAR components by the moment of writing this deliverable. The internal architecture of the AI Cyber Defence module covers both the Training/Testing phases for building the necessary baseline models to be used, as well as the actual deployment of the end-model which will be responsible for the detection of the adversarial attempts during the run-time phase of the manufacturing floors. The following paragraphs elaborate on the scope and the interactions of the individual components of Figure 2.

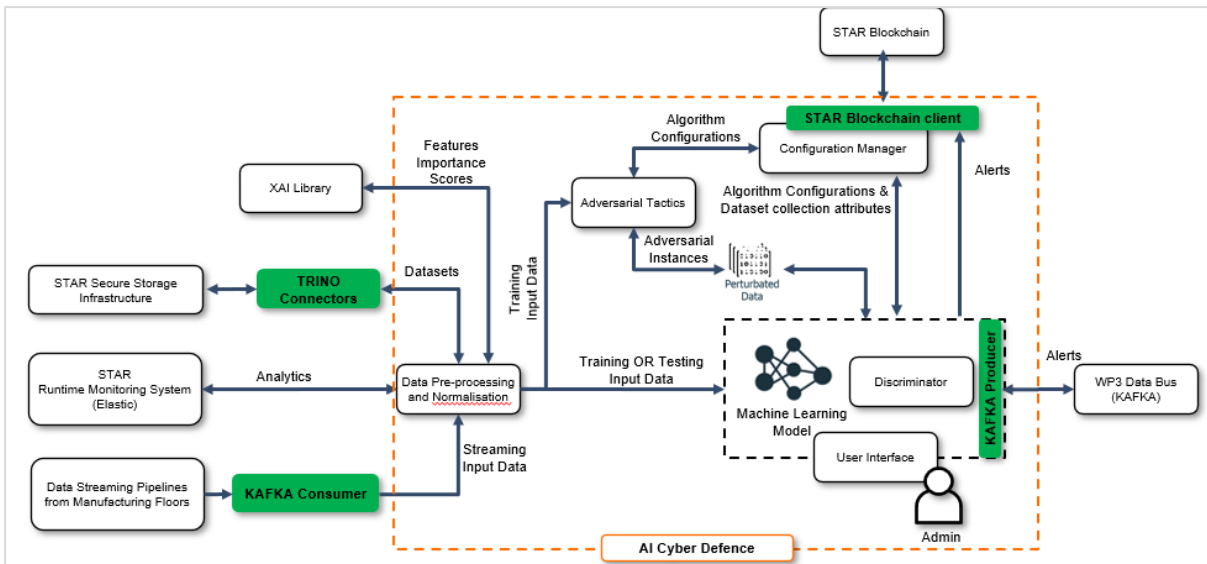


Figure 2 AI Cyber Defence Module architecture

3.2.1 Data Input Sources

The following three data sources have been identified:

- **STAR Secure Storage infrastructure:** This entity is a vital component in the STAR project architecture as it provides a unified datalake for all AI-enabled STAR components to consume data mainly for training reasons. That is, the AI Cyber Defence module will acquire Binary (images, videos, etc.), Tabular (CSV, xls, etc.), Database (SQL, Mongo DB), and Time-series data that may be needed to be evaluated for the identification of poisoning and evasion attacks. To establish this communication, the AI Cyber Defence tool will integrate the necessary TRINO connectors to query the respective resources to acquire the data required to perfume the training.
- **STAR Runtime Monitoring system:** As will be also explained in Section 3.4.1, the runtime monitoring system will be used for the acquisition of statistical measurements stemming directly from the core systems of the manufacturing floor. These measurements may indicate the presence of anomalous behaviours that could be used as inputs in the inference process. AI Cyber Defence will use a specific connector in order to acquire information, if needed, from the Elastic (ELK) Stack used by the Runtime Monitoring system.
- **Data streaming pipelines:** This entity represents the data sources that will be used during the actual deployment of the STAR platform to the pilot sites. Towards this phase of the project, the AI Cyber Defence tool takes the necessary steps in order to be able to handle input data stemming from pilots in a streaming mode. Pilot data will be fed to the end-model of the AI Cyber Defence in order to detect potential evasion attacks against the AI systems used in the context of the business process which is being monitored/evaluated in a pilot. A KAFKA consumer might have the technical solution to enable the acquisition of data in streaming mode and in fact some pilot partners already utilised KAFKA for this purpose. However, the choice of this solution will vary depending on the adopted technologies by the pilot partners.

3.2.2 Data pre-processing and normalisation

Depending on the AI model that will be training and utilised for the detection of adversarial attempts against the STAR systems, data pre-processing is an important step in order to manipulate the data and apply the necessary transformations in order to make them fit to the specific model. In addition, through the pre-processing, we are able to provide to the model with a sufficient set of features for the training task and even create new or uncover hidden information so that to contribute to the overall performance of the end-model.

Crucially, this component contributes to the interoperability of the data pipelines which are formed among the various data sources of STAR and the AI Cyber Defence tool. As the data sources may be used for multiple STAR components, it would be unmanageable to delegate the data pre-processing to the technical components that provide the data. Thus, the AI Cyber Defence module receives the raw data and applies the necessary transformations in order to feed its internal pipelines. More details on the specific pre-processing steps that have been evaluated so far are given in Section 4.3.

As aforementioned in the previous sub-section, there are multiple data sources to be considered and feed the training or the inference process of the AI Cyber defence module, both for the cases of the detection of poisoning and evasion attacks. That is, this component also undertakes the data manipulation for creating augmented datasets as the results of data merging actions. For example, training instances coming from the Secure Storage

Infrastructure might be augmented by the importance scores of features that will be produced by the XAI library. Then, this augmented dataset will be used as input to the AI Cyber Defence module.

Note that, the pre-processing and normalisation component is also used during the inference stage of the module. In fact, the same data processing steps that took place during the training of a model needs to be replicated for the data pushed into the module in streaming mode during the runtime phase of the model.

3.2.3 Adversarial Tactics

This is the technical component that undertakes the generation of adversarial examples, based on the tactic and the methodology to be followed in the training of the end-model of the AI Cyber Defence module. The purpose of this tool is really important for the establishment of the end model as, based on this, we are in a position to utilise the off-the-shelf and well-established approaches in order to generate adversarial example for specific test cases and help us understand the techniques followed by the attackers.

Thus, this component gets as input the dataset instances from the data input sources, described in Section 3.2.1, in order to generate the respective adversarial examples.

3.2.4 Configuration Manager

In the context of the Security and Data Governance for AI Systems in Manufacturing, the STAR Blockchain network will be used to provide a decentralized infrastructure for the provenance and tracking of industrial data and for the configuration of algorithms used in AI systems. In this context, the AI Cyber Defence module utilises the Configuration Manager component in order to interface with the STAR Blockchain and manage the process of validating and acquiring for the blockchain the correct configuration and data that the AI Cyber Defence requires for each case. For example, a specific Adversarial tactic with a defined parameterisation may be required in the context of a specific use case of a pilot. Thus, in order to ensure that there is no malicious intervention or tampering in the configuration of this component, the latter may acquire the correct configuration from the blockchain prior to triggering the generation of adversarial examples. The same applies to the configuration setup of the discriminator model.

3.2.5 Discriminator

The discriminator is a model that tries to distinguish real data from the data created by the attacker or in other words the adversarial examples. This component is the heart of the AI Cyber Defence module. The discriminator is a term that we use in order to refer to the model that discriminates the legitimate from the adversarial examples. However, the actual model depends always on the use case, the dataset and the adversarial approach we aim to detect. Thus, in order to build the discriminator, we feed its model with perturbed data as generated by the adversarial tactic component, as well as with training data coming from the data sources of STAR. The end goal to the discriminator coincides with the goal of the AI Cyber Defence module per se, which is the generation of an alert that indicates the presence of a poisoning or evasion attack against the STAR AI components and the manufacturing environment. More technical details and evaluation results are given in Section 4.7.3. Again, as was the case for the adversarial tactics component, the Discriminator, depending on the case, may utilise a different algorithm configuration. That is, through the Configuration management, a new configuration is applied for given cases.

3.2.6 Alert Data Output

Upon the detection of a poisoning or an evasion attack, the AI Cyber Defence module generates an alert. An alert includes all the necessary information for an attack, the targeted system and dataset, the timestamp that indicates when the detection took place, as well as the confidence level of the detection.

In order to enable the interoperability of the STAR framework, from the early stages of the WP3 actions, the partners have worked towards the definition of specific data models that will ensure the smooth integration of the WP3 tools. It is expected that the defined data models will be enhanced, while new data models will be defined as a result of the definitions of more fine-grained interactions among the WP3 tools, and the integration developments of WP6. Figure 3 offers the data model of the alert, while Figure 4 offers an indicative example.

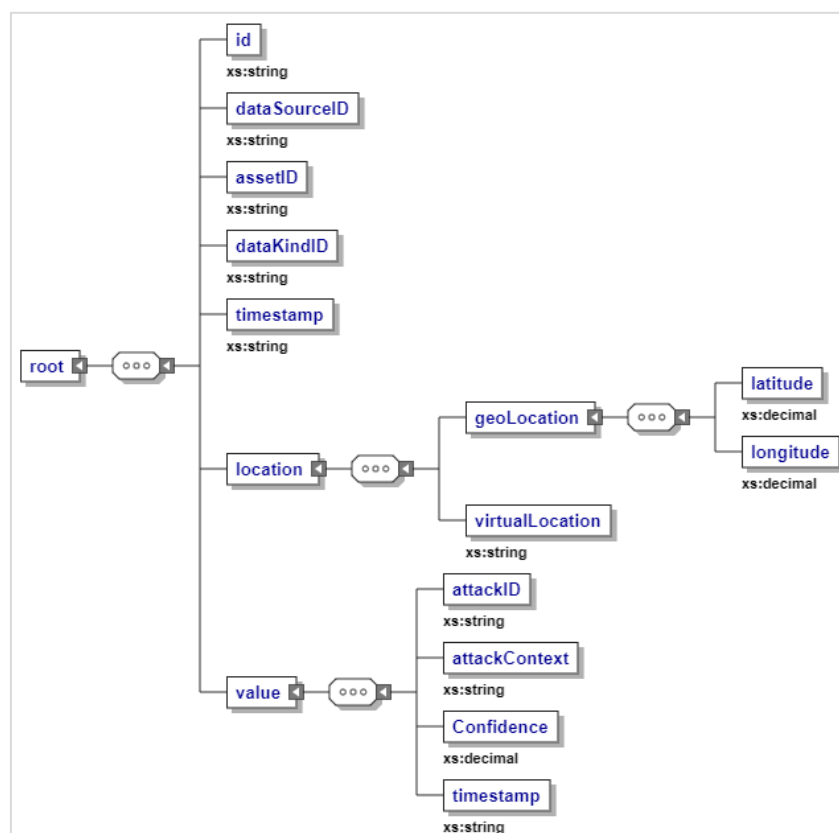


Figure 3 AI Cyber defence Alert Datamodel

```
{
  "id": "1ee5f356-632e-11ec-90d6-0242ac120003",
  "dataSourceID": "3341e5b2-632e-11ec-90d6-0242ac120003",
  "assetID": "4daef4f2-487e-48e1-8f8f-d526d36aa5cd",
  "dataKindID": "65a7604e-9a94-4a74-9a34-3e44c6cebd49",
  "timestamp": "2022-01-10 13:01:29.709071",
  "location": {
    "geoLocation": {
      "latitude": "53.107731",
      "longitude": "6.088499"
    }
  }
}
```

```

    },
    "virtualLocation": "8.162.203.200"
  },
  "value": {
    "attackID": "f777d14b34c3cdff92468fbfa55aeeddd8298745",
    "attackContext": "Evasion attack",
    "Confidence": "90.12",
    "timestamp": "2022-01-10 13:01:22.709095"
  }
}
}

```

Figure 4 AI Cyber defence Alert output example

The generated alert will be sent to the Data Bus component which will be used as an integration point for most of the WP3 components. The data bus is an Apache KAFKA which will be used for the sharing of data among the tools. The AI Cyber Defence module integrates a KAFKA producer in order to push the alert to KAFKA. For this purpose, specific topics will be defined, and tools will act as consumers and producers for those topics. Thus, the alerts of the AI Cyber Defence will be consumed by the STAR policy manager for further processing and correlation with other collected data.

3.3 Functional Requirements for AI Cyber Defence module

The tables given below summarise the functional requirements related to the AI Cyber Defence module, as those have been initially reported in D2.1. The requirements are enumerated below, while more details are given in the tables below.

- Req-01) Detection of poisoning attacks
- Req-02) Interaction with STAR blockchain
- Req-03) Interaction with XAI
- Req-04) Generation of alerts
- Req-05) Detection of evasion attacks
- Req-06) Detection of malformed instances
- Req-07) Adversarial training for robust model derivation

Based on the description given in Section 3.2 for the architecture of the AI Cyber Defence module, one can notice that the architecture and the defined interactions are aligned with the requirements defined in the early stages of the project. More specifically, Req-02 is addressed through the acquisition of proper algorithm configurations via the Configuration Manager of the tool. Req-03 is addressed as the XAI libraries will be exploited by the AI Cyber Defence tool to acquire the importance of features and exploit additional features in the process of the poisoning attack detection. Req-04 is met by the operation of the Discriminator tool which generates the respective alerts, as described in Section 3.2.6. Req-01, Req-05, Req-06 and Req-07, will be addressed through the utilization, evaluation and development of the state-of-the-art techniques, as will be documented in Section 4, where our experimental setup and a first evaluation of adversarial tactics and defences are given.

The purpose of this section is not to claim that all the requirements have been met and can be considered as complete, but to justify that all the technical decisions taken are aligned with the defined requirements and define the means to achieve them. It is expected that all

requirements will be met with the 2nd (and final) release of the AI Cyber Defence module in D3.4.

