

**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

### D3.4 – Cyber-Defence Mechanisms against Poisoning and Evasion Attacks – Final Version

<b>Dissemination level</b>	PU -Public
<b>Type of Document</b>	Report
<b>Contractual date of delivery</b>	31/03/2023
<b>Deliverable Leader</b>	UBI
<b>Status - version, date</b>	Final – v1.0, 19/09/2023
<b>WP / Task responsible</b>	WP3 / Tasks 3.2 and 3.3
<b>Keywords:</b>	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 final version of the AI Cyber Defence Tool 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.

This deliverable builds on top of D3.3 and extends the experimental testbed by conducting experiments using new datasets stemming from the pilot environments. More specifically, we take advantage of the *Soother* and *Assembly of Horizontal Lamellas* datasets, provided by the Philips and IBER partners respectively. The deliverable offers a description of the AI Cyber Defence Tool and the technology enablers which are combined for the delivery of the tool. In addition, the deliverable offers a discussion of the experimental results on applying a wide spectrum of attacks and defences over the pilot datasets.

Finally, D3.4 extends the work conducted in D3.3 for the delivery of a solution which utilises XAI to enhance the trustworthiness of data in sensory environments, dealing both with poisoning and evasion attacks. The new prototype is called XFSD and has been published in [REF-43].

<b>Deliverable Leader:</b>	Dimitris Papamartzivanos (UBI)
<b>Contributors:</b>	Thanassis Giannetsos, Entso Veliou (UBITECH) George Makridis, Ioannis Makridis, Dimosthenis Kyriazis, Spyros Theodoropoulos (UPRC) Erik Koehorst, Jorn Beukinga (Philips) Mihail Fontul, Joao Almeida (IBER)
<b>Reviewers:</b>	Jože Rožanec (JSI) Mihail Fontul (IBER)
<b>Approved by:</b>	Charalampos Ipektsidis (INTRA)

Document History			
Version	Date	Contributor(s)	Description
V0.1	14/04/2023	UBI	Table of contents
V0.2	25/05/2023	UBI, PCL, IBER	Information on the datasets
V0.3	10/06/2023	UBI	Cyber defence module architecture updates
V0.4	25/06/2023	UPRC, IBER, PCL	Contributions of partners in Section 2 and 3
V0.5	30/06/2023	UBI	Section 4, towards enhanced data trustworthiness using XAI
V0.7	25/07/2023	UBI	Experimental setup
V0.9	04/08/2023	UBI	Final Refinements
V0.95	04/09/2023	JSI, IBER	Quality Review
V0.97	19/09/2023	UBI	Refinements after review process
V1.0	19/09/2023	UBI, INTRA	Final version and submission

# Table of Contents

<b>EXECUTIVE SUMMARY .....</b>	<b>2</b>
<b>TABLE OF CONTENTS.....</b>	<b>4</b>
<b>TABLE OF FIGURES.....</b>	<b>6</b>
<b>LIST OF TABLES.....</b>	<b>7</b>
<b>DEFINITIONS, ACRONYMS AND ABBREVIATIONS .....</b>	<b>8</b>
<b>1 INTRODUCTION.....</b>	<b>9</b>
1.1 SCOPE AND PURPOSE .....	9
1.2 RELATION TO OTHER WPS AND DELIVERABLES .....	10
1.3 METHODOLOGY.....	10
1.4 DELIVERABLE STRUCTURE .....	11
<b>2 STAR AI CYBER DEFENCE TOOL .....</b>	<b>12</b>
2.1 POSITIONING IN THE STAR ARCHITECTURE .....	12
2.2 OVERVIEW OF AI CYBER DEFENCE INTERNAL ARCHITECTURE & INTERACTIONS .....	13
2.2.1 <i>Summary of the AI Cyber Defence workflow in the STAR architecture.....</i>	<i>14</i>
2.3 ENABLING TECHNOLOGIES OF THE AI CYBER DEFENCE TOOL .....	15
2.3.1 <i>Kafka and Zookeeper .....</i>	<i>15</i>
2.3.2 <i>Redis.....</i>	<i>16</i>
2.3.3 <i>Python Flask Backend.....</i>	<i>16</i>
2.3.4 <i>React frontend .....</i>	<i>17</i>
2.4 APIS.....	18
2.4.1 <i>External APIs .....</i>	<i>18</i>
2.4.2 <i>Internal APIs.....</i>	<i>19</i>
2.5 DESCRIPTION OF THE FUNCTIONAL LIFE CYCLE OF THE AI CYBER DEFENCE TOOL .....	19
2.6 FUNCTIONAL REQUIREMENTS FOR AI CYBER DEFENCE MODULE .....	21
<b>3 EXPERIMENTAL SETUP .....</b>	<b>22</b>
3.1 TESTBED SETUP .....	22
3.2 DATASETS.....	23
3.2.1 <i>Soothers dataset.....</i>	<i>23</i>
3.2.2 <i>Assembly of Horizontal Lamellas Dataset.....</i>	<i>24</i>
3.3 BASE MODEL TRAINING AND EVALUATION .....	25
3.4 CREATING ADVERSARIAL EXAMPLES USING ATTACKS FROM ADVERSARIAL ROBUSTNESS TOOLBOX .....	26
3.5 DEFENCE STRATEGIES.....	27
3.5.1 <i>Adversarial Training .....</i>	<i>27</i>
3.5.2 <i>ART Defence strategies .....</i>	<i>27</i>
3.6 EVALUATION RESULTS .....	28
3.6.1 <i>Soother Dataset Results .....</i>	<i>29</i>
3.6.2 <i>Assembly of Horizontal Lamellas Dataset Results.....</i>	<i>31</i>
3.7 SUMMARY .....	34
<b>4 EXPLAINABLE ARTIFICIAL INTELLIGENCE TO ENHANCE DATA TRUSTWORTHINESS IN CROWD-SENSING SYSTEMS.....</b>	<b>35</b>
4.1 INTRODUCTION.....	35
4.2 RELATED WORK.....	36
4.3 TOWARDS TRUSTWORTHY MCS TASKS .....	37
4.3.1 <i>Threat modelling .....</i>	<i>37</i>
4.3.2 <i>Explainable AI .....</i>	<i>38</i>

4.4	XFSD CONCEPTUAL ARCHITECTURE .....	38
4.4.1	<i>Training Phase</i> .....	39
4.4.2	<i>Testing phase</i> .....	40
4.5	EXPERIMENTAL SETUP.....	41
4.5.1	<i>Adversarial Behaviour</i> .....	41
4.6	RESULTS.....	43
4.6.1	<i>Results of XFSD Distribution attacks</i> .....	44
4.6.2	<i>Results of XFSD positional attacks</i> .....	47
4.6.3	<i>Advanced data poisoning</i> .....	50
4.6.4	<i>Aggregation Method</i> .....	51
4.7	SUMMARY .....	52
<b>5</b>	<b>DISCUSSION AND CONCLUSION</b> .....	<b>53</b>
	<b>REFERENCES</b> .....	<b>54</b>

# Table of Figures

FIGURE 1 STAR AI SECURITY AND DATA PROTECTION LAYER ARCHITECTURE (WP3) .....12

FIGURE 2 AI CYBER DEFENCE MODULE ARCHITECTURE .....13

FIGURE 3 AI CYBER DEFENCE TOOL UI.....21

FIGURE 4 EXPERIMENTAL TESTBED FLOW .....22

FIGURE 5 PHILIPS AVENT SOOTHERS .....23

FIGURE 6 SOOTHERS DATASET - EXAMPLES OF GOOD CHERRIES.....24

FIGURE 7 SOOTHERS DATASET - EXAMPLES OF FAULTY CHERRIES .....24

FIGURE 8 AIR VENT PARTS OF THE IBER'S PRODUCTION LINE .....25

FIGURE 9 EXAMPLE OF OK (LEFT) AND NOT OK (RIGHT) PRODUCT OF THE IBER DATASET .....25

FIGURE 10 APPLIED ATTACKS ON THE SOOTHER DATASET (PHILIPS) .....27

FIGURE 11 APPLIED ATTACKS ON THE ASSEMBLY OF HORIZONTAL LAMELLAS DATASET (IBER).....27

FIGURE 12 APPLIED DEFENCES AGAINST ATTACKS FOR SOOTHER DATASET (PHILIPS).....28

FIGURE 13 APPLIED DEFENCES AGAINST ATTACKS FOR ASSEMBLY OF HORIZONTAL LAMELLAS DATASET (IBER).....28

FIGURE 14 COLLECTIVE EXPERIMENTAL RESULTS ON PAIRWISE EVALUATION OF ATTACKS-DEFENCES USING SOOTHER DATASET (PHILIPS) .....30

FIGURE 15 COLLECTIVE EXPERIMENTAL RESULTS ON PAIRWISE EVALUATION OF ATTACKS-DEFENCES USING THE ASSEMBLY OF HORIZONTAL LAMELLAS DATASET (IBER) .....33

FIGURE 16 DATASET STRUCTURE .....37

FIGURE 17 TRAINING PHASE OF XFSD.....39

FIGURE 18 STRUCTURE OF Z MATRIX.....40

FIGURE 19 TESTING PHASE OF XFSD .....41

FIGURE 20 DISTRIBUTIONS OF LEGITIMATE SAMPLES (CLASS 0) VS. ADVERSARIAL SAMPLES (CLASS 1) FOR TWO FEATURES WITH  $m=0$  AND  $\sigma = 1.0$ . (A) ATTACK CASE I, AND (B) ATTACK CASE II.....42

## List of Tables

TABLE 1 API FOR POSTING IMAGES TO THE AI CYBER DEFENCE TOOL.....	18
TABLE 2 CACHING API DELIVERING THE PROCESSED IMAGES AND RESULTS TO THE FRONTEND. ....	19
TABLE 3 RESULTS OF VARIOUS XAI ALGORITHMS + FSD FOR ATTACK CASE I IN PRE-TRAINING ATTACK STRATEGY ..	45
TABLE 4 RESULTS OF VARIOUS XAI ALGORITHMS + FSD FOR ATTACK CASE II IN PRE-TRAINING ATTACK STRATEGY .	45
TABLE 5 RESULTS OF VARIOUS XAI ALGORITHMS + FSD FOR ATTACK CASE I IN POST-TRAINING ATTACK STRATEGY.	46
TABLE 6 RESULTS OF VARIOUS XAI ALGORITHMS + FSD FOR ATTACK CASE II IN POST-TRAINING ATTACK STRATEGY	47
TABLE 7 F-MEASURES FOR ATTACK CASE III (BENIGN FIRST) WITH ATTACK SAMPLES GENERATED BY ATTACK CASE I IN PRE-TRAINING FOR DIFFERENT XAI ALGORITHMS. ....	48
TABLE 8 F-MEASURES FOR ATTACK CASE IV (MALICIOUS FIRST) WITH ATTACK SAMPLES GENERATED BY ATTACK CASE I IN PRE-TRAINING .....	48
TABLE 9 F-MEASURES FOR ATTACK CASE V WITH ATTACK SAMPLES GENERATED BY ATTACK CASE I IN PRE-TRAINING	48
TABLE 10 F-MEASURES FOR ATTACK CASE III (BENIGN FIRST) WITH ATTACK SAMPLES GENERATED BY ATTACK CASE I IN POST-TRAINING.....	49
TABLE 11 F-MEASURES FOR ATTACK CASE IV (MALICIOUS FIRST) WITH SAMPLES GENERATED BY ATTACK CASE I IN POST- TRAINING .....	49
TABLE 12 F-MEASURES FOR ATTACK CASE V POST-TRAINING ATTACK STRATEGY WITH ATTACK SAMPLES GENERATED BY ATTACK CASE I .....	50
TABLE 13 OPTIMIZED DATA POISONING AVAILABILITY ATTACK CASE VI FOR THE SCENARIO OF $\mu$ .....	50
TABLE 14 OPTIMIZED DATA POISONING AVAILABILITY ATTACK CASE VI FOR THE SCENARIO WITH $\Sigma$ .....	51
TABLE 15 OPTIMIZED DATA POISONING TARGET ATTACK CASE VII FOR THE SCENARIO WITH $\mu$ .....	51
TABLE 16 OPTIMIZED DATA POISONING TARGET ATTACK CASE V II FOR THE SCENARIO OF $\Sigma$ .....	51
TABLE 17 F-MEASURES FOR ATTACK CASE I PRE-TRAINING ATTACK STRATEGY WHERE THE DEFAULT AGGREGATION IS APPLIED BEFORE CONCATENATION OF FIXED LENGTH VECTORS. ....	52

## Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
<b>AI</b>	Artificial intelligence
<b>ART</b>	Adversarial Robustness Toolbox
<b>ATLAS</b>	Adversarial Threat Landscape for Artificial-Intelligence Systems
<b>CAM</b>	Class Activation Mapping
<b>CNN</b>	Convolutional Neural Network
<b>CPS</b>	Cyber-Physical Systems
<b>DL</b>	Deep Learning
<b>DNN</b>	Deep Neural Networks
<b>FGSM</b>	Fast Gradient Sign Method
<b>FSD</b>	Fake Sequential Data
<b>IDS</b>	Intrusion Detection System
<b>JSMA</b>	Jacobian-based Saliency Map Approach
<b>LIME</b>	Local Interpretable Model-agnostic Explanations
<b>LSTM</b>	Long-Short Term Memory
<b>MCS</b>	Mobile Crowd Sensing
<b>MCS</b>	Mobile Crowd Sensing
<b>ML</b>	Machine Learning
<b>NLP</b>	Neuro-Linguistic Programming
<b>PGD</b>	Projected Gradient Descent
<b>RMS</b>	Runtime Monitoring System
<b>RNN</b>	Recurrent Neural Network
<b>SHAP</b>	SHapley Additive exPlanations
<b>SoTA</b>	state-of-the-art
<b>WP</b>	Work Package
<b>XAI</b>	Explainable Artificial Intelligence
<b>XFSD</b>	Explainable Artificial Intelligence Fake Sequential Data
<b>ZOO</b>	Zeroth Order Optimisation

# 1 Introduction

## 1.1 Scope and purpose

The aim of this deliverable is to document the research performed on the attack and defence strategies against AI systems with particular focus on data poisoning and evasion attacks, as a result 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, using the datasets provided by the use case partners of STAR.

In this context, this deliverable complements the work done in D3.3 and concludes the development actions on adversarial AI tools and methods that empower the AI Cyber Defence tool of STAR. Thus, this deliverable (D3.4) documents the new developments and updates of T3.2 and T3.3 since the delivery of D3.3 until the completion of WP3 actions.

As the AI Cyber Defence tool aims to support the detection services against poison and evasion attacks, the deliverable reports on the results of applying various attacks and defences utilising datasets stemming from the STAR pilots' environments. In D3.3, UBITECH team experimented with the *Decocap* and *Shavershell* datasets. The work conducted in D3.3 led also to the respective publication [REF-19]. Now, D3.4 extends the experimental testbed of D3.3 by considering two new datasets, as those have been generated by the Philips and IBER technical teams.

In the context of this deliverable, which is the final 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. In addition, we document the technical developments for offering the AI Cyber Defence tool as-a-service, which can be placed in the STAR's architecture and operate in synergy with the rest of the STAR tools.

Based on our experimental results, we highlight the advantages and disadvantages of the state-of-the-art approaches, and we concluded that *Adversarial AI* is not only a hot research challenge, but it is an actual threat that can affect existing production-level AI-enabled systems in the manufacturing domain.

Last but not least, in continuation to the long-term research plan for enhancing the data trustworthiness for AI systems, which was presented in D3.3 and in the respective publication [REF-01], in D3.4 we proceed to the exploitation and integration of XAI in order to evaluate their effectiveness on detecting data inconsistencies generated by colluding adversaries. Our research endeavour of applying explainable artificial intelligence to enhance data trustworthiness in sensing systems, such as those that can be found in the manufacturing domain, is reported in section 4, while the respective publication can be found in [REF-43].

It has to be noted that the detailed interactions of the AI Cyber Defence tool, in the context of the overall STAR framework and the Security and Data Governance layer of WP3, have been documented in D3.3. Thus, we prompt the interested reader to refer to D3.3 for more details.

## 1.2 Relation to other WPs and Deliverables

The deliverable at hand 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 have been documented in D2.1. D3.3 documented how those requirements are met by the developed tool. The updates reported in D3.4 are aligned with the requirements provided in deliverable D2.1.

**D2.2 Reference Scenarios and Use Cases for AI in Manufacturing:** The reference scenarios of D2.2 are considered to guarantee that the development from the AI Cyber Defence tool would meet the expectation and the needs of the STAR pilots. In fact, the AI models empowering the AI Cyber Defence tool have been built using datasets stemming from the Philips and IBER partners.

**D2.4 and D2.5 Data Models and Data Collection-Initial and Final versions:** 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 and D2.5. Thus, this deliverable exploits the dataset collected in D2.5 in order to feed the experiments on the evaluation of the adversarial and defence strategies.

**D3.2 Decentralized Reliability for Industrial Data and Distributed Analytics-Initial version:** This deliverable documents the blockchain-based service of STAR that aims 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.

