Autonomous and Self-organized Artificial Intelligent Orchestrator for a Greener Industry 4.0

Deliverable

D4.3 Final Reusability, Explainability, Trustworthiness, Security & Privacy mechanisms

Actual submission date: 29/08/2025

Funded by
the European Union

Project Number:	101070181		
Project Acronym:	TALON		
Project Title:	Autonomous and Self-organized Artificial Intelligent Orchestrator for a Greener Industry 4.0		
Start date:	October 1st, 2022	**Duration:**	36 months
D4.3 Final Reusability, Explainability, Trustworthiness, Security & Privacy mechanisms			
Work Package:	WP4		
Lead partner:	SIDROCO Holdings Ltd (SID)		
Author(s):	Konstantinos Kyranou (SID)		
Reviewers:	UBITECH, INNOCUBE		
Due date:	29/08/2025		
Deliverable Type:	DEM	**Dissemination Level:**	PU
Version number:	1.0		

Revision History

Version	Date	Author	Description
0.1	05/05/2025	SID	First release of the Table of Contents
0.2	30/07/2025	SID	Integrated provided contributions from partners
0.3	04/08/2025	SID	Finalised the initial draft
0.5	05/08/2025	INNO	Review of the contents
0.5	06/08/2025	UBI	Review of the contesnts
1.0	29/08/2025	ENG	Final coordinator review and submission

Table of Contents

List of Figures

List of Tables

Definitions and acronyms

AI	Artificial Intelligence
ABAC	Attribute-Based Access Control
ADJP	Adjective Phrase
ADVP	Adverb Phrase
API	Application Programming Interface
AR	Augmented Reality
ARIEC	Anonymous Repository of Incidents
BLSTM	Bidirectional Long Short-Term Memory
CA	Consortium Agreement
CDN	Content Delivery Network
CS	Construction Safety
CNN	Convolutional Neural Network
CRF	Conditional Random Fields
DP	Digital Policies
DoA	Description of Action
DNN	Deep Neural Network
DT	Decision Tree
E2C	Edge-to-cloud
EC	European Commission
EU	European Union
ELMo	Embeddings from Language Models
FedAvg	Federated Average
FedProx	Federated Proximal
FedAdam	Federated Adam
FedAdagrad	Federated Adagrad
FedYogi	Federated Yogi
FL	Federated Learning
FSL	Few-Shot Learning
GA	Grant Agreement
GDPR	General Data Protection Regulation
IAM	Identity and Access Management
ImAM	Image Anonymization Module
IG	Information Gain Attribute Evaluation
IoT	Internet of Things
IOB	Inside Outside and Beginning
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory
MAML	Model-Agnostic Meta Learning
mAP	Mean Average Precision
ML	Machine Learning
MSE	Mean Squared Error
MAE	Mean Absolute Error
NER	Named Entity Recognition
NIC	Network Interface Controller
NP	Neural Processes
NLP	Natural Language Processing
PPE	Personal Protection Equipment
PC	Project Coordinator
PCA	Principal Component Analysis
PEP	Policy Enforcement Point
PIP	Policy Information Point
PDP	Policy Decision Point
POS	Part-of-Speech
PRP	Prepositional Phrase
PII	Personally Identifiable Information

QoS	*Quality of Service*
RBAC	*Role Based Access Control*
RFE	*Recursive Feature Elimination*
RMSE	*Root Mean Squared Error*
RNN	*Recurrent Neural Network*
ROI	*Region of Interest*
SSO	*Single sign-on*
TC	*Technical Coordinator*
TCP	*Transmission Control Protocol*
TL	*Transfer Learning*
TR	*Trust Level*
UC	*Use Case*
VP	*Verb Phrase*
VR	*Virtual Reality*
WP	*Work Package*
XAI	*eXplainable Artificial Intelligence*
YOLO	*You Only Look Once*

Disclaimer

Executive Summary

The “Final Reusability, Explainability, Trustworthiness, Security & Privacy Mechanisms” deliverable offers a comprehensive evaluation of the concluding technical developments within the TALON project. It thoroughly presents the final iterations of TALON’s modules, mechanisms and algorithms that are specifically designed to meet the project’s core requirements regarding design scalability, model comprehensibility, operational credibility, threat resistance and privacy-preserving principles. This document functions as a demonstrator type of manuscript, showcasing TALON’s novel strategies and methodologies in achieving its overarching goals.