Section	Description
Id	Req-AI-Cyber-Defence-01
Type	FUNC
Short name	Detection of poisoning attacks.
Description & quantification	This component needs to detect attempts of adversaries to train ML models using malformed training instances.
Additional information	The requirements will be covered through the investigation of state-of-the-art techniques and tools like ART (Adversarial Robustness Toolbox)
Priority	MAN

Section	Description
Id	Req-AI-Cyber-Defence-02
Type	FUNC
Short name	Interaction with STAR blockchain.
Description & quantification	This component needs to combine data from the blockchain-enabled provenance mechanisms in order to identify potentially fraudulent data sources. Ideally, this piece of information should be considered as additional external features that can enrich the inference mode of the algorithms empowering the AI Cyber defence module.
Additional information	N/A
Priority	MAN

Section	Description
Id	Req-AI-Cyber-Defence-03
Type	FUNC
Short name	Interaction with XAI.
Description & quantification	This component needs to consider the importance of explainable AI features in order to identify potential poisoning attack attempts. Ideally, this piece of information should be considered as additional external features that can enrich the inference mode of the algorithms

	empowering the AI Cyber defence module.
Additional information	N/A
Priority	MAN

Section	Description
Id	Req-AI-Cyber-Defence-04
Type	FUNC
Short name	Generation of alerts.
Description & quantification	This component needs to raise an alert upon the detection of training instances of poisoning attack attempts. The alerts need to be accompanied by a level of detection/prediction confidence for each of the instances that will be detected as malicious.
Additional information	N/A
Priority	MAN

Section	Description
Id	Req-AI-Cyber-Defence-05
Type	FUNC
Short name	Detection of evasion attacks.
Description & quantification	This component needs to detect evasion attacks that may target the deep neural networks during the inferencing stage.
Additional information	N/A
Priority	MAN

Section	Description
Id	Req-AI-Cyber-Defence-06
Type	FUNC
Short name	Detection of malformed instances.

Description & quantification	This component needs to detect adversarial examples that cause misclassification on the context of evasion attacks. The malformed instances, along with the confidence level of the detection will be combined into an alert to be shared with the security policy manager.
Additional information	N/A
Priority	MAN

Section	Description
Id	Req-AI-Cyber-Defence-07
Type	FUNC
Short name	Adversarial training for robust model derivation.
Description & quantification	This component needs to use adversarial training in order to increase the robustness of the utilized model by retraining the model with adversarial examples. This approach will be used to create a discriminator which will be in position to differentiate malevolent instances from benign ones. This detector will generate alerts on the detection of malevolent instances.
Additional information	N/A
Priority	MAN

3.4 Interactions with other STAR tools

3.4.1 Runtime monitoring system

As mentioned above and depicted in Figure 1, one of the AI Cyber Defence component data sources is the Runtime Monitoring System (RMS). RMS is a Data collection framework which provides the specifications and relevant implementation to enable a real time data collection, transformation, filtering, and management service to facilitate data consumers (i.e., AI Cyber Defence). In STAR the framework will be applied in collecting system and user data to facilitate the detection of abnormal behaviour and make the data available to the AI algorithms and Security Policy Manager. For example, the solution may be used to collect security related data (e.g., network, system, solution proprietary, etc.) from monitored IoT systems and store them to detect patterns of abnormal behaviour by applying simple (i.e., filtering and pre-processing) or more elaborated mechanisms (i.e., AI algorithms of the Cyber Defence system). One of the key features of the framework is that it is detached from the underlying infrastructure by employing a specialized data model for modelling the solution's Data Sources, Processors and Results which facilitates the data interoperability discoverability and configurability of the offered solution. Especially for the results, the

Observation model will be followed which is described in section 3.2.6 above and in D3.1 section 4.2.4. More details for the RMS can be found in deliverable D3.5.

3.4.2 STAR Blockchain network

As mentioned above and depicted in Figure 1, the AI Cyber Defence component interacts with the Data Provenance and traceability services in two ways:

- For persisting/retrieving the AI algorithms configurations metadata which are capable of describing an algorithm type along with its various instantiation configurations across time by using the Analytics Engine Configuration (AEC) service (see D3.1 section 3.3.1). Information about the exposed API, data models and usage can be found in section 4.2.3 of D3.1, and
- For persisting AI algorithm results by utilizing the Analytics Results Publishing (ARP) service (see D3.1 section 3.3.2) using the Observation data structure as described in section 3.2.6 above and in D3.1 section 4.2.4. Information about the exposed API can be found in section 4.2.4 of D3.1.

Using the Distributed Ledger as the distribution channel, STAR will ensure a truly decentralized but also reliable system, as analytics manifests are "signed, sealed and timestamped" so that no forgery or tampering is possible. Moreover, the AI Cyber Defence component will be able to share analytics results on the Distributed Ledger infrastructure, thus contributing to a common data set representing the combined results across the entire distributed system. The virtues of such workflow lie in immutability and non-repudiation.

3.4.3 Explainable AI (XAI)

XAI algorithms, as shown in D4.1, will play an important role in many STAR scenarios. Among them, XAI aims to contribute to the detection of poisoning and evasion attack as one of the benefits that can be brought to STAR for Secure and Trustworthy AI. Due to that fact, the XAI part will be constructed as library of algorithms that can be used in many different cases, one of which is the AI Cyber Defence module, which is the focal point of this deliverable. Steinhardt et al. emphasized on the significance of this component against the poisoning attacks by stating that even with a strong "defence", a poisoning of a 3% of the total training dataset can lead to a drop of up to 11% in accuracy [REF-13].

As far as the XAI contribution to the AI cyber defence component is concerned, the state-of-the-art XAI models have been gathered and examined by the T4.1 in the early stages of the project in order to point out the way that XAI methods can be utilized against the most likely types of poisoning attacks.

3.4.3.1 XAI on the detection of adversarial attempts

Nowadays, Deep Learning and Machine Learning models have been a necessary part of many tasks in the industrial sector. Despite that fact, the Industrial sector's state-of-the-art AI development is still far from utilizing the most sophisticated machine and deep learning competencies [REF-17]. The Defence against poisoning attacks along with the Explainability and Interpretability of AI algorithms can be considered as essential parts for "triggering" the wide use of such methods in production. The approach of detecting adversarial examples will be further examined in the T4.1 targeting to an enhanced XAI library with poisoning and evasion attack detection competences.

In this research domain, a lot of focus has been placed on creating successful and self-resilient XAI algorithms against poisoning attacks [REF-07]. Many researchers, for example, have attempted to defend the XAI model from poisoning attacks (e.g., Heo et al. exposed the vulnerabilities of state-of-the-art saliency-map based systems by deceiving the system with adversarial model manipulation [REF-08].)

Muddamsetty et al. have also proposed a SIDU XAI visual explanation algorithm, which can effectively localize the total object regions responsible for prediction to a large extent. This algorithm appears to be more solid in the presence of adversarial attacks compared with black-box models as its performance can be better understood [REF-09]. The performance of SIDU is compared to GradCam in [REF-10]. When compared to the fixation maps, the results show that GradCam outperforms SIDU; however, when the algorithms are compared to noise, the results switch, indicating that SIDU is more solid to adversarial attacks.

The research reported by Dombrowski, et al., which is similar to T4.1's vision, indicated that adversarial manipulations of the input data can substantially modify the explanation maps [REF-11]. By simplifying the explanation process, the authors were also able to strengthen the system's robustness to attacks.

Another innovative detection way is to distinguish between normal and adversarial inputs by using Shapley Additive Explanations (SHAP) values generated for the internal layers of a DNN classifier [REF-12].

Most approaches for identifying adversarial data sought to find anomalies in the input data, internal layer activations, or model architecture, while others took a more active approach to increase detector performance by manipulating the inputs [REF-14] or changing the training process [REF-15].

According to Ilyas et al. [REF-16], the presence of adversarial cases is an inherent aspect of the dataset. They make the distinction between robust and non-robust features. Ilyas et al. presented that the adversarial examples existence and their transferability across different classification models, arises naturally from the existence of non-robust features, which allow small changes in the input to cause large changes in the value of these highly predictive features. As a result, adversarial evasion attacks alter the values of non-robust characteristics while substantially maintaining the values of robust features [REF-16]. This is since implementing an effective alteration to robust characteristics necessitates major input changes.

The STAR XAI, will be based on a method for detecting adversarial examples as documented in T4.1. It is based on Fidel et al. recommended technique [REF-12]. SHAP [REF-18] is used to calculate the significance scores of the neurons in the penultimate layer of the classification model for each input to be classified as adversarial or normal. The adversarial example detector then uses these weighted scores as features.

Micromodels are a unique take on anomaly detection. For cleaning training data for network intrusion detectors, the Micromodels defence was recommended [REF-19]. The defence examines classifiers on the training set by training them on non-overlapping epochs of the training set (micromodels). Training cases are classified as safe or suspicious based on a majority vote of the micromodels.

If the "poisonous" data was injected in the same "space" as the data injection in a poisoning attack. We can use the following strategy in this direction:

STAR XAI Approach (XAI Global vs Local Explanations)

Most machine learning approaches were created to work on certain problem sets where the training and test data came from the same statistical distribution. Adversaries may provide data that contradicts that statistical assumption when such models are used in reality. This data could be adjusted to exploit specific flaws and manipulate the results. In order to provide Subset Aggregate Feature Importance or XAI metrics against data poisoning attacks, the XAI model may give a specific implementation based on random sampling and GAs.

As shown in Figure 5, a number of folds were generated, each consisting of some known data "golden" and a distinct subset of unknown data.

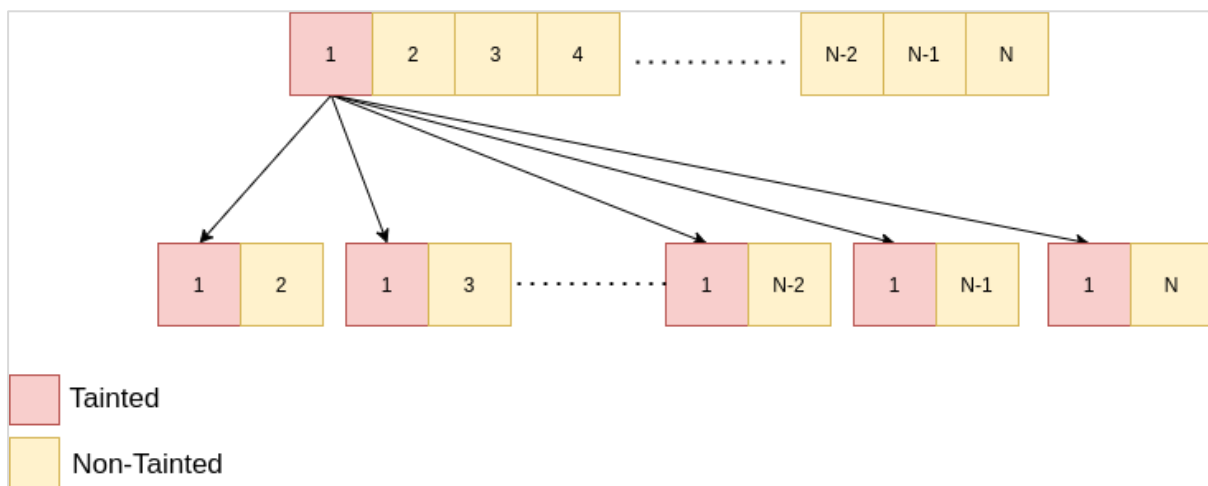


Figure 5 Data partitioning in batches for local explanation testing

The Genetic Algorithm (GA) is a search-based optimization technique based on genetics and natural selection principles. It is used to identify optimal or near-optimal solutions to computational problems that take a long time to solve. They have since been used to solve a variety of optimization issues with great success.

Furthermore, the same approach has been used multiple times, with subsets created at random. At each "epoch" a ratio denoting the "status" (i.e., adversarial or not) of each batch of images will be noted, and the average "score" for each image will be returned to the cyber-defence component (either per image or per batch of images) in order reduce the likelihood and impact of an adversarial split (i.e., to minimize the likelihood that the initial random choice will severely impact the output). To summarize it, the more times the experiments are repeated, the more reliable the results become.

Algorithm 1 in Figure 6 describes the entire process. Moreover, the method considered all of the GA's parameters as well as the ML model's hyper-parameters as inputs.

Algorithm 1: GENETIC ALGORITHM for feature selection

Input: A tabular dataset $data$, a $classifier$, the $features_array$

(i.e. discrete sequence to a sequence of ones and zeros)

Output: A vector of the indexes of the selected features.

```

1 for  $n \leftarrow range(number\_of\_permutations)$  do
2   Split the  $data$  into balanced  $subsets$ 
3   for  $subset = 1, subsets$  do
4     Initialize the Population  $P$  with size  $P_{length}$ 
5     for  $i \leftarrow range(P_{length})$  do
6       Initialize an specific selection of features as Individual  $I$ 
7        $P_i = I$ 
8     end
9     Evaluate the fitness score  $P_{score}$  of the  $P$ 
10    for  $i \leftarrow 0, range(L)$  do
11      Select the features of the dataset based on the  $I$ 
12       $data = data[features \times I]$ 
13      Split the dataset in  $train, validation, test$ 
14       $model.fit(train, validation)$ 
15       $prediction = model.predict(test)$ 
16       $I_{score} = F1(test, prediction)$ 
17    end
18     $P_{score} = average([I_{score}])$ 
19    if  $i == 0 \vee P_{score} > P_{score\_best}$  then
20      Set best score  $P_{score\_best}$  and best Population  $P_{best}$ 
21       $P_{score\_best} \leftarrow P_{score}$ 
22       $P_{best} \leftarrow P$ 
23       $no\_improve = 0$       28
24    end
25    Check for termination criterion
26  end
27 end

1 if  $no\_improve < termination\_criterion$  then
2   Selection
3   Select top  $k$  best Individuals  $I$  of the  $P$  with probability
    $I_{score}/sum(I_{scores})$ 
4   Crossover
5   for  $k \in P_{length}/2$  do
6      $cross\_index = randint(I_{length})$ 
7      $P[k + population_{length}/2] \leftarrow P[k]:$ 
    $cross\_index] + P[k + 1][cross\_index :]$ 
8      $P[k + population_{length}/2 + 1] \leftarrow P[k][cross\_index :]$ 
    $+ P[k + 1]:$ 
    $cross\_index]$ 
9   Mutation
10  for  $m \in P_{length}$  do
11     $flag_{mutation} = rand()$ 
12    if  $mutation\_flag < mutation_{threshold}$  then
13       $mutation\_index = randint(I_{length})$ 
14       $P[m][mutation\_index] = 1 - P[m][mutation\_index]$ 
15  else
16    return  $I_{best}, I_{score\_best}$ 
17  return  $I_{best}, I_{score\_best}$ 

```

Figure 6 Pseudo-algorithm of the XAI method of STAR

3.4.3.2 Experiments/Outputs

The type of data, that will be chosen, will be critical in determining which XAI method should be used for each task. The project's XAI framework solutions will be classified according to the input data categories, as defined in D4.1. As a result, we investigate and create XAI algorithms specialized for time series, tabular, text, and image data.

