

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

D4.2 – Library of XAI Algorithms – Final version

Dissemination level	PU - Public
Type of Document	Demonstrator, pilot, prototype, plan designs
Contractual date of delivery	31/12/2022
Deliverable Leader	UPRC
Status - version, date	Final – v1.0, 19/01/2023
WP / Task responsible	WP4
Keywords:	Library of Explainable AI Algorithms

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

The purpose of this document is to describe the progress and the outcomes of task T4.1 and cover the time from month 12 to month 24 of the project. T4.1 applied and validated various explainable AI techniques in manufacturing environments and applications, focusing on methods particularly suited for manufacturing use cases, such as XAI techniques for image classification (i.e., visual inspection) and time-series classification (i.e., sensor data). The outcomes include, among other techniques that, identify the dominant features used by AI systems and Prediction Difference Analysis to explain and interpret the project's quality management and agile production use cases.

The task focused on Image and Timeseries data as these types seem prevalent in industrial processes. Along with implementing and validating these techniques, the task also integrated them with other project components based on the overall architecture, including AI security techniques for protecting against data poisoning attacks.

The main activities carried out during T4.1 between M3, and M12 were: 1) a comprehensive state-of-the-art analysis; 2) the development of XAI models for Image classification; 3) an analysis of UC requirements regarding AI models explainability. These activities were documented in D4.1.

From M12 to M24, the task focused on proving the utility of Image Classification on Philips UC regarding the visual inspection. Then the focus was paid to time-series explainability in a way to be comprehensive by non-expert end human users. Finally, there was effort paid to exploit XAI knowledge against adversarial attacks.

The prototype library has been provided and is currently available at [GitHub](https://github.com/GeorgeMakridis/STAR_XAI_LIBRARY), as show in Fig.1.

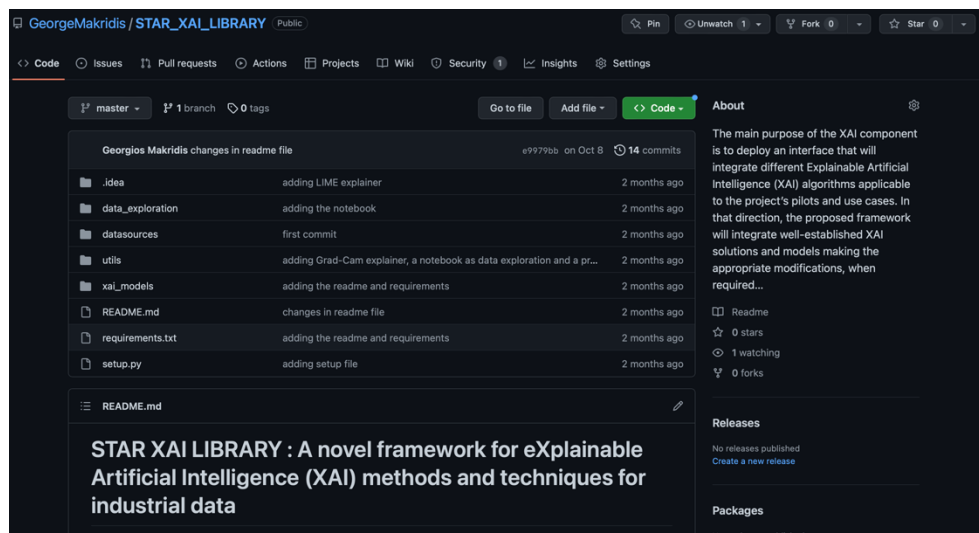


Figure 1 STAR XAI library on GitHub

Deliverable Leader:	UPRC
Contributors:	UPRC, QLE, DFKI
Reviewers:	GFT, THA
Approved by:	Charalampos Ipektsidis, John Soldatos (INTRA)

Document History			
Version	Date	Contributor(s)	Description
0.1	20/12/2022	UPRC	Table of Contents.
0.2	30/12/2022	UPRC	Results 12M-24M - XAI Algorithms – 1 st Draft
0.3	13/01/2023	UPRC	1 st Revision - 2 nd Draft
0.4	16/01/2023	UPRC	Consolidated draft
1.0	19/01/2023	INTRA	QA and creation of the submitted version

Table of Contents

1	INTRODUCTION	7
1.1	DELIVERABLE SUMMARY	8
1.2	DOCUMENT SCOPE	8
1.3	DOCUMENT STRUCTURE	10
2	XAI FOR IMAGE DATA	11
3	XAI FOR TIMESERIES DATA	16
3.1	LAGS IMPORTANCE.....	16
3.2	TIME-SERIES TO IMAGE TRANSFORMATION	18
3.3	TIME-SERIES TO TEXT TRANSFORMATION.....	21
4	XAI FOR TEXT AND TABULAR DATA	22
4.1	XAI FOR TABULAR DATA.....	22
4.2	XAI FOR TEXT DATA (NLP)	22
5	XAI ENHANCING CYBER DEFENCE AGAINST ADVERSARIAL ATTACKS IN VISUAL INSPECTION APPLICATION	23
5.1	INVESTIGATION WITH LIME	24
5.2	INVESTIGATION WITH GRAD-CAM.....	26
5.3	INVESTIGATION WITH SALIENCY MAP	26
6	CONCLUSION	29
	REFERENCES	30

Table of Figures

FIGURE 1 STAR XAI LIBRARY ON GITHUB 2

FIGURE 2 GRAD-CAM OF 3 IMAGE EXAMPLES, THE FIRST IMAGE IS A GOOD PRINT, THE SECOND THE INTERRUPTED ONE, AND THE THIRD A DOUBLE PRINT ONE.....12

FIGURE 3 GRADCAM HEATMAP OF 2 EXAMPLES IN TWO LAYERS OF THE CNN MODEL13

FIGURE 4 EXAMPLE OF LIME IMPORTANT SUPERPIXELS14

FIGURE 5 EXAMPLE OF SALIENCY MAPS15

FIGURE 6 IMPORTANCE OF LAGS OF A TIMESERIES, FOR CLASSIFICATION CALCULATED WITH SHAP17

FIGURE 7 IMPORTANCE OF LAGS OF A TIMESERIES, FOR CLASSIFICATION CALCULATED WITH LIME17

FIGURE 8 EXAMPLE OF LIME IMPORTANT SUPERPIXELS WHERE WE CAN NOTICE THE RED AREAS THAT SEEM NOT RELEVANT TO THE PREDICTED CLASS18

FIGURE 9 EXAMPLE OF GRADCAM HEATMAP AND THE OVERLAP OF HEATMAP TO THE ORIGINAL IMAGE WHERE WE CAN NOTICE THE AREAS OF THE IMAGE THAT THE MODEL FOCUSED19

FIGURE 10 SAMPLES OF GAF TRANSFORMED TIME-SERIES20

FIGURE 11 EXAMPLE OF LIME IMPORTANT SUPERPIXELS WHERE WE CAN NOTICE THE RED AREAS THAT SEEM NOT RELEVANT TO THE PREDICTED CLASS20

FIGURE 12 EXAMPLE OF GRADCAM HEATMAP AND THE OVERLAP OF HEATMAP TO THE ORIGINAL GAF IMAGE WHERE WE CAN NOTICE THE AREAS OF THE IMAGE THAT THE MODEL FOCUSED20

FIGURE 13 EXAMPLE OF IMPORTANT "WORDS" WHERE WE CAN NOTICE THE RED AREAS THAT SEEM NOT RELEVANT TO THE PREDICTED CLASS21

FIGURE 14 EXAMPLE OF LIME PERTURBED IMAGES AGAINST ADVERSARIAL ATTACKS.....25

FIGURE 15 HOW LIME CAN BE USED IN THE PREDICTION PHASE TO MAKE A MODEL MORE ROBUST AGAINST ADVERSARIAL ATTACKS.25

FIGURE 16 BAR PLOT SHOWING THE LOSS OF THE CNN IMAGE CLASSIFIER AGAINST ADVERSARIAL ATTACKS OF VARIOUS LEVELS (X-AXIS) ON DIFFERENT TYPES OF INPUT DATA.....27

FIGURE 17 BAR PLOT SHOWING THE ACCURACY OF THE CNN IMAGE CLASSIFIER AGAINST ADVERSARIAL ATTACKS OF VARIOUS LEVELS (X-AXIS) ON DIFFERENT TYPES OF INPUT DATA27

List of Tables

TABLE 1 MARKET RESEARCH (BY OMNIXAI) REGARDING THE OFF-THE-SHELF XAI FRAMEWORKS AND THE UNDERLYING DATA TYPES THAT THEY COVER..... 9