This document outlines all the technical enhancements made to the components developed during the final phase of the TALON project. These components align with the four core objectives that form the foundation of the TALON platform’s integrated approach. It presents the complete suite of modules designed to promote Reusability, Explainability, Trustworthiness, Security & Privacy mechanisms, alongside the improvements implemented during the project’s second phase. Furthermore, the impact of these implementations is documented here, demonstrating the significant contribution of the applied technologies to the project’s pilot deployments and use cases (UCs).

This work represents the concluding phase of TALON’s development lifecycle and highlights the demonstration of the platform’s final technical advancements. All modules have now reached a similar level of maturity, reflecting steady and meaningful progress across the entire platform toward meeting TALON’s goals and obligations.

I Introduction

1.1 Purpose of the Deliverable

This deliverable, titled "Final Reusability, Explainability, Trustworthiness, Security & Privacy Mechanisms," presents the final outcomes of the technical activities carried out under Work Package 4 (WP4). It focuses on four TALON's core principles of reusability, explainability, trustworthiness, privacy and security. The document highlights how these key concepts have been addressed within the TALON framework and showcases the final technological results in each area:

i. TALON's AI cognition modules for dynamic reusability
ii. Developed Trustworthy data and AI models
iii. TALON's eXplainable Artificial Intelligence (XAI) framework
iv. Security and Privacy related mechanisms

The deliverable begins with a high-level overview of the ethical and regulatory considerations surrounding the developed modules, demonstrating that TALON has successfully integrated technologies in line with ethical principles, GDPR, and relevant EU regulatory frameworks. The following sections provide a detailed exploration of the platform's reusable data modules, explainable AI capabilities, trustworthy data and AI models, and privacy-preserving mechanisms supported by real results originating from pilot activities/implementations.

1.2 Relation with other Work Packages, Tasks and Deliverables

Tasks 4.1 Task 4.2, Task 4.3 and Task 4.4 have been developed based on key insights from two basic documents: D2.1 "Use Case, KPIs, Requirements, Specification, Slices & Technology Enablers Definition Report" and D5.1 "Installation & Demonstration Planning, Evaluation Methodology & KPIs Definition Report."

D2.1 plays an important role outlining the technical requirements of the Use Cases (UCs) and offering an early assessment of the challenges expected in each pilot. This helped us shape the development strategy and provided the essential directions for the technical work within WP4. In parallel, D5.1 provides a reference framework for defining KPIs and describes the methods for installing and evaluating the TALON components.

The outcomes of Deliverable D4.3 feed directly into several activities within Work Package 5. Specifically, it supports:

- Task 5.1 which focuses on demonstration planning, evaluation processes and KPI tracking
- Task 5.2 which involves setting up, operating and maintaining the TALON platform and ensuring continuous integration of its components

In addition to these core operational tasks, D4.3 informs the project's pilot activities, including:

- Task 5.3 "Automatic UATV Coordination"
- Task 5.4 "I5.0 Automation & Planning"
- Task 5.5 "AR/VR for Training & Maintenance"
- Task 5.6 "Human Robot Collaboration"

These pilots directly apply the technologies and functionalities delivered through WP4.

Beyond serving as a technical input, D4.3 represents the final consolidated evolution of D4.1 and D4.2. It brings together the initial module implementations from the first phase of the project and details the final, mature versions achieved during the second phase completed with real-world demonstrations and pilot deployment results.