Up to this point, we've concentrated on image data because, as D2.4 and D2.1 show, most use-cases employ image recognition techniques for various tasks. As a result, the remaining deliverables will be primarily oriented by image type data.

Bellow, there is a figure that depicts the SHAP values of the shavershell dataset provided by PHILIPS. This will be represented in an array of scalar numbers and will be fed into the cyber defence component.

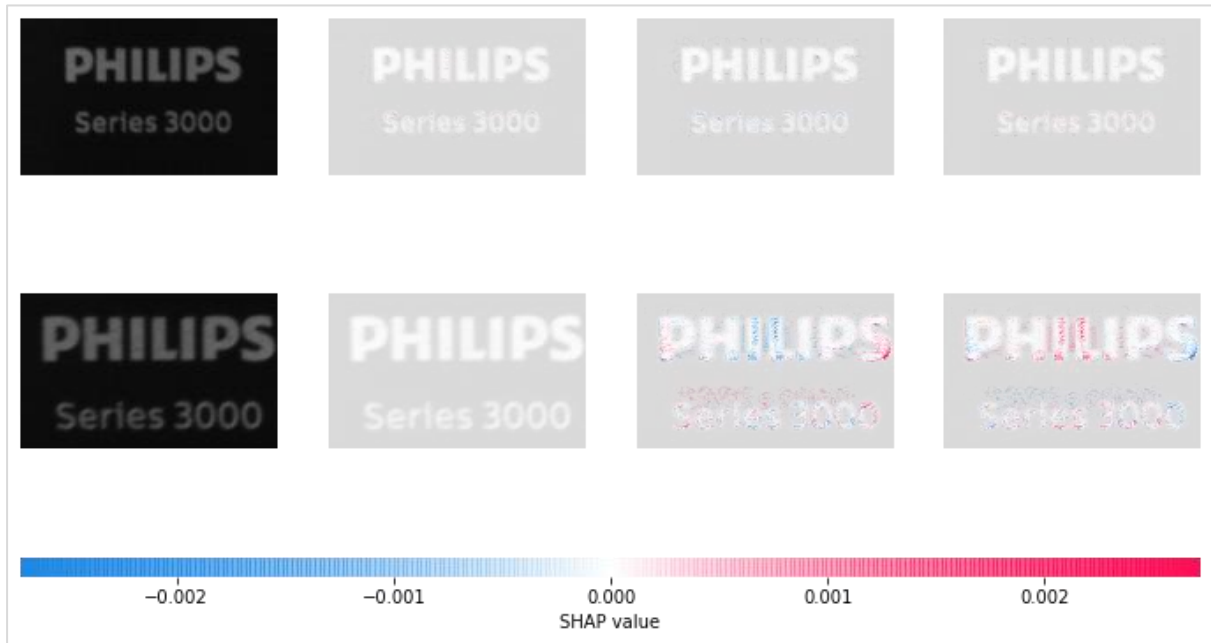


Figure 7 SHAP results on image dataset

3.4.3.3 Integration with AI Cyber Defence

A key point that may need extension or modification of the underlying models is the type of the output of the XAI models, with respect to the end-component (e.g., input to other modules such poisoning/evasion attack detection (WP3) or simulated reality (WP4)) may differ from the one that will be represented to a human user).

In task (T4.1) we propose a method for helping to detect adversarial examples. It is based on the proposed approach of Fidel et al. [REF-12]. For each input to be classified as adversarial or normal, SHAP [REF-18] is used to compute the importance scores of the neurons of the penultimate layer of the classification model. Then, these importance scores are used as features for the adversarial example detector as depicted in Figure 2. This will be represented in an array of scalar numbers and will be fed into the AI Cyber Defence module. Until the moment of writing this deliverable, the methodologies of both the XAI component and the AI Cyber Defence are under construction and conceptualisation. It is clear that the AI Cyber Defence module will exploit the importance scores generated by the XAI in order to use them as indicators to boost the detection accuracy of the respective end-model. It is expected that on the 2nd release of the framework in D3.4, more details on the interactions will be documented.

4 Experimental setup

4.1 Testbed setup

In order to evaluation and applicability of existing adversarial techniques and to test corresponding defences, in the context of this deliverable we built an experimental testbed based on the data provided by Philips for Pilot 1. In this direction, we analysed and processed Decocap and Shavershell datasets which are presented in the following section.

Two different datasets of images were separated into three of two subcategories (e.g., good/interrupted print/doubleprint, and good/bad) in order to enable both the analysis under the prism of a multi-classed and binary classification problem. Our goal was to build a Neural Network that would be able to predict in which subgroup, an unknown instance (i.e., image) to our model, would be classified. In this way, we imitate the behaviour of an AIN-enabled visual inspection system, as part of a quality assurance process running in a production line.

Figure 8 presents the experimental flow followed. It is a common practice in computer vision to perform a pre-processing on images before using them. This pre-processing helps to identify important features in the data. Thus, we train a basemodel and we evaluate its accuracy in order to create a reference model and use it as the baseline for assessing at a later stage the impact that an adversarial technique will bring on the model. To do so, we form a cross-validation loop where we use the pre-processed data with the attack algorithms. For this initial version of the Cyber Defence mechanisms of STAR, we make use of the Adversarial Robustness Toolbox to define different attackers i.e., attackers deploying different attack techniques. Using these attackers, we can create and manipulate our adversarial examples. To this end, we define three divergent scenarios, which have the same general goal, to achieve the increase of the performance of our attacked model. In the next sections, these processes and scenarios will be explained in detail.

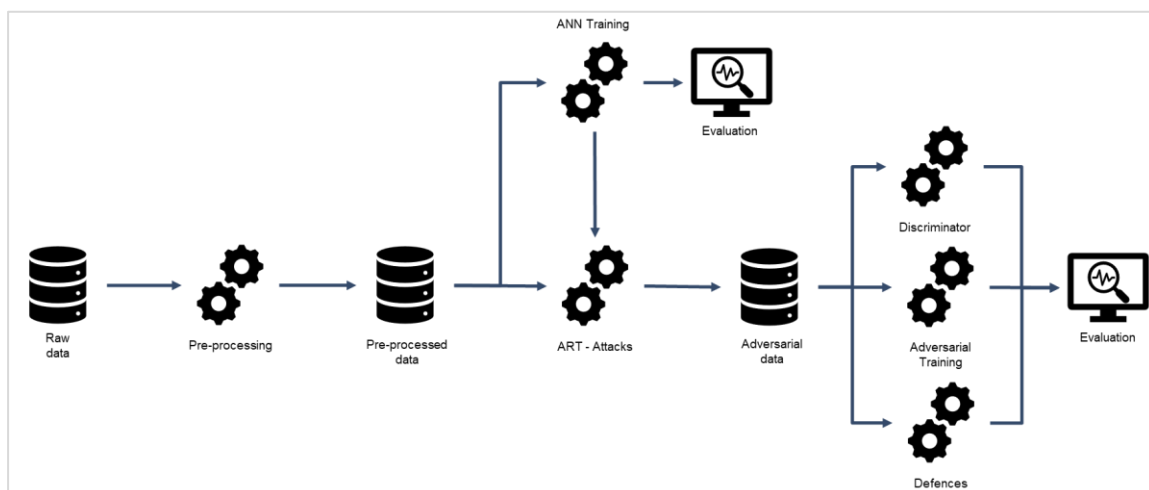


Figure 8 Experimental testbed flow

4.2 Datasets

AI solutions are data driven and this implies that the methodology that empowers an AI system needs to be tailored to the data and the environment in which the system is destined

to operate. In the context of STAR, the AI Cyber Defence module will be a core artifact of the data governance and security layer and is expected to provide its protection capabilities in the majority of the pilot use cases. Thus, taking advantage of the developments of T2.4 on the collection and documentation of STAR datasets, we gathered datasets from the pilot sites in order to perform the evaluation of the AI Cyber Defence considering the peculiarities of the pilot sites.

At this stage of the project, as given the maturity of the STAR pilots, the evaluation was focused on the datasets provided by PHILIPS, as part of the Human-CoBot Collaboration for Robust Quality Inspections pilot. Visual inspection is the most indicative case to demonstrate the applicability of adversarial attacks and defences, as the most prominent adversary tactics, techniques of the literature are demonstrated in such kind of application domains.

Given the above, in this deliverable we report our developments based on the following datasets:

- Dataset1: Decocap
- Dataset2: ShaverShell

Because of the peculiarity of each dataset, we had to process each of them with a different approach and to apply different techniques in order to maximise the potential of the AI solutions to be evaluated and get the maximum of them in terms of performance. In other words, the Decocap dataset made it difficult even for the human eye to distinguish the differences among the two out of three categories. On the other hand, the ShaverShell dataset was easier to handle and perform the feature extraction.

4.2.1 Decocap dataset

The decocap dataset was a set of 592 samples which contains 203 good samples, 198 samples of defective product images having flowlines, and 191 samples having marks, as shown in the image below. The classes of the dataset imply that we can treat the classification problem both as multi-classified one, having 3 distinct classes, as well as a Binary problem by merging the flowlined and the marked samples in a broader class of defective products.

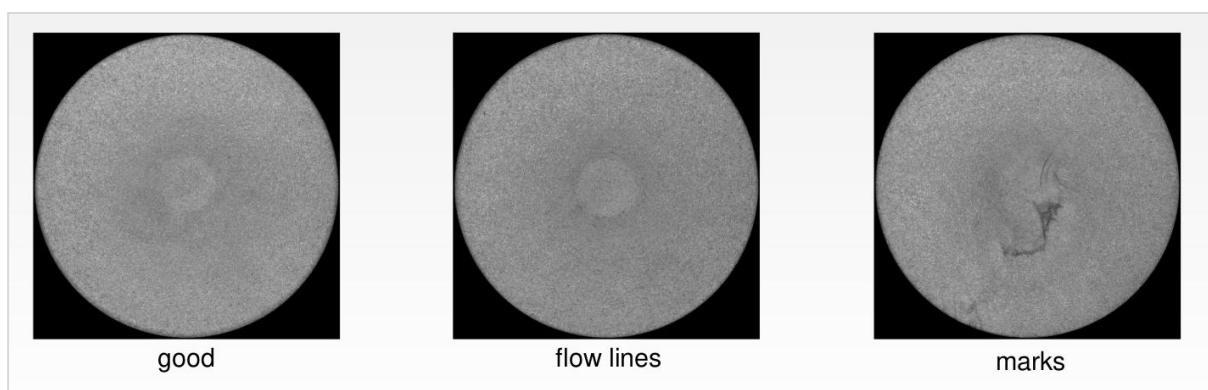


Figure 9 Decocap dataset sample

In order to get a more informative basis for our experiments we performed pre-processing steps on the decocap images.

Initially, we transformed the images to arrays using Keras library. In order to make sure that all the features would be visible and straightforward for our model to use, filtering techniques were applied. More specifically we used exposure filters and an established technique, namely HOG (Histogram of Oriented Gradients). HOG is a method of feature extraction, mostly used in computer vision and image processing applications for object detection. This method counts events of gradient orientation in a specific portion of an image or a region of interest that we choose. For the need of our experiments, we defined HOG as shown in the following code snippet. Finally, we normalized the arrays of the images in ranges [0,1], in order to get the output shown in Figure 10.

```
sample_image_features, img = hog(img, orientations=5, pixels_per_cell =
(2,2), cells_per_block=(1,1), visualize=True, block_norm='L2-Hys')
```

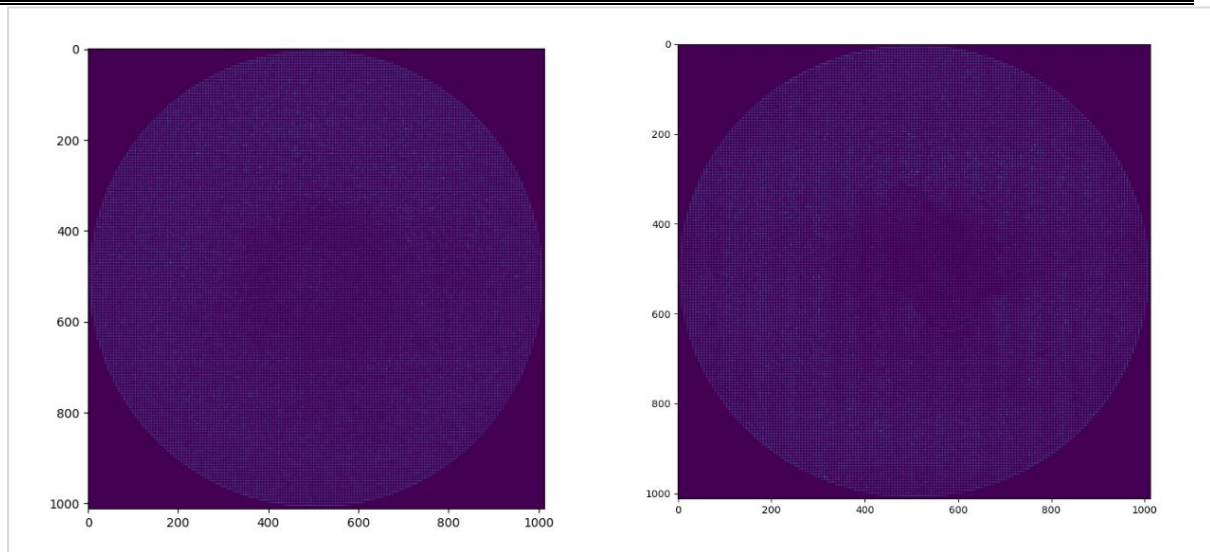


Figure 10 Decocap image samples after applying HOG method

4.2.2 Shavershell dataset

The shavershell dataset contains 2700 good samples, 244 samples of double printed labels on Philips’ shavers and 620 of interrupted prints, as show in the figure below. The classes of the dataset imply that we can treat the classification problem both as multi-classed one, having 3 distinct classes, as well as a Binary problem by merging the double-printed and interrupted print images in a broader class of defective products.



Figure 11 Shavershell dataset image samples

Again, the pre-processing of the shavershell dataset made sure that all the features would be visible and straightforward for our model and filtering techniques were applied. In this dataset because of its simplicity, it was easier to decide which image belonged to which category. That is, we only used exposure and Otsu's Thresholding Technique to make image features even more visible. Next, a normalization in range of $[0,1]$ was made on our data, in order to get image samples shown in Figure 12.