TABLE 2 CONFUSION MATRIX FOR RESULTS OF THE CUSTOM CNN MODEL ON ORIGINAL DATA.....23

TABLE 3 RESULTS OF LOSS AND ACCURACY IN VARIOUS XAI ENHANCED DATASET24

TABLE 4 CONFUSION MATRIX FOR RESULTS OF THE CUSTOM CNN MODEL ON GRADCAM ENHANCED DATA26

TABLE 5 CONFUSION MATRIX FOR RESULTS OF THE CUSTOM CNN MODEL ON SALIENCY MAPS ENHANCED DATA26

Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
GAF	Gramian Angular Field
GRAD-CAM	Gradient-weighted Class Activation Mapping
IT	Information Technologies
LIME	Local Interpretable Model-Agnostic Explanations
ML	Machine Learning
NN	Neural Network
PAA	Piece-wise Aggregate Approximation
SAX	Symbolic Aggregate ApproXimation
SHAP	SHapley Additive exPlanations
UC	Use Case
WP	Work Package
XAI	eXplainable Artificial Intelligence

1 Introduction

The current day and age, also known as the Digital or Information Age, is characterized by complex computing systems which generate enormous amounts of data daily. The digital transformation of industrial environments leads to the fourth industrial revolution -Industry4.0 [Makridis20], with Artificial Intelligence (AI) being the key facilitator of the Industry4.0 era by enabling innovative tools and processes, including predictive quality management (Quality4.0) [Bai18]. Despite that, the Industrial sector’s state-of-the-art AI development is still far from utilizing the most sophisticated machine and Deep Learning (DL) competencies [Makridis20]. Numerous challenges must be addressed to implement AI in real-world tasks successfully. Protecting against poisoning attacks and ensuring the explainability and interpretability of AI algorithms are essential for the widespread adoption of these methods in production environments.

Explainable AI (XAI) research is focused on creating techniques that allow for more transparent and understandable artificial intelligence (AI) models while still maintaining high-performance levels in tasks such as searching, learning, planning, and reasoning. XAI aims to enable human users to understand, trust, and effectively manage AI systems. In recent years, many different approaches to XAI have been proposed, including frameworks that focus on defining what makes an AI model explainable, developing methods for understanding and explaining the behavior of AI models, and evaluating the performance of models in terms of their explainability. These approaches can be applied to various AI systems in different sectors, including machine learning, robotics, and deep learning. They can be used in finance, healthcare, and manufacturing industries.

Task 4.1 of the STAR project is focused on exploring and demonstrating the potential of machine learning and artificial intelligence technologies in industrial and manufacturing settings. The intended audience for the project is mostly domain experts from the fields of industry and production lines. In these demanding industrial environments, there is a need for computing systems to analyze large amounts of data quickly. While human beings can make reliable decisions based on specific sets of information, they have limitations in terms of the speed and efficiency with which they can process that information quickly. As a result, domain experts rely on computers or machines to perform real-time analysis.

The frameworks and tools incorporated for XAI library are the following:

Framework	Version	Usage
pandas	1.1.5	Data analysis
numpy	1.19.5	Mathematical functions
scikit-learn	0.24.1	Predictive data analysis
scipy	1.5.4	Mathematical functions
reliability	0.8.3	Reliability engineering and survival analysis
tensorflow	2.7.1	Engineering and Deep Learning Models
Flask	1.1.2	REST applications
unicorn	20.0.4	Web server portal interface for deployment

ELI5		XAI framework
shap		XAI framework
lime		XAI framework
Matplotlib, plotly		Visualizations and plots
docker		
Python	3.6	Programming language

1.1 Deliverable Summary

This deliverable presents the research and the outcomes of XAI models for manufacturing processes. Furthermore, it describes how the XAI can contribute to STAR's cyber defense (WP3) against adversarial attacks. To this end, T4.1 provides an eXplainable AI (XAI) library where a user can use a set of techniques that help develop more explainable models while at the same time preserving their high-performing learning, planning, search, and reasoning functionalities in real-world manufacturing environments and applications. The ultimate end of XAI is to increase human users' understanding and trust in AI systems so that they can harmoniously coexist in the emerging era of ubiquitous AI. Within this library, some models are generic, model-agnostic XAI techniques such as SHAP and LIME (to boost reusability and adaptability), and class activation mapping techniques for deep neural networks. We also envision highlighting the types of data that can be "explained" by the T4.1 and defining "how" XAI can enhance image datasets against data poisoning attacks.

In terms of implementation, we envision the XAI component as a library of algorithms that supports different data modalities and AI algorithms and their technical implementations as they are developed in support of the project's use cases. The library will be usable and accessible by various STAR components across multiple work packages and will hold a central role in the project's architecture. More details on the deployment or migration is presented in Section 6.

1.2 Document Scope

The purpose of this document is to describe the progress and the outcomes of task T4.1 and cover the time from month 12 to month 24 of the project. T4.1 applied and validated various explainable AI techniques in manufacturing environments and applications, focusing on techniques particularly suited for manufacturing use cases, such as XAI techniques for image classification (i.e., visual inspection) and time-series classification to support time-dependent pilot datasets such as the sensor data. The outcomes include, among other methods that, identify the dominant features used by AI systems and Prediction Difference Analysis to explain and interpret the project's quality management and agile production use cases.

This document include the XAI models, techniques, and corresponding meta-XAI models that have been under study. The main purpose of the XAI component is to deploy an interface that will integrate different XAI algorithms applicable to the project's pilots and use cases. In that direction, the proposed framework will integrate well-established XAI solutions and

models making the appropriate modifications, when required, adding value to STAR-specific pilots. Furthermore, part of the library will be used by the cyber-defence component.

During the first months of the project within the T4.1, the aim was to collect the state-of-the-art XAI models and identify the most suitable to STAR use-cases. It should be mentioned that the suitability of an XAI algorithm is dependent on the type of provided data from the manufacturing partners and the related AI-enabled components of STAR leveraged by each pilot. Each XAI algorithm can be used with specific datasets (i.e. tabular, text data, images, or time-series) and coupled with different ML/DL models for more accurate performance.

Thus, we study and develop specialized XAI algorithms for, tabular and text type of underlying data.

Bellow, a table taken from the omnixai website, presents the main tools and solutions they offer, and to which type of data they correspond. As we can notice, there is a gap in the time-series data as omnixai only offer two models, which is relatively fewer than the number of model offered for other types of data. This market research also defines and proves the scientific contribution Task 4.1 offers regarding the research of explainability of time-series ML/DL models.

Table 1 Market research (by Omnixai) regarding the off-the-shelf XAI frameworks and the underlying data types that they cover

Data Type	Method	OmnixAI	InterpretML	AIX360	Eli5	Captum	Alibi	explainX
Tabular	LIME	✓	✓	✓		✓		
	SHAP	✓	✓	✓		✓	✓	✓
	Partial dependence plot	✓	✓					
	Sensitivity analysis	✓	✓					
	Integrated Gradients	✓				✓	✓	
	Counterfactual	✓					✓	
	L2X	✓						
	Linear models	✓	✓	✓	✓		✓	✓
Image	Tree models	✓	✓	✓	✓		✓	✓
	LIME	✓				✓		
	SHAP	✓				✓		
	Integrated Gradients	✓				✓	✓	
	Grad-CAM	✓			✓	✓		
	Contrastive explanation	✓		✓			✓	
	Counterfactual	✓					✓	
	L2X	✓						
Text	LIME	✓			✓	✓		
	SHAP	✓				✓		
	Integrated Gradients	✓				✓	✓	
	Counterfactual	✓						
	L2X	✓						
Timeseries	SHAP	✓						
	Counterfactual	✓						

1.3 Document Structure

This document is structured in 7 sections. Section 1 provides a summary of the deliverable together with the description and scope of T4.1.

Section 2 provides the work done and the result regarding Image data types.

Section 3 describes the outcomes regarding the implementation of XAI Library on time-series data.

Section 4 follows the implementation of XAI techniques for tabular and text data.

Section 5 presents the research and implementation of XAI models to enhance an image dataset against adversarial attacks.

Finally, Section 6 includes the conclusions and lessons learned from T4.1.

2 XAI for Image data

XAI can be applied in industrial processes through visual inspection systems. Visual inspection systems use cameras or other sensors to capture images of products or processes and then use machine-learning algorithms to analyze the photos and identify defects or abnormalities. However, these machine-learning algorithms can sometimes make mistakes or produce results that are difficult for humans to understand. XAI techniques can be used to create more transparent and interpretable visual inspection systems. For example, an XAI system might use techniques such as feature importance analysis or decision tree visualization to help explain how the machine learning algorithms arrived at their conclusions. This can help the operators and engineers understand the reasoning behind the AI system's decisions and identify potential problems.