**D3.3 Cyber-Defence Mechanisms against Poisoning and Evasion Attacks-Initial version:** D3.3 was the initial deliverable documented the developments and the design of the methodologies for defending against poisoning and evasion attacks. D3.4 builds upon D3.3 and finalises this series of deliverables for the AI Cyber Defence tool.

Overall, D3.4 provides the final 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 is documented in D3.6.

## 1.3 Methodology

UBITECH, 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 has 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 literatures (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.

D3.4 completes the above-mentioned actions. Steps 1-4 were already achieved in the context of D3.3. The integration of STAR blockchain of Step 5 was also achieved in D3.3, while the integration of XAI methods is now documented in D3.4 in Section 4. Finally, our experimental testbed now considers new datasets from pilot partners, more specifically one new dataset from Philips and one from IBER, concluding the experimental process described in Step 6. Further refinements are expected in the context of WP6, where some adaptations to the models may be required as a results of the integration of the tools in the pilot's environments and the STAR integrated framework.

During this period, UBITECH has been in close collaboration with the pilot leaders in order to be aligned with the datasets that are going to provide and the work towards meeting the expected goal of their pilots regarding the cyber defence of the AI systems.

## 1.4 Deliverable structure

The structure of the document is as follows:

- Section 2 provides a comprehensive documentation of the AI Cyber defence tool, by discussing its placement in the STAR architecture, the enabling technologies, and the developed APIs.
- Section 3 provides the documentation of the evaluation results of the experimental testbed that UBI created in order to evaluate different adversarial and defence methods utilising data stemming from the pilot sites.
- Section 4 presents the work of UBITECH towards exploiting XAI to enhance the trustworthiness of data.
- Section 5 concludes the deliverable, outlining the main outcomes of the work performed in the context of Tasks 3.2 and 3.3.

## 2 STAR AI Cyber Defence Tool

### 2.1 Positioning in the STAR architecture

The AI Cyber defence module aims to defend the Philips and IBER 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 are susceptible to attacks that may target both the training (i.e., poisoning) and the operational (i.e., evasion) phases of Deep Neural Networks (DNNs). The interested reader can refer to D3.3 for a thorough analysis of the state-of-the-art.

The AI Cyber defence tool is positioned at the AI Security and Data protection layer of the STAR architecture and is destined to receive data from the manufacturing environment in order to detect the presence of an AI adversary. 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.

The AI Cyber Defence tool is positioned in the middle, between the manufacturing floors (left) and the Security Policy 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 Policy Manager and the Risk Assessment & Mitigation Engine for further analysis and for informing the security administrator about the detected incidents.

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 latest version of the architecture is given in D3.6.

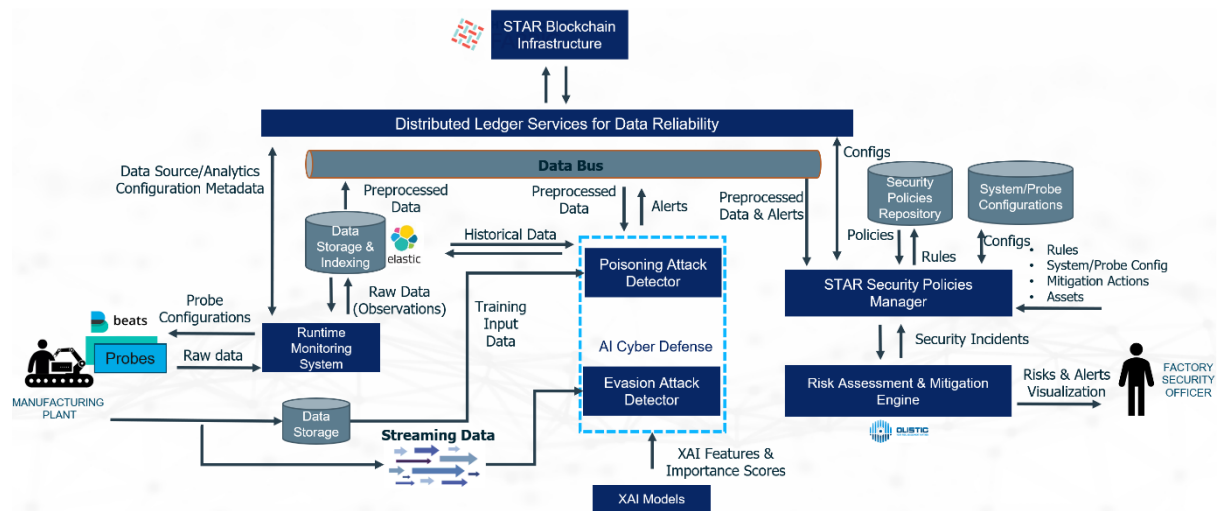


Figure 1 STAR AI Security and Data protection layer Architecture (WP3)

## 2.2 Overview of AI Cyber Defence 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. The architecture of the tool remains the same as presented in D3.3, but it brings updates regarding 1) the incorporated AI models that perform the detection of adversarial examples for each of the manufacturing floors of the Pilots, as well as 2) the use of backend technologies that support the operation of the tool. The details of these updates will be documented in the following sections.

All the individual internal components of the AI Cyber Defence tool have been described in D3.3. For purposes of completeness, we provide a high-level description of the tool, but for more details the interested reader can always refer to D3.3.

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.

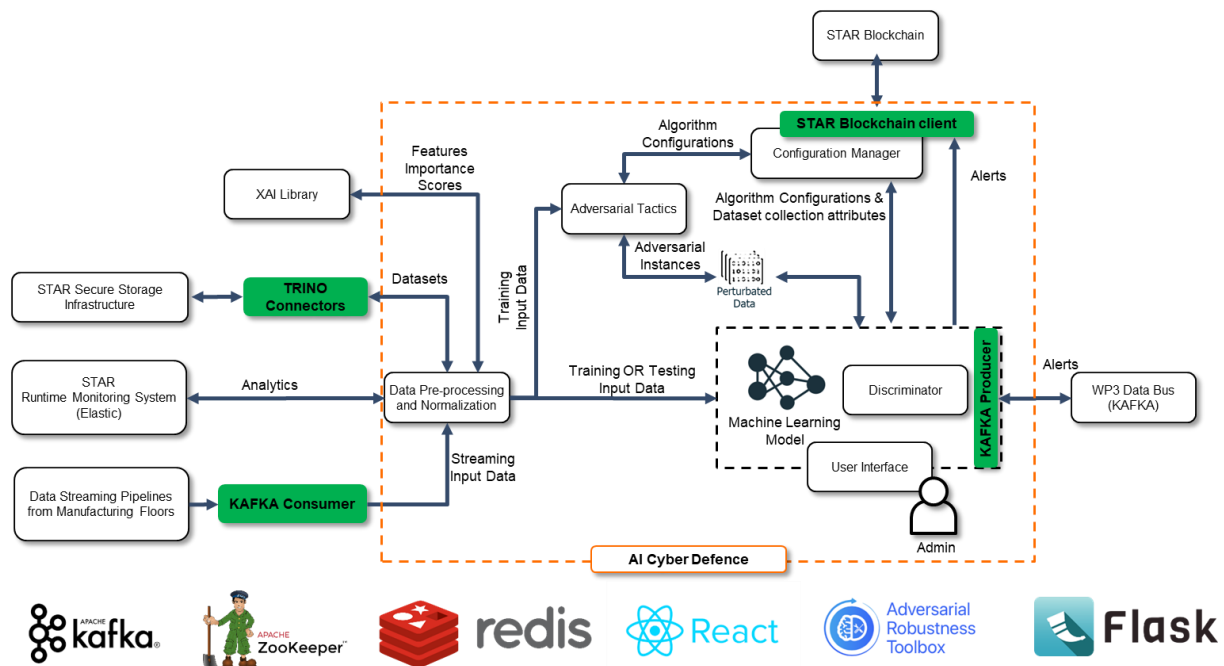


Figure 2 AI Cyber Defence Module architecture

In D3.3 we provided a detailed description of the entire workflow of the components, as well as for the interactions that the AI Cyber Defence tool has with the rest of the STAR components. Thus, the following paragraphs offer just a summary of the aforementioned points, as the main objective of this deliverable is to highlight the technical updates of the tool and the advancements on the experimentation testbed for the Adversarial AI methods.

## 2.2.1 Summary of the AI Cyber Defence workflow in the STAR architecture

The AI Cyber defence tool utilises the inputs of the:

- **STAR Secure Storage infrastructure:** Utilising the TRINO connectors (see D2.5) the tool can query data stored on the STAR storage infrastructure. Those data are used for training purposes of the tool.
- **STAR Runtime Monitoring system:** As will be also explained in Section 4, in the context of our research endeavour we have developed a methodology which takes advantage of XAI to detect attacks in ecosystems fed by sensory data. Under this concept, we illustrate the connection of AI Cyber Defence with the Runtime Monitoring System as a dataflow that could be materialised as an extra feature of the AI Cyber Defence functionality in the future.
- **Data streaming pipelines:** This entity represents the data sources which are used to feed the AI Cyber defence tool during runtime for the detection of evasion attacks. This pipeline is materialised using the APIs exposed by the tools, as reported in section 2.4. 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.

Given the inputs, a ***data pre-processing and normalisation*** step takes place depending on the AI model that will be training and utilised for the detection of adversarial attempts against the STAR systems. It is necessary to manipulate the data and apply the necessary transformations in order to make them fit to the specific model.

After bringing the data to the necessary format, the ***Adversarial Tactics*** component 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 subcomponent is important for the establishment of the end-model as, based on this, we are in 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. Section 3 of this deliverable elaborates on the methods, the datasets, and the results of applying a set of adversarial tactics in the context of the Philps and IBER pilots.

Given the development of the Adversarial Tactics, the ***Discriminator*** is the final 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. The output of the discriminator is an alert which will be conveyed to the SPM through the KAFKA data bus. In addition, in this version of the tool a web-based GUI is offered in order to illustrate to the user image received in the pipeline, as well as the detection result of the discriminator. More details are given in Section 2.5.

In parallel with the workflow described above, 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 provenance and tracking of industrial data and the 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 that the model used for the detection is the correct configuration and ensure that there is no malicious intervention or tampering in the configuration of the tool.

Finally, upon the detection of a poisoning or an evasion attack, the AI Cyber Defence module generates an alert. This alert includes all the necessary information for an attack. The alert is sent to the SPM of STAR through the data bus, in order to trigger the generation of the respective attack scenario that will be visualised in the Risk Assessment and Mitigation Engine of STAR. More details on the alert's data model are given in D3.3.

## 2.3 Enabling Technologies of the AI Cyber Defence Tool

Since the deliverable at hand reports on the final version of the AI Cyber Defence tool, this section elaborates on the various technologies used for building the tool and for supporting its functional objectives.

### 2.3.1 Kafka and Zookeeper

Kafka and ZooKeeper are two separate components often used in combination for building distributed streaming platforms. Those components were used in AI Cyber Defence tool in order to deliver a consumer/producer operational approach. In what follows, we first provide an overview of the roles of Kafka and Zookeeper and then we discuss why a consumer/producer architecture was the suitable solution for streaming image data objects.

**Kafka:** Kafka is a distributed streaming platform designed to handle large-scale real-time data streams. It is built to be highly scalable, fault-tolerant, and durable. Kafka allows to publish and subscribe to streams of records, which can be anything from simple text messages to more complex data like images or videos. Kafka provides a persistent, distributed, and fault-tolerant message queue that enables the decoupling of producers (data senders) from consumers (data receivers).

**ZooKeeper:** ZooKeeper is a centralized service used to manage distributed systems. It provides coordination, synchronization, and configuration maintenance for distributed applications. Kafka uses ZooKeeper for distributed coordination among its brokers (the servers that manage Kafka topics) and to maintain metadata, leader election, and other administrative tasks.

The AI Cyber Defence tool has been built in order to support image streaming, as it is destined to perform in the context of visual quality inspection systems. These are the characteristics that make the consumer and producer architecture the right choice for our tool.

- **Decoupling and Scalability:** Using a consumer and producer architecture with Kafka allows for the decoupling of image producers from consumers. Image producers, such as cameras or other sources of image data, can publish images to Kafka topics without needing to know who or how many consumers are consuming the data. This decoupling enables easy scalability, as we can add more consumers to handle the image stream as the demand grows.
- **Fault Tolerance:** Kafka's distributed nature and replication mechanisms ensure fault tolerance. If one consumer goes down, other consumers can continue to receive the image stream without interruption. Similarly, if a consumer is slow or experiences issues, Kafka can buffer and manage the data flow, preventing data loss and providing reliable data delivery.

- **Real-time Processing and Analytics:** A consumer and producer architecture allows consumers to process image data in real-time. Multiple consumers can consume the same image stream and perform different processing tasks, or even run a different model over the received data, simultaneously.
- **Data Storage and Historical Analysis:** Kafka retains data for a configurable period, allowing consumers to access historical image data and perform analysis if needed.
- **Asynchronous Communication:** The use of Kafka enables asynchronous communication between producers and consumers. This means producers can publish images to Kafka quickly, and consumers can process the images at their own pace, based on their processing capabilities and the complexity of the AI model destined to perform the analysis, without directly affecting the image producers.

In summary, Kafka and ZooKeeper provide the infrastructure for building scalable, fault-tolerant, and real-time image streaming systems. Using a consumer and producer architecture with Kafka allows for decoupling, scalability, fault tolerance, real-time processing, and historical analysis, making it a popular choice for image streaming applications, such as the one required for the AI Cyber Defence tool.

### 2.3.2 Redis

For the UI of the Cyber AI defence tool and in order to provide *real time image streaming data analysis*, Redis technology and base64 image conversion has been implemented. Caching base64 images with Redis can provide significant performance benefits and reduce the load on the image storage or generation systems. Here are the benefits brought to the AI Cyber Defence Tool by using caching with Redis and using base64 image conversion:

- **Fast Data Retrieval:** Redis is an in-memory data store, and it is known for its high-performance data retrieval capabilities. When you cache base64 images in Redis, the images are stored in memory, allowing for quick and efficient retrieval when requested. This reduces the need to fetch the images from slower storage systems, such as databases or file systems, every time they are needed.
- **Reduced Processing Overhead:** Converting an image to a base64 representation can be computationally intensive, especially for large images. By caching the base64 representation of images in Redis, we avoid the need to repeatedly perform the base64 encoding process for the same image. This reduces the processing overhead and improves the overall response time of the tool.
- **Lower Bandwidth Usage:** When images are requested frequently, serving them directly from Redis avoids unnecessary data transfer between the application server and the image storage. This helps to reduce bandwidth usage.
- **Expire and Eviction Policies:** Redis allows to set expiration time for cached data. This feature ensures that cached base64 images are automatically invalidated and removed from Redis after a specific period.
- **Scaling and Load Distribution:** In case the application or the operational environment grow, Redis can be scaled by using replication or setting up new Redis clusters. This ensures that the cached images remain available even during high-traffic scenarios, ensuring the overall system's performance and reliability.

### 2.3.3 Python Flask Backend

Python Flask is the technology of the backend system of the AI Cyber Defence Tool, and it is a popular choice for web development projects. The Flask server is the one which exposes

the APIs that can be triggered in order to send data objects to the AI Cyber Defence tool. In addition, it plays the role of the Kafka producer which then sends the received data to the internal Kafka in order to scale up and parallelise the processing on the received data. More details on the operational workflow of the tool are given later in section 2.5. We choose to use python flask to take advantage of the following beneficial characteristics:

- **Lightweight and Simple:** Flask is a micro-framework, meaning it is designed to be lightweight and minimalistic. It provides just the essentials for building web applications, which makes it easy to understand and quick to get started.
- **Flexibility:** Flask gives the freedom to structure our application the way we want as it does not dictate the way to organise out code unlike some other web frameworks. Thus, its flexibility allows us to tailor the AI Cyber Defence architecture to our specific needs.
- **Extensibility:** Flask is designed to be extensible and allows to add functionality through third-party extensions or to write our own custom extensions. Its modularity makes it easy to integrate with various databases and other libraries so that to enhance the capabilities of the application.
- **RESTful APIs:** Flask's design makes it well-suited for building RESTful APIs (Application Programming Interfaces). Its lightweight nature and ease of handling HTTP requests and responses enabled us to create the necessary APIs for the frontend application (See Section 2.4) and for exposing the necessary APIs to the use case partners of STAR for posting the data in the tool's engine.
- **Rich Python Ecosystem:** Flask benefits the use of Python libraries. Thus, we leverage the power of Python to handle tasks beyond web development. In fact, we integrate into the Flask server the machine learning part and execute the Adversarial AI code and business logic which was built using the python-based ART tool [REF].
- **Deployment Options:** Flask applications can be easily deployed on various platforms and hosting providers. Whether we choose to deploy on traditional servers, containerized environments (e.g., Docker), or cloud services (e.g., AWS, Google Cloud, Azure), Flask's simplicity ensures a smooth deployment process. Given that, the AI Cyber Defence tool may be needed to be installed to different infrastructures and premises of the Pilot partners, and this characteristic ensures the easy integration with the pilot systems.