Figure 12 Otsu's Thresholding Technique result on shavershell image sample

As it will be shown later, after a series of experiments it was found that our models could have better results when the data were separated in two categories instead of three. Considering this, we built separated experiments for binary and multiclass datasets by setting different labels to the images each time. To exemplify this, our datasets were separated into "good" and "bad" samples, in which the "bad" samples indicate the merging of both sample of the defective products.

4.3 Pre-processing

Prior to any experiment, pre-processing of the images needs to be performed, as the original images were not able to provide to the model a sufficient set of features for the training task. Thus, a pre-processing approach was designed in such a way that each dataset reveals its main features. The pre-processing was done using Keras pre-processing library [REF-23] in order to load the data and transform the images into arrays.

After visualising and exploring the data, we applied established techniques of computer vision filtering. Normalization of the data was implemented in a specific range, as well as categorization of the images in specific categories, to be used later for model training. In order to explore the efficacy of the attacks and defences and to pinpoint the optimal evaluation strategy we separated the data in different ways, by dividing them into two or three subcategories (binary and multi-classes classification approaches).

4.4 Base Model Training and evaluation

According to Figure 8, the next step is to build and train a base model. This model will be used for triggering the ART - attacks, so that we can create our adversarial examples and it will be used as the baseline of our evaluations and statistical reports. The training of the base model is actually a simulation of a conventional AI-based visual inspection system, which tries to differentiate good from faulty products. We assume that this is the model which is being attacked by an adversary who tries to poison or evade the corresponding quality assurance system.

In this evaluation we built the model using Keras layers, Sequential, Dense, Flatten, Conv2D, MaxPooling2D, Activation and Dropout, with activation functions that are slightly different whenever we have to deal with a binary dataset or a multi-classed one. The Keras Classifier, as provided by ART tool, is used to train and evaluate our model. For the evaluation we used confusion matrices and classification reports, but we used accuracy as our main comparison metric. To evaluate our model, we used test sets that our model had never seen before.

4.5 Creating Adversarial Examples using attacks from Adversarial Robustness Toolbox

Having trained the baseline model, we use it as a main parameter to form an attacker. Until the moment of writing this deliverable, we have tested several attackers from the ART (Adversarial Robustness Toolbox), so that we can compare the accuracies of our model on the generated adversarial examples. The notable adversarial techniques tested in the context of the initial version of the cyber defence mechanisms are the following:

- Fast Gradient Sign Attack (FGSM) [REF-04]
- DeepFool [REF-48]
- NewtonFool [REF-49]
- Projected Gradient Descent (PGD) [REF-50]
- BasicIterative using PGD [REF-51]
- SpatialTransformation [REF-52]
- SquareAttack [REF-53]
- CarliniLIn Method [REF-54]
- CarliniL2 Method [REF-54]
- Universal Perturbation using EAD-elastic-net attack [REF-55]

These functions were designed in such a way that will offer the option to the user to determine how effective (or not) the attack wants to be. An important step in this process is that the attackers, i.e., the object instances created as part of our codebase, are taking as a parameter the baseline classifier build on the on the previous step and the original pre-processed data to be able to create the adversarial examples. By defining and executing these attackers, we acquire the adversarial examples that are similar to our original pre-processed images but this time including noise according to the respective technique deployed by the attacker. An illustration of one original and its adversarial example is given in Figure 13 and Figure 14.

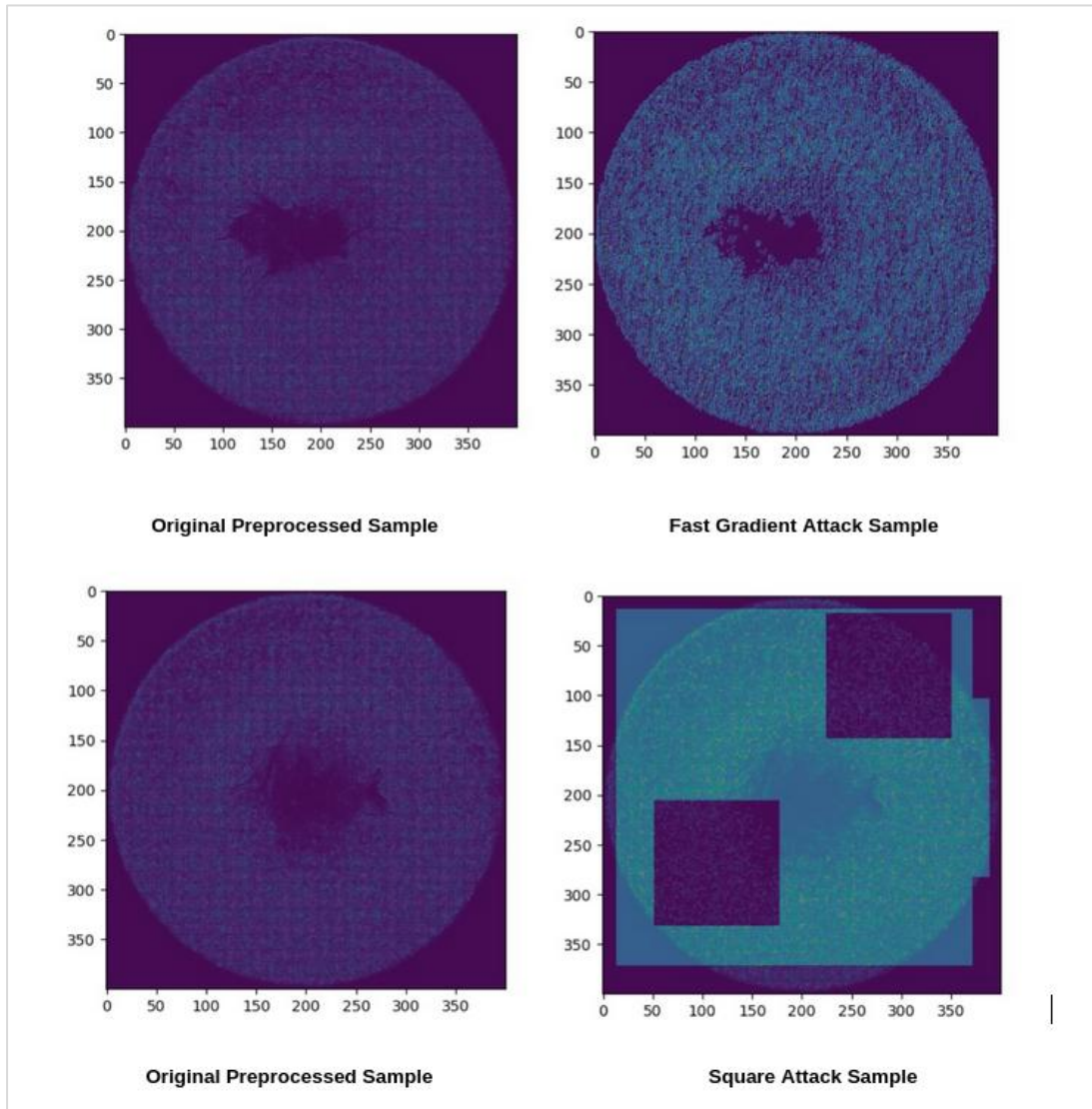


Figure 13 Example of adversarial examples for decocap dataset

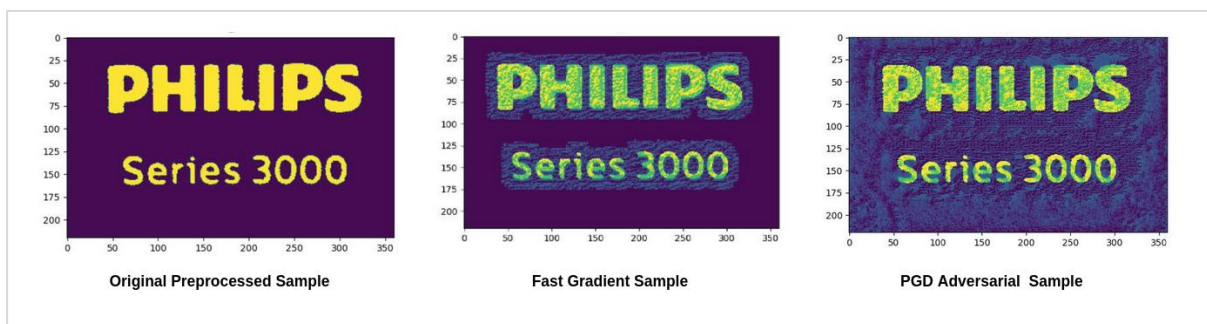


Figure 14 Example of adversarial examples for shavershell dataset

In order to compare our baseline model's accuracy when it has to classify adversarial examples, we measure its accuracy. It is expected that, if the attack is successful, the accuracy will be declined dramatically, depending on the robustness of the baseline model against the noise which is injected in the images.

4.6 Defence Strategies

4.6.1 Adversarial Training

Adversarial training is perhaps the most prominent defence strategy to be applied to an attacked model. The basic idea is to simply create adversarial examples that later on will be used in the training process. In other words, since we have a decrease of accuracy when we evaluate our model on adversarial examples, what is needed is to train the model on a few more adversarial examples.

In the context of the initial version of our framework, we append our original data with the adversarial examples created by our attackers, to build a new larger dataset. We use this new dataset that contains our original data and the adversarial examples to train our model. Afterwards, we re-evaluate our model and, if the adversarial training proves to be an effective strategy, it is expected and increase to the accuracy of the model.

In our experiments we applied the adversarial training as a defence against each attack separately in order to have adversarial examples from only one attacker.

4.6.2 ART Defence strategies

The adversarial training is one form of defence that implies that need to create adversarial examples and inject them in the training process on a model. However, there have been several research attempts to provide defence techniques that aim to either enhance the robustness of the model per se, or to focus on the data and apply specific pre-processing methods.

Thus, in STAR we also consider different defence approaches. More specifically, we make use of specific defences that have been proposed in the literature and offered from ART, in order to evaluate our models again after an attack and after the application of a defence so that to observe the differences and perceive the efficacy of the latter.

In this initial version of the AI Cyber Defence tool, we evaluated the following defences:

- FeatureSqueezing [REF-06]
- JpegCompression [REF-46]
- SpatialSmoothing [REF-16]
- TotalVarMin [REF-47]

We need to note that because of the structure of the algorithms we only used the adversarial example test set to evaluate the defences and not both the train and test adversarial example sets. To make sure that the defences differentiated the adversarial examples in any way, at least observable with the naked eye, we used visualization tools to illustrate the images. An example of adversarial examples and images produced after the defence methods can be seen in Figure 15 and Figure 16.

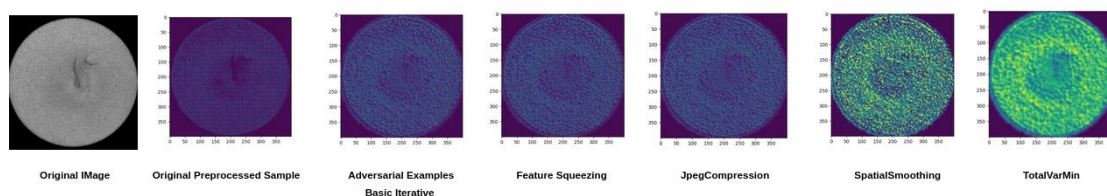


Figure 15 Application of defence methods on the decocap dataset



Figure 16 Application of the defence methods on the shavershell dataset

4.7 Evaluation results

This section elaborates on the results of the evaluation after running and collecting statistical data on the models' performance. Our evaluation and discussion are based on collective plots that present an overview on the efficacy of the attacks and defences tested.

That is, these plots were specially designed to visualize the comparison of the accuracy of a model, before any attack, on the attacked data and on the defensive scenario that we had chosen each time. More specifically, depending on the scenario applied, we evaluated the model using the accuracy metric, i) on the adversarial training, ii) on the attacked data that were subjects to the defending algorithms, and iii) the discriminator model on original and adversarial samples.

The evaluation and the produced plots occurred using a standard validation technique known as K-fold Cross Validation. K-fold Cross Validation is an expansion of the simple Cross Validation which is used in machine learning to estimate how good or bad the accuracy of the model will perform in different portions of the dataset. By using this technique, we also managed to avoid overfitting of the model on the data, as before the model training the dataset is split in different train and testing sets. More specifically, in our experiments we used K-fold to divide the dataset into 5 for Shavershell and 10 for Decocap, groups of different train and test sets.

In addition, as we mentioned before, our evaluation testbed is built on the basis of both a Binary and a multi-classed classification problem that imitates an AI-based visual inspection system for quality assurance in a production line.

4.7.1 Binary classification approach

During the evaluation of our model performance, the data were separated in two categories instead of three. However, according to the accuracies collected, it seems that our model had slightly better performance on binary classification over multiclass classification in both datasets. This is something that it was expected, as it is easier for a model to make correct predictions as it was trained on a larger number of samples of the "bad" category.

4.7.1.1 Adversarial Training

Taking into account the plots in Figure 17 and Figure 18, on the Decocap dataset the adversarial tactics (orange bars) had without a doubt a greater impact on the baseline model's accuracy than on the Shavershell dataset. Intuitively, one can understand why this happens, as the descriptive features in the decocap images are not as obvious as in the shavershell, the noise injection of the attacker can fool the model easier. When it comes to the effectiveness of the Adversarial training, in most of the cases, this approach is in position to revitalise the model and achieve an accuracy level which is close to the original model, and in some cases to even surpass it. However, in some cases the adversarial

training model achieves an accuracy score around 65%, which is a proof that some adversarial attacks can have considerable impact even against robust defences.