We mainly focused on using XAI to improve the automated quality inspection process for Philips UC. The goal is to reduce the cost and time of manual inspection and allow workers to focus on more meaningful and less repetitive tasks. To make this possible, we will use XAI to bridge the gap between AI and human operators. XAI will explain the predictions made by convolutional neural networks (CNNs) that analyze image data to identify defects in manufactured parts. The use of XAI is intended to increase confidence in AI decisions and assist workers in handling challenging inspection tasks that require human intervention.

The main concept of this component is to provide explanations in terms of attribution scores. Given a trained model, such as a neural network, the component will output the importance of each input feature for a particular prediction. When dealing with image data, feature importance can be translated into the importance of each pixel to the output forecast. The latter can also be visualized into a heatmap where the importance of each feature (pixel) can be displayed with different colours and can work as an excellent explanation for the human operator. In the context of the UC, we envisioned using XAI algorithms to hint to the user where the model believes a defect can be located and thus speed up the defect detection in a manual revision context. We set up multiple experiments with a variety of off-the-shelf XAI libraries that have or can be leveraged for the implementation of this component or can alternatively provide baselines for comparisons.

Some notable ones are the following:

- **lime [Lime]:** Accessible through pip, the library implements the Local Interpretable Model-agnostic Explanations family of methods for multiple modalities such as vectors, images and text. Receives the `model.predict()` method as input and is not dependent on underlying implementations. No GPU is needed unless the model is really complex, and many instances need to be interpreted.
- **shap [Shap]:** Implements Shapley additive explanations and is from a usage viewpoint similar to lime without being tied to a specific model and ML/DL framework. Also extends to images by using random masking. It should be mentioned that this solution was not applied on Image data rather it served as baseline for explainability of the timeseries. SHAP is designed to work with tabular data and models that output scalar values. It may not be well-suited for explaining the predictions of image classification models, which typically output a probability distribution over multiple

classes. Additionally, SHAP values for image data may be difficult to interpret, as the model's decisions are based on the presence or absence of patterns in the image, rather than on specific features of the input data.

- **Grad-CAM [GradCAM]:** A family of methods specific to CNNs. Many variants have been implemented in the pip grad-cam package, currently implemented in PyTorch and also supports GPU usage. Note that these methods most often require some access to the actual model (e.g., weights, architecture) to perform gradient calculations and do not treat the model as a black box.

The abovementioned algorithms that are integrated into the component are mainly developed on Python Libraries and can easily be applied to the STAR project's use cases. Once again LIME is one of the handiest techniques (with interpretable visualizations) for image type of input data.

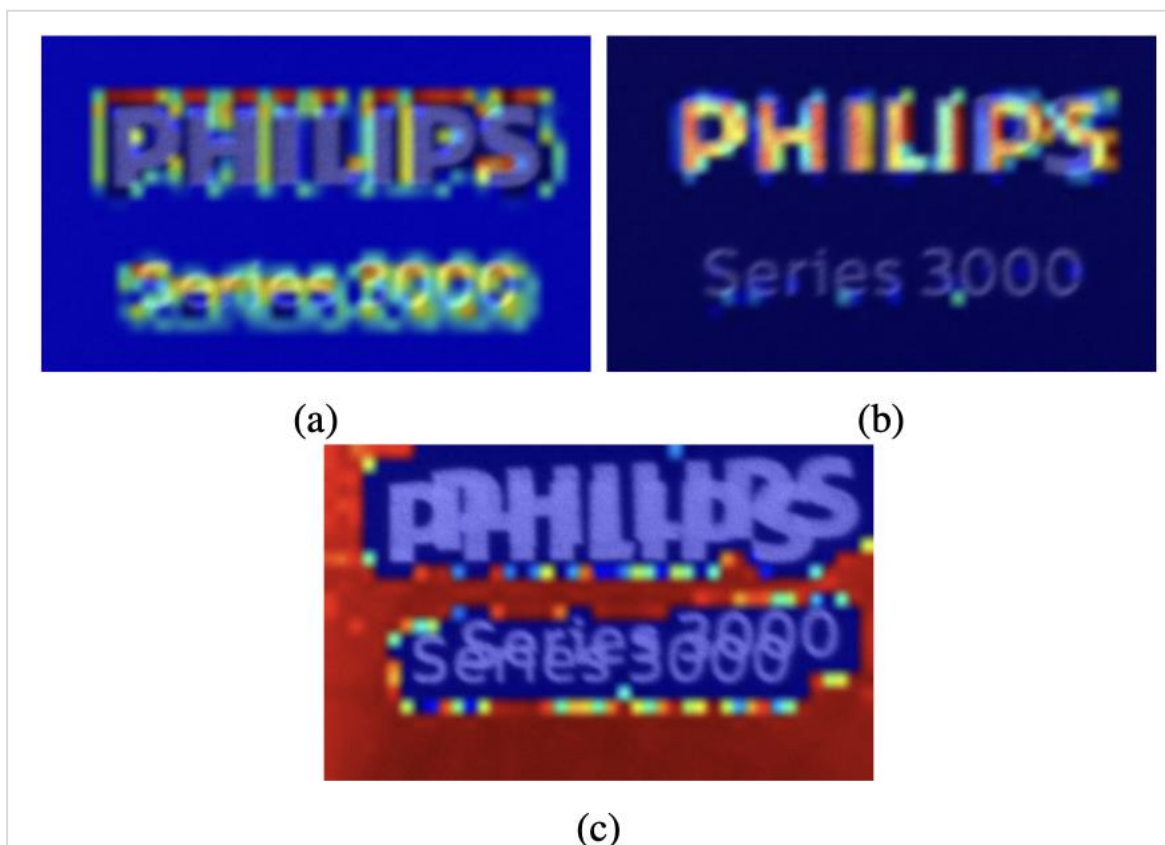


Figure 2 Grad-Cam of 3 image examples, the first image is a good print, the second the interrupted one, and the third a double print one.

We can see the original image and the explanation of grad-cam methods where there are highlighted parts of the image that the CNN model is focusing on in each layer. This is a technique for visualizing the regions in an image that are most important for a convolutional neural network (CNN) to classify an image correctly. It does this by overlaying a heatmap on top of the original image, where the intensity of the colour in the heatmap indicates how important that region is for the classification. To create the heatmap, Grad-CAM first computes the gradient of the output of the CNN with respect to the feature maps of a convolutional

layer. It then weighs these gradients by the importance of the feature maps to the class prediction and produces a weighted combination of the feature maps. Finally, it resizes the resulting map to the size of the input image and overlays it on top of the original image to produce the heatmap. Overall, the heatmap produced by Grad-CAM can be used to understand which regions of an image are most important for a CNN's prediction and to identify potential areas of the image that may be causing the CNN to make incorrect predictions. Fig. 2 shows 3 examples of the process described regarding the explainability of the model. We can see 3 examples of the initial dataset and how they are masked. While Fig. 3 depicts the GradCam heatmap of 2 examples in two layers of the CNN model.



Figure 3 GradCam heatmap of 2 examples in two layers of the CNN model

The output of Grad-CAM is a heatmap overlaid on the original image, where the intensity of the colour at each pixel indicates the relative importance of that region for the prediction. To interpret the output of Grad-CAM, you can look at which regions of the image are highlighted in the heatmap. These regions are likely the most important for the model's prediction, and may provide insight into the features that the model is using to make its decision.

Additionally, you can try to compare the Grad-CAM heatmap of different classes, to understand which regions of the image are more important for one class than for the other. It's important to keep in mind that Grad-CAM is a post-hoc interpretability method, it doesn't give you information about the model's internal process, but it gives you information about which regions of the image are relevant to the model. On the other hand, LIME is a technique for explaining the predictions of any machine learning model by approximating it locally with an

interpretable model. It does this by perturbing the input data and creating a new, simpler model that is fit only to the perturbed data. This process is repeated many times, each time with a different set of perturbations, and the results are aggregated to create a global explanation of the model's prediction.

The colours of the Grad-CAM heatmap typically represent the relative importance of each region of the image for the model's prediction. The colour scale used in the heatmap can vary depending on the implementation, but in most cases, the highest importance is represented by the brightest or most intense colour (e.g. white or yellow), while the lowest importance is represented by the darkest or least intense colour (e.g. black or blue). The heatmap is generated by taking the dot product of the last convolutional layer's output feature maps and the gradient of the class score with respect to the feature maps. Pixels in the heatmap with high values are the ones that were highly activated by the model and had the largest positive impact on the class score. Pixels with low values had little impact on the class score. The heatmap is overlaid on the original image, so you can see which areas of the image are most important for the model's prediction. It's important to keep in mind that these heatmap are generated from a single image, and the interpretation of the heatmap is limited to this image.