In summary, Python Flask was a great fit for building a web application and the necessary RESTful APIs for the AI Cyber Defence Tool due to its simplicity, flexibility, extensibility, and strong Python ecosystem support.

### 2.3.4 React frontend

The AI Cyber Defence Tool is using React for frontend development, as it offers numerous benefits. There are also several other web frameworks that could have been chosen instead of React. However, it is our choice to utilize the React framework due to the following key advantages:

- **Component-Based Architecture:** React follows a component-based architecture, allowing us to break down the UI into reusable, self-contained components. This modularity promotes code reusability, maintainability, and scalability.
- **Virtual DOM:** React utilizes a virtual DOM, which is a lightweight copy of the actual DOM. When there are changes in the data or state, React calculates the minimal update required and efficiently updates only the necessary parts of the actual DOM.

This approach significantly improves performance and reduces unnecessary reflows and repaints on the frontend,

- **Declarative Syntax:** React uses a declarative syntax, making it more straightforward to describe how the UI should look based on the current application state.
- **High Performance:** Due to its efficient rendering using the virtual DOM and one-way data flow, React delivers excellent performance even for complex applications. It minimizes unnecessary updates and re-renders, resulting in faster and more responsive user interfaces.
- **JSX - JavaScript XML:** React uses JSX, which allows developers to write HTML-like syntax directly within JavaScript code. This combination of HTML and JavaScript makes the code more readable and maintainable.
- **Mobile App Development:** React Native, built on top of React, enables the development of cross-platform mobile applications for both iOS and Android. This could allow the potential development of the tool for mobile devices so that to provide the AI Cyber Defence Tool results to security officer on the go.

In summary, React's component-based architecture, virtual DOM, and declarative syntax contribute to its popularity and success in frontend development, and these are the characteristics that led us to choose React to build a high-performance, scalable, and maintainable web application for our tool.

## 2.4 APIs

In this section we document the developed APIs that enable the integration of the AI Cyber Defence tool with the rest of the STAR components of the STAR architecture. In terms of interactions, the AI Cyber Defence tool had to expose an interface in order to consume information from the pilot sites and to enable the communication with the Blockchain infrastructure of STAR.

### 2.4.1 External APIs

Table 1 documents the API used for posting images to the AI Cyber Defence Engine in order to trigger the Discriminator to infer whether the posted image is an adversarial example or not. This is the main endpoint triggered from the pilot’s environment in order to stream data to our tool in parallel to the production process.

*Table 1 API for posting images to the AI Cyber Defence tool.*

Functionality		<b>Posting Image to AI Cyber Defence tool</b>	
API Type	Form-data (File)		
API	POST /starImage		
Description	HTTP Form Data is a way of sending data from a client to a server using the HTTP protocol. When it comes to sending images via HTTP Form Data, it involves encoding the image data in a format that can be easily transmitted over the web. This is commonly used when building APIs that allow users to upload images or other files to a server. This API is listening to a streaming of images, the images then are streamed to the internal Kafka component of the AI Cyber Defence tool that undertakes parallel execution of the AI model that performs the detection of adversarial instances.		
Inputs	<b><u>Input/s</u></b>		<b><u>Output/s</u></b>

	<b>starFile</b> type: file (jpeg or png)	"Success or Fail"
--	---	-------------------

Functionality **Update Blockchain Configuration**

API Type	RESTful (JSON data types)	
API	POST /updateProcessorConfig	
Description	This API enables us to change the hash of the AI model in the blockchain. Once our AI model has been uploaded in the AI Cyber Defence Tool, the hash of the model needs to be verified in every transaction in order to see if the model has been compromised by an attacker. This hash is updated from our administrator once the AI model is uploaded.	
Inputs	<b>Input/s</b>	<b>Output/s</b>
	<b>modelHash</b> type: string	"Success or Fail"

### 2.4.2 Internal APIs

This section documents that internal APIs which are used internally in the tool in order to enable the communication and data transmission between the backend and the frontend.

*Table 2 Caching API delivering the processed images and results to the frontend.*

Functionality **Get Processed Images**

API Type	RESTful (JSON data types)	
API	POST/GET /getStarImages	
Description	This API provides a list of the images, formed in base64 string format, with the results of the AI model process. The API gets the data directly from the Redis Caching and is only used by the frontend in order to get the necessary data to be visualised. The result is a list of images which have been processed by the tool.	
Inputs	<b>Input/s</b>	<b>Output/s</b>
		<b>Images:</b> List [imagesBase64, ImagesResult] (JSON)

## 2.5 Description of the functional life cycle of the AI Cyber Defence Tool

The AI Cyber Defence tool exposes the API documented in Table 1. This is the endpoint which is exposed by the Flask server and acts as the entry point of the data into the internal pipeline of the tool. It has to be noted that, in the context of STAR, our application is used in order to process imagery data to support the visual quality inspection setups of the STAR pilots. However, our application can also support other types of data and we have created additional endpoints for supporting the exchange of sound and tabular (.csv) data files, but those fall out of the scope of the tool.

Once an image is being streamed to the respective interface it is initially transformed into a byte array. This action triggers the creation of a separate processing thread in the backend engine, which means that another image can be sent simultaneously. Then, this byte array of the image is produced into the Kafka. We then have a consumer listening to images produced from the Kafka, with the help of the zookeeper. Once the image is consumed by the consumer the image is recreated to its original format from the byte array and it is saved locally in a directory with all the input images. The reasoning behind using Kafka at this point, as explained also in Section 2.3, is that it creates an asynchronous data processing behaviour at the backend that helps the application to scale up and handle multiple processing requests without draining out the resource of the server. This design enables parallel process but, at the same time, ensures that the service will remain up & running in cases of high workload and keep up in case where the production rate in the pilot sites is increased. After reaching this point, the images are ready for processing.

An API call is being made towards the blockchain to check the validity of the AI model hash with the one that is on site in the blockchain infrastructure. Once the validity check is passed, then two simultaneous actions in separate threads are triggered, 1) the image is processed in base64 and is saved to Redis Caching infrastructure and 2) the AI model processing of the image takes place. Once the AI processing is complete, the results are then sent to the Kafka infrastructure (Data bus) of the AI Security and Data protection layer with the JSON result, to facilitate the rest of the pipeline of the tool. The JSON result also will be saved locally in the log file and will be also saved in the Redis caching with a key that can match the result with the image base64. In this way, a direct map between the image and the AI processing result is generated and maintained in the engine.

At the other end the frontend of the AI Cyber Defence Tool is listening in real time for image base64 keys in combination with the result. Once a match is found, the UI receives directly from the caching API (Table 2) the pair of base64 and result which the user then sees in the frontend of the AI Cyber Defence tool. The keys have been set to expire every 200 seconds in order to have a better streaming experience. However, this threshold can be adjusted accordingly. The image below (Figure 3) shows the UI of the Cyber Defence tool. On the upper part, the image that was captured - or it has been maliciously injected - during the production process in the manufacturing floor is illustrated. This image was processed by the Discriminator in order to infer whether it is a normal instance or an adversarial input. The result of the tool is illustrated in the lower part of the image, along with the necessary details and the results of the process (e.g., Attack type and confidence level). The UI provides the necessary buttons to navigate among the generated results, as highlighted in the green boxes in Figure 3.

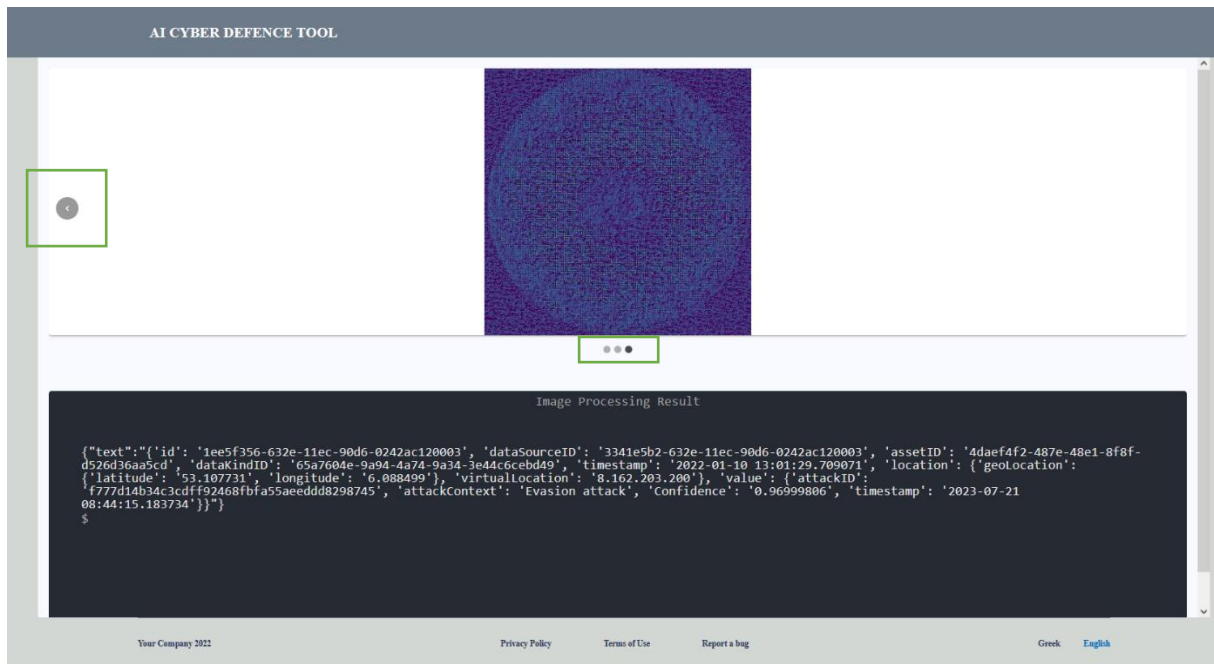


Figure 3 AI Cyber Defence Tool UI

## 2.6 Functional Requirements for AI Cyber Defence module

In this section we summarise the functional requirements related the AI Cyber Defence module, as those have been initially reported in D2.1. The requirements are enumerated bellow.

- 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 in Section 2 for the architecture of the AI Cyber Defence module, one can notice that the defined interactions are aligned with the requirements defined in the early stages of the project. More specifically, Req-02 is addressed though the acquisition of proper algorithm configurations via the Configuration Manager of the tool. Req-03 is addressed as the XAI libraries are exploited in the context of the actions described in Section 4. Req-04 is met by the operation of the Discriminator tool which generates the respective alerts. Req-01, Req-05, Req-06 and Req-07, are addressed as a result of the development and evaluation of the state-of-the-art techniques for developing AI attacks and defences, as have been documented in D3.3, in Section 3 of this deliverable, and in our publications in [REF-43][REF-44][REF-01] and [REF-19].

### 3 Experimental setup

#### 3.1 Testbed setup

In D3.4 we took advantage of the experimental testbed that we built in D3.3, but this time we apply the developed algorithms to the newly available datasets from the STAR pilot partners. Our testbed aims to evaluate and test the applicability of existing adversarial techniques and evaluate corresponding defences. That is, in D3.4 we analysed and processed the *Smoother* dataset (Philips) and *Assembly of Horizontal Lamellas* Dataset (IBER) which are presented in the following sections.

The aforementioned datasets are treated under a binary classification problem, meaning that the two classes considered in the “OK” (representing a capture of a good product) and “NOK” (representing a faulty product). The choice of treating the datasets under the binary classification approach was due to our intention to simplify the classification process but mainly because of the absence of data for the distinct classes of faulty products.

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 AI-enabled visual inspection system, as part of a quality assurance process running in a production line.

Figure 4 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 base model, 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. To build our testbed we take advantage 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.

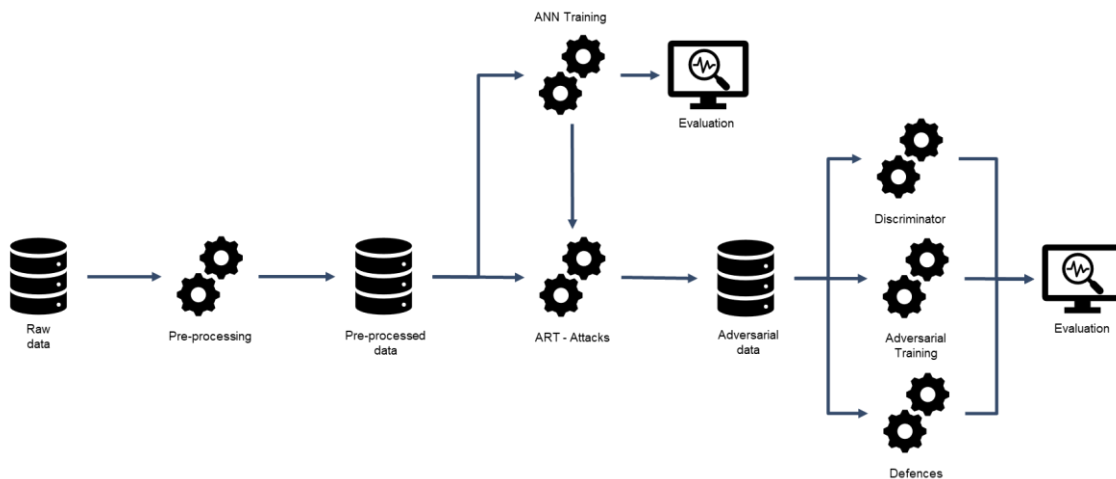


Figure 4 Experimental testbed flow

## 3.2 Datasets

In continuation of the experimentation testbed built in D3.3, D3.4 considers two new datasets stemming from the Philips and IBER use case partners and reflect the data generated in the context of the visual quality inspection performed during the production. Visual inspection is the most indicative case to demonstrate the applicability adversarial attacks and defences, as the most prominent adversary tactics, techniques of the literature are demonstrated in such kind of application domains. In addition, it was one of the main objectives of the pilot partners to safeguard the visual inspection process as it is rather critical for ensuring the high quality of their products, but also for ensuring the continuity of the supply chains in which they act as stakeholders, and for avoiding potential financial losses, as a result of an evasion attack against the quality inspection system.

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

- Dataset1: Soother dataset (Partner: Philips)
- Dataset2: Assembly of Horizontal Lamellas Dataset (Partner: IBER)

### 3.2.1 Soothers dataset

Philips Avent soothers are produced with the highest focus on quality. The main part where the visual inspection process is focusing in the “cherry”, i.e., the part of the soother which is in the baby's mouth and may never fail. Each cherry is inspected individually and Philips goal is to have this inspection process being conducted in a fully automated manner in the future. The Soother product is illustrated in Figure 5, while instances of the dataset images that represent normal and faulty products are given in Figure 6 and Figure 7 respectively. The interested reader can find more information of the datasets and the production line in D2.5 and in the WP6 deliverables. The data provided by Philips in order to be able to conduct our experiments in the context of D3.4 was: 163 OK and 81 Not-OK instances, while 17.5% of those were used for the testing phase of the AI models.



*Figure 5 Philips Avent soothers*

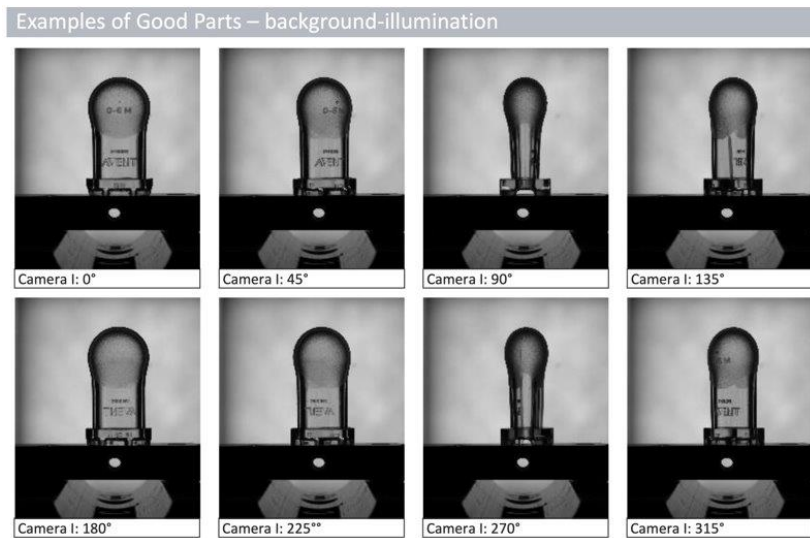


Figure 6 Soothers Dataset - Examples of good cherries

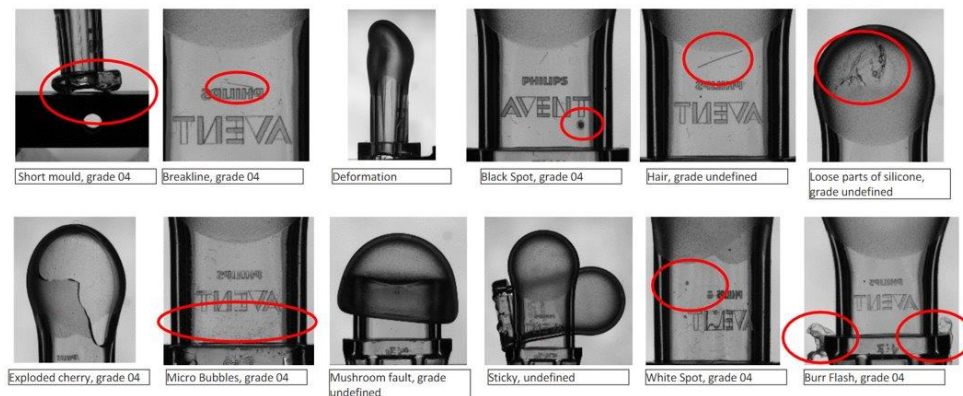


Figure 7 Soothers Dataset - Examples of faulty cherries

### 3.2.2 Assembly of Horizontal Lamellas Dataset

IBER produces plastic air vents for cars. During the assembly process, the horizontal parts of the air vents (black-coloured parts) are assembled with the Lamella (yellow-coloured part), and then are injected into the housing part (blue part), as shown in Figure 8. However, this process is not free of faults, as the during the assembly several fault cases have been recognised, as illustrated indicatively in Figure 9. The interested reader can find more information of the datasets and the production line in D2.5 and in the WP6 deliverables, respectively. The data provided by IBER in order to be able to conduct our experiments in the context of D3.4 was: 2164 OK and 78 Not-OK instances, while 17.5% of those were used for the testing phase of the AI models.