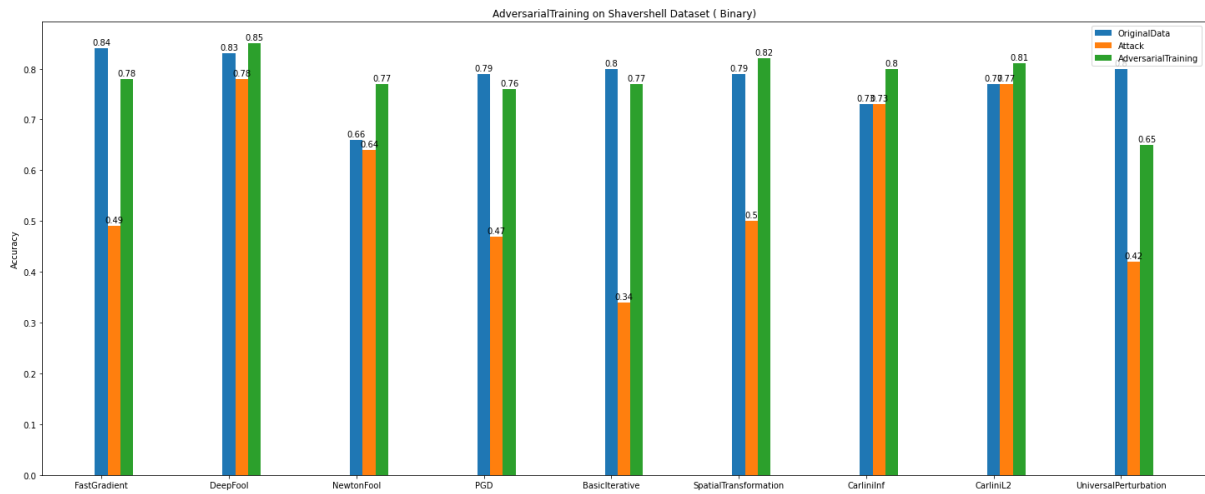


Figure 17 Collective results using Adversarial Training for the Shavershell Dataset in Binary classification mode

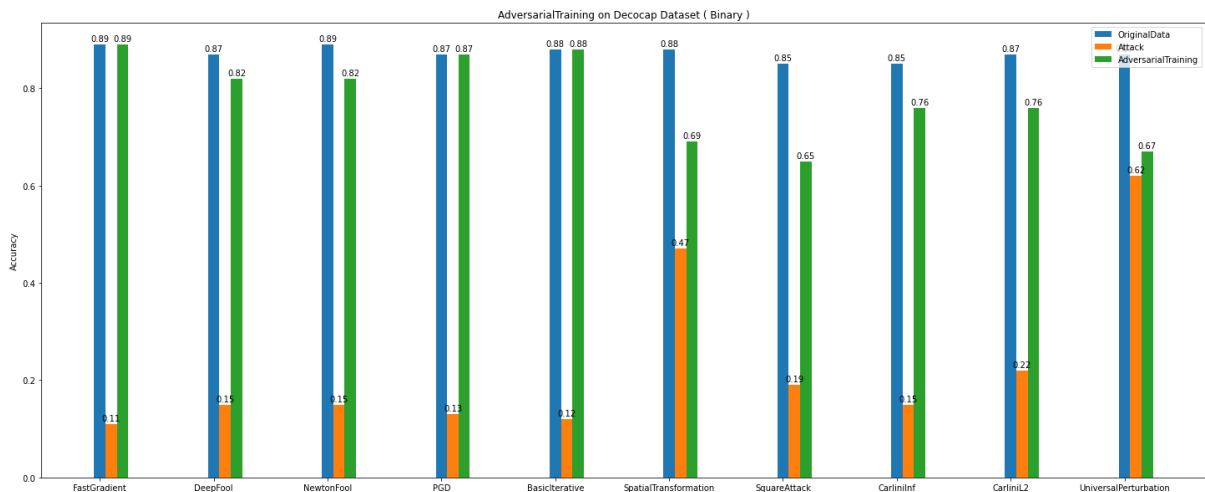


Figure 18 Collective results using Adversarial Training for the Decocap Dataset in Binary classification mode

4.7.1.2 ART Defence Strategies

Under the binary classification approach, off-the-shelf defence methods have been evaluated and the collective results are given in Figure 19 and Figure 20.

For the Shavershell dataset, the top defensive algorithm is the TotalVarMin. Another notable point is that the Carlini's family attackers do not have major impact on this dataset, so the accuracy of our model is easier to increase as the model is trained on even more data. SpatialSmoothing is noted as the defence with the best performance.

For the Decocap dataset, the SpatialSmoothing's defensive algorithm seems to compete with TotalVarMin. More specifically this reaction is noted against the following attack algorithms: NewtonFool, SpatialTranformation, CarliniLInf, CarliniL2, UniversalPerturbation (EAD). Among the defences, the JpegCompression is the one that achieves better results.

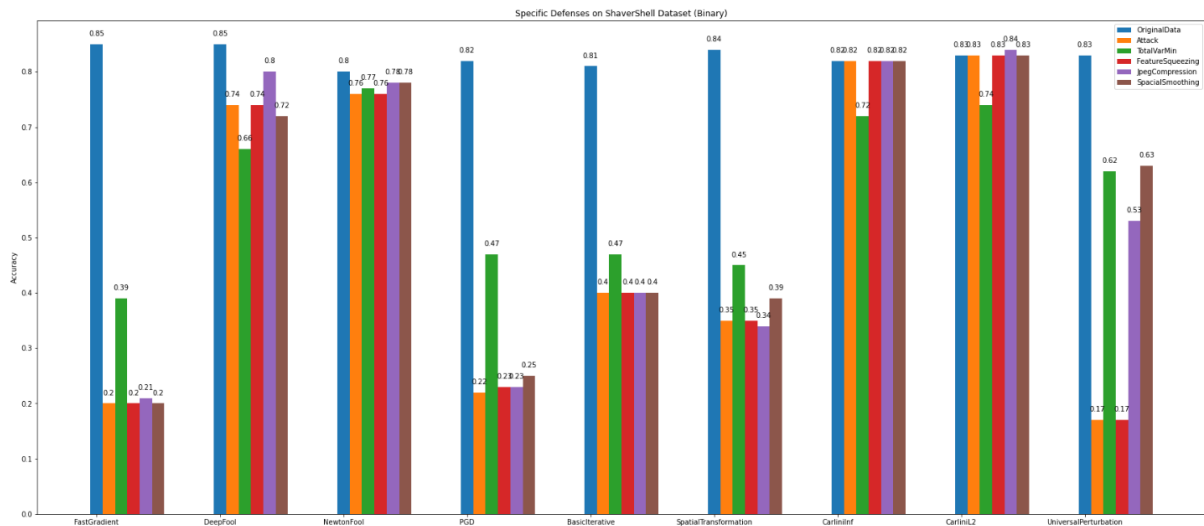


Figure 19 Collective results using ART defence strategies for the Shavershell Dataset in Binary classification mode

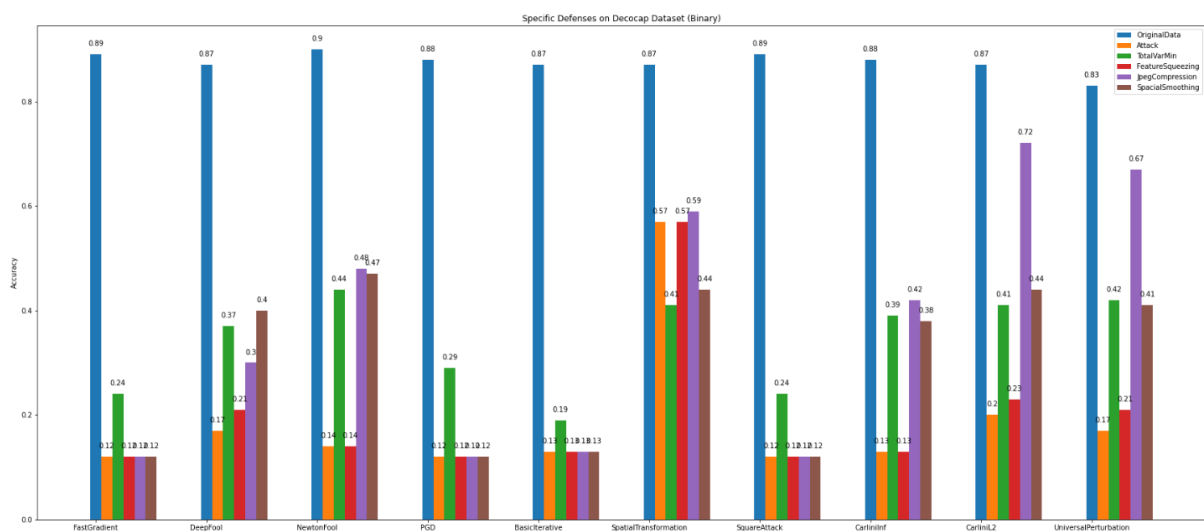


Figure 20 Collective results using ART defence strategies for the Decocap Dataset in Binary classification mode

4.7.2 Multi-classed classification approach

During the evaluation of the models' performance, the data kept the initial categories, as reported in Section 4.2, thus, three classes were used during the following rounds of experiments. The multi-classed approach was proved a more challenging testbed, as the addition of the classes makes it more difficult to make correct predictions.

4.7.2.1 Adversarial Training

Regarding the multi-classed case of the shavershell dataset under the adversarial training approach, Figure 21 offers the collective results. It can be seen that, it is difficult for some attack algorithms, e.g. DeepFool, NewtonFool, CarliniLInfMethod, CarliniL2Method, to achieve their goal, which is to destroy the robustness of the model. On the other hand,

Decocap dataset is more vulnerable. It is obvious that the attack algorithms can modify the samples and as a result the models' accuracy is decreased. Again, the adversarial training, as a defence strategy, seems that is able to revitalise the accuracy of the end-model and in some case to even perform better than the original model. As aforementioned, the Decocap dataset proved a challenging case and the same applies for adversarial training results, as for most of the cases in Figure 22 the results revolve around the 75%, showing that even such a robust method does not always suggest a credible method.

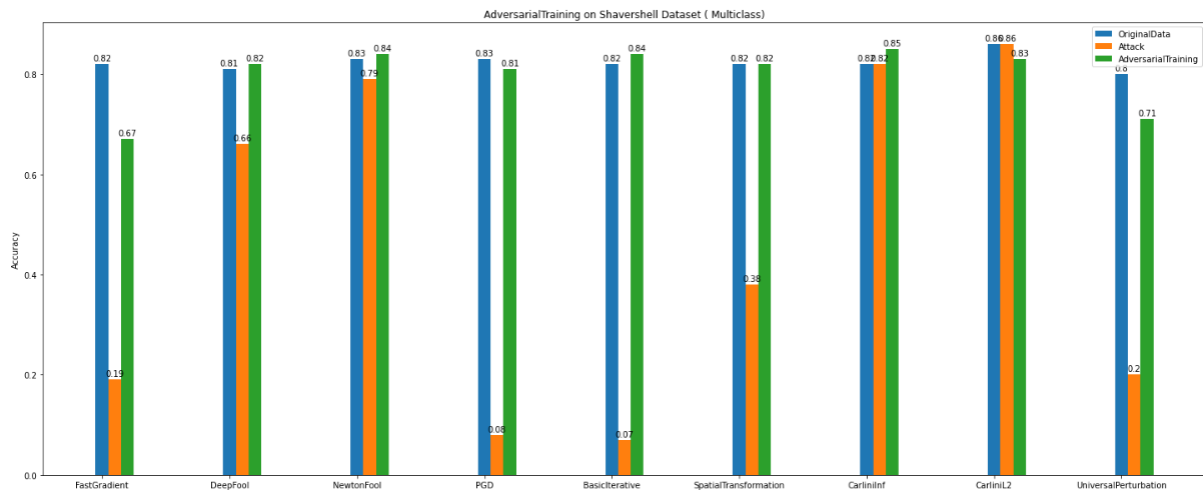


Figure 21 Collective results using Adversarial training strategies for the Shavershell Dataset in multi-classified classification mode

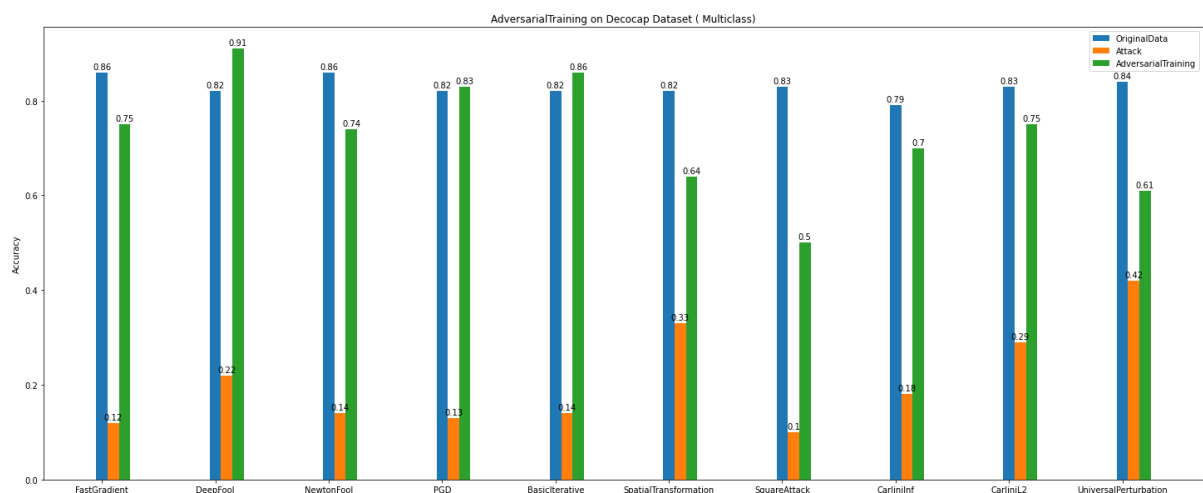


Figure 22 Collective results using Adversarial training strategies for the Decocap Dataset in multi-classified classification mode

4.7.2.2 ART Defence Strategies

As was the case for the ART defence strategies in the binary task, we notice again that it is hard for the defence algorithms to cope with the attacks.

The evaluated defences on the shavershell dataset have failed to defend the model, as can be seen in Figure 23. For the cases, that the attacks did not significantly affect the accuracy

of the model, the defences make a slight contribution in the model’s improvement. The SpatialSmoothing is the defence that achieves the higher delta between the attacked model and the one after the defence approach. For the Decocap dataset jpegCompression, was able to resist against the NetonFool, the CarliniL2Method and the UniversalPerturbation (EAD), as shown in Figure 24.