In the case of image data, LIME can be used to explain the prediction of a CNN or other image classification model by creating a heatmap that shows which parts of the image are most important for the model's prediction. The heatmap is created by perturbing the pixels of the image and creating a new, simpler model that is fit only to the perturbed data. This process is repeated for different parts of the image, and the results are aggregated to create the final heatmap.

Overall, the heatmap produced by LIME can be used to understand which parts of an image are most important for a model's prediction and to identify potential areas of the image that may be causing the model to make incorrect predictions.

In figure 4, we have the same output with the only difference in the XAI heatmap output, where we can see the green and red areas of the image. This means that our model classifies us because of these parts of super-pixels; the size of super-pixels coloured in green are the ones that increase the probability of our image belonging to the predicted class, while the super-pixels coloured in red are the ones that decrease the likelihood.

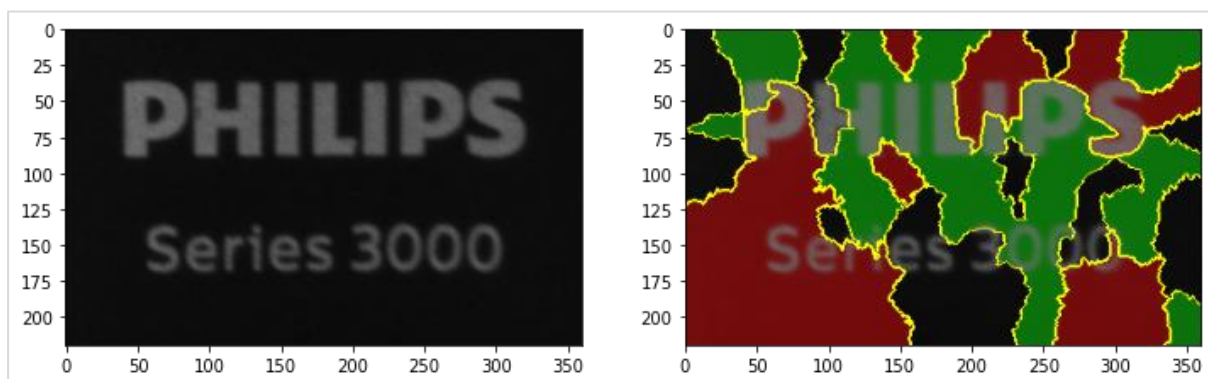


Figure 4 Example of LIME important superpixels

Grad-CAM (Gradient-weighted Class Activation Mapping) and saliency maps are both techniques for visualizing the regions in an image that are most important for a convolutional neural network (CNN) to make a prediction. However, there are some key differences between the two:

1. Scope: Grad-CAM focuses on the feature maps of a specific convolutional layer, while saliency maps consider the entire input image.
2. Gradient calculation: Grad-CAM calculates the gradient of the output of the CNN with respect to the feature maps of a specific convolutional layer, while saliency maps calculate the gradient of the prediction with respect to the input image.
3. Heatmap calculation: Grad-CAM produces a heatmap by weighing the gradients of the feature maps by their importance to the class prediction, while saliency maps use the gradient of the prediction as the heatmap.

Overall, Grad-CAM can be used to understand which regions of an image are most important for a specific convolutional layer in a CNN, while saliency maps can be used to understand which parts of the entire input image are most important for the CNN's prediction.

In figure 5, we present the overlaying heatmap on top of the original image, where the intensity of the colour in the heatmap indicates how important that region is for the prediction.

To create a saliency map, the CNN is first fed the input image and its prediction is made. The gradient of the prediction with respect to the input image is then calculated. This gradient represents the importance of each pixel in the image for the prediction. The gradient is then resized to the size of the input image and overlaid on top of the original image to create the saliency map.

Overall, a saliency map can be used to understand which parts of an image are most important for a CNN's prediction and to identify potential areas of the image that may be causing the CNN to make incorrect predictions. It can also be used to understand how a CNN processes an image and which features it is focusing on to make a prediction.



Figure 5 Example of Saliency Maps

3 XAI for timeseries data

It is well known that IoT and sensor data are of major importance to industrial applications. According to D2.1 and D2.4, both Demonstrator#2 and #3 have posed a requirement of applying ML and DL models to timeseries data. To this end we created the XAI for timeseries component that it was tested on the UCR 1 dataset. The UCR Time Series Archive - introduced in 2002, has become an important resource in the time series data mining community, with at least one thousand published papers making use of at least one data set from the archive. Specifically the dataset used was the FordA dataset for timeseries classification

There has been significant research on developing methods for explaining the predictions of machine learning models in computer vision and natural language processing. However, there is still much work to be done in explaining models applied to time series data. This may be because it is more difficult for humans to understand and interpret temporal data at first glance than visual or textual data. Temporal data is present in many forms, such as sound, but humans are not naturally accustomed to representing this type of data as a signal that changes over time. It may require expert knowledge or additional methods to extract the underlying information from temporal data. This challenge may explain why many existing ways of explaining model predictions in computer vision focus on identifying the parts of the input most important for the forecast.

3.1 Lags Importance

The methods presented below were applied to time-series classification applications. First, we report post-hoc methods that approach the behaviour of a model by exporting relationships between feature values and predictions; in this case, features (lags) of time series also play a role. At the same time, the ante-hoc methods incorporate the explanation in the structure of the model, which is, therefore, already explainable at the end of the training phase. Figures 6 and 7 present the importance of lags of a time-series for classifying this timeseries to a class. Fig 6 presents the SHAP method and Fig 7 LIME, both techniques for explaining the predictions of machine learning models. They both work by approximating the model locally with an interpretable model and creating a heatmap to show which parts of the input data are most important for the prediction. However, there are some key differences between the two:

1. Scope: SHAP considers the entire input data and how it affects the prediction, while LIME only considers a small, local region around the input data.
2. Interpretable model: SHAP uses the Shapley value from game theory to determine the importance of each feature, while LIME uses a simple, interpretable model that is fit only to the perturbed data.
3. Aggregation: SHAP combines the results from different perturbations using the Shapley value, while LIME aggregates the results using a weighted average.

Overall, SHAP is a more comprehensive approach that considers the entire input data and how it affects the prediction, while LIME is a local approach that only considers a small region around the input data.

¹ <http://www.timeseriesclassification.com/description.php?Dataset=FordA>

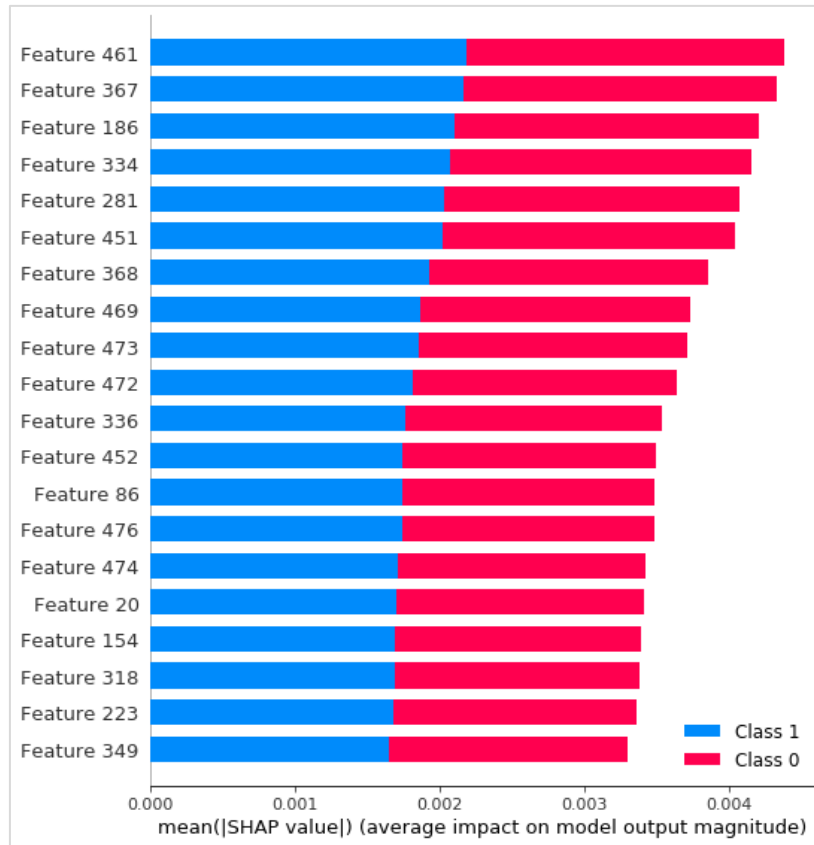


Figure 6 importance of lags of a timeseries, for classification calculated with SHAP