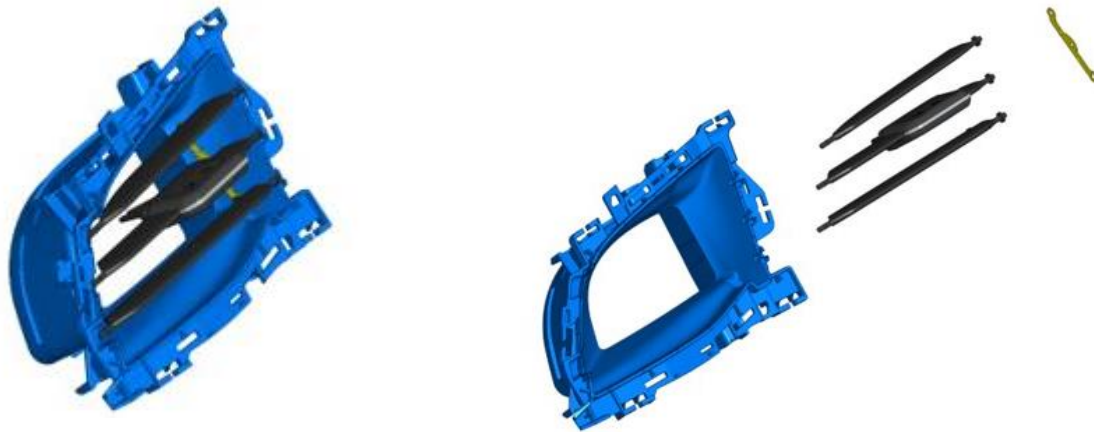


Figure 8 Air vent parts of the IBER's production line

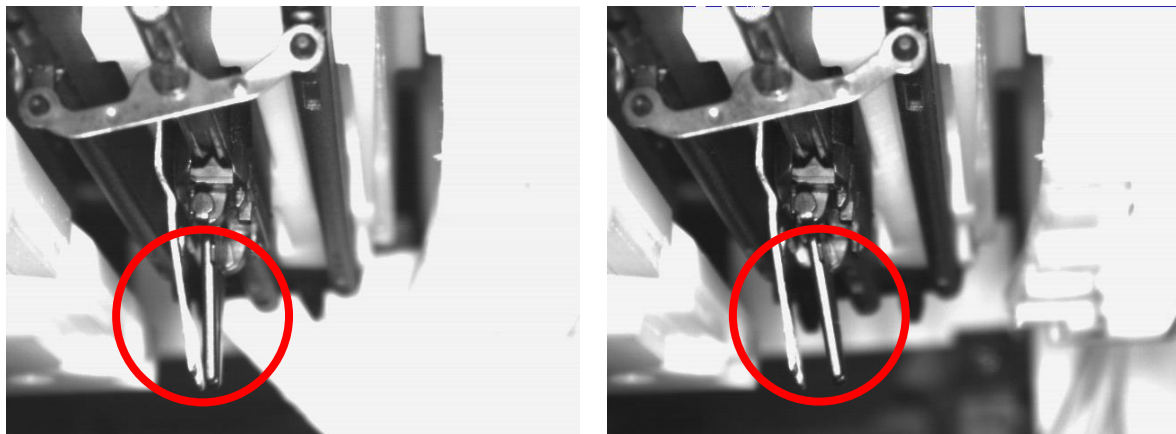


Figure 9 Example of OK (left) and Not OK (right) product of the IBER dataset

### 3.3 Base Model Training and evaluation

Following the setup of Figure 4, 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 Relu activation function. 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. A snippet of the AI model setup is given below.

```
model = Sequential()
model.add(Conv2D(8, (3, 3), padding="same", input_shape=(512, 256, 1)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(8, (3, 3), padding="same"))
```

```

model.add(Activation("relu"))
model.add(Conv2D(8, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128))
model.add(Activation("relu"))
model.add(Dropout(0.25))
model.add(Dense(10))
model.add(Activation(activationfun))
model.compile(loss=loss_, optimizer="adam",
metrics=[metrics.MeanSquaredError(), metrics.AUC(), 'acc'])

```

### 3.4 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. We opt to use the same set of attacks as those used in D3.3, and to consider 3 additional ones, namely the Zeroth Order Optimisation (ZOO) [REF-86], HopSkipJump Attack [REF-87] and Boundary Attack [REF-88]. Thus, in total we have applied 12 different attacks in order to test the applicability of the attacks on the STAR pilot environments and data. The attacks have been implemented by taking advantage of the ART (Adversarial Robustness Toolbox). The adversarial techniques tested in the context of this deliverable are the following:

- Fast Gradient Sign Method (FGSM) [REF-02]
- DeepFool [REF-07]
- NewtonFool [REF-08]
- Projected Gradient Descent (PGD) [REF-09]
- BasicIterative using PGD [REF-10]
- SpatialTransformation [REF-11]
- CarliniLIn Method [REF-12]
- CarliniL2 Method [REF-12]
- Universal Perturbation using EAD-elastic-net attack [REF-13]
- Zeroth Order Optimisation (ZOO) [REF-40]
- HopSkipJump Attack [REF-41]
- Boundary Attack [REF-42]

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 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. The following images show the result of applying the attacks on the available datasets.

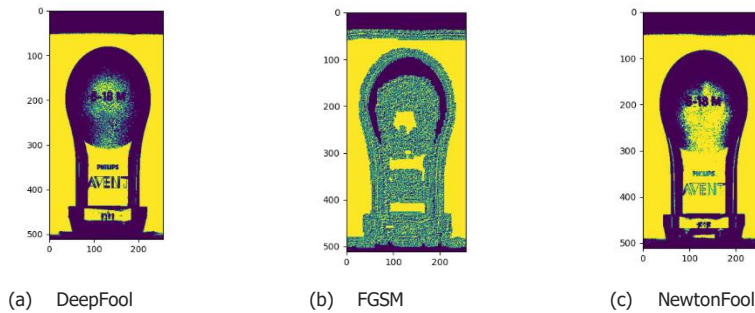


Figure 10 Applied Attacks on the Soother Dataset (Philips)

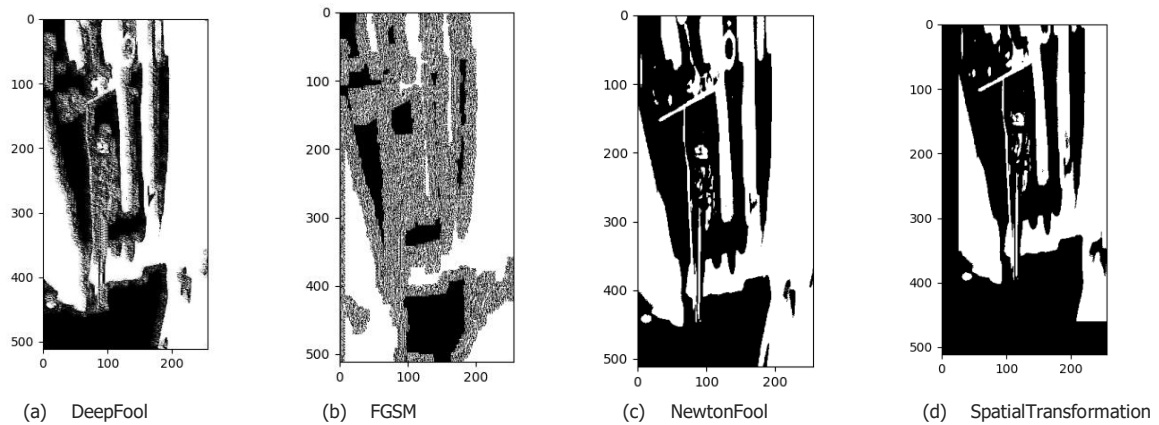


Figure 11 Applied Attacks on the Assembly of Horizontal Lamellas Dataset (IBER)

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 should decline dramatically, depending on the robustness of the baseline model against the noise which is injected in the images.

## 3.5 Defence Strategies

### 3.5.1 Adversarial Training

As was the case also in D3.3, Adversarial training is the prominent defence strategy that we aim to test and evaluate in the context our scenarios in order to defend the pilot environments against the advanced AI attackers. The basic idea behind adversarial training is to simply create adversarial examples and later to use them in the training process. In fact, adversarial training was proved the most robust defence based on the experimental results acquired in D3.3.

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.

### 3.5.2 ART Defence strategies

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

with the experiments conducted in D3.3, we evaluate our models again after an attack and after the application of a defence so that to evaluate their defences’ efficacy.

In D3.4 we opt to evaluate the following defences:

- FeatureSqueezing [REF-03]
- JpegCompression [REF-04]
- SpatialSmoothing [REF-04]
- TotalVarMin [REF-06]

An example of adversarial examples and images produced after the defence methods can be seen in the following figures.

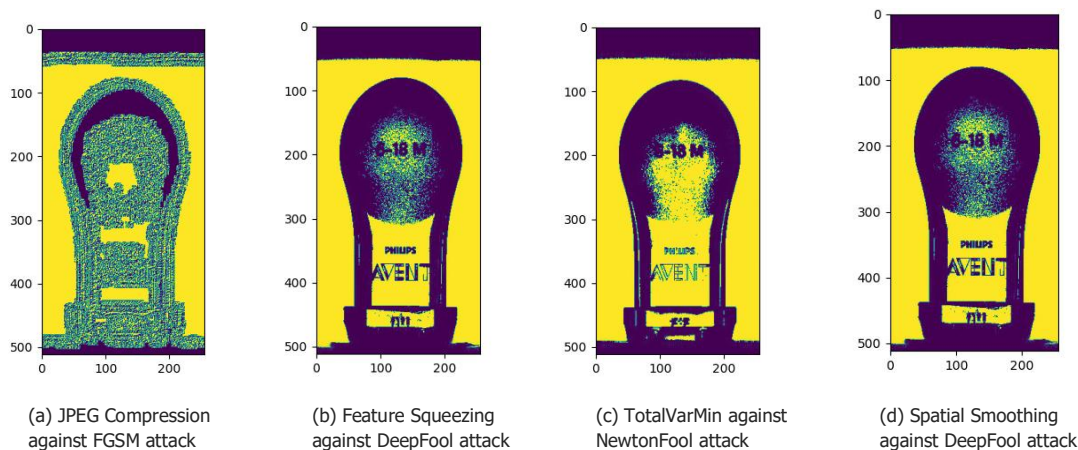


Figure 12 Applied Defences against attacks for Soother Dataset (Philips)

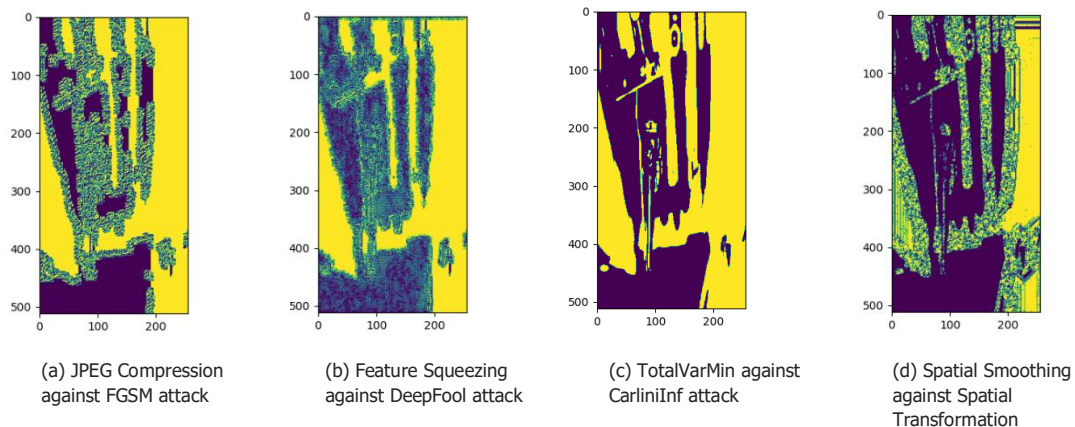


Figure 13 Applied Defences against attacks for Assembly of Horizontal Lamellas Dataset (IBER)

### 3.6 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.

Keep a note that, the evaluation and the produced plots occurred using a standard validation technique known as K-fold Cross Validation. In addition, as we mentioned before, our evaluation testbed is built on the basis of a Binary classification problem that imitates an AI-based visual inspection system for quality assurance in a production line.

### 3.6.1 Soother Dataset Results

The goal of the testbed was to quantify the impact of adversarial attacks on classification models performing a visual inspection and evaluate how effective the defences against such attacks are. Figure 14 presents collectively all the experimental results of the pairwise evaluation of the Attacks-Defences that we conducted for the Soother dataset.

To build the testbed, we used the Adversarial Robustness Toolbox, and we applied 11 different attacks, 4 off-the-shelf defences and Adversarial Training. The aim was to utilize these well-documented adversarial methods to derive crafted instances that can be used to attack the baseline classification model.

We summarize the results of the pairwise evaluation of the attacks and defences in Figure 14. The results are grouped into eleven sets based on the attack strategy. We initially trained the baseline classifier for each of the four experiments (see tag "Training") to get the perception of the accuracy level that the quality inspection algorithm can achieve. The baseline model achieved an accuracy between 88% and 98%.

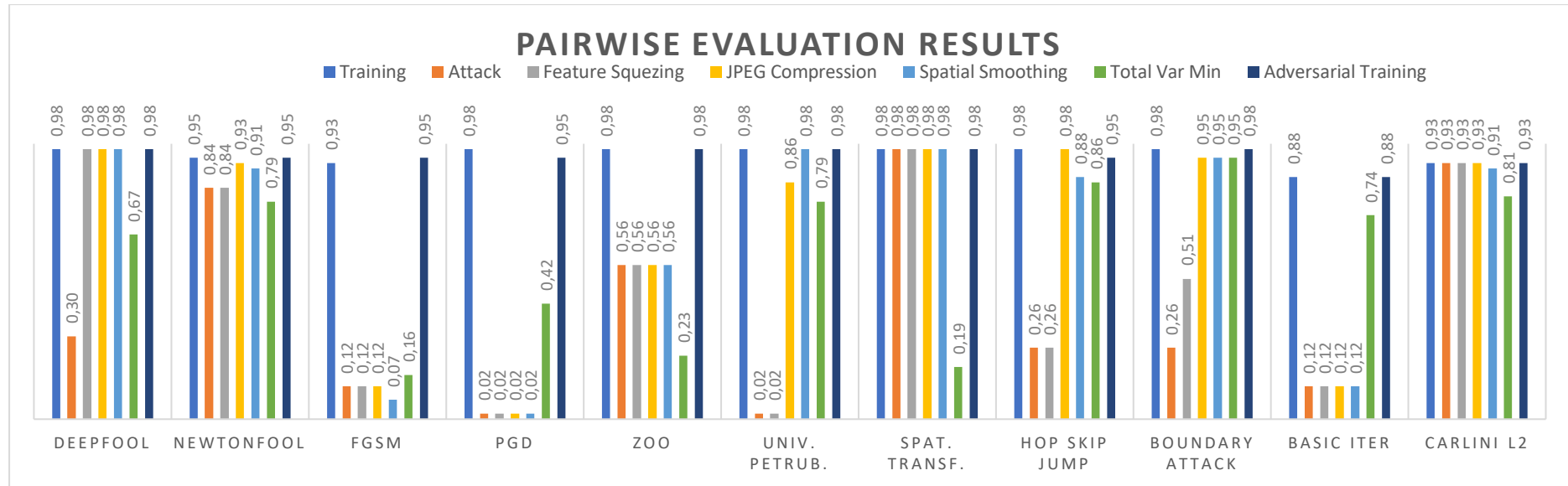


Figure 14 Collective experimental results on pairwise evaluation of Attacks-Defences using Soother Dataset (Philips)

The "Attack" bar indicates the accuracy of the classifier when posed against the adversarial attack. We found the DeepFool, FGSM, PGD, ZOO, Universal Perturbation, HopSkipJump and Basic Iterative attacks strongly affected the classifier, causing the model's accuracy to drop below 30%. This was not the case for the NewtonFool attack, Spatial Transformation and Carlini L2 attacks, where the classifier's accuracy dropped to 84%.