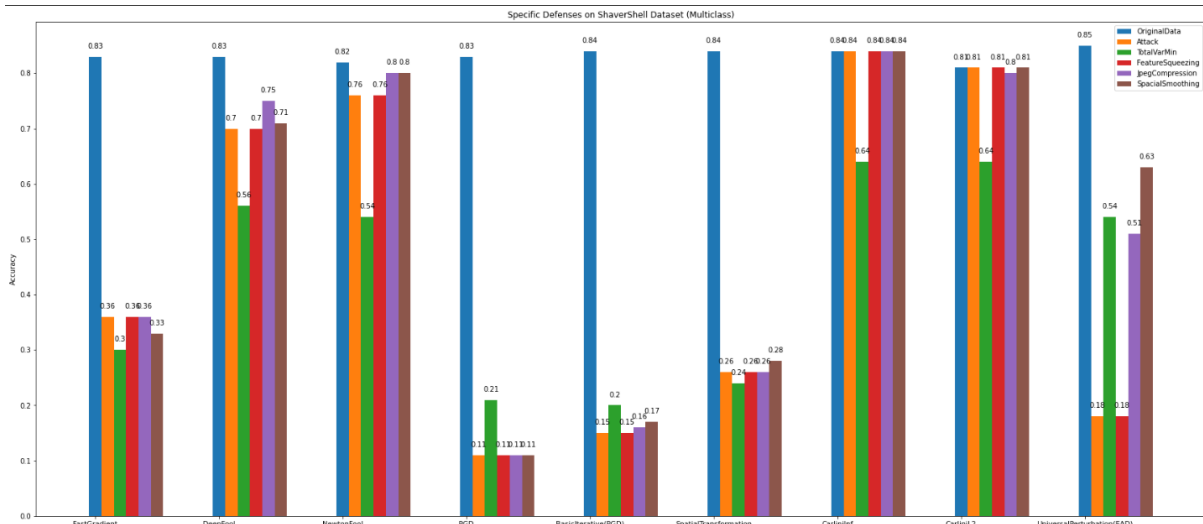


Figure 23 Collective results using ART defences strategies for the Shavershell Dataset in multi-classed classification mode

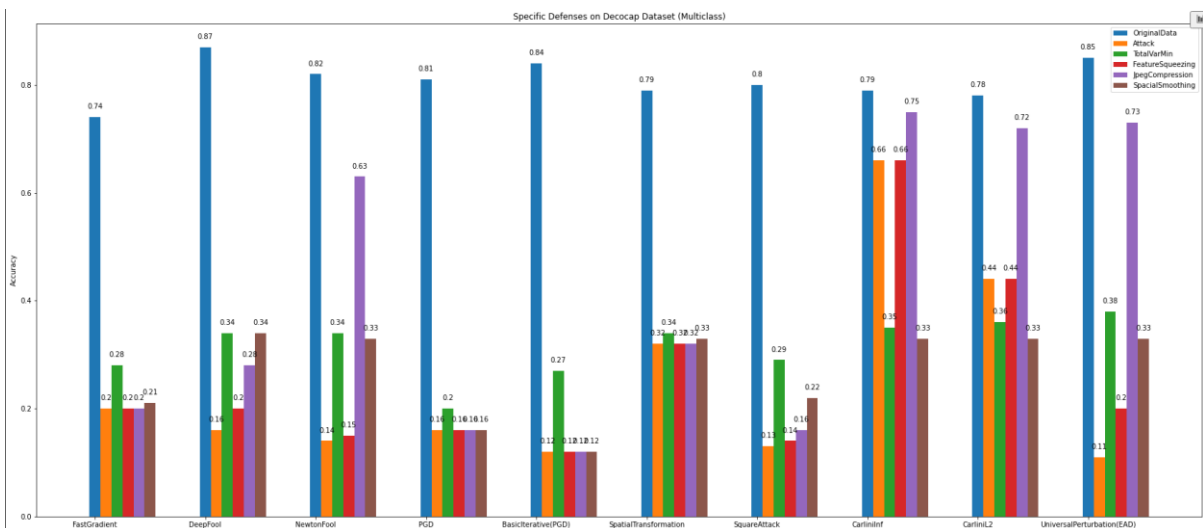


Figure 24 Collective results using ART defences strategies for the Decocap Dataset in multi-classed classification mode

4.7.3 Discriminator

As an extension of the adversarial training approach detailed in Section 4.6.1, the discriminator is a model that tries to distinguish real data from the data created by the attacker or in other words the adversarial examples. The discriminator is destined to play an

important role in the STAR WP3 architecture, as it will be a core AI model that will empower the AI Cyber Defence module.

For the creation of the Discriminator, one could use any neural network architecture appropriate to the type of data to be classified. In this evaluation testbed, we used the baseline model as a discriminator, but this time by compiling a dataset with the only difference that after the attacker creates the adversarial examples, we label them with a different label.

More specifically, in the previous scenarios of the adversarial training approach, even if our sample was an adversarial example, we still labelled it with its original label (e.g., good, flowlined, marked). In this experiment we assign a new and distinct label to the adversarial examples, so that the model will be trained and then be able to predict if a sample is an adversarial sample or not and from which specific category of our original dataset the adversarial example has been generated from.

This is how the evaluation data are separated for training the Discriminator. Initially, the training of the baseline model takes place considering the basic N categories of a dataset. Then, the generation of the adversarial examples is performed in order to derive another N categories of data. By combining these two datasets, we form superset on 2 x N classes including both the original images and the original images that were perturbed so that to generate the adversarial examples.

Taking a look over the collective results of Figure 25 referring to the Decocap’s Multiclass Discriminator bar plot, the overall performance of our model meets our expectations. There are some cases e.g., NewtonFool, CarliniInf, CarliniL2, UniversalPerturbation(EAD), where the results are not as good to meet the goal of the experiment. However, for some cases, e.g., for the PGD attack algorithm, the discriminator appears to be fully capable of predicting and understanding the differences between the original and the adversarial examples data.

Figure 26 illustrates the results for the Decocap dataset, where, again, depending on the selected attack the Discriminator can achieve remarkable results (e.g., PGD, BasicIterative, SpatialTransformation). Of course, in some cases the model was not able to perform adequately.

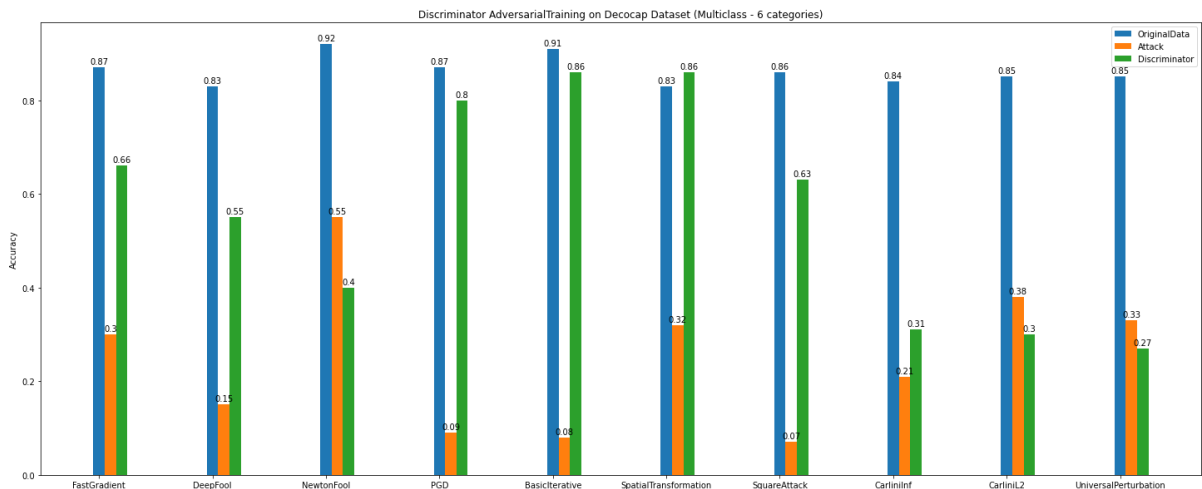


Figure 25 Collective results using the Discriminator approach for the Decocap Dataset in multi-classed classification mode

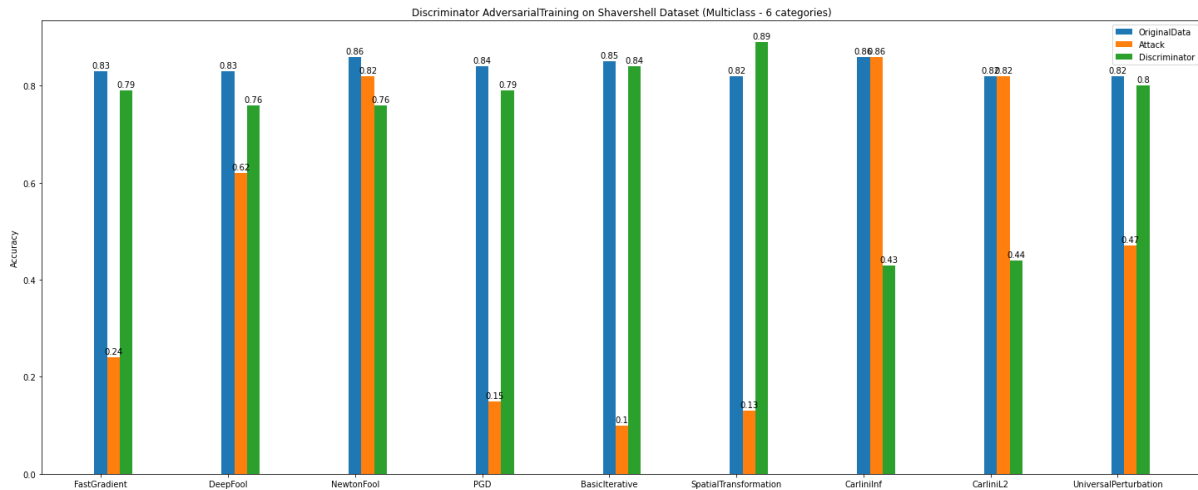


Figure 26 Collective results using the Discriminator approach for the Shavershell Dataset in multi-classed classification mode

4.7.4 Summary

By summarizing the evaluation performed in this section, we can conclude that each case and dataset may require a different defence strategy to be deployed in order to safeguard a baseline model. Different adversarial approaches can be addressed by different defences, but there is no clear indication that there is a defence strategy that can cover the wide range of attack techniques in an adequate manner.

As expected, between the adversarial training and the tested ART defences, the former seems to be the most effective way for the recovery of the overall model robustness. According to the results of the plots that demonstrate on average the increase of the accuracy after the adversarial training technique, one can observe that it has better performance than the evaluated off-the-shelf defences offered by ART. Of course, this does not imply that these defences should not be considered in the future experiments of STAR, but there is evidence suggesting that, at least for the decocap and shavershell datasets, they cannot limit the impact of an adversarial technique.

The evaluation testbed was extended also by the discriminator component in Section 4.7.3. Given the above, we extended the experimental testbed to build a component, namely Discriminator, in order to adopt the advantages of the adversarial training but in parallel to provide a setup which is acceptable from a business point of view, where the adversarial samples would not be part of the good samples, as is the case for the adversarial training. Again, the challenging nature of the problem that T3.2 and T3.3 aim to tackle, i.e., the detection of poisoning and evasion attacks, is verified. As reported by the results, each dataset and each different attack imply the need for the placement of different defence strategy.

5 Towards enhanced data trustworthiness for AI systems

This section offers a summary of the work published in [REF-03], as an outcome of the research actions conducted in tasks 3.2 and 3.3, and as a research direction of the long-term vision of UBITECH for developing AI defence methods against poisoning and evasion attacks.

5.1 Introduction

The advent of IoT edge devices aid in the digitisation of the manufacturing domain and lead to the emergence of a wide gamut of services and applications which are based on unprecedented quantity of data. Manufacturing floors are subjected to inherent data trustworthiness challenges due to factors such as malevolent input and faulty sensors.

Thus, there has been a plethora of proposed solutions, some of them have been already evaluated in Section 4, but there are still a number of open issues to be considered: how to cope with the presence of strong, colluding adversaries while at the same time efficiently managing this high influx of incoming data. That is, the long-term vision of UBITECH's team is to meet these challenges by proposing the hybrid use of Deep Learning schemes (i.e., LSTMs) and conventional Machine Learning classifiers (i.e., One-Class Classifiers) for detecting and filtering out false data points. A prototype implementation coupled with a detailed performance evaluation under various (attack) scenarios, employing both real and synthetic datasets was published in [REF-03], as an outcome of the work conducted in the context of STAR. This work will serve as the basis for further research in the context of T3.2 and T3.3 of STAR.

In distributed environments, comprising from multiple devices and sensors, the main hurdle is how well ML mechanisms can operate, as many untrustworthy data sources providing falsified data may be part of the (either as the result of an attack or a malfunctioning sensor) environment.

Compounding this issue, Banerjee et. al [REF-02] studied how concept drifts and false data can masquerade the existence of intelligent attackers and their impact on the accuracy of both the clustering and (unsupervised) classification processes; however, they did not investigate the integration of more advanced Deep Learning schemes. What is needed is a set of advanced deep learning mechanisms for assessing and sifting faulty data without any assumption on the trustworthiness of their sources while coping with intelligent adversaries that try to significantly decrease the overall performance and cause targeted misclassification.

Towards this goal, we investigate for a data verification framework employing deep learning techniques to address the issue of possible data poisoning. We define malicious data as falsified data exhibiting different statistical properties (from the real data provided by benign sources), and our approach is leveraging sequential relationships of sensory data towards detecting malicious samples generated by adversaries or faulty components.

More specifically, our solution named False Sequential data Detection (FSD) offers:

- (i) Data verification by combining Deep Learning sequential architecture of Long Short-Term Memories (LSTMs) with conventional one-class classifiers for distinguishing between "false" and "real" samples
- (ii) A Proof-of-concept implementation evaluated under various testing scenarios using both real and synthetic datasets in order to have more flexibility on the type of experiments. FSD demonstrates high accuracy even when the distribution of data comes from adversaries that demonstrate very similar behaviour with the legitimate user data, i.e., strong colluding adversaries by composing two main attack strategies that represent different aspects of adversarial machine learning.

5.2 Conceptual Architecture

The main motivation behind our endeavour is to benefit from the relationships between samples of different time orders so as to accurately predict the values of the next time step. It uses the distance metrics for comparing the predicted data and real data, in each time step, in order to prepare samples for the one-class classification approach, i.e., draw a boundary around the allowed deviations between predicted and real data values.

Our approach consists of two main phases, namely the Training and Testing phases. In the Training Phase, we build a predictor for each sensory data as well as a final one-class classifier whereas during the Testing Phase, we feed a combination of malicious and benign testing samples in order to evaluate how good our trained models behave in separating malicious data from benign values.