1.3 Structure of the Document

Following this introduction, the document is structured in the following way:

- Section 2: This section highlights TALON's efforts to align its AI technologies with ethical standards and EU regulatory frameworks such as the GDPR and the proposed AI Act. It outlines the project's implementation of privacy-preserving techniques, bias mitigation strategies and security measures to ensure the development of secure, fair and trustworthy systems.
- Section 3: This section highlights the reusability of TALON's AI-driven predictive analytics and data processing functionalities.
- Section 4: In this part the detailed overview of the Explainable AI (XAI) modules integrated with enhancements into the platform is showcased
- Section 5: This section covers the finalised data workflows and AI methodologies implemented to ensure the trustworthiness of the TALON system.
- Section 6: This part encapsulates the latest enhancements related to privacy and security within TALON's platform
- Conclusion: Concludes the document

2 TALON Methodology regarding Reusability Explainability,Trustworthiness and Security & Privacy

The TALON project integrates a range of artificial intelligence methods and technologies to develop the intended framework. In particular, this work package (WP4) has focused on outlining some of the key technical components and innovations developed over the course of the project. While the technical requirements have been examined in depth and documented consistently, ethical considerations have also been addressed with the same degree of attention, particularly those related to the secure, fair and trustworthy application of AI. In particular several meetings and recommendations have been provided by the appointed ethical advisor of the project. Given the project's alignment with the broader EU regulatory environment including the AI Act, GDPR and ethics guidelines from the European Commission, it is essential that these dimensions are more clearly highlighted at this section.

Securing the data generation and augmentation processes is a critical concern, since unauthorised access or manipulation at this stage could compromise the integrity of the synthetic data and in extention also the performance and reliability of the AI/ML models trained on it. This risk highlights the need for strong technical safeguards, aligned with Article 10 of the proposed AI Act[1], which emphasises data governance and quality as essential requirements for high-risk AI systems. In response, the project implements controls to ensure traceability, integrity and security across the data lifecycle including mechanisms such as Attribute-Based Access Control (ABAC) and Role-Based Access Control (RBAC). These mechanisms significantly reduce the risk of data breaches or leaks.

In a similar context, the incorporation of privacy-preserving techniques is strongly recommended to protect individual identities, particularly when handling sensitive or personal data. These practices are consistent with the principles set out in the General Data Protection Regulation (GDPR, Regulation (EU) 2016/679)[2], specifically Article 25 (Data protection by design and by default) and Article 32 (Security of processing). To this end, the TALON project employs Federated Learning (FL) models, which inherently improve data privacy. In this paradigm, model training occurs locally on edge devices or decentralised nodes, meaning that raw data never leaves its source. Instead, only model updates or parameters are shared and aggregated centrally. This, combined with the deployed access control mechanisms, significantly reduces the risk of data exposure and guarantees that personal or sensitive information remains under local control, thus aligning with GDPR requirements and strengthening the project’s overall privacy posture.

Special attention must be given to potential biases embedded within training datasets and AI models, as these can undermine model reliability and fairness. Specifically, imbalances in image datasets where some classes are underrepresented could lead to models that favor majority classes and perform poorly on the rest ones, limiting their generalisation ability to real-world data. Similarly, outliers in time-series data can introduce noise and distort prediction accuracy. More importantly, if the resulting predictive capabilities of the AI models is biased, this could lead to systematically unfair or discriminatory decisions that could compromise ethical and legal principles. To address these issues, TALON employs the Explainable AI (XAI) framework. This framework promotes transparency in AI model functionality and provides human interpretable explanations of the AI decisions/outcomes. These technical measures are reinforced by the consortium’s adherence to the Ethics Guidelines for

[1] *https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202401689*
[2] *https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679*

Trustworthy AI[3] and the EU Guidelines on Ethics in Artificial Intelligence: Context and Implementation[4], which advocate fairness, inclusivity, and non-discrimination.

In terms of data privacy and governance, TALON adopts the best available practices and remains responsive to the evolving regulatory landscape and emerging technological risks. Engagement with external stakeholders including end-users, domain experts and regulatory bodies is maintained throughout the project lifecycle to ensure transparency and continuous improvement, as emphasised in the Ethics Guidelines for Trustworthy AI under the requirement of stakeholder participation.