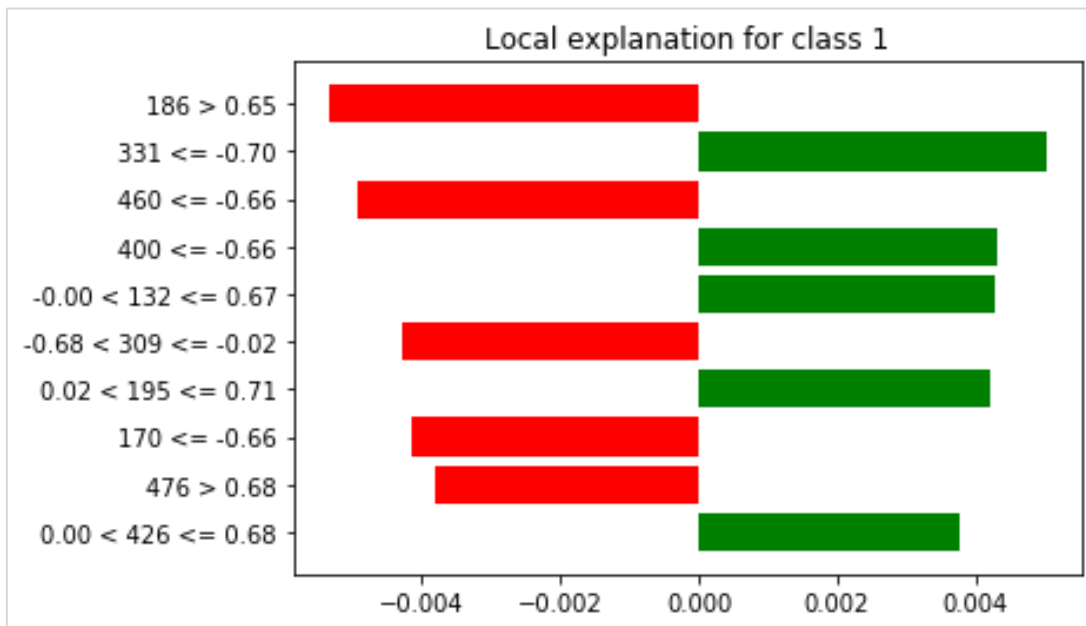


Figure 7 importance of lags of a timeseries, for classification calculated with LIME

3.2 Time-series to Image transformation

Afterwards, we applied image classification with a convolutional neural network (CNN) to a time series. One way to do this is to first plot the time series as an image and then input the image into the CNN for classification.

To plot the time series as an image, you can use a 2D plot with the time axis on the x-axis and the value of the time series on the y-axis. Or we can then use a color map to encode the values of the time series, where different colors represent different values. We examined both of these two ways, and it is obvious that in terms of explainability the first one is more preferred as a human can see the comprehensive visualization.

Once you have plotted the time series as an image, you can input the image into a CNN for classification. The CNN will process the image and use its learned features to make a prediction about the time series.

In figure 8, we see the output of a XAI model that have been applied to a plot of a timeseries based on LIME methods, where we can see the green and red areas of the image. This means that our model classifies an instance because of these parts of super-pixels; the size of super-pixels coloured in green are the ones that increase the probability of our image belonging to the predicted class, while the super-pixels coloured in red are the ones that decrease the likelihood. There we can notice the “common” patterns of the timeseries that belong to the same class.

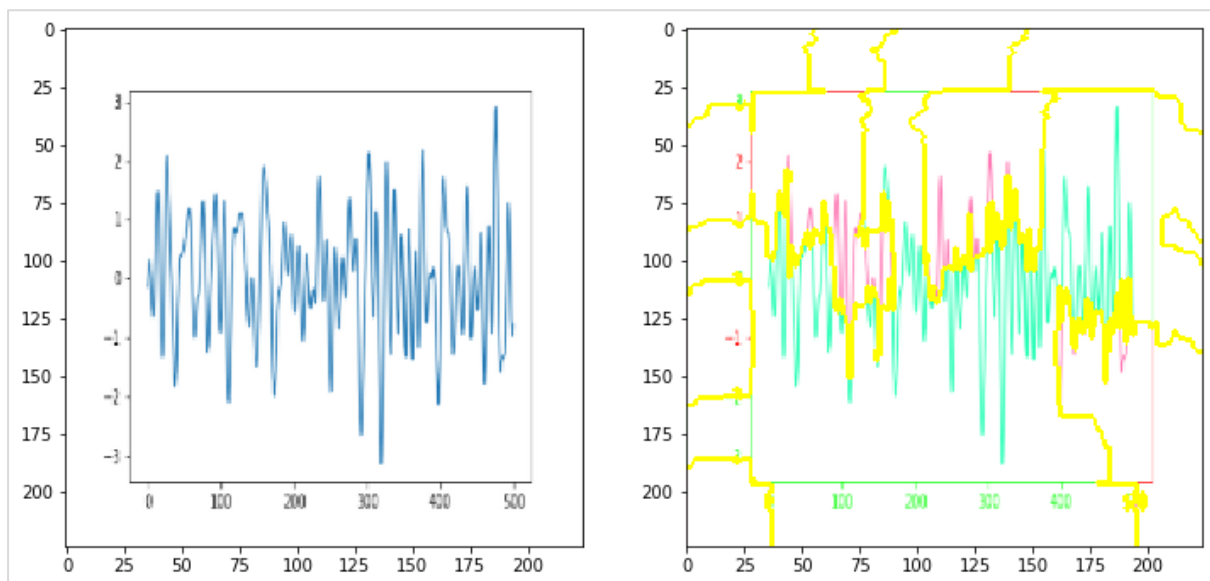


Figure 8 Example of LIME important superpixels where we can notice the red areas that seem not relevant to the predicted class

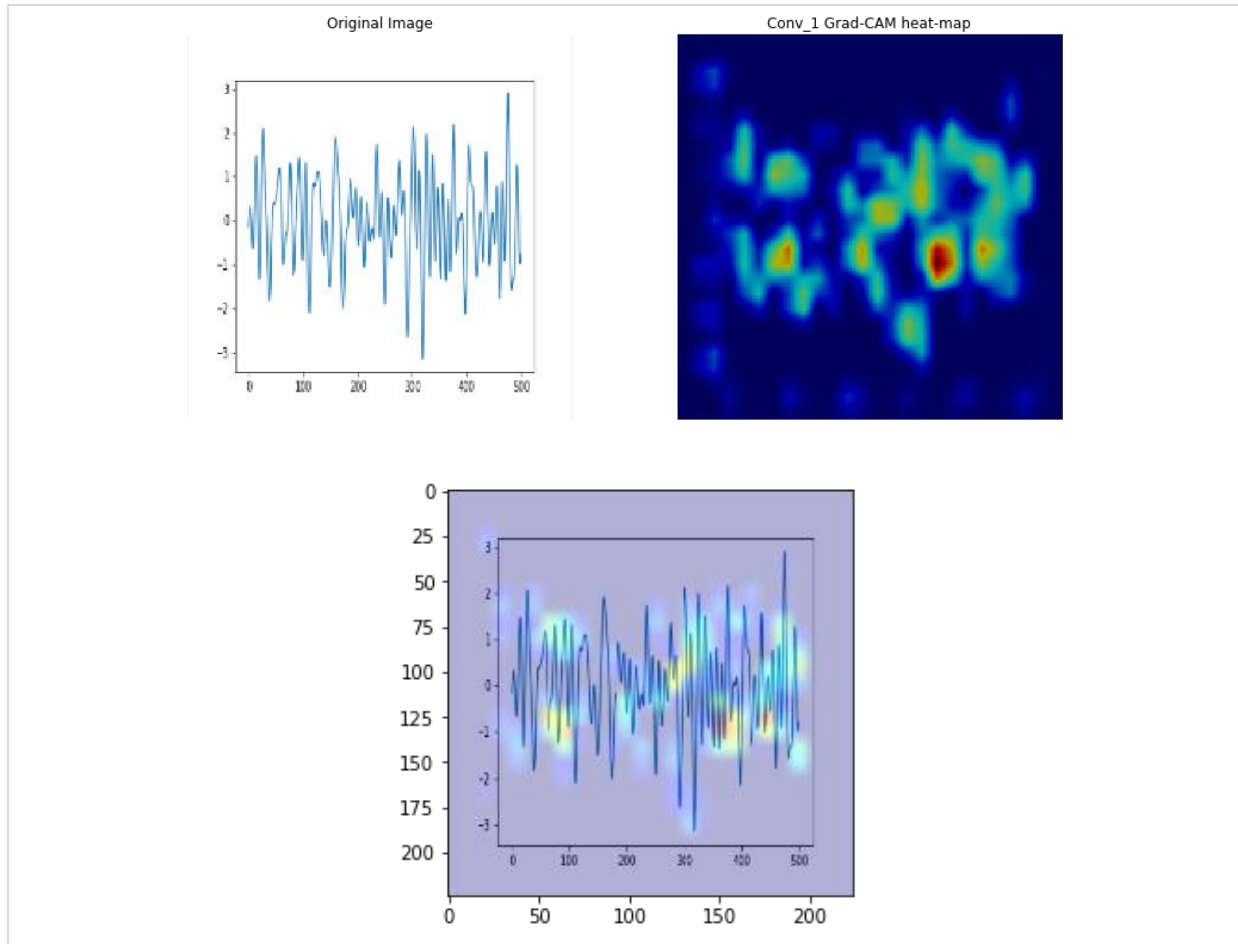


Figure 9 Example of GradCam heatmap and the overlap of heatmap to the original image where we can notice the areas of the image that the model focused

Then we applied GradCam to the timeseries plots and we can notice the sections of the timeseries where the CNN model focus in the same way that this method is applied to image type data.