5.2.1 Training phase

Figure 27 depicts the underpinnings of the training phase comprising of three steps: 1) training of LSTMs as sequential predictors for each sensor, 2) measuring deviations between predicted and real data vectors, for each time step, and 3) training an one-class classifier on vectors of the observed deviations in order to detect anomalies according to real and expected predicted values (again for each time step).

1. **Training LSTMs:** In this step, FSD trains separate LSTM models, M , for predicting values of each sensor, for each time step. As LSTM is useful for processing, classifying, and making predictions based on time series, we employ LSTMs on time series of sensor data. This step results in m trained models based on m sensors, and M_j would be a trained model on S_j for predicting the $p_{i,j}$ value of sensor j at time step i .
2. **Measuring deviations between predicted and real values:** After training our LSTM models (M_s), our method feeds all S_s in parallel (S_j to M_j) and store all predicted values ($p_{i,j}$) in matrix P where each row contains all predicted values by M_j on S_j for each time step. P_j is column j of P showing the predicted values for sensor j for time step 1 to n , and $P_i = [p_{i,1}, \dots, p_{i,m}]$ is a vector of all predicted values for m sensors at time step i referring to rows of P .

At time step i , we calculate deviations between predicted vector (P_i) and real vector (V_i) into a new vector $\delta_i = [d_{i,1}, \dots, d_{i,K}]$ using K distance/similarity metrics such as Euclidean distance, cosine similarity, Manhattan distance, or other customized metrics with the ability of calculating one single scalar to show the deviation between

the values of the two input vectors. Calculating the differences of predicted and real values we would have a new dataset Δ as a set of δ vectors with size K where $\delta_i = [d_{i,1}, \dots, d_{i,k}, \dots, d_{i,K}]$ and $d_{i,k}$ is the deviation between vector (V_i) and (P_i) calculated by distance metric k (e.g., Euclidean distance).

3. **Training one-class classifier:** After feeding all legitimate data to our trained models (M_s) and calculating δ_i for each time step i , we have $\Delta = [\delta_1, \dots, \delta_i, \dots, \delta_n]$ containing all allowed deviations between the predictions and the real data for all time step. Dataset Δ is suitable for one-class classifiers trying to find a boundary around the training samples in order to separate them from the data from other distributions, where M_{occ} in Figure 27 would be the trained one-class classifier.

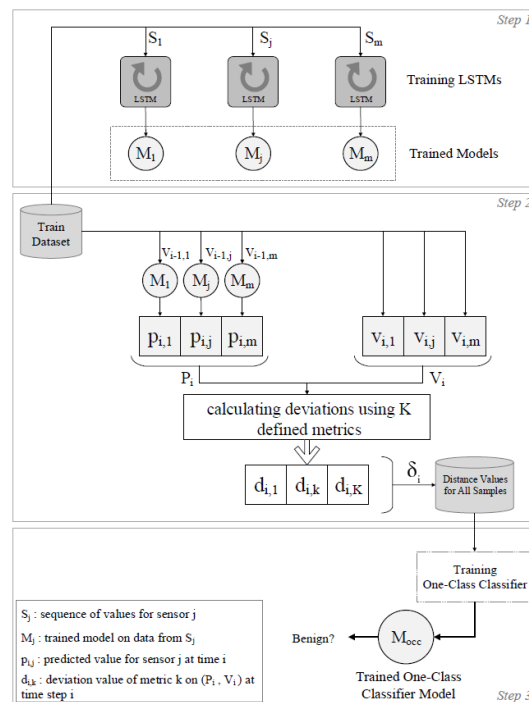


Figure 27 Training Phase of FSD

5.2.2 Testing phase

Figure 28 shows how our approach works in real time to detect adversarial samples. For each testing time step i , our method predicts P_i using trained models (M_s), and calculates δ_i as the distance vector between V_i and P_i . Then, M_{occ} classifies δ_i as Benign or Malicious. As M_{occ} is a trained one-class classifier on legitimate deviations between real and predicted values, the output dictates whether the vector of calculated deviations is legitimate.

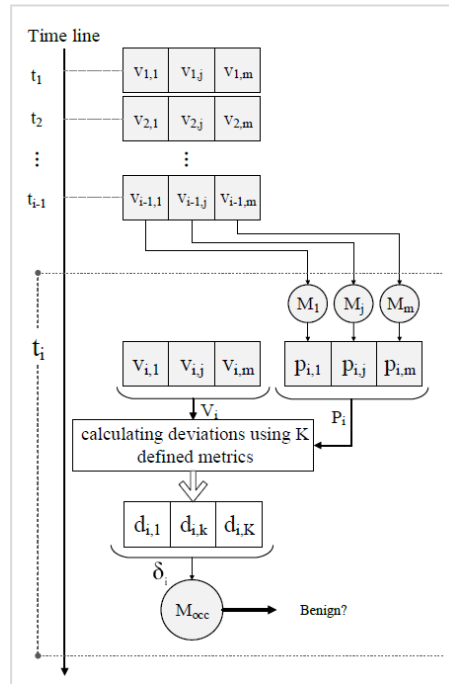


Figure 28 Testing Phase of FSD

5.3 Experimental Setup

In this section, we proceed with presenting our setup for implementing and evaluating FSD under various attack strategies. The dataset originates from real-world measurements collected from Data Sensing Lab [REF-45] by sensors deployed at the Strata Clara convention center in 2013 [REF-01]. The dataset contains sensor data from 5 different sensors namely, Temperature (Temp), Humidity (Hum), Pir, Motion and Microphone (Mic). Table 2 shows the distributions of each sensor type.

Table 2 Mean (μ) and standard deviation (σ) of base dataset

	Temp	Hum	Pir	Motion	Mic
μ	22.5150	36.1389	0.1514	-0.000386	5.2275
σ	1.6171	6.0058	0.3583	0.3544	5.1334

LSTM and One-class Classifier Architectures: As we are dealing with sequential sensor data with specific time steps, Recurrent Neural Networks based techniques – like the leveraged LSTMs - provide an attractive root of trust. In this context, in order to better justify the reasoning behind moving from Multi-Layer Perceptron (MLPs) towards more complex time-oriented classification techniques, we compare the best Root Mean Square Errors (RMSE) achieved by MLPs and LSTMs. Table 3 shows the RMSE values between predicted and real corresponding values for each sensor. We name LSTMs with 20 units as LSTM_{best} as it showed the best performance in our experiments with different number of units. LSTM with one LSTM unit (LSTM₁) and conventional Neural Networks with 10 hidden layers (MLP_{best}) are reported to compare with LSTM_{best}.

Table 3 Comparing RMSEs achieved by the best trained MLP model, one-layer LSTM, and best trained LSTMs.

	Temp	Hum	Pir	Motion	Mic
MLP_{best}	0.208	0.350	0.797	1.152	2.322
$LSTM_1$	0.037	0.081	0.347	0.521	1.832
$LSTM_{best}$	0.029	0.080	0.301	0.216	1.639

Adversarial Behaviour: As FSD is a framework for characterizing between inlying and outlying sensory data reports in the presence of adversarial malicious users, **we assume that colluding adversaries are attacking the data collection process (poisoning attack) or the classification process (evasion attack)**. Therefore, we have effectively categorized attack strategies into pre-training and post-training attack strategies respectively. We measure the performance of the FSD framework based on different levels of distortion that colluding adversaries try to impose on the data trustworthiness. We consider two main attack approaches for each attack strategy:

1. *Distribution attacks (Attack Cases I & II)*, which manipulate mean or standard deviation to generate adversarial samples which are different from the benign ones.
2. *Position attacks (Attack Cases III, IV & V)*, which manipulate the position of injecting adversarial samples in the sequence of values which in turn targets the order of samples. Position attacks can only be applied after a distribution attack for changing the order of samples to be injected in the sequence.

Attack Case I: Adversaries may affect the system uncertainty by modifying the true value of the sensory data by setting $\sigma' = \sigma$ and $\mu' \neq \mu$, which represents a malicious standard deviation equal to the standard deviation of the legitimate data points with smaller/larger μ as Equation (1) where λ is a scalar value as the deviation factor. Adversarial samples generated by $\lambda > 2$ are not attractive in our data as they are easier to detect due to their lower overlap with benign samples.

$$\begin{cases} \mu' = \mu + (\lambda \times \sigma) & 0 < \lambda \leq 2 \\ \sigma' = \sigma \end{cases} \quad (1)$$

Figure 29(a) compares the distributions of legitimate samples and adversarial samples generated by Attack Case I for a simple dataset with only two features, where $\lambda = 1.0$.

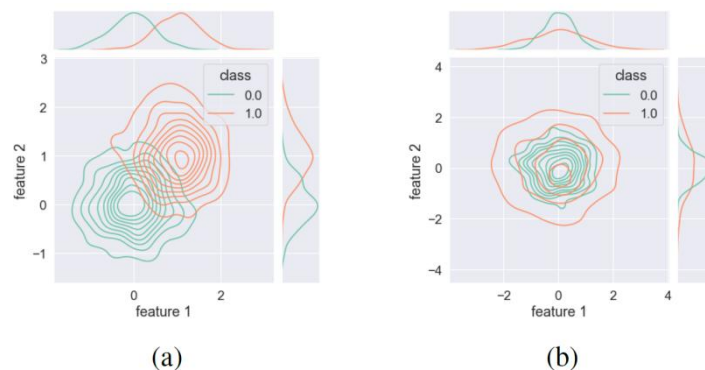


Figure 29 Distributions of legitimate samples (class 0) vs. adversarial samples (class 1) for two features with $\mu = 0$ and $\sigma = 1.0$; (a) Attack Case I, and (b) Attack Case II.

Attack Case II: Adversaries may affect the system uncertainty by selecting an adversarial distribution based on equation 2 where the adversary's goal is to keep the same mean but changing the standard deviation.

$$\begin{cases} \mu' = \mu \\ \sigma' = \sigma + (\lambda \times \sigma) \quad 0 < \lambda \leq 2 \end{cases} \quad (2)$$

Figure 29 (b) shows the distributions of a dataset with legitimate and adversarial data from Attack Case II with $\lambda = 1.0$. This attack case is designed to better reflect the real-world case scenarios where adversaries attempt to gradually change the classification behaviour by performing concept drifting by modifying the standard deviation over time.

Attack Case III: Adversaries may affect the system uncertainty by selecting a different distribution by changing the order of legitimate and malicious data points in the sequence of data. In this case all legitimate data comes before malicious data points in the sequence.

Attack Case IV: In this case all malicious data come before benign data points in the sequence as opposed to Case III.

Attack Case V: Adversaries may affect the system uncertainty by putting malicious batches of data in between legitimate data batches.

5.4 Evaluation

In [REF-03], we evaluated the performance of FSD in the presence of strong adversaries under the two aforementioned attack strategies (pre-training and post-training). We used F-Measure as the evaluation metric since it directly considers True/False Positive Rates as well as Recall and Precision metrics. The dataset is partitioned into two sub-sets of training and testing sets with 60% and 40% of data respectively. The interested reader can refer to [REF-03] in order to get more details on the results as it is out of the scope of this deliverable to replicate the entirety of the publication. This section aims only to highlight the methodology and provide a summary of the key outcomes.

5.4.1 Strategy 1: Impact of Adversaries in Pre-training

Summary of results on Distribution Attacks (Cases I & II): When it come to the attack cases I & II, the achieved performance results are significant in comparison to conventional machine learning techniques, as FSD achieved F-measures of 0.556 for Attack Case I and 0.516 for Attack Case II with a lot of malicious samples (40%) and a significant deviation $\lambda = 2.0$, which is higher than the random guessing. The results also show that FSD is more robust against the changes in mean (Attack Case II) in comparison to changes in standard deviation (Attack Case I).

Summary on Position Attacks (Cases III, IV & V): FSD achieved higher F-Measures in detecting samples generated by Attack Case III in comparison to the two other Attack Cases. In addition, an outcome to be highlighted is that the superiority of FSD in detecting samples from Attack Case III diminishes by increasing λ or the rate of malicious samples. FSD was more robust against attacks with one concept drift (Attack Cases III & IV) in comparison to attacks that try to trigger more drifts on the data preventing the models to learn enough knowledge from legitimate data. FSD achieved high performance in cases with

high rates of malicious samples implying superiority over conventional machine learning studied at [REF-02].

Table 4 An overview of the F-Measures of FSD under different Attack Cases III, IV and V in pre-training

	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
CaseIII-CaseIV	0.033	0.030	0.027	0.023	0.017	0.017	0.015	0.013	0.012
Case V-Case III	-0.018	-0.022	-0.023	-0.039	-0.047	-0.053	-0.062	-0.067	-0.067
Case V- Case IV	0.014	0.008	0.004	-0.015	-0.030	-0.036	-0.047	-0.053	-0.055

To give a comprehensive overview of FSD under various position Attack Cases, Table 4 compares the Attack Cases III, IV and V. To calculate values of Table 4, we subtracted and averaged all values of two comparing Attack Cases. Negative values in the table shows that FSD achieved higher F-Measures for the second attack case.

5.4.2 Strategy 2: Impact of Adversaries in Post-training

This strategy generates samples for bypassing the classifier during testing time (evasion attacks) without accessing the training dataset. Therefore, we first trained the FSD on legitimate data to evaluate this strategy. In post-training strategy, it is easier for a trained classifier to detect non-overlapping malicious samples as the classifier has the knowledge to cover the legitimate area. Therefore, an adversary with this strategy tends for lower λ since they have to follow the distributions of legitimate data with minor changes for a successful evasion attack.

Distribution Attacks (Cases I & II): FSD achieved higher F-Measures when we increased the rate of malicious examples (Adversarial Rate) because it was trained on legitimate data, so detecting malicious samples would be easier if the adversary injects more samples. This is because malicious data may create a new cluster with new distribution, so it is easier to detect them in cases with enough adversarial samples (higher rates of adversarial samples). Opposite to pre-training attacks, increasing the rate of adversarial samples may make it easier for the classifier to separate malicious data in post-training strategy.

The FSD achieved higher performance in cases where adversaries change σ' without updating μ' (Attack Case II). Similar to Attack Case I, there is an increase in performance of FSD along with the increase in λ and number of malicious data points in the post-training attack strategy.

Position Attacks (Case III, IV & V): When all benign samples appeared before malicious samples (Case III) in post-training More adversarial samples with higher deviation means more distance between benign and malicious clusters. FSD achieved high F-measure (0.903) in post-training strategy when the adversaries injected a lot of malicious samples (45%).