Furthermore, the project would benefit from advancing data anonymisation techniques, especially in the context of image and facial data. By enhancing methods that balance individual privacy with the analytical needs of AI systems, TALON has strengthened compliance with both the GDPR and the AI Act, particularly in use cases where biometric data may fall under the category of high-risk processing.

Finally, the application of Distributed Ledger Technologies (DLTs) for securing AI/ML model weights could offer significant advantages in terms of transparency and integrity. Demonstrating the viability of such methods in real-world industrial scenarios would not only secure AI systems but also contribute to the broader goals of trustworthy and resilient digital infrastructures, as encouraged by the European Commission's strategic priorities in digital transformation.

[3] *https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai*
[4] *https://www.europarl.europa.eu/RegData/etudes/BRIE/2019/640163/EPRS_BRI(2019)640163_EN.pdf*

3 AI Cognition Modules for Dynamic Reusability

3.1 Transfer, Few-shot and Hybrid Learning Framework

3.1.1 Key Technologies & Technical Updates

As part of TALON's AI Cognition Layer, the transfer learning, few-shot learning (FSL) and hybrid learning functionalities presented in this deliverable build upon the foundation established in D4.1, *"Initial Reusability, Explainability, Trustworthiness Report."* In D4.1, a transfer learning (TL) approach was introduced, which involved finetuning specific layers of a deep-learning-based object detector. This approach was experimentally validated and demonstrated promising results for object detection in data-scarce scenarios, showcasing its potential for FSL.

Building on this methodology, we have extended our analysis to incorporate hybrid strategies that combine transfer learning with data augmentation (DA) techniques within the FSL framework. The outcomes of this extended analysis are presented in this deliverable.

In this subsection, we describe the novel components introduced in the final version of the study, along with the methods developed. Specifically, we provide:

- a brief background on relevant DA techniques,
- a detailed analysis of the proposed hybrid methods that integrate TL and DA,
- an overview of the YOLOv8 model used as the examined object detector in our work,
- an overview of the two evaluation settings used to assess FSL performance,
- a novel finetuning pipeline tailored for training the proposed approaches,
- two new metrics designed to evaluate the trade-off between performance and energy consumption and
- a discussion connecting the proposed methods to their real-world applicability.

Background on DA Methods

Before looking deeper into the specifics of the hybrid methods examined in this study and how they extend the previously presented TL approaches, it is important to first understand the role of DAs in the context of FSL. Given that the methodologies explored here focus on object detection, our analysis primarily concentrates on DA techniques designed for image-based tasks, particularly those commonly used in computer vision such as image classification and object detection.

DA has become a fundamental component in deep learning pipelines for computer vision, particularly when addressing tasks like object detection. Traditional DA techniques rely on manually designed transformations tailored to specific datasets or model architectures. These typically include geometric transformations (e.g., rotation, translation, shearing), occlusion-based methods (e.g., random erasing, Cutout), and image-level synthesis approaches where multiple images are combined to produce new samples, such as in MixU [1].

More recently, a new wave of augmentation strategies has emerged that treat DA policy selection as a search problem within a discrete space. This shift has led to the development of automated methods for discovering effective augmentation strategies, leveraging techniques such as reinforcement learning (e.g., AutoAugment [2]), random policy sampling (e.g., RandAugment [3]), and consistency-regularized approaches (e.g., AugMix [4]), where augmentations are selected to maintain semantic consistency with the original input.

Building on these developments, further work has introduced advanced approaches that automate and optimize augmentation selection. These include methods that apply inference-time augmentation policies optimized through Bayesian techniques [5], gradient-based algorithms that enable differentiable policy search [6], and strategies that rely on gradient matching to identify optimal augmentation schemes [7]. In addition, some recent approaches explore the simultaneous optimization of both DA policies and neural network architectures, combining Neural Architecture Search (NAS) with DA policy learning for improved performance under constrained data scenarios [8].

Examined DA methods

Our work explores two main approaches to DAs in the context of FSOD:

- **Custom strategies**, where augmentations are manually defined.
- **Automated strategies**, where augmentations are selected through search algorithms.