When considering defence strategies, we found that Feature Squeezing, JPEG Compression, and Spatial Smoothing can defend against the DeepFool attack: for the given dataset, they led to an accuracy of 98%. However, TotalVarMin failed to defend the model.

We also found that all the defences failed against the FGSM and the PGD attacks. The same applies for the ZOO attack, where all the defences revitalised the model, but the gained accuracy level does not differ from a random classifier.

Based on the acquired results of the pairwise evaluations, it becomes clear that no clear mapping exists between types of attacks and defences. Therefore, it can be inferred that defending against intelligent AI attacker is a really challenging task and it is difficult for the defenders to plan a strategy to cope against any attack successfully. This outcome advocates the criticality and the challenge of defending against adversarial AI attacks. In fact, this outcome is in line with the experimental results that we conducted in the context of D3.3, when using other datasets from the pilot partners. Thus, we witness a same behaviour among our experiments.

While the off-the-shelf and state-of-the-art defences cannot perform in a stable manner under different adversarial methods, the Adversarial Training approach seems to be robust. Our results agree with the literature, advocating that Adversarial Training can be a robust solution that can cope with adversaries despite its simplicity. More specifically, among the 11 different attacks that we launched against the model, when the model had the knowledge about the attack to be faced it was able in all cases to achieve an accuracy level similar to the initial model.

The results presented in Figure 14 provide a clear indication that the model which will empower the *Discriminator* component of the AI Cyber Defence tool will be the one built using the Adversarial Training approach.

Part of the results presented in this section on the evaluation of Soother dataset have been published in [REF-44].

### 3.6.2 Assembly of Horizontal Lamellas Dataset Results

This section reports on the results generated throughout the experimental process when using the Assembly of Horizontal Lamellas Dataset from IBER. The results are presented in Figure 15. In this case we have applied 9 different attacks (depicted in the respective column sets in Figure 15), 4 off-the-shelf defences and Adversarial Training.

Overall, one can notice that the results of Figure 15 reflect almost the same behaviour noted in the previous section. In other words, we cannot be certain that a specific attack can be mitigated by a specific defence, while there is no defence that can be considered capable of mitigating all the attacks. Hence, the acquired results confirm that defending against the adversarial tactics is a really challenging task.

By taking a closer look at the results of Figure 15 one can notice that the DeepFool, NewtoolFool FGSM, PGD, Spatial Transformation, Universal Perturbation and Basic Iterative had a significant effect to the accuracy level of the base model. The only attacks that failed to compromise the base model was the ones of the Carlini family. Again, it is evident that also for the case of the IBER datasets, an AI adversary can significantly affect a visual inspection system and, hence, disrupt the manufacturing process. Most of the attacks were successful, denoting the need for defending the manufacturing process and the AI-based quality inspection system.

The results of the applied defences showed that in some case the attacks were mitigated. For instance, all the defences were capable to revitalise the model against the NewtonFool attack and the Spatial Transformation. On the downside, the off-the-shelf defences were capable to defend the model against the FGSM, PGD, Universal Perturbation and Basic Iterative. This outcome is aligned to the results acquired in the previous section and to the outcomes of D3.3, advocating that there is no single defence that can mitigate all the attacks.

On the bright side, Adversarial Training seems again, also for the case of the IBER dataset, a prominent solution to empower the AI Cyber Defence tool and defend the against the various utilised attacks. More specifically, according to the results of Figure 15, Adversarial Training was able to keep high accuracy rates for all the cases, except for the case on the Universal Perturbation, making evident that it remains the most robust solution to defend against poisoning and evasion attacks.

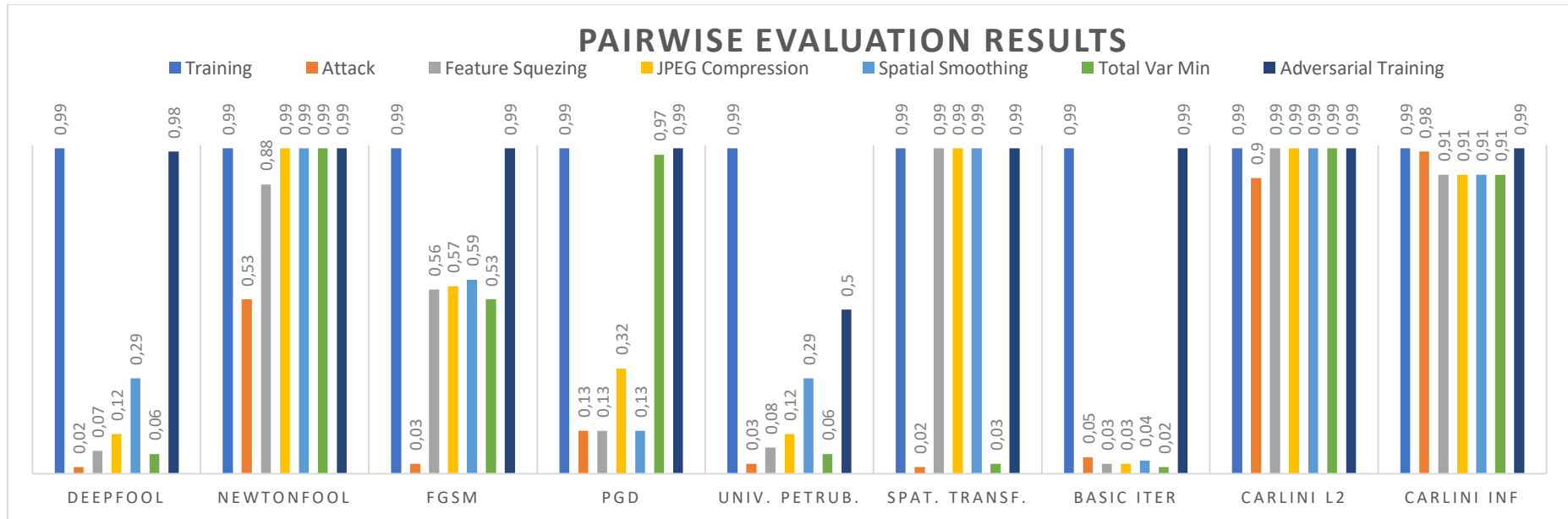


Figure 15 Collective experimental results on pairwise evaluation of Attacks-Defences using the Assembly of Horizontal Lamellas Dataset (IBER)

### 3.7 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. This is an outcome that comes in agreement with the results of the 1<sup>st</sup> experimentation phase which was conducted in D3.3.

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. Hence, the adversarial training model will be the one that will empower the Discriminator component of the AI Cyber Defence tool in the context of the pilots.

## 4 Explainable Artificial Intelligence to Enhance Data Trustworthiness in Crowd-Sensing Systems

This section offers a summary of the work published in [REF-43] and is the extension of the work presented in D3.3 and in [REF-01]. This section is an outcome of the research actions conducted in tasks 3.2 and 3.3 and comes as a research direction of the long-term vision of UBITECH for developing AI defence methods against poisoning and evasion attacks. This work addressed the part of the architecture of the AI Cyber Defence tool which implies the connection with the RMS for the acquisition of sensory data from the manufacturing floors, and the exploitation of XAI methods for assisting the detection of adversarial attacks. It has to be noted that this dependency with the RMS and XAI has been only approach from a scientific perspective. Hence, this series of works in [REF-01] and [REF-43] aim to evaluate the applicability of XAI in the battle against poisoning and evasion attacks, when focusing specifically to sensory data (in the form of time series data).

Furthermore, due to the lack of sensory data from the STAR pilot partners by the time that these works in [REF-01] and [REF-43] were under conceptualisation and development, we turned to open source datasets of sensory data in order to investigate the applicability the developed defence methods. Our vision is to investigate the possibility of applying the developed methods in the future if enough data are generated and documented by the STAR pilot partners, and to continue our research actions even after the completion of the STAR project.

Given the above, the motivation of this work lies on the fact that around the world there has been an advancement of IoT edge devices, that in turn have enabled the collection of rich datasets as part of the Mobile Crowd Sensing (MCS) paradigm, which in practice is implemented in a variety of safety critical applications. Despite the advantages of such datasets, there exists an inherent data trustworthiness challenge due to the interference of malevolent actors. In this context, there has been a great body of proposed solutions which capitalize on conventional machine algorithms for sifting through faulty data without any assumptions on the trustworthiness of the source. However, there is still a number of open issues, such as how to cope with strong colluding adversaries, while in parallel managing efficiently the sizable influx of user data. Thus, in this section we suggest that the usage of explainable artificial intelligence (XAI) can lead to even more efficient performance as we tackle the limitation of conventional black box models, by enabling the understanding and interpretation of a model's operation. Our approach enables the reasoning of the model's accuracy in the presence of adversaries and has the ability to shun out faulty or malicious data, thus, enhancing the model's adaptation process. To this end, we provide a prototype implementation coupled with a detailed performance evaluation under different scenarios of attacks, employing both real and synthetic datasets. Our results suggest that the use of XAI leads to improved performance compared to other existing schemes.

### 4.1 Introduction

Temporal data can be quite unpredictable to humans as it can be hard for someone to understand and explain why changes over time happen by just observing the data as an instance in the form of a picture or text. That is why it can be tricky to represent temporal data as a signal that varies as a function of time due to the ubiquitous nature of temporal data. To enable knowledge extraction out of such data we need to leverage additional methods along with expert knowledge [REF-14].

The abundance of devices worldwide has created a mobile sensing landscape that requires users to feed information about sensory data of the environment, within a sensing paradigm, known as mobile crowd sensing (MCS) [REF-15]. The generated data often take the form of a temporal data set. In terms of security, several issues arise on data quality and integrity due to the openness of these platforms [REF-16], while the same issue affects significantly other domains which capitalize on the collection of data from sensors, such as the smart manufacturing domain [REF-17][REF-18][REF-19]. Previous works aim to tackle these issues by making use of Machine Learning. ML systems are trained using MCS datasets that are assumed to be benign, whereas malicious actors attempt to poison them by targeting training data through simple or advanced manipulations or can directly evade models' inference process using "adversarial examples" aiming to deceive completely the model.

There has been a plethora of research works trying to address such security issues using adversarial machine learning leveraging conventional ML-based techniques for distinguishing between malicious and benign data [REF-20]. However, those conventional techniques have proven to be inefficient especially in cases where the discrimination between concept drift and cases where untrustworthy data sources provide falsified data (either as the result of an attack or a malfunctioning sensor) is hard. In fact, the synthesis of concept drift and false data injection can enable an intelligent attacker to hide in the shadows and affect significantly the accuracy of both clustering and classification processes.

In our research path, we have previously worked on applying advanced Deep learning schemes [REF-21] that were capable of distinguishing malicious and benign data, without any assumptions regarding the trustworthiness of data sources, whilst being robust against many colluding intelligent adversaries. In this paper, we take a step forward by addressing the plausible relation between the sensory input from devices over time via the application and integration of explainable artificial intelligence (XAI). XAI is exploited for the purpose of gaining knowledge and insights regarding the temporal data set that typically impose restrictions on human comprehension [REF-14]. Capitalizing on the pipeline of FSD [REF-21], we further expand our method by introducing the aspect of XAI which enables the more exhaustive investigation of the aforementioned security challenges, leading to the proposed XFSD framework.

## 4.2 Related Work

A great mass of works applied Machine learning and Deep Learning (DL) in the mobile crowd sensing infrastructures, addressing issues pertaining to concept drift, data integrity, sparse sensing [REF-16][REF-22][REF-23]. In this paper, we focus on works that applied XAI to sequential time-series data so we can possibly incorporate it into the existing FSD framework [REF-21][REF-24]. There is a variety of works that apply XAI to time series but it can be broken down to the distinct categories of Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and non-sequential Neural Network based methodologies [REF-14].

RNNs are a great fit for time series datasets which also encompass the application of LSTM model that retains the whole history of the time data. The XAI methodology used typically within this context is some form of SHAP which is a model-agnostic explanation method. Kim et al. [REF-25] were the first to suggest using SHAP algorithms to explain the output of RNN used on time series data to produce highly accurate outcomes. SHAP has been applied successfully to financial time series prediction and explanation [REF-26][REF-27][REF-28][REF-29].

Regarding CNN-based methods, the prevalent approach of applying XAI to time series is class-activation mapping. This is a post-hoc method that highlights regions in the input data which affect the CNN output with quite accurate results [REF-30][REF-31][REF-32]. Within the category of CNNs, there are other interesting approaches, such as ConvTimeNet which occludes part of the time series and computes the probability of the predicted class [REF-33]. Another approach is Gradient\*Input to identify the contribution of the input raw data in time series classification tasks [REF-34]. In terms of non-sequential methods, explanations can be generated using LIME. This approach is less accurate than the sequential explanatory methods but it is more efficient [REF-35][REF-36][REF-37].

### 4.3 Towards Trustworthy MCS Tasks

Mobile Crowd Sensing (MCS) information platforms allow users to contribute by uploading their data to a central server, where the collected information can be processed by analytic algorithms for sharing knowledge with other applications of different fields [REF-38]. To put it mathematically, for a specific time window with  $n$  time steps and  $m$  sensors, we gather a dataset  $\mathbf{D}$  containing a sequence ( $\mathbf{S}$ ) of values  $\mathbf{v}$  for each sensor  $\mathbf{j}$ , having  $S_j = [v_{1,j}, v_{2,j}, \dots, v_{i,j}, \dots, v_{n,j}]$ .  $v_{i,j}$  represent the value of sensor  $\mathbf{j}$  at time step  $\mathbf{i}$ .  $\mathbf{D}$  in Figure 16 shows how  $\mathbf{S}$  sequences build the matrix as the dataset. The sequential structure of the data makes the use of LSTMs an interesting approach for extracting useful insights from the relationships between the various sequence values.

$$D = \begin{bmatrix} v_{1,1} & \dots & v_{1,j} & \dots & v_{1,m} \\ \vdots & & \vdots & & \vdots \\ v_{i,1} & \dots & v_{i,j} & \dots & v_{i,m} \\ \vdots & & \vdots & & \vdots \\ v_{n,1} & \dots & v_{n,j} & \dots & v_{n,m} \end{bmatrix}$$

Figure 16 Dataset structure

#### 4.3.1 Threat modelling

Colluding adversaries aim to mislead MCS apps by incorporating an adversarial agent for generating malicious measurement values that seem to be legitimate to the application.

There are two main adversarial models which must be taken into consideration for such a platform [REF-20], namely the: 1) pre-training (poisoning) attacks, and 2) post-training (evasion) attacks. With pre-training attacks, the adversaries attempt to inject malicious data to poison the training dataset and, thus, build a model which is formed based on misleading instances. The post-training attacks occur in testing time, i.e., during the inference mode of the model, where the adversaries can craft adversarial data that prevent a machine learning model from correctly identifying the contents of the data, i.e., to mislead the already trained model to mis-classify the injected samples.

In principle, in any ML approach we try to minimize False Positives and False Negatives. On the contrary, the adversarial objective is to maximize the impact of the attack by maximizing these metrics. For the remainder of the paper, we refer to the adversarial data as the positive class, and the legitimate data as negative one.

### 4.3.2 Explainable AI

Explainable AI (XAI) is a set of processes and methods that allows human users to comprehend and trust the results and output created by machine learning algorithms in contrast to typical black box approaches of some algorithms, especially neural networks. Machine learning has a long history of being applied to time series data, but most of the solutions make the interpretation difficult, and without a quantitative assessment it may be hard to understand areas of temporal data which may affect the prediction. XAI can tackle this issue by adding additional knowledge that may strengthen the overall model prediction ability [REF-14][REF-35][REF-39].

There is a wide spectrum of algorithms to use when applying XAI on time series information. In this paper, we deal with sequential time series data and we exploit two of the most common XAI algorithms, namely the SHAP (SHapley Additive exPlanations) and Class Activation Mapping (CAM). Moreover, we evaluate the impact of a well-known non-sequential based XAI algorithm called LIME (Local Interpretable Model-agnostic Explanations) to study the difference between sequential and non-sequential approaches.