Under Attack Case IV in post-training the attacker attempts to affect the initial weights of LSTMs and as a result lower performance in comparison to Attack Case III was reported. An outcome to be highlighted is that FSD is more efficient if the adversary positions all benign samples at the beginning of testing data.

For the case where the adversary positions the malicious data in between benign samples, FSD showed better performance under this attack in comparison to Attack Case IV. However, the results of Attack Case V are more or less similar to Attack Case III.

Table 5 gives an overview of F-Measures for Attack Cases III, IV and V. The first row of the table shows that FSD achieved higher F-measures when the adversaries put benign samples before the attacking samples. The second row of table shows that increasing the rate of adversarial samples to more than 25% in Attack Case V makes the detection easier for FSD to detect malicious samples in comparison to Attack Case III with adversarial rate of $\lambda > 25\%$.

Table 5 An overview of the F-Measures of FSD under different Attack Cases III, IV and V in post-training

	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Case III-Case IV	0.097	0.070	0.065	0.062	0.061	0.060	0.059	0.057	0.055
Case V-Case III	-0.044	-0.016	-0.006	-0.003	0.000	0.001	0.003	0.005	0.012
Case V-Case IV	0.053	0.054	0.059	0.059	0.061	0.061	0.062	0.063	0.067

5.5 Summary

Data trustworthiness is a main challenge the STAR project aims to address. In this direction our work published in [REF-03] made the first step for the development of an AI-based defensive technique against poisoning and evasion attacks. Our approach is based on the use of LSTMs and One-Class SVM to build FSD. FSD improves data trustworthiness by providing the capability of detecting falsified data generated by adversaries under both pre-training and post-training attack strategies. In our experiments we designed five attack cases consisting of two distribution attack cases and three position attack cases to evaluate both attack strategies. FSD achieved higher performance when it visits more benign samples before malicious samples in both studied strategies. We prompt the interested reader to refer to [REF-03] to acquire more details on the evaluation results. As a future step, we aim to apply the developed method to the data provided by the pilot partners of STAR in order to explore the potential to deploy FSD as an additional module in the context of the AI Cyber Defence module.

6 Discussion and conclusion

This deliverable summarized the work performed in the context of T3.2 and T3.3 for the design, development and evaluation of methodologies for detecting poisoning and evasion attacks. In this context, the deliverable provides the initial version of the AI Cyber Defence module, this offers the aforementioned detection services as a main function of the AI security and data protection layer which aims to boost the safety, reliability and transparency of the functionalities of the upper operational layers of STAR.

Thus, this deliverable performed an analysis of state-of-the-art tools, libraries and methodologies that need to be considered in order to drive the development of the AI Cyber Defence module. Having this analysis as a solid background, the deliverable documented the positioning of the module in the STAR AI security and data protection layer, as well as the internal architecture of the module along with the defined interactions that provide an overview of its mode of operation. However, the AI Cyber Defence is part of a broader architecture where other tools operate to guarantee the decentralized data provenance and reliability for industrial data. Thus, the deliverable documented the details of the integration of the AI Cyber Defence module with the Runtime Monitoring System, the STAR Blockchain infrastructure and the XAI libraries of STAR.

Apart from the description of the system per se, a core contribution of the deliverable is the conduction of several experiments on the applicability of diverse AI attack and defence strategies, utilizing data stemming from the STAR pilot environments. Several evaluation results acquired and proved the challenging nature of the problem for which the STAR project aims to make a contribution. The results advocate that there is no single solution, "panacea", to deal with advanced adversarial techniques, as the robustness of a defense depends on the peculiarities of the data and the attack utilized each time by the adversary. However, we note the exceptional performance of the adversarial training approach as a solution that could empower a Discriminator that can distinguish between benign and adversarial instances.

Our research efforts, lead to the design of a prototype, called FSD, as reported in section 5. This section provided a summary of the methodology that constitutes the long-term research vision of UBITECH in the domain. Section 5 highlighted the conceptual architecture of FSD and the methodology which is based on the use of LSTMs and One-Class SVM for the detection of poisoning and evasion attacks. More results can be found in [REF-03].

As future action, and towards the 2nd and final release of the AI Cyber Defence module, we aim to perform more experiments by extending our evaluation testbed including more datasets and data types (e.g., text, sensory data, etc.) stemming from the STAR pilot environments, as well as by including more defense methods to be evaluated. Finally, we aim to extend the FSD prototype and apply it on the STAR pilots' datasets.

References

Reference	Name of document
[REF-01]	T. Giannetsos, S. Gisdakis, and P. Papadimitratos, "Trustworthy people-centric sensing: Privacy, security and user incentives road-map," in 2014 13th Annual Mediterranean Ad Hoc Networking
[REF-02]	N. Banerjee, T. Giannetsos, E. Panaousis, and C. Cheong Took, "Unsupervised learning for trustworthy IoT," 07 2018, pp. 1–8.
[REF-03]	Afzal-Houshmand, S., Homayoun, S., & Giannetsos, T. (2021, September). A Perfect Match: Deep Learning Towards Enhanced Data Trustworthiness in Crowd-Sensing Systems. In 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom) (pp. 258-264). IEEE.
[REF-04]	I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.
[REF-05]	N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defence to adversarial perturbations against deep neural networks," in 2016 IEEE Symposium on Security and Privacy (SP). IEEE, 2016, pp. 582–597.
[REF-06]	W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," arXiv preprint arXiv:1704.01155, 2017.
[REF-07]	X. Yuan, P. He, Q. Zhu, X. Li, Adversarial examples: Attacks and defenses for deep learning, IEEE transactions on neural networks and learning systems 30 (9) (2019) 2805–2824.
[REF-08]	J. Heo, S. Joo, and T. Moon, "Fooling neural network interpretations via adversarial model manipulation," arXiv preprint arXiv:1902.02041, 2019.
[REF-09]	T. B. M. Satya M. Muddamsetty, Mohammad N. S. Jahromi, "Sidu: Similarity difference and uniqueness method for explainable ai," accepted in IEEE International Conference on Image Processing(ICIP) - Jan 2020.
[REF-10]	Fenoy, L. M., and A. Ciontos. "Performance evaluation of Explainable AI methods against adversarial noise."
[REF-11]	A.-K. Dombrowski, M. Alber, C. J. Anders, M. Ackermann, K.-R. Müller, and P. Kessel, "Explanations can be manipulated and geometry is to blame," arXiv preprint arXiv:1906.07983, 2019.
[REF-12]	Fidel, G., Bitton, R., & Shabtai, A. (2020, July). When explainability meets adversarial learning: Detecting adversarial examples using SHAP signatures. In 2020 international joint conference on neural networks (IJCNN) (pp. 1-8). IEEE.
[REF-13]	Steinhardt, Jacob, Pang Wei W. Koh, and Percy S. Liang. "Certified defenses for data poisoning attacks." Advances in neural information processing systems 30 (2017).
[REF-14]	W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," arXiv preprint arXiv:1704.01155, 2017.
[REF-15]	K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: A statistical test for detecting adversarial examples," in International Conference on Machine Learning, 2019, pp. 5498–5507.
[REF-16]	A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," arXiv preprint arXiv:1905.02175, 2019.
[REF-17]	Makridis, Georgios, Dimosthenis Kyriazis, and Stathis Plitsos. "Predictive maintenance leveraging machine learning for time-series forecasting in the maritime industry." 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2020
[REF-18]	S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran

	Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
[REF-19]	Cretu, Gabriela F., et al. "Casting out demons: Sanitizing training data for anomaly sensors." 2008 IEEE Symposium on Security and Privacy (sp 2008). IEEE, 2008
[REF-20]	Croce, Francesco & Hein, Matthias. (2020). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks.
[REF-21]	Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2016.
[REF-22]	Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints, abs/1605.02688, May 2016.
[REF-23]	François Chollet. Keras. GitHub repository: https://github.com/fchollet/keras , 2015.
[REF-24]	Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.
[REF-25]	Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pages 372–387. IEEE, 2016.
[REF-26]	Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, George Loukas, A taxonomy and survey of attacks against machine learning, Computer Science Review, Volume 34, 2019, 100199, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2019.100199 .
[REF-27]	S. Afzal-Houshmand, S. Homayoun and T. Giannetsos, "A Perfect Match: Deep Learning Towards Enhanced Data Trustworthiness in Crowd-Sensing Systems," 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), 2021, pp. 258-264, doi: 10.1109/MeditCom49071.2021.9647554.
[REF-28]	Mahbooba, Basim & Timilsina, Mohan & Sahal, Radhya & Serrano, Martin. (2021). Explainable Artificial Intelligence (XAI) to Enhance Trust Management in Intrusion Detection Systems Using Decision Tree Model. Complexity. 2021. 11. 10.1155/2021/6634811.
[REF-29]	Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, and Jaehoon Amir Safavi. 2017. Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach. Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. Association for Computing Machinery, New York, NY, USA, 103–110. DOI: https://doi.org/10.1145/3128572.3140450
[REF-30]	Lurski, N., Younis, M. (2022). Application and Mitigation of the Evasion Attack against a Deep Learning Based IDS for IoT. In: Renault, É., Boumerdassi, S., Mühlethaler, P. (eds) Machine Learning for Networking. MLN 2021. Lecture Notes in Computer Science, vol 13175. Springer, Cham. https://doi.org/10.1007/978-3-030-98978-1_6
[REF-31]	Ziyi Bao, Mitigating Evasion Attacks against Machine Learning Systems through Dimensionality Reduction and Denoising. MEng Individual Project Final Report, Imperial College, London Department of Computing, June 2018.
[REF-32]	Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19). Association for Computing Machinery, New York, NY, USA, 793–803. DOI: https://doi.org/10.1145/3292500.3330961
[REF-33]	S. Al-Eidi, Y. Chen, O. Darwishand and A. M. S. Alfosoool, "Time-Ordered Bipartite Graph for Spatio-Temporal Social Network Analysis," 2020 International Conference

	on Computing, Networking and Communications (ICNC), 2020, pp. 833-838, doi: 10.1109/ICNC47757.2020.9049668.
[REF-34]	Laghaout, A. (2020). Supervised learning on heterogeneous, attributed entities interacting over time. <i>ArXiv</i> , abs/2007.11455.
[REF-35]	Kim, A.C., Park, W.H. and Lee, D.H. (2013). A study on the live forensic techniques for anomaly detection in user terminals. <i>International Journal of Security and Its Applications</i> , 7(1), 181–187.
[REF-36]	Chandola, V., Banerjee, A. and Kumar, V. (2009). Anomaly detection. <i>ACM Computing Surveys</i> , 41(3), 1–58.
[REF-37]	Garcia, R.F., Rolle, J.L.C. and Castelo, J.P. (2011). A review of SCADA anomaly detection systems. <i>Advances in Intelligent and Soft Computing</i> , 87, 405–414.
[REF-38]	Rabatel, J., Bringay, S. and Poncelet, P. (2011). Anomaly detection in monitoring sensor data for preventive maintenance. <i>Expert Systems with Applications</i> , 38, 7003–7015.
[REF-39]	Zhang, Jiliang and Li, Chen. (2019). Adversarial Examples: Opportunities and Challenges. In <i>IEEE Transactions on Neural Networks and Learning Systems</i> .
[REF-40]	Behzadan, Vahid and Munir, Arslan. (2017). Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. <i>arXiv:1701.04143</i> .
[REF-41]	https://theconversation.com/the-cyberattack-on-ukraines-power-grid-is-a-warning-of-whats-to-come-52832
[REF-42]	https://atlas.mitre.org/
[REF-43]	Soldatos, J., & Kyriazis, D. (2021). Trusted Artificial Intelligence in Manufacturing; Trusted Artificial Intelligence in Manufacturing: A Review of the Emerging Wave of Ethical and Human Centric AI Technologies for Smart Production; A Review of the Emerging Wave of Ethical and Human Centric AI Technologies for Smart Production.
[REF-44]	E Veliou, D Papamartzivanos, SA Menesidou, P Gouvas, T Giannetsos (2021) Artificial Intelligence and Secure Manufacturing: Filling Gaps in Making Industrial Environments Safer; Trusted Artificial Intelligence in Manufacturing: A Review of the Emerging Wave of Ethical and Human Centric AI Technologies for Smart Production. Book Chapter
[REF-45]	D. S. Lab, "strata santa clara dataset". [Online]. Available: http://datasensinglab.com/data/
[REF-46]	Dziugaite, G. K., Ghahramani, Z., & Roy, D. M. (2016). A study of the effect of jpg compression on adversarial images. <i>arXiv preprint arXiv:1608.00853</i> .
[REF-47]	Guo, C., Rana, M., Cisse, M., & Van Der Maaten, L. (2017). Countering adversarial images using input transformations. <i>arXiv preprint arXiv:1711.00117</i> .
[REF-48]	Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In <i>Proceedings of the IEEE conference on computer vision and pattern recognition</i> (pp. 2574-2582).
[REF-49]	Uyeong Jang, Xi Wu, and Somesh Jha. 2017. Objective Metrics and Gradient Descent Algorithms for Adversarial Examples in Machine Learning. In <i>Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC 2017)</i> . Association for Computing Machinery, New York, NY, USA, 262–277. DOI: https://doi.org/10.1145/3134600.3134635
[REF-50]	Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. <i>arXiv preprint arXiv:1706.06083</i> .
[REF-51]	Kurakin, A., Goodfellow, I. J., & Bengio, S. (2018). Adversarial examples in the physical world. In <i>Artificial intelligence safety and security</i> (pp. 99-112).

	Chapman and Hall/CRC.
[REF-52]	Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., & Madry, A. (2019, May). Exploring the landscape of spatial robustness. In International Conference on Machine Learning (pp. 1802-1811). PMLR.
[REF-53]	Andriushchenko, M., Croce, F., Flammarion, N., & Hein, M. (2020, August). Square attack: a query-efficient black-box adversarial attack via random search. In European Conference on Computer Vision (pp. 484-501). Springer, Cham.
[REF-54]	Carlini, N., & Wagner, D. (2017, May). Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP) (pp. 39-57). IEEE.
[REF-55]	Moosavi-Dezfooli, S. M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1765-1773).
[REF-56]	Papamartzivanos, D., Mármol, F. G., & Kambourakis, G. (2019). Introducing deep learning self-adaptive misuse network intrusion detection systems. <i>IEEE Access</i> , 7, 13546-13560.
[REF-57]	Papamartzivanos, D., Mármol, F. G., & Kambourakis, G. (2018). Dendron: Genetic trees driven rule induction for network intrusion detection systems. <i>Future Generation Computer Systems</i> , 79, 558-574.