Custom DAs are designed to address the challenges of FSL, particularly the risk of overfitting due to limited training data. We introduce two augmentation sets:

1) **Set 1** uses **Sharpen**, **Solarize**, and **Superpixel** transformations. Each is applied independently to images in the novel dataset, tripling the data and resulting in a dataset four times larger. The goal is to increase sample quantity to counter data scarcity.
2) **Set 2** applies one of eight transformations, **Rotation**, **Translation**, **Scaling**, **Shearing**, **Perspective**, **Flipping**, **HSV shift**, and **Mosaic**, to each image with specific probabilities. Here, augmented images replace the originals, maintaining the dataset size but enhancing image diversity to reduce overfitting.

While custom DAs have long been the standard in computer vision, recent advances have introduced automated approaches that adapt DA policies based on the specific task and dataset. In our work, we examine three such methods: **AutoAugment**, **RandAugment**, and **AugMix**.

1) **AutoAugment** [2] formulates DA policy selection as a discrete optimization problem. A controller model (based on an LSTM) selects which augmentations to apply, as well as their strength and probability. The controller is trained using reinforcement learning (RL), where feedback is based on the performance of a target model trained with the selected augmentations. This allows the system to discover task-specific augmentation policies automatically but comes with significant computational overhead.
2) **RandAugment** [3] simplifies AutoAugment by removing the need for a controller model. Instead, it randomly applies a fixed number (N) of augmentation operations from a predefined pool to each image, all at a fixed strength (M). This drastically reduces computational cost while maintaining strong performance, as it removes the need for an RL-based policy search.
3) **AugMix** [4] takes a different approach by combining multiple DA chains into a single image. Specifically, several randomly generated augmentation pipelines are applied to the original image, then blended together using randomly sampled weights. The final image is produced as a weighted combination of the original and these augmented variants. To maintain semantic consistency, AugMix encourages similarity between the model's predictions on original and augmented inputs using a regularization term based on Jensen-Shannon divergence.

These automated methods offer a scalable and adaptive alternative to manually designed augmentations, particularly valuable when dealing with diverse datasets or when domain-specific DA knowledge is limited. An overview of the three automated DA selection strategies examined in our work is presented in Figure 1.