Generally, the pipeline of applying XAI to the numerical values of time series aim to identify the significance that the input values have on the model prediction result. In our model, we take as input the raw data and we obtain a vector of fixed length, determined by a sliding-window, fed into a DL model so that to acquire a prediction. Then, by using an XAI method, we can calculate impact scores which reflect the impact that an input value has on the prediction at a given time. The outcome is a numerical representation which denotes whether a given step in the raw dataset affects the prediction negatively or positively considering the input and the predicted values. In this way, XAI is applied in a holistic manner to the model and explains which are the areas of importance of the input data. Then, this outcome, i.e., the value obtained by the XAI model, can be used in order to enrich or adjust the NN model accordingly. The rationale behind this approach is that the XAI values are also affected by the adversarial attack strategies and, intrinsically, they bear the information for the presence of malicious data points in a dataset. In this way, XAI can contribute to the identification of attack vectors, which has been applied to the system and the data, by utilizing the generated XAI values. Thus, having these values, we can apply an aggregation model to generate enhanced data vectors that combine the predictions and the XAI importance values, aiming to make the adversarial inputs more distinguishable. In fact, this is the key difference between the FSD approach presented in [REF-21] and the enhanced XAI-assisted method, namely the XFSD, presented here.

## 4.4 XFSD Conceptual Architecture

The Fake Sequential Data (FSD) detection framework [REF-21] was created to benefit from the time-based relationships between dataset samples so as to accurately predict the values of the next time step. That system utilized distance metrics for comparing the predicted data and real data, at each time step, in order to enable a one-class classification approach to essentially draw a boundary around the allowed (expected) deviations between predicted and real data values.

The XFSD approach augments the existing data with numerical values from the XAI models which represents the importance of each feature over time. The input to XAI is the raw sequence of values for each feature along with a trained model, while the output is a numerical vector which denotes the importance score of the feature. Then, we apply an aggregation

model to concatenate the predictions and the XAI importance values to one dataset. Like most deep learning frameworks, our approach consists of two main phases, namely the *Training* and *Testing* phases which are detailed below.

### 4.4.1 Training Phase

Figure 17 shows the steps followed by XFSD to train the final one-class classifier to distinguish benign samples (not generated by adversaries). Similar to FSD [REF-21], XFSD framework trains a sequence predictor ( $P_j$ ) for every sensor  $j$  that takes an input sequence and predicts the next step value (Step 1 of Figure 17). Step 2, for each timestep, uses the trained predictor ( $P_j$ ) to estimate the expected value ( $e_j$ ) at time step  $i$  by feeding the sequence ( $S_j$ ) of all values of sensor  $j$  from timestep 0 to timestep  $i-1$ .  $a_i$ , then, is the distance vector between the real values and the predictions.  $a_i$  is of size  $K$  which corresponds to the number of  $K$  distance metrics (e.g., Cosine Distance, etc) used. At the same time, feeding the sequence  $S_j$  and the predictor  $P_j$  to the XAI explainer  $X_j$  will generate a vector of values ( $\beta_{i,j}$ ) as the explanation of the data at the current step. Concatenating and flattening  $a_i$  and all  $\beta_{i,j}$  will result to  $\vec{z}_i$  vector (for row  $i$ ) of  $Z$  (see Figure 18).

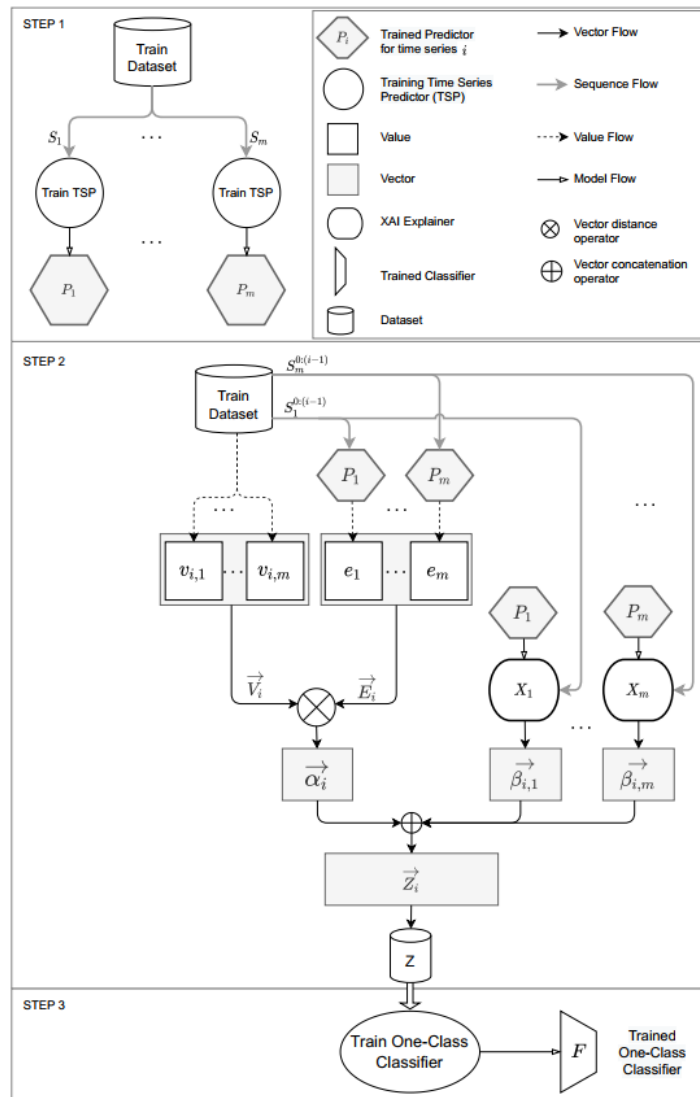


Figure 17 Training phase of XFSD

#### 4.4.1.1 Step1: Train time-series predictors

This step trains a predictor  $P$  for each sensor that is able to predict sensor value of timestep  $\mathbf{i}$  based on the sequence of values at previous timesteps. Although this step is not limited to sequential based algorithms, we recommend using Recurrent neural networks (RNNs) and Convolutions neural networks (CNNs) as they are widely used in predicting time series. The output of Step 1 (*Predictors*) will be used later in Step 2.

#### 4.4.1.2 Step2: Explainable feature extraction

We apply an XAI algorithm (any of the SHAP, CAN, LIME) which effectively calculates a numerical representation of the important areas of the data input based on the sequential model predictor. Step 2 of Figure 17 shows the procedure of computing  $\vec{a}_i$  and  $\vec{\beta}_i$  into  $\vec{Z}_i$ . The output of this step is matrix  $Z$  that will be used in Step 3 to train the final one class classifier. In  $Z$ ,  $q = K + m \times a$ , as shown in Figure 18.

$$Z = \begin{bmatrix} z_{1,1} & \dots & z_{1,k} & \dots & z_{1,q} \\ \vdots & & \vdots & & \vdots \\ z_{2,1} & \dots & z_{i,k} & \dots & z_{i,q} \\ \vdots & & \vdots & & \vdots \\ z_{n,1} & \dots & z_{n,k} & \dots & z_{n,q} \end{bmatrix}$$

Figure 18 Structure of  $Z$  matrix

#### 4.4.1.3 Step3: Training one-class classifier

The completion of step 2 means that all legitimate data have been fed to the *Predictors* and the system calculated  $Z$  for each time step  $\mathbf{i}$  containing all *allowed deviations* between the predictions and the real data plus the XAI values for all time steps.  $Z$  matrix is suitable for a one-class classifiers as it is synthesized by feeding only benign data to the predictors and the explainers. The one-class classifier attempts to find a boundary around the training samples (based on benign samples) in order to distinguish them from data that follow other distributions.  $\mathbf{F}$  in Figure 17 represents the trained one-class classifier.

#### 4.4.2 Testing phase

Figure 19 shows how XFSD works in real time to detect adversarial samples. For each testing time step  $\mathbf{i}$ , XFSD uses predictor  $P_j$  to produce  $e_j$  for sensor  $j$ . Then it concatenates and flatten  $\vec{a}_i$  and  $\vec{\beta}_i$  into  $\vec{Z}_i$  so that to feed the one-class classifier  $\mathbf{F}$  in order to specify if the data received in the current time step follows the distribution of benign data. As  $\mathbf{F}$  is a trained one-class classifier on legitimate deviations between real and predicted values as well as the XAI values, the output dictates whether  $\vec{Z}_i$  falls within the boundaries of the legitimate behaviour or not. If  $\mathbf{F}$  suggests that  $\vec{Z}_i$  is not in the distribution of benign samples, then  $\vec{V}_i$  will be flagged as malicious, as one or more sensor values at time step  $\mathbf{i}$  in  $\vec{V}_i$  were generated by adversaries.

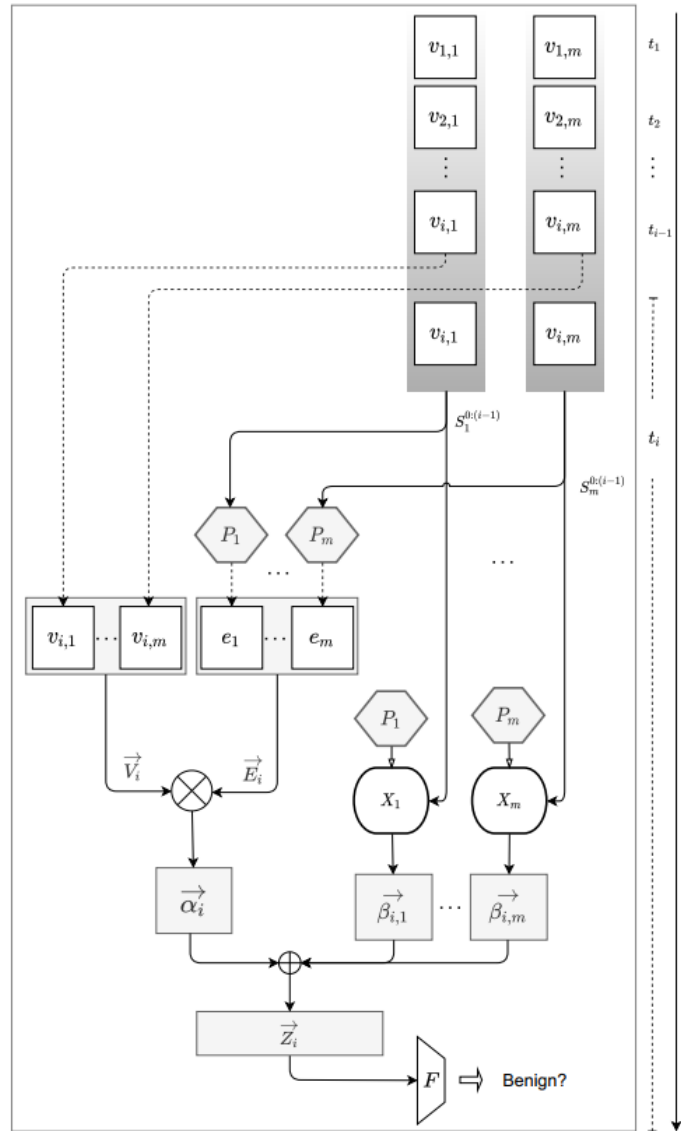


Figure 19 Testing phase of XFSD

## 4.5 Experimental Setup

XFSD is an extension of FSD which introduces the use of XAI on numerical values as a way of improving the detection rate of malicious data. We offer an experimental testbed with more advanced attack strategies in order to witness the performance improvements of XFSD.

### 4.5.1 Adversarial Behaviour

Following the threat model presented in section 4.3.1, the adversary may attempt to feed malicious data points which are then included in the training of the deep learning model used for the classification or sends malicious data after the model and the classifier has been trained on the real benign data points. Therefore, we categorize the attack strategies into pre-training and post-training attack strategies, respectively.

The performance of the framework was evaluated based on different levels of distortion. In XFSD we consider two main attack approaches for each attack strategy: 1) distribution attacks (Attack Cases I & II)} and (Attack Cases VI & VII), which manipulate the mean or standard deviation to generate adversarial samples which are different from the benign ones; and 2)

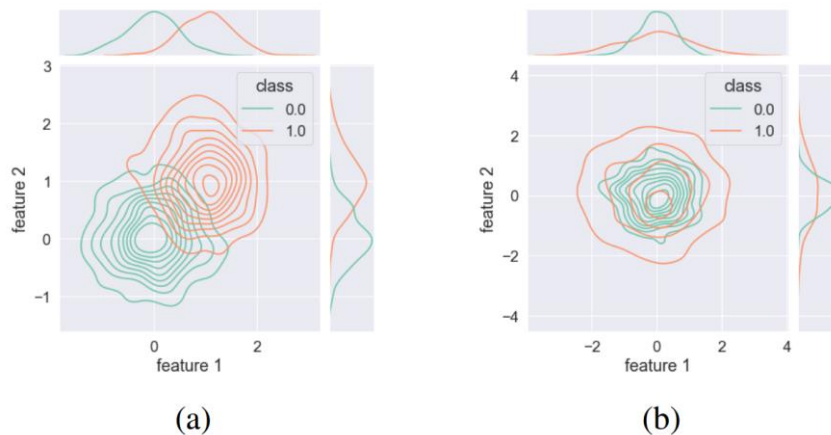
position attacks (Attack Cases III, IV & V)}, which manipulate the position of injecting adversarial samples in the sequence of values, which, in turn, targets the order of samples. Position attacks can only be applied after a distribution attack towards changing the order of samples to be injected.

**Attack Case I:** The adversaries may affect the system uncertainty by setting  $\sigma' = \sigma$  and  $\mu' \neq \mu$  which represents a malicious standard deviation equal to the standard deviation of the legitimate data points, but with smaller/larger  $\mu$ . In Equation 1,  $\lambda$  is a scalar value as the deviation factor. Adversarial samples generated with  $\lambda > 2$  are not attractive as they are easier to detect due to their lower overlap with benign samples. That is, we limit  $\lambda$  between 0 and 2 to study more realistic data distributions.

*Equation 1  $\mu$  and  $\sigma$  for Attack Case I*

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

To further illustrate what the actual structure of this attack we refer to Figure 20 where there is a comparison of the distributions of legitimate samples and adversarial samples generated by Attack Case I for a simple dataset with only two features, where  $\lambda = 1.0$ . The adversaries may choose the  $\lambda$  value depending on the attack strategy (pre-training or post-training) and the number of attacking samples they want to inject/test notated as a percentage of the total number of samples. As seen in the figure, changing the mean value would have a greater impact on changing the distribution as it will not have a huge overlapping region with the actual distribution of the legitimate data. The intuition is that trained classifiers should actually work better in detecting malicious data if they were trained solely on a legitimate distribution.



*Figure 20 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 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 (SD).

### Equation 2 $\mu$ and $\sigma$ for Attack Case II

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

Similar to case I, we refer to Figure 20 where one can see 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 behavior by performing concept drifting and by modifying the SD.

**Attack Case III:** In this case of a positional attack strategy, the adversaries may affect the system 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:** The opposite of case III, in this case all malicious data come before benign data points in the sequence.

**Attack Case V:** The adversaries attempt to affect the system uncertainty by putting malicious batches of data in between legitimate data batches. Variations of this approach inspired us for the expansion of positional attacks including more specific patterns including randomization, normal and different sized windows of malicious data batches inserted.

**Attack Case VI:** In this case we perform an availability attack which aims to maximize the error at the point where the adversaries are not detected by the model. Practically, this means that we feed the system a dataset during the pre-training which will broaden the boundaries that the classifier can draw for benign samples, and thus, making adversarial inputs indistinguishable during the testing phase. The focus is on increasing the variance in the data poisoning process. To do so, we use bilevel optimization approach so that to define the optimal attack strategy, i.e., to define the optimal shift, distribution, and the number of malicious samples which will shift the outcome as much as possible from the truth. We apply this method on cases I and II i.e., shifting the  $\mu$  and  $\sigma$ .

**Attack Case VII:** In this case we perform a targeted attack focusing on steering the model estimation towards a certain prediction. In this concept, the attacker poisons the model so that to target a pre-determined outcome. In contrast to availability attack of case VI, where the attacker's objective is to deviate the response of the model in an untargeted manner, in the targeted attack, the attacker aims to have control over the model's prediction, and this requires to develop an optimized adversarial strategy. Practically, we generate a dataset for a specific pre-determined target, we feed it during pre-training and we observe the effects through the testing phase. To achieve this, we make use of bilevel optimization to obtain the optimal variable to be manipulated in order to steer the model to the pre-determined target. We apply the targeted attack considering the cases I and II i.e., shifting the  $\mu$  and  $\sigma$ .

## 4.6 Results

In this section, we evaluate the performance of XFSD. For this purpose, we utilize F-measure metric as it considers True/False positive rates, recall and precision. We divided the dataset into two subsets with a 60%-40% split for the training and testing respectively. The dataset

considered in this paper originates from real world measurements collected from Data Sensing Lab [REF-45] by sensors deployed at the Strata Clara convention center in 2013.

In addition, before converging to a specific configuration setup for our testbed, we performed a number of experiments in order to define the “optimal” size of the sliding-window. A sliding window of 1000 instances was the best fit for our model based on the acquired F-measure performance results. Thus, with the optimal setup and by incorporating the XAI methods, we report the results, and we perform a comparative analysis against FSD [REF-21] in order to advocate the merit of XFSD.