It is important to note that this approach may not be the most effective way to classify time series data, as it does not take into account the temporal relationships between the data points. Other methods, such as using a recurrent neural network (RNN) or a transform-based model, may be more suitable for analyzing time series data. One way to address this challenge is to plot timeseries with Gramian Angular Field (GAF).

GAF is a method for visualizing and analyzing time series data. It is based on the idea of creating a matrix representation of the time series, called the Gramian matrix, and then applying a transformation to the matrix to create a visual representation of the data.

To create a GAF plot, the time series is first transformed into a matrix, called the Gramian matrix, by multiplying the time series by its transpose. The Gramian matrix contains information about the correlations and dependencies between different parts of the time series.

Next, the Gramian matrix is transformed using a nonlinear function, such as the tangent function, to create a visual representation of the time series. The resulting plot, called the GAF

plot, encodes the correlations and dependencies between different parts of the time series as colors, with warmer colors (such as red and yellow) representing strong correlations and cooler colors (such as blue and green) representing weak or no correlations.

GAF plots are useful for identifying patterns and relationships in time series data, such as periodic patterns and dependencies between different parts of the time series. They can also be used to compare different time series and identify similarities and differences between them. In figure.

We transformed the timeseries with GAF and 10 samples are depicted in Fig.10.



Figure 10 Samples of GAF transformed time-series

Once you have plotted the time series as GAF image, you can input the image into a CNN for classification. The CNN will process the image and use its learned features to make a prediction about the time series. In simple words we follow the process like the previous one. Then we present the figures (Fig.11 and Fig.12) of implementing XAI on these plots.

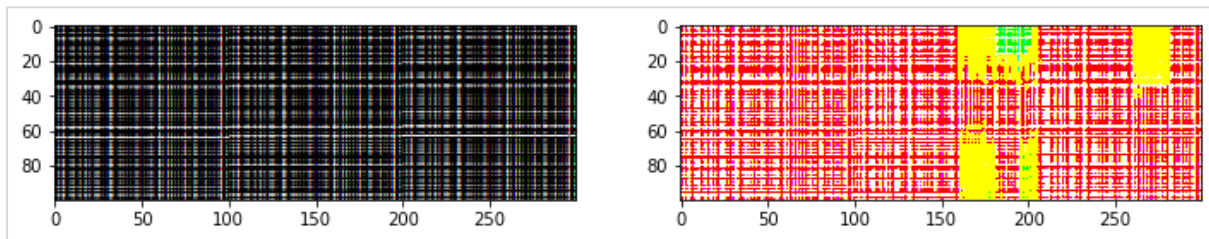


Figure 11 Example of LIME important superpixels where we can notice the red areas that seem not relevant to the predicted class

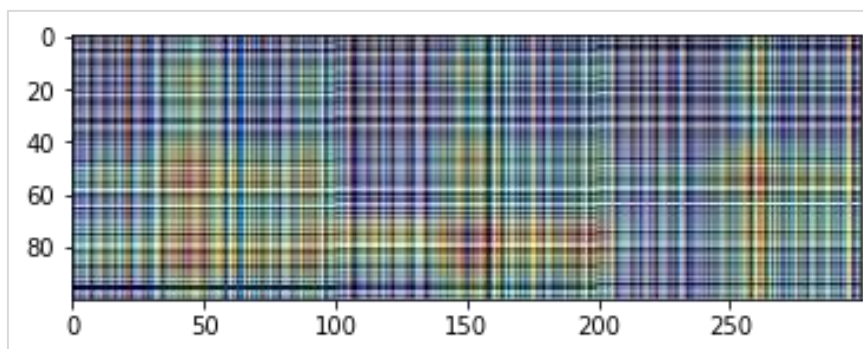


Figure 12 Example of GradCam heatmap and the overlap of heatmap to the original GAF image where we can notice the areas of the image that the model focused

3.3 Time-series to Text transformation

SAX is a technique for converting a time series data set into a sequence of symbols. It does this through two steps: first, it breaks the time series into segments. It averages the values within each segment to create a piece-wise aggregate approximation (PAA) representation, and then it assigns symbols to each segment. The symbols are chosen to be equally likely to be assigned to each segment based on the assumption that the input data follows a Gaussian distribution. The resulting sequence of symbols can be used to identify recurring patterns in the original data.

SAX is a technique that can be used for time series classification, an ML task that involves predicting the class or category of a time series based on its characteristics. By using SAX to convert the time series into a set of interpretable high-level features, selecting the most relevant features for each classification problem can improve the classifier's performance. Additionally, these approaches, based on SAX, can be applied to variable-length time series and are typically easier to interpret than deep learning methods. In other words, they provide both good performance and interpretability, which can be particularly useful in certain situations.

ELI5 is a Python library that provides tools for visualizing and debugging machine learning models. It has a unified API and supports a wide range of models, including those from scikit-learn, Keras, XGBoost, and LightGBM. ELI5 can be used to explain both white-box models, which are transparent and easy to understand, and black-box models, which are more complex and difficult to interpret. For white-box models, ELI5 provides tools for both global and local interpretation, while for black-box models it only provides global interpretation. ELI5 offers several functions for explaining the models, including `eli5.show_weights()` and `eli5.show_prediction()`. It also has a formatter module that can generate explanations in various formats, such as HTML, JSON, and a pandas DataFrame.

The result of the application of time series to text transformation and the classification and the XAI is depicted in Fig.13. Where the green and red colors used are used to indicate the importance of individual words or phrases in the model's predictions. Typically, a word or phrase that is colored green is considered to have a positive influence on the prediction, while one that is colored red is considered to have a negative influence.

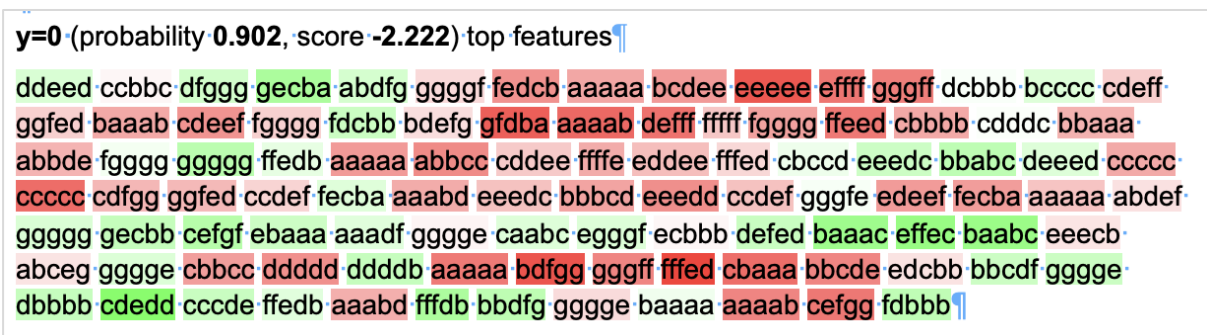


Figure 13 Example of important "words" where we can notice the red areas that seem not relevant to the predicted class

4 XAI for Text and Tabular data

4.1 XAI for Tabular Data

For tabular data, which consists of rows and columns of structured data, there are several approaches that can be used to explain the predictions made by a machine learning model. One approach is to use feature importance measures, which evaluate the relative contribution of each input feature to the model's predictions. These measures can help identify the most important features for a given prediction, and can provide insight into the underlying patterns and relationships in the data that the model has learned. Another approach is to use example-based explanations, which provide a detailed breakdown of how a particular prediction was made by the model. For tabular data, this can involve examining the input feature values for a specific example and showing how they influenced the model's prediction. There are also more general techniques, such as model distillation, which involve training a simpler model to mimic the behaviour of a more complex model in order to make the latter more interpretable. Overall, the use of XAI techniques can be beneficial in a number of situations, including improving trust and confidence in the model, aiding in model debugging and development, and facilitating the use of machine learning models by domain experts who may not have a strong background in data science.