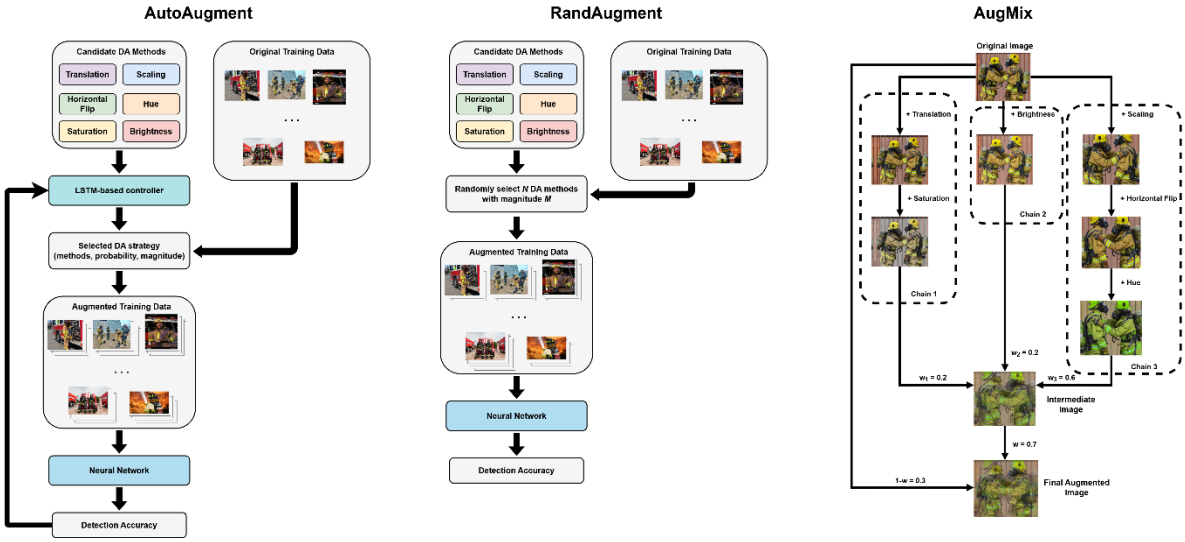

Figure 1: Overview of the three automated DA selection strategies examined

Real-Time Object Detection with YOLOv8

Compared to standard image classification, object detection is significantly more computationally demanding. This is due not only to the need to classify objects, but also to locate them within the image using bounding boxes. Object detection models typically include additional components for handling localization and foreground/background separation, which increases computational load, especially when processing real-world data where images often contain multiple objects.

To address these challenges, we adopt the **YOLOv8** architecture for our experiments. YOLOv8 offers a strong trade-off between accuracy and runtime efficiency, making it well-suited for real-time object detection tasks [9].

The model consists of two primary components:

- A **backbone network** for feature extraction, built on a custom version of CSPDarknet53. This architecture combines the 53-layer Darknet53 CNN [10] with a Cross Stage Partial Network [11] (CSPNet), which improves gradient flow and reduce redundancy using a split-and-merge strategy. It also incorporates a Feature Pyramid Network [12] (FPN) to capture information across multiple scales, allowing detection of objects of different sizes.
- A **prediction head**, composed of a CNN "neck" and three detection modules. These modules receive features from different FPN levels, enabling multi-scale detection. This design supports accurate classification and localization with minimal overhead.

Overall, YOLOv8's architecture provides the necessary efficiency and precision for evaluating FSOD/LSOD strategies under practical, resource-constrained conditions such as those encountered in TALON's industrial use cases.

Problem Formulation

In our work, we focus on the application of few-shot learning for object detection, commonly referred to as few-shot object detection (FSOD). This approach is particularly relevant to TALON's UC4, where the objective is to detect the presence or absence of personal protective equipment (PPE) worn by workers in industrial environments. In such settings, gathering large, annotated datasets for every possible variation of PPE and work condition is often impractical, making few-shot methods an effective solution.

FSOD aims to train object detectors that can quickly adapt to new tasks using only a small number of labeled examples from previously unseen classes. To support this capability, the detector is typically first trained on a base dataset containing many examples from a set of known object categories. This base training phase allows the model to learn useful visual representations and inductive biases that facilitate generalisation.

Once the model is pretrained on this base dataset, it is then finetuned or adapted using a novel dataset that includes only a few annotated examples for each of the target object categories that were not present in the base training phase. Notably, the base and novel categories do not overlap, ensuring the evaluation truly tests the model's ability to generalise.

In our work, we introduce two different approaches to structuring the novel dataset in the FSOD literature:

- **Image-Based FSL**: In this setup, a small number of images are selected for each novel class, and all objects present in those images are used during finetuning. This approach reflects natural image composition and minimizes the risk of training the model to misclassify true foreground objects as background. However, it may lead to class imbalances if some object categories are more frequently co-occurring or underrepresented in the selected images.
- **Object-Based FSL**: This alternative approach also selects a fixed number of images per novel class but considers only a single annotated object per image for model adaptation which is the object that defines the class of interest. While this enforces more controlled sampling and balances class representation, it may discard useful contextual information present in multi-object scenes.

For TALON's purposes, particularly in monitoring PPE compliance across varied industrial scenarios, adopting an image-based few-shot setup offers a more flexible and realistic training paradigm. Nonetheless, care must be taken to manage the potential class imbalance issues that may arise when not all object classes are equally represented in the selected adaptation samples. Consequently, both approaches are examined and evaluated.

In real-world, industrial datasets, such as those relevant to TALON's UC4, images often contain multiple objects from different categories. In these scenarios, the distinction between image-based and object-based settings becomes more pronounced. Consequently, one can expect a performance gap to emerge depending on which approach is used during model adaptation, since object-based FSL may discard useful contextual or multi-object information present in the image.