Note that for attack cases VI and VII we have considered only pre-training attacks so that to evaluate the performance of XFSD framework against more advanced poisoning attacks compared to the simpler poisoning approaches of cases I and II. The following sections are structured as follows: Firstly, we focus on distribution attack cases (i.e., I and II). An analysis on the positional attacks follows (i.e., attack cases III, IV and V). The analysis concludes with a section focusing on the advanced attack cases (i.e., VI and VII) where both cases include manipulation of both  $\mu$  and  $\sigma$ .

#### 4.6.1 Results of XFSD Distribution attacks

##### 4.6.1.1 XFSD case I and II in pre-training attack strategy

With the pre-training attack strategy, the adversaries try to poison the training datasets. It is generally harder to detect malicious samples in the pre-training than the post-training since classifiers are trained to recognize the adversarial samples as legitimate. Increasing the rate of adversarial inputs in pre-training has a higher impact on classifiers. For the same reason an attacker would choose a higher deviation factor as the end-goal is to mislead the classifier's perception in that scenario. Table 3 and Table 4 summarize the results for attack cases I and II, respectively for the pre-training attack strategy. Each row of the tables shows F-measure for a specific  $\lambda$  value used by the attacker to generate malicious data. It is expected that by increasing  $\lambda$  or the adversarial data rate, this should lead to less accurate classification in the pre-training strategy, as this would skew the perception of the classifier. This F-measure degradation tendency is in fact reflected in the results of Table 3 and Table 4.

All these reflections are as expected from what we know from FSD without the XAI values. The interesting aspect is the positive impact of applying different XAI methods. As showcased in the tables the tendencies are more or less the same across the different algorithms, implying that the impact of different attack cases in pre-training strategies is the same regardless of the algorithm applied. The only difference is the scale of accuracy among their different XAI methods. Generally, the introduction XAI to FSD provides a proportional improvement as degradation is the same with respect to the different configurations of adversarial attacks between FSD and XFSD. However, the F-measure is clearly higher for XFSD showcasing a significant improvement when adding XAI values, advocating the motivation of this work for introducing XAI values as the additional information to improve the overall performance of classifiers.

*Table 3 Results of various XAI algorithms + FSD for Attack Case I in pre-training attack strategy*

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.963	0.929	0.941	0.933	0.915	0.881	0.865	0.848	0.826
	1	0.952	0.882	0.909	0.889	0.895	0.856	0.823	0.804	0.771
	2	0.914	0.837	0.876	0.849	0.828	0.819	0.792	0.769	0.719
CAM	0.5	0.886	0.841	0.8	0.791	0.777	0.741	0.725	0.707	0.686
	1	0.841	0.81	0.769	0.749	0.752	0.716	0.681	0.664	0.631
	2	0.794	0.774	0.735	0.708	0.686	0.674	0.652	0.629	0.579
LIME	0.5	0.85	0.805	0.764	0.755	0.741	0.705	0.689	0.671	0.65
	1	0.805	0.774	0.733	0.713	0.716	0.68	0.645	0.628	0.595
	2	0.758	0.738	0.699	0.672	0.65	0.638	0.616	0.593	0.543
FSD	0.5	0.843	0.797	0.756	0.747	0.733	0.695	0.682	0.663	0.642
	1	0.797	0.765	0.725	0.703	0.707	0.672	0.637	0.619	0.587
	2	0.749	0.729	0.691	0.664	0.641	0.629	0.608	0.583	0.533

*Table 4 Results of various XAI algorithms + FSD for Attack Case II in pre-training attack strategy*

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.875	0.861	0.839	0.821	0.802	0.782	0.762	0.718	0.684
	1	0.836	0.811	0.802	0.789	0.769	0.753	0.719	0.669	0.635
	2	0.799	0.789	0.784	0.779	0.742	0.727	0.684	0.638	0.589
CAM	0.5	0.866	0.852	0.83	0.811	0.792	0.772	0.752	0.708	0.675
	1	0.827	0.801	0.792	0.779	0.759	0.743	0.709	0.659	0.625
	2	0.789	0.779	0.776	0.769	0.732	0.718	0.676	0.627	0.579
LIME	0.5	0.858	0.844	0.822	0.803	0.784	0.764	0.744	0.7	0.667
	1	0.819	0.793	0.784	0.771	0.751	0.735	0.701	0.651	0.617
	2	0.781	0.771	0.768	0.761	0.724	0.71	0.668	0.619	0.571
FSD	0.5	0.791	0.777	0.755	0.736	0.717	0.697	0.677	0.633	0.6
	1	0.752	0.726	0.717	0.704	0.684	0.668	0.634	0.584	0.55
	2	0.714	0.705	0.701	0.694	0.657	0.644	0.601	0.552	0.504

The reasoning behind this improved performance is that the adversarial strategies also impact the explanation values obtained by XAI. That is, the explanation values are a valuable addition for better distinguishing between legitimate and malicious instances. Furthermore, in its foundation, XAI provides values that explain the impact of input to a NN model. With that knowledge it is possible to adjust weights that may improve a model mitigating issues like concept-drift.

Another part of the experimentation of this project was to find the best XAI algorithm to employ. We had some insights, why certain algorithms may perform better given the dataset we are working with. In fact, we expected SHAP to outperform the other methods as it is designed for scenarios using RNN or LSTM, which is the basis of FSD, which is widely recognized as the conventional way of handling time-series information. To illustrate this performance let us take an example of  $\lambda = 2$  and 40% adversarial rate for all algorithms in case I in pre-training strategy. SHAP is clearly the most accurate method in terms of F-measure. However, it is interesting to note that LIME achieves worse performance than FSD. The reasoning is that LIME is a non-sequential algorithm which indicate that there is a significant amount of information that can be extracted from the sequential order of our data, and LIME fails to capture this behaviour.

#### 4.6.1.2 XFSD case I and II in post-training attack strategy

In post-training scenario the attacker attempt to generate samples for bypassing the classifiers during testing time without any access to the training data. Thus, for this attack strategy the models are trained on legitimate data and then the overall classification is evaluated using manipulated data. In the post-training strategy, it should be easier for a classifier to detect non-overlapping malicious samples as the classifier has the knowledge to cover the legitimate area. Therefore, adversaries will tend to not deviate the incoming poisoned dataset from the legitimate dataset i.e., not change  $\lambda$  much.

As shown in Table 5 and Table 6 we can see that we achieve higher F-measure scores when the malicious samples are increased and the deviation  $\lambda$  is larger. This makes sense since the malicious data points in those cases are far from the boundaries of the legitimate behaviour, making it easier to be detected by the classifier. Again, these are expected results for XFSD from what we have previously learned from FSD and, as in the scenario of pre-training strategy, the post-training strategy shows similar tendencies across models for all configurations in both attack cases in post-training.

When focusing on the behaviour of the different XAI algorithms, unsurprisingly the tendencies are the same as they were for the pre-training cases. XFSD-SHAP is the strongest and performs with a higher F-measure than base FSD proving that adding XAI-values can have a positive effect as discussed throughout this paper. The dominance of SHAP is evident across the results of Table 5 and Table 6 regardless the  $\lambda$  and the adversarial rate configurations. The same highlights can be extracted for the post-training case as for the pre-training one, suggesting that SHAP is more accurate, as it is destined to work well with RNN and time series data, while the contribution of XAI is evident based on the performance results of XFSD against FSD.

*Table 5 Results of various XAI algorithms + FSD for Attack Case I in post-training attack strategy*

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.698	0.709	0.738	0.759	0.782	0.804	0.832	0.862	0.901
	1	0.72	0.738	0.755	0.774	0.796	0.839	0.853	0.905	0.945
	2	0.729	0.747	0.763	0.795	0.815	0.857	0.871	0.939	0.986
CAM	0.5	0.647	0.66	0.687	0.71	0.734	0.755	0.782	0.813	0.853
	1	0.671	0.69	0.708	0.725	0.747	0.79	0.803	0.855	0.896
	2	0.678	0.699	0.713	0.746	0.766	0.807	0.823	0.89	0.947
LIME	0.5	0.579	0.592	0.619	0.642	0.666	0.687	0.714	0.745	0.785
	1	0.603	0.622	0.64	0.657	0.679	0.722	0.735	0.787	0.828
	2	0.61	0.631	0.645	0.678	0.698	0.739	0.755	0.822	0.879
FSD	0.5	0.578	0.609	0.623	0.633	0.654	0.675	0.703	0.723	0.769
	1	0.599	0.622	0.646	0.669	0.705	0.714	0.74	0.761	0.789
	2	0.617	0.627	0.669	0.693	0.724	0.727	0.771	0.791	0.842

*Table 6 Results of various XAI algorithms + FSD for Attack Case II in post-training attack strategy*

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.668	0.688	0.712	0.739	0.791	0.843	0.868	0.897	0.942
	1	0.738	0.756	0.769	0.787	0.807	0.856	0.877	0.921	0.967
	2	0.78	0.787	0.801	0.817	0.841	0.872	0.888	0.932	0.982
CAM	0.5	0.665	0.685	0.709	0.737	0.788	0.842	0.865	0.895	0.94
	1	0.735	0.755	0.766	0.783	0.805	0.855	0.875	0.918	0.965
	2	0.779	0.784	0.798	0.815	0.839	0.869	0.884	0.928	0.978
LIME	0.5	0.651	0.671	0.695	0.723	0.774	0.828	0.851	0.881	0.926
	1	0.721	0.741	0.752	0.769	0.791	0.841	0.861	0.904	0.951
	2	0.765	0.77	0.784	0.801	0.825	0.855	0.87	0.914	0.991
FSD	0.5	0.568	0.588	0.612	0.64	0.691	0.745	0.768	0.798	0.843
	1	0.638	0.658	0.669	0.686	0.708	0.758	0.778	0.821	0.868
	2	0.682	0.687	0.701	0.718	0.742	0.772	0.787	0.831	0.908

#### 4.6.2 Results of XFSD positional attacks

With the architecture settled in terms of scenario and algorithms, we are using similar visualization of aggregates to review the difference of different positional attacks.

##### 4.6.2.1 Attack case III, IV, V in pre-training

Our approach takes advantage of the sequential relationship between the data samples from different time steps. That is, an adversary could launch positional attacks to bypass the classifier. To emulate this behaviour, we consider that in attack case III all legitimate data appear before the malicious data, in case IV all legitimate samples appear after the malicious data, while in case V malicious data distributed in between legitimate samples.

Depending on the adversarial rate, the model may ignore parts of adversarial samples, e.g., with 5% of adversarial data, a model for attack case III learns 95% of legitimate data before learning from the 5% of malicious data, which has less impact on the final performance in comparison to 55% legitimate and 45% malicious samples. This impact is shown in Table 7, Table 8, and Table 9. It is evident that the higher adversarial rates cause higher negative impact on the model.

Among the different attack cases, the lowest impact is for attack case IV, in which the model learns first on malicious data. The highest negative impact appears for attack case III, while attack case V, as expected is in between the other two attack cases. Overall, based on the experimentation results across all cases, XFSD is proved to perform better, advocating the beneficial impact of the XAI over the FSD approach.

Table 7 F-measures for Attack Case III (benign first) with attack samples generated by Attack Case I in pre-training for different XAI algorithms.

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.971	0.865	0.83	0.816	0.801	0.78	0.775	0.755	0.69
	1	0.882	0.846	0.804	0.788	0.764	0.755	0.754	0.73	0.645
	2	0.847	0.817	0.78	0.76	0.745	0.737	0.725	0.694	0.611
CAM	0.5	0.901	0.856	0.815	0.806	0.792	0.756	0.74	0.722	0.701
	1	0.856	0.825	0.784	0.764	0.767	0.731	0.696	0.679	0.646
	2	0.809	0.789	0.75	0.723	0.701	0.689	0.667	0.644	0.594
LIME	0.5	0.83	0.785	0.744	0.735	0.721	0.685	0.669	0.651	0.63
	1	0.785	0.754	0.713	0.693	0.696	0.66	0.625	0.608	0.575
	2	0.738	0.718	0.679	0.652	0.63	0.618	0.596	0.573	0.523
FSD	0.5	0.842	0.736	0.701	0.687	0.672	0.651	0.646	0.626	0.561
	1	0.753	0.717	0.675	0.659	0.635	0.626	0.625	0.601	0.516
	2	0.718	0.688	0.651	0.631	0.616	0.608	0.596	0.565	0.482

Table 8 F-measures for Attack Case IV (malicious first) with attack samples generated by Attack Case I in pre-training

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.958	0.894	0.858	0.851	0.838	0.822	0.813	0.787	0.729
	1	0.904	0.858	0.824	0.814	0.797	0.777	0.786	0.756	0.672
	2	0.877	0.822	0.787	0.766	0.76	0.76	0.746	0.734	0.646
CAM	0.5	0.841	0.796	0.755	0.746	0.732	0.696	0.68	0.662	0.641
	1	0.796	0.765	0.724	0.704	0.707	0.671	0.636	0.619	0.586
	2	0.749	0.729	0.69	0.663	0.641	0.629	0.607	0.584	0.534
LIME	0.5	0.77	0.725	0.684	0.675	0.661	0.625	0.609	0.591	0.57
	1	0.725	0.694	0.653	0.633	0.636	0.6	0.565	0.548	0.515
	2	0.678	0.658	0.619	0.592	0.57	0.558	0.536	0.513	0.463
FSD	0.5	0.783	0.719	0.683	0.676	0.663	0.647	0.638	0.612	0.554
	1	0.729	0.683	0.649	0.639	0.622	0.602	0.611	0.581	0.497
	2	0.702	0.647	0.612	0.591	0.585	0.585	0.571	0.559	0.471

Table 9 F-measures for Attack Case V with attack samples generated by Attack Case I in pre-training

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.954	0.907	0.871	0.82	0.796	0.782	0.773	0.743	0.664
	1	0.93	0.855	0.818	0.795	0.765	0.746	0.739	0.706	0.615
	2	0.889	0.827	0.783	0.761	0.735	0.713	0.683	0.658	0.593
CAM	0.5	0.814	0.769	0.728	0.719	0.705	0.669	0.653	0.635	0.614
	1	0.769	0.738	0.697	0.677	0.68	0.644	0.609	0.592	0.559
	2	0.722	0.702	0.663	0.636	0.614	0.602	0.58	0.557	0.507
LIME	0.5	0.77	0.725	0.684	0.675	0.661	0.625	0.609	0.591	0.57
	1	0.725	0.694	0.653	0.633	0.636	0.6	0.565	0.548	0.515
	2	0.678	0.658	0.619	0.592	0.57	0.558	0.536	0.513	0.463
FSD	0.5	0.782	0.735	0.699	0.648	0.624	0.610	0.601	0.571	0.492
	1	0.758	0.683	0.646	0.623	0.593	0.574	0.567	0.534	0.443
	2	0.717	0.655	0.611	0.589	0.563	0.541	0.511	0.486	0.421

#### 4.6.2.2 Attack case III, IV, V in Post-training

The results are reflected in Table 10, Table 11 and Table 12. As before, the comparative tendencies reveal that XAI algorithms do not differ much depending on the order or the distribution of malicious samples. SHAP is the most accurate variant showcasing the best performance especially for the attack case V.