Regarding our library we use SHAP and LIME methods as described in the previous section.

4.2 XAI for Text Data (NLP)

In addition to the main objective mentioned above, it is of interest that NLP modules can be incorporated in other parts of the architecture, so that designers of AI solutions for the industry can apply it to their use cases. Natural Language Processing is the topic of how computers interact with human (natural) languages, notably how to train computers to process and analyze huge volumes of natural language data. The main research topics include question answering, conversational agents, natural language interpretation, and generation. To that aim, T4.1 will explain the objective of employing text data in STAR use-cases as well as the provided text dataset.

Within the context of T4.1 we use the ELI5 framework in cases of text explainability.

5 XAI enhancing cyber defence against adversarial attacks in visual inspection application

In this task, a lot of attention has been given to developing self-resilient XAI algorithms that can withstand poisoning attacks. Many researchers have tried to protect XAI models from these types of attacks (for example, [Heo19] demonstrated the vulnerabilities of state-of-the-art saliency-map-based systems by tampering with the system's adversarial model). The Similarity Difference And Uniqueness method (SIDU) [Muddamsetty20] was designed as an XAI algorithm that provides visual explanations and can identify the parts of objects that most affect the model's prediction outcome. Additionally, SIDU is more resistant to adversarial attacks compared to "black-box" models because its performance can be better understood and explained by domain experts. When compared to fixation maps, SIDU performs better; however, when the algorithms are tested with noisy inputs, the results are reversed, indicating that SIDU is more robust to adversarial attacks. As [Dombrowski19] pointed out, explanation maps are vulnerable to adversarial attacks and manipulations that aim to corrupt the input data. By simplifying the explanation process, [Dombrowski19] was also able to increase the system's resistance to these attacks. Many approaches to detecting adversarial data have treated it as an anomaly detection task in both the input space and the internal activation layer and model architecture. However, [Roth19] has also researched ways to improve detector performance by altering the inputs or training process. According to [Ilyas19], the presence of adversarial examples is an inherent characteristic of the underlying dataset. Therefore, it is important to identify which features are robust and which are not. Additionally, [Ilyas19] also noted that the presence of adversarial examples in the input space can affect more than just a single classification model. As a result, in the presence of non-robust features, the predictive values may be compromised.

Table 2 Confusion matrix for results of the custom CNN model on original data

	precision	recall	f1-score	support
double print	0.98	1	0.99	46
good	0.97	0.99	0.98	540
interrupted print	0.95	0.88	0.91	126
accuracy	0.97			712
macro avg	0.97	0.96	0.96	712
weighted avg	0.97	0.97	0.97	712

Initially, we trained a CNN model for image classification. Following the standard evaluation scheme, 70-30 for train-testsplit the results of the evaluation on the test set of images are depicted in Table 2 where we can notice that this model achieves an accuracy of 97% in this 3-class classification task when normal data are used. This model will be the AI model that will be used to test its robustness against adversarial attacks. To this end we used adversarial data created with the Gradient-based evasion attack [Biggio13] method and, evaluated this model Loss and accuracy on the test set. That means that the model was trained on normal data (train and validation set) and then we poisoned the test set using epsilon from 0 to 0.2

and measure the degradation of the accuracy. These are the results in Table 3 under the label “Normal Images”. As we can see, the accuracy falls heavily after the value of epsilon of 0.02.

Table 3 Results of Loss and Accuracy in various XAI enhanced dataset

Adversarial - epsilon	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
0	0.874	0.9677	0.1098	0.9705	0.448	0.9312
0.001	0.8741	0.9677	0.1283	0.9663	0.4532	0.9284
0.02	0.8838	0.9621	6.1446	0.6742	0.7465	0.8525
0.04	0.9125	0.9551	12.6796	0.6643	1.409	0.7331
0.06	0.9791	0.9424	18.6595	0.6643	2.0485	0.7331
0.09	1.0812	0.9298	24.2798	0.6643	2.64	0.7331
0.1	1.2043	0.9185	29.793	0.6643	3.2418	0.7331
0.13	1.3412	0.9143	35.163	0.6643	3.9503	0.7331
0.16	1.5038	0.9101	40.0465	0.6643	4.8538	0.7331
0.18	1.7087	0.9087	44.1805	0.6643	5.9688	0.7331
0.2	1.9866	0.9073	47.5323	0.6643	7.2715	0.7331

5.1 Investigation with LIME

Figure 14 shows three examples of the process described regarding the explainability of the LIME model. We can see three examples of the initial dataset and how they are masked by the LIME based on the most “important” super-pixels of the images for their category. These examples are labeled as perturbed. Then we can notice the images that are labeled as reconstructed. These are the results of an encoder-decoder DNN that is trained to generate the results of the LIME model. The inspiration behind the usage of this encoder-decoder is depicted in Figure 15. Afterwards, we performed a poisoning attack on the reconstructed images again with the same levels of poisoning and the results are presented in Table III under the label “Pertrubed Images”. We notice that these images are almost as efficient as the initial ones when no adversarial data are present and more robust when in the presence of an attack. For the purposes of this particular study, the correct classification of poisoned samples is of paramount importance, and therefore this approach performs up to standard for the poisoned/malicious test set (as it has an accuracy of 90% when epsilon=0.2). Nevertheless, the original set of images drops to 66% when epsilon = 0.04, which lead us to conclude that the LIME mask approach is effective against Gradient-based evasion attack.



Figure 14 Example of LIME perturbed images against adversarial attacks

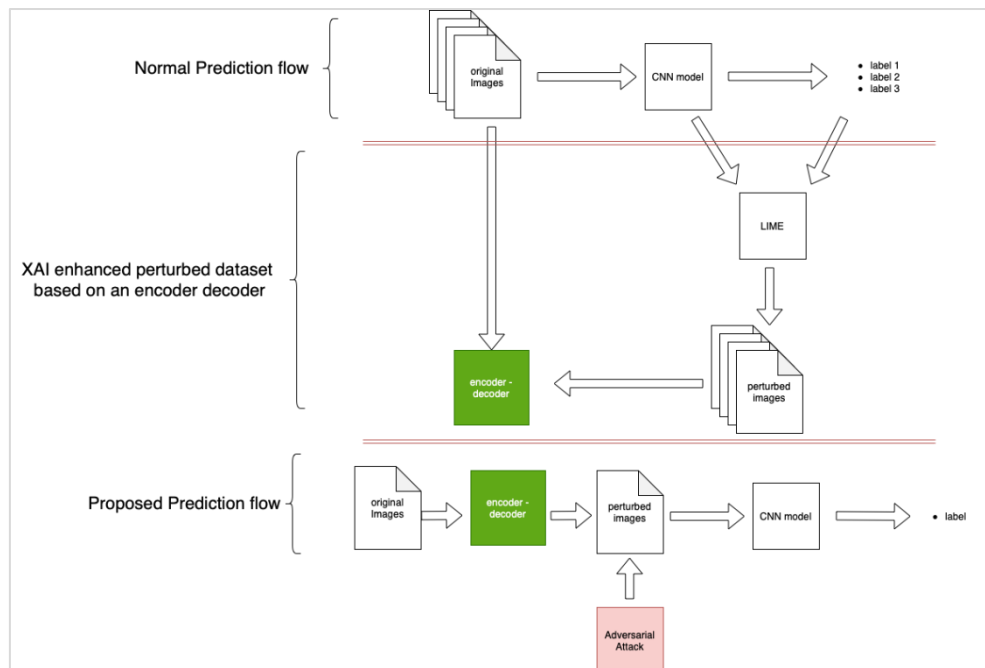


Figure 15 How LIME can be used in the prediction phase to make a model more robust against adversarial attacks.

5.2 Investigation with Grad-Cam

Figure 2 illustrates the process of explaining the model's behaviour with three examples. The figure shows three examples of the initial dataset and how they are masked. Next, the Image Classifier model is applied to the modified images. The results of the classification on the test set are shown in Table 4. It can be seen that the model does not perform well when the inputs are enhanced with highlights, so our initial hypothesis is not supported. Therefore, we used these images as additional information and trained a Siamese type of DNN with two identical networks that are similar to the original one. However, the overall model did not achieve an accuracy of more than 50%, indicating that the modified image acts as noise for the original image.