Additionally, the novel dataset used during adaptation includes not only a small *support set* (used for finetuning the model) but also a *query set*, which consists of separate images used to evaluate the adapted model's performance on the novel object classes.

A common and practical strategy for adapting a pretrained object detector to the novel set involves finetuning the model on the support set, typically by updating only the final classification layer while keeping the rest of the model either fixed or partially trainable. This finetuning can be done in a *full* manner, where all model parameters are updated, or in a *partial* manner, where some layers (e.g.,

the feature extraction backbone) remain frozen. The adapted model is then evaluated on the query set to assess its ability to detect previously unseen classes using only limited training data.

Training Pipeline for FSOD

As outlined in the problem formulation subsection, the object detector is initially trained on a base dataset comprising a large number of images from known object categories. This pretraining phase enables the model to learn rich visual representations and inductive biases that support generalization. After this stage, the model is finetuned using a novel dataset, which contains only a few annotated examples for each target object category not present in the base training set.

In finetuning-based methods, a common practice is to train the full model on the base dataset and then update only selected layers during adaptation to the novel dataset, keeping the remaining layers frozen to prevent overfitting. We adopt a similar strategy: after pretraining on the base dataset, only the detection modules are finetuned on the novel dataset, while the backbone remains unchanged. This approach is further supported by preliminary results reported in D4.1, which show that finetuning only the detection modules of the YOLOv8 object detector results in minimal additional energy consumption. This is a critical consideration in our work, as the models are intended for deployment on edge devices in industrial settings with constrained compute and power resources.

Before finetuning, the novel dataset is augmented using one of the DA strategies described earlier to address data scarcity and mitigate the risk of overfitting due to the limited number of examples. The complete training pipeline of the proposed hybrid approach, which combines TL and DAs, consists of the following three steps:

1. **Step 1 - Pretraining:** The entire model is pretrained on a large base dataset.
2. **Step 2 - Data Augmentation:** The support set of the novel dataset is augmented using a selected DA strategy.
3. **Step 3 - Finetuning:** The detection modules of the model are finetuned on the augmented novel dataset.

Figure 2 highlights the training procedure, highlighting the specific model components involved in each step, as well as the intermediate DA stage between pretraining and finetuning.

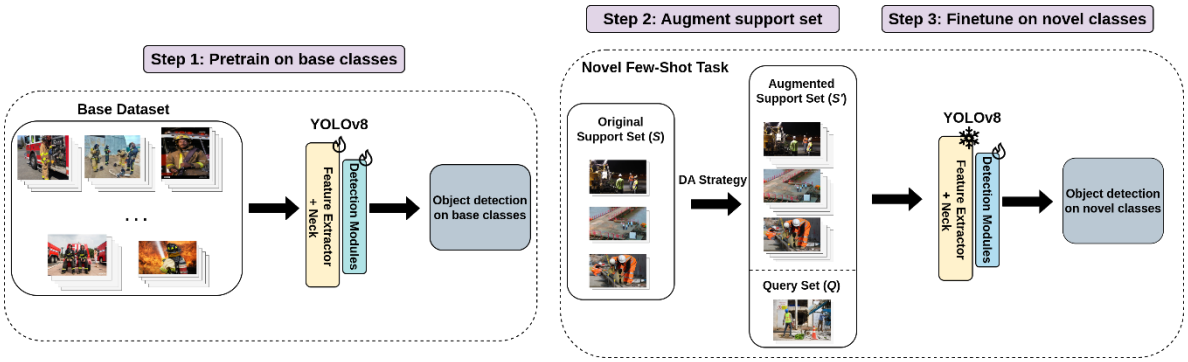

Figure 2: Overview of the training procedure of the YOLOv8 model in the FSOD setting

Quantifying the performance – efficiency trade-off with the EF and mEF metrics

While achieving high detection accuracy is a primary goal in object detection, it often comes with increased energy consumption due to the need for large training datasets and extended training

times. This poses a challenge for deploying deep learning models in resource-constrained environments, such as TALON's industrial setting in UC4.