*Table 10 F-measures for Attack Case III (benign first) with attack samples generated by Attack Case I in post-training*

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.698	0.714	0.75	0.78	0.814	0.833	0.88	0.899	0.931
	1	0.731	0.754	0.78	0.812	0.858	0.876	0.909	0.932	0.965
	2	0.777	0.78	0.801	0.833	0.873	0.894	0.931	0.95	0.981
CAM	0.5	0.656	0.669	0.696	0.719	0.743	0.764	0.791	0.822	0.862
	1	0.68	0.699	0.717	0.734	0.756	0.799	0.812	0.864	0.905
	2	0.687	0.708	0.722	0.755	0.775	0.816	0.832	0.899	0.956
LIME	0.5	0.585	0.598	0.625	0.648	0.672	0.693	0.72	0.751	0.791
	1	0.609	0.628	0.646	0.663	0.685	0.728	0.741	0.793	0.834
	2	0.616	0.637	0.651	0.684	0.704	0.745	0.761	0.828	0.885
FSD	0.5	0.599	0.615	0.651	0.681	0.715	0.734	0.781	0.800	0.832
	1	0.632	0.655	0.681	0.713	0.759	0.777	0.810	0.833	0.866
	2	0.678	0.681	0.702	0.734	0.774	0.795	0.832	0.851	0.903

*Table 11 F-measures for Attack Case IV (malicious first) with samples generated by Attack Case I in post-training*

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.617	0.666	0.718	0.739	0.767	0.796	0.826	0.873	0.915
	1	0.664	0.727	0.731	0.782	0.835	0.848	0.897	0.899	0.93
	2	0.727	0.737	0.778	0.809	0.852	0.871	0.913	0.929	0.981
CAM	0.5	0.596	0.609	0.636	0.659	0.683	0.704	0.731	0.762	0.802
	1	0.62	0.639	0.657	0.674	0.696	0.739	0.752	0.804	0.845
	2	0.627	0.648	0.662	0.695	0.715	0.756	0.772	0.839	0.896
LIME	0.5	0.525	0.538	0.565	0.588	0.612	0.633	0.66	0.691	0.731
	1	0.549	0.568	0.586	0.603	0.625	0.668	0.681	0.733	0.774
	2	0.556	0.577	0.591	0.624	0.644	0.685	0.701	0.768	0.825
FSD	0.5	0.487	0.536	0.588	0.609	0.637	0.666	0.696	0.743	0.785
	1	0.534	0.597	0.601	0.652	0.705	0.718	0.767	0.769	0.800
	2	0.597	0.607	0.648	0.679	0.722	0.741	0.783	0.799	0.851

*Table 12 F-measures for Attack Case V post-training attack strategy with attack samples generated by Attack Case I*

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.699	0.731	0.778	0.803	0.849	0.869	0.9	0.923	0.98
	1	0.712	0.751	0.796	0.824	0.868	0.891	0.934	0.958	0.981
	2	0.726	0.781	0.8	0.852	0.891	0.91	0.959	0.979	0.989
CAM	0.5	0.616	0.629	0.656	0.679	0.703	0.724	0.751	0.782	0.822
	1	0.64	0.659	0.677	0.694	0.716	0.759	0.772	0.824	0.865
	2	0.647	0.668	0.682	0.715	0.735	0.776	0.792	0.859	0.916
LIME	0.5	0.535	0.548	0.575	0.598	0.622	0.643	0.67	0.701	0.741
	1	0.559	0.578	0.596	0.613	0.635	0.678	0.691	0.743	0.784
	2	0.566	0.587	0.601	0.634	0.654	0.695	0.711	0.778	0.835
FSD	0.5	0.579	0.611	0.658	0.683	0.729	0.749	0.780	0.803	0.860
	1	0.592	0.631	0.676	0.704	0.748	0.771	0.814	0.838	0.878
	2	0.606	0.661	0.680	0.732	0.771	0.790	0.839	0.859	0.901

### 4.6.3 Advanced data poisoning

Taking inspiration from Miao et al. [REF-46] we added more advanced adversarial techniques based on their optimal attack framework which uses a bilevel optimization. The two types of data poisoning attacks i.e., the availability attack (case VI) and the target attack (case VII) were applied to evaluate XFSD effectiveness. The outcomes which are mainly on the pre-training strategy (data poisoning) showcase that the optimization and these types of attacks are highly efficient, even with the addition of XAI values, rendering the system beneath random guessing during some adversarial configurations. The order in which the results are presented in Table 13, Table 14, Table 15 and Table 16 are 1) SHAP, 2) CAM, 3) LIME, and 4) FSD.

The outcome for this experimentation using optimal standard deviation and different adversarial rates based on the newly introduced optimized strategy is a great opportunity to gain more insight into the robustness of our model. Our model is still performing well within the area, but for these challenging attack cases it dips beneath random guessing in certain configurations. It is evident again that XAI gives an advantage to the XFSD approach, while SHAP remains the most prominent XAI method. It is noted that the advanced attack strategies with the optimization is more effective in skewing the perception of the classifiers.

*Table 13 Optimized Data poisoning availability attack case VI for the scenario of  $\mu$*

$\lambda$	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Optimal	0.701	0.681	0.642	0.615	0.593	0.581	0.559	0.536	0.486
Optimal	0.686	0.666	0.627	0.6	0.578	0.566	0.544	0.521	0.471
Optimal	0.674	0.654	0.615	0.588	0.566	0.554	0.532	0.509	0.459
Optimal	0.661	0.641	0.602	0.575	0.553	0.541	0.519	0.496	0.446

*Table 14 Optimized Data poisoning availability attack case VI for the scenario with  $\sigma$*

$\lambda$	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Optimal	0.687	0.677	0.674	0.667	0.63	0.616	0.574	0.525	0.477
Optimal	0.674	0.664	0.661	0.654	0.617	0.603	0.561	0.512	0.464
Optimal	0.662	0.652	0.649	0.642	0.605	0.591	0.549	0.5	0.452
Optimal	0.647	0.637	0.634	0.627	0.59	0.576	0.534	0.485	0.437

*Table 15 Optimized Data poisoning target attack case VII for the scenario with  $\mu$*

$\lambda$	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Optimal	0.788	0.757	0.716	0.696	0.699	0.663	0.628	0.611	0.578
Optimal	0.638	0.607	0.566	0.546	0.549	0.513	0.478	0.461	0.428
Optimal	0.625	0.594	0.553	0.533	0.536	0.5	0.465	0.448	0.415
Optimal	0.613	0.582	0.541	0.521	0.524	0.488	0.453	0.436	0.403

*Table 16 Optimized Data poisoning target attack case VII for the scenario of  $\sigma$*

$\lambda$	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Optimal	0.717	0.707	0.704	0.697	0.66	0.646	0.604	0.555	0.507
Optimal	0.702	0.692	0.689	0.682	0.645	0.631	0.589	0.54	0.492
Optimal	0.677	0.667	0.664	0.657	0.62	0.606	0.564	0.515	0.467
Optimal	0.662	0.652	0.649	0.642	0.605	0.591	0.549	0.5	0.452

#### 4.6.4 Aggregation Method

In an effort to reduce the dimensionality of the problem and to explore whether a different aggregation approach of the XAI values may have a positive impact to the model, we extended attack case I, as reported in Table 17. We proceeded to an aggregation before the concatenation of the fixed length vectors from the time window. However, as it can be seen in Table 17, no improvement is reported, implying that a different aggregation approach cannot be considered a prominent method for improving the model.

*Table 17 F-measures for Attack Case I pre-training attack strategy where the default aggregation is applied before concatenation of fixed length vectors.*

	$\lambda$	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.865	0.813	0.825	0.817	0.799	0.765	0.749	0.732	0.71
	1	0.836	0.766	0.793	0.773	0.779	0.74	0.707	0.688	0.655
	2	0.798	0.721	0.76	0.733	0.712	0.703	0.676	0.653	0.603
CAM	0.5	0.77	0.725	0.684	0.675	0.661	0.625	0.609	0.591	0.57
	1	0.725	0.694	0.653	0.633	0.636	0.6	0.565	0.548	0.515
	2	0.678	0.658	0.619	0.592	0.57	0.558	0.536	0.513	0.463
LIME	0.5	0.734	0.689	0.648	0.639	0.625	0.589	0.573	0.555	0.534
	1	0.689	0.658	0.617	0.597	0.6	0.564	0.529	0.512	0.479
	2	0.642	0.622	0.583	0.556	0.534	0.522	0.5	0.477	0.427
FSD	0.5	0.724	0.681	0.64	0.631	0.617	0.579	0.566	0.547	0.526
	1	0.681	0.649	0.609	0.587	0.591	0.556	0.521	0.503	0.471
	2	0.633	0.613	0.575	0.548	0.525	0.513	0.492	0.467	0.417

## 4.7 Summary

The purpose of XFSD was to improve the existing FSD framework. The landscape of MCS for IoT present a plethora of research challenges with one of the more prominent ones being to address data trustworthiness. With FSD we took a first step, while with XFSD we attempt to find a way for improving the robustness of the models that can be derived from the framework by leveraging advanced deep learning capabilities with explainability qualities. The XAI extension provided a notable improvement, making classifiers able to distinguish between legit and malicious samples. In our work we showcased that, for time series data, the use of LSTM/RNN in combination with SHAP was more accurate than applying it with CAM. This can be explained by the way CAM works in combination with CNN for time-series i.e., CAM considers areas of importance, whereas SHAP considers the whole ordinal information sequence. This is observed to be advantageous in dataset such as ours in which one has to consider the issues pertaining to concept drift, i.e., unforeseen changes over time. The information that can be extracted from sequential order of data is proven important considering the improvement in the accuracy of the classifiers when introducing XAI values and the better performance when compared to non-sequential algorithms, such as LIME which failed to contribute in the XFSD setup. As future work, we aim to explore more robust approaches in XFSD that could tackle the challenging availability and targeted adversarial attacks.

## 5 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 provided the final version of the AI Cyber Defence tool, which 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 builds on top of D3.3 and extends the experimental testbed by conducting experiments using new datasets stemming from the pilot environments. More specifically, we took advantage of the *Soother* and *Assembly of Horizontal Lamellas* datasets, provide by Philips and IBER partners respectively. Apart from the description of the AI Cyber Defence Tool and the technologies enablers which are combined for the delivery of the tool, a core contribution of the deliverable is the conduction of several experiments on the applicability of diverse AI attack and defence strategies. Several evaluation results acquired and proved the challenging nature of the problem for which the STAR project aims to make a contribution. As was the outcome of the 1<sup>st</sup> experimental period in D3.3, the results of D3.4 also advocate that there is no single solution, “panacea”, to deal with advanced adversarial techniques, as the robustness of a defence 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.

In addition, our research efforts led to the design of a prototype, called XFSD, as reported in section 4. This section provided a summary of the methodology that constitutes the long-term research vision of UBITECH in the domain. Section 4 highlighted the conceptual architecture of XFSD, which extends the one presented previously in D3.3 and in [REF-01]. XFSD takes advantage of XAI method in order to enhance the trustworthiness of sensory data against intelligent AI attackers, addressing in parallel the requirement of exploiting XAI method for the detection of poisoning and evasion attacks. More results can be found in [REF-43].

This deliverable completes the actions of T3.2 and T3.3 with the final release of the AI Cyber Defence tool, along with the extensive experimental testbed that was built in order to evaluate a wide spectrum of AI attacks and Defences using data stemming from the STAR manufacturing floors.

## References

Reference	Name of document
[REF-01]	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-02]	I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.
[REF-03]	W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," arXiv preprint arXiv:1704.01155, 2017.
[REF-04]	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-05]	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-06]	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-07]	Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2574-2582).
[REF-08]	Uyeong Jang, Xi Wu, and Somesh Jha. 2017. Objective Metrics and Gradient Descent Algorithms for Adversarial Examples in Machine Learning. In Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC 2017). Association for Computing Machinery, New York, NY, USA, 262–277. DOI:https://doi.org/10.1145/3134600.3134635
[REF-09]	Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.
[REF-10]	Kurakin, A., Goodfellow, I. J., & Bengio, S. (2018). Adversarial examples in the physical world. In Artificial intelligence safety and security (pp. 99-112). Chapman and Hall/CRC.
[REF-11]	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-12]	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-13]	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-14]	T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Diaz Rodriguez, "Explainable artificial intelligence (XAI) on timeseries data: A survey," 04 2021
[REF-15]	S. Gisdakis, T. Giannetsos, and P. Papadimitratos, "Shield: a data verification framework for participatory sensing systems," 07 2015.
[REF-16]	N. Banerjee, T. Giannetsos, E. Panaousis, and C. Cheong Took, "Unsupervised learning for trustworthy iot," 07 2018, pp. 1–8.
[REF-17]	E. Veliou, D. Papamartzivanos, S. A. Menesidou, P. Gouvas, T. Giannetsos et al., "Artificial intelligence and secure manufacturing: Filling gaps in making industrial environments safer," Trusted Artificial Intelligence in Manufacturing, p. 30, 2021.

[REF-18]	J. M. Rozanec, D. Papamartzivanos, E. Veliou, T. Anastasiou, J. Keizer, B. Fortuna, and D. Mladenic, "Machine beats machine: Machine learning models to defend against adversarial attacks," 2022.
[REF-19]	T. Anastasiou, S. Karagiorgou, P. Petrou, D. Papamartzivanos, T. Giannetsos, G. Tsirigotaki, and J. Keizer, "Towards robustifying image classifiers against the perils of adversarial attacks on artificial intelligence systems," <i>Sensors</i> , vol. 22, no. 18, 2022.
[REF-20]	N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," <i>Computer Science Review</i> , vol. 34, p. 100199, Nov. 2019.
[REF-21]	S. Afzal-Houshmand, S. Homayoun, and T. Giannetsos, "A perfect match: Deep learning towards enhanced data trustworthiness in crowdsensing systems," in 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), 2021, pp. 258–264
[REF-22]	B. Guo, Z. Yu, X. Zhou, and D. Zhang, "From participatory sensing to mobile crowd sensing," in 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS), 2014, pp. 593–598.
[REF-23]	X. Li, K. Xie, X. Wang, G. Xie, D. Xie, Z. Li, J. Wen, Z. Diao, and T. Wang, "Quick and accurate false data detection in mobile crowd sensing," <i>IEEE/ACM Transactions on Networking</i> , vol. 28, no. 3, pp. 1339–1352, 2020
[REF-24]	L. Cheng, L. Kong, C. Luo, J. Niu, Y. Gu, W. He, and S. Das, "Deco: False data detection and correction framework for participatory sensing," in 2015 IEEE 23rd International Symposium on Quality of Service (IWQoS). IEEE, Jun. 2015.
[REF-25]	J.-Y. Kim and S.-B. Cho, "Electric energy consumption prediction by deep learning with state explainable autoencoder," <i>Energies</i> , vol. 12, p. 739, 02 2019.
[REF-26]	K. E. Mokhtari, B. P. Higdon, and A. Basar, "Interpreting financial time series with shap values," in Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering, ser. CASCON '19. USA: IBM Corp., 2019, p. 166–172
[REF-27]	P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," <i>Entropy</i> , vol. 23, p. 18, 12 2020
[REF-28]	D. Dataman. Explain your model with the shap values. [Online]. Available: <a href="https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d">https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d</a>
[REF-29]	B. Khaleghi. The how of explainable ai: Post-modelling explainability. . [Online]. Available: <a href="https://towardsdatascience.com/the-how-of-explainable-ai-post-modelling-explainability-8b4cbc7adf5f">https://towardsdatascience.com/the-how-of-explainable-ai-post-modelling-explainability-8b4cbc7adf5f</a>
[REF-30]	Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 1578–1585
[REF-31]	H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Accurate and interpretable evaluation of surgical skills from kinematic data using fully convolutional neural networks," 08 2019

[REF-32]	F. Oviedo, Z. Ren, S. Sun, C. M. Settens, Z. Liu, N. T. P. Hartono, R. Savitha, B. L. DeCost, S. I. P. Tian, G. Romano, A. G. Kusne, and T. Buonassisi, "Fast classification of small x-ray diffraction datasets using data augmentation and deep neural networks," ArXiv, vol. abs/1811.08425, 2018
[REF-33]	K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "ConvtimeNet: A pre-trained deep convolutional neural network for time series classification," 07 2019, pp. 1–8
[REF-34]	L. Zhou, C. Ma, X. Shi, D. Zhang, W. Li, and L. Wu, "Saliency-cam: Visual explanations from convolutional neural networks via saliency score," 07 2021, pp. 1–8
[REF-35]	M. T. Ribeiro, S. Singh, and C. Guestrin, "'why should i trust you?': Explaining the predictions of any classifier," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1135–1144. [Online]. Available: <a href="https://doi.org/10.1145/2939672.2939778">https://doi.org/10.1145/2939672.2939778</a>
[REF-36]	L. Hulstaert. Understanding model predictions with lime. [Online]. Available: <a href="https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b">https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b</a>
[REF-37]	A. Sharma. Decrypting your machine learning model using lime. [Online]. Available: <a href="https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5">https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5</a>
[REF-38]	R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and using lime. [Online]. Available: <a href="https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5">https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5</a>
[REF-39]	D. Mercier, A. Dengel, and S. Ahmed, "Patchx: Explaining deep models by intelligible pattern patches for time-series classification," 02 2021
[REF-40]	Chen, Pin-Yu, et al. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models." Proceedings of the 10th ACM workshop on artificial intelligence and security. 2017.
[REF-41]	Chen, J., Jordan, M. I., & Wainwright, M. J. (2020, May). Hopskipjumpattack: A query-efficient decision-based attack. In 2020 IEEE Symposium on Security and Privacy (SP) (pp. 1277-1294). IEEE.
[REF-42]	Brendel, W., Rauber, J., & Bethge, M. (2017). Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. arXiv preprint arXiv:1712.04248.
[REF-43]	Sam Afzal-Houshmand, Dimitrios Papamartzivanos, Sajad Homayoun, Entso Veliou, Christian D. Jensen, Athanasios (Thanos) Voulodimos, Thanassis Giannetsos - "Explainable Artificial Intelligence to Enhance Data Trustworthiness in Crowd-Sensing Systems", 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), DOI 10.1109/DCOSS-IoT58021.2023.00093 (To appear online)
[REF-44]	Joze M. Rozanec and Elias Montini and Vincenzo Cutrona and Dimitrios Papamartzivanos and Timotej Klemencic and Blaz Fortuna and Dunja Mladenec and Entso Veliou and Thanassis Giannetsos and Christos Emmanouilidis, "Human in the AI loop via xAI and Active Learning for Visual Inspection", Book Chapter, (To Appear)

[REF-45]	D. S. Lab, "strata santa clara dataset." [Online]. Available: <a href="http://datasensinglab.com/data/">http://datasensinglab.com/data/</a>
[REF-46]	C. Miao, Q. Li, H. Xiao, W. Jiang, M. Huai, and L. Su, "Towards data poisoning attacks in crowd sensing systems," 06 2018, pp. 111–120