Table 4 Confusion matrix for results of the custom CNN model on GradCam enhanced data

	precision	recall	f1-score	support
double print	0.05	0.52	0.09	46
good	0	0	0	540
interrupted print	0.52	0.94	0.67	126
accuracy	0.2			712
macro avg	0.19	0.49	0.25	712
weighted avg	0.09	0.2	0.12	712

5.3 Investigation with Saliency Map

Figure 5 shows three examples of the process described regarding the explainability of the Saliency map visualization of important parts of an image explaining why the model classified each image. We see 3 examples of the initial dataset and how they are masked by the Saliency method. Specifically, one can notice the saliency map, the original image, and their superimposition.

Table 5 Confusion matrix for results of the custom CNN model on Saliency Maps enhanced data

	precision	recall	f1-score	support
double print	0.61	1.00	0.76	46
good	0.97	0.99	0.98	540
interrupted print	0.93	0.66	0.77	126
accuracy	0.93			712
macro avg	0.84	0.88	0.84	712
weighted avg	0.94	0.93	0.93	712

Afterward, we applied the Image Classifier model to the Saliency-enhanced images. The classification results of the test set are shown in Table 5. Furthermore, we used poisoning attacks to the saliency-enhanced image with the same poisoning levels. The results are in Table 3 under the label "Saliency Images." We can notice that these images are almost as efficient as the initial ones when no adversarial data are present and more robust when there

is an attack. In this study, it is important to classify the poisoned samples accurately, so this approach seems to perform better for small epsilon, while the performance drops significantly for $\epsilon > 0.04$. It still performs better than the original images but worse than the LIME-masked images.

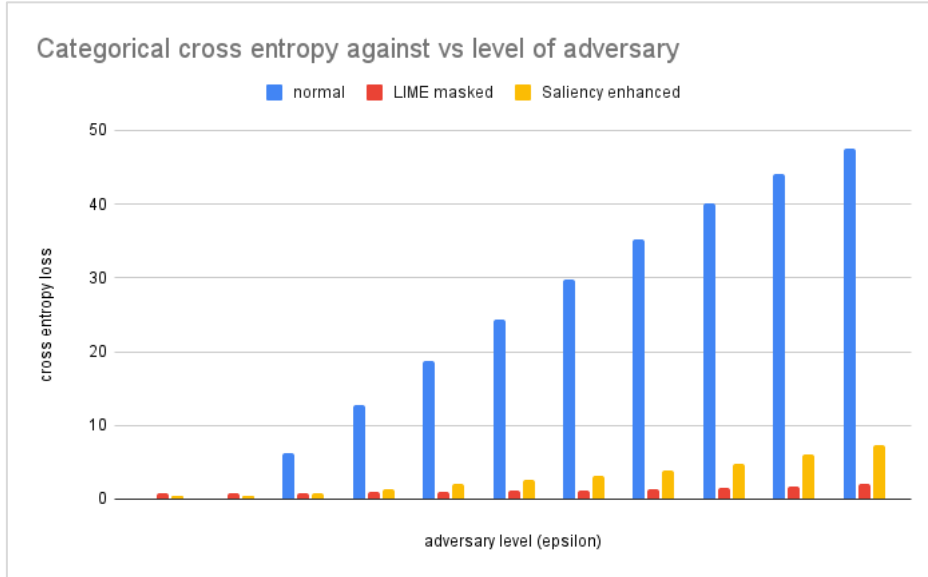


Figure 16 Bar plot showing the Loss of the CNN image classifier against adversarial attacks of various levels (x-axis) on different types of input data

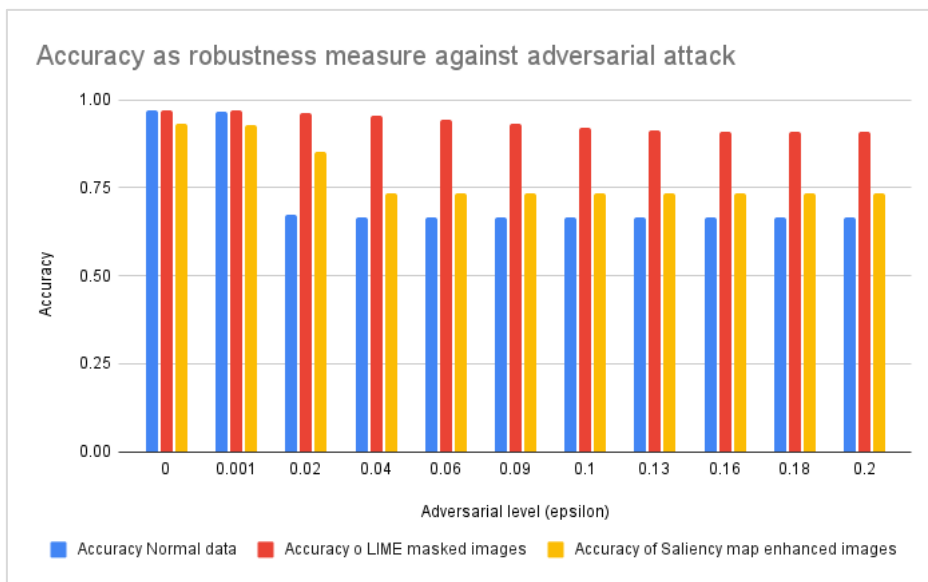


Figure 17 Bar plot showing the accuracy of the CNN image classifier against adversarial attacks of various levels (x-axis) on different types of input data

This task examined an XAI-enhanced DNN for addressing the problem of cyber defense against adversarial attacks for manufacturing image classification tasks. LIME, Saliency maps, and Grad-Cam techniques are used to find the most informative parts of each image with respect to enhancing a trained image classification DL model offering higher explainability and

tolerance against adversarial data attacks. These experiments highlighted that masked/perturbed images (i.e., most important super-pixels) are the most tolerant XAI-based transformation/enhancement to a custom CNN image classifier against Gradient-based evasion attack. The outcomes of our experiments show that LIME in reconstruction is better than the other methods as occlusions make the classifier learn more robust features and not be affected by small perturbations. Our defences rely on the insight that LIME-masked images can serve as a noise reduction process, helping reduce the magnitude of adversarial perturbations.

Moreover, this task proposes a framework that leverages XAI techniques to enhance DL approaches for image classification in industrial use cases. The framework consists of an 'encoder-decoder' architecture for decoding an image in a novel masked/perturbed image that depicts the most predictive part that is less vulnerable to adversarial variations.

6 Conclusion

This document aimed to provide an overview of the methodologies and algorithms that have been selected and/or will be extended for the purposes of STAR project. Although many XAI approaches are under evaluation, an interesting perspective for extension or modification of the underlying models is the User Interfaces or visualizations which will serve as the XAI component's output. Each specific type of XAI solution may have a different representation of the results regarding the stakeholders or the usage of it. The output of a XAI model may as well be injected in another STAR-component (i.e., input to other modules such poisoning/evasion attack detection (WP3) or simulated reality (WP4)) and should differ from the one that will be represented to domain experts.

Thus, one particular goal of the next phase of the requirement-gathering process should be to identify which types of visualizations were needed most by the pilots and for what purpose. When that information becomes available, we will exploit the most common visualizations to explore XAI results and to communicate them.

As a future roadmap, we will be starting to deploy this component to AI4EU platform.

References

Reference	Name of document
[Bai18]	Y. Bai, Z. Sun, J. Deng, L. Li, J. Long, and C. Li, "Manufacturing quality prediction using intelligent learning approaches: A comparative study," <i>Sustainability</i> , vol. 10, no. 1, p. 85, 2018
[Makridis20]	Makridis, G., Kyriazis, D., & Plitsos, S. (2020, September). Predictive maintenance leveraging machine learning for time-series forecasting in the maritime industry. In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC) (pp. 1-8). IEEE
[Heo19]	J. Heo, S. Joo, and T. Moon, "Fooling neural network interpretations via adversarial model manipulation," <i>Advances in Neural Information Processing Systems</i> , vol. 32, 2019.
[Muddamsetty20]	S. M. Muddamsetty, N. J. Mohammad, and T. B. Moeslund, "Sidu: similarity difference and uniqueness method for explainable ai," in 2020 IEEE International Conference on Image Processing (ICIP). IEEE, 2020, pp. 3269–3273
[Dombrowski19]	A.-K. Dombrowski, M. Alber, C. Anders, M. Ackermann, K.-R. Müller, and P. Kessel, "Explanations can be manipulated and geometry is to blame," <i>Advances in Neural Information Processing Systems</i> , vol. 32, 2019
[Roth19]	K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: A statistical test for detecting adversarial examples," in <i>International Conference on Machine Learning</i> . PMLR, 2019, pp. 5498–5507.
[Ilyas19]	A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," <i>Advances in neural information processing systems</i> , vol. 32, 2019.
[Biggio13]	B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in <i>Joint European conference on machine learning and knowledge discovery in databases</i> . Springer, 2013, pp. 387–402