With the growing emphasis on Green AI, which promotes energy-efficient and environmentally sustainable AI practices, it becomes essential to evaluate whether the energy cost of improving model performance is justified. As a result, the development of FSOD models involves a critical trade-off between maximizing performance and minimizing energy consumption during model training and finetuning.

To quantify this trade-off, we propose the **Efficiency Factor (EF)**, a metric that consolidates both performance and energy usage into a single interpretable value. This allows for direct comparisons between different models, configurations, and DA strategies without relying on multiple separate metrics. Specifically, EF is defined as follows:

$$EF = \frac{AP_{50}}{1 + EC}$$

Where $\mathbf{AP_{50}}$ is the Average Precision at 50% IoU threshold, ranging from 0 to 100, that captures the model's ability to correctly detect and localise objects and **EC** is the energy consumption measured during training (in watt-hours).

This formulation penalizes models that consume excessive energy relative to their accuracy, promoting solutions that are both effective and sustainable. However, one limitation of this metric is that it assumes a linear relationship between energy usage and accuracy gain, which does not align with real-world observations.

Recent works [13] and our own experimental findings [14] [15] demonstrate that energy consumption tends to grow exponentially, while performance improvements grow more slowly as the training dataset scales. This means that beyond a certain point, increasing training effort yields diminishing returns in terms of accuracy. To better capture this dynamic, we propose an enhanced metric, named **modified Efficiency Factor (mEF)**, that is defined as follows:

$$mEF = \frac{AP_{50}}{1 + \ln(1 + EC)}$$

The mEF metric introduces a logarithmic scaling of the energy consumption term. The natural logarithm compresses large values of EC, making the metric more resilient to excessive energy usage and more reflective of real-world deployment trade-offs. Overall, mEF demonstrates the following key characteristics:

- It remains bounded within the range [0, 100), since EC is always positive.
- It gives a more balanced weighting to performance and energy use, especially when comparing models trained under different data or augmentation settings.
- It emphasizes practical efficiency, rewarding models that deliver high accuracy while keeping training energy costs low.

In total, by reporting both EF and mEF, we provide a dual perspective on model efficiency since EF captures strict energy-awareness, penalizing high energy usage more heavily while mEF captures realistic scaling behavior and encourages practical, deployable solutions.

Real-World Applicability of the Proposed Methodology

Although YOLOv8 and the evaluated DA strategies may not reflect the state-of-the-art in detection accuracy, they are widely adopted in real-world applications due to their computational efficiency, deployment simplicity and stable performance.

These methods strike a practical balance between accuracy and resource use, making them well-suited for industrial and resource-constrained AI systems where constraints like energy and computational capacity are critical. Unlike studies focused solely on benchmark performance, our work emphasises the trade-off between detection performance and energy consumption, a key consideration for real-world deployments.

Additionally, to further ground the evaluation of the proposed methods, in section 3.1.3 we have included evaluations using real-world data from TALON's UC4, featuring drone imagery from an actual industrial setting.

Overall, our work complements state-of-the-art efforts by providing guidance on selecting energy-efficient detection pipelines that are viable for resource-constrained environments, aligning with growing needs in industrial computer vision.

3.1.2 Scientific and Technical Results

In the next section we present a comprehensive benchmarking study that follows the methodology outlined in Section 3.1.1. We first describe the datasets, evaluation metrics, and implementation details of our experimental setup. The results are then organised into two broad categories: object-based FSL and image-based FSL. For each setting we report performance, energy consumption, and performance-efficiency trade-offs across a varying number of available training samples. Finally, we analyse the findings and provide a short overall discussion based on them.

Experimental Setting

To support the development of energy-efficient object detectors suitable for real-world deployment, our experiments are conducted on datasets derived from industrial object detection scenarios. Specifically, and in alignment with the work presented in D4.1, we utilize the same CS and PPE datasets used in that study. The datasets are originally split as follows:

- **CS Dataset:** 997 training images, 119 validation images, and 90 testing images
- **PPE Dataset:** 280 training images, 34 validation images, and 31 testing images

For each dataset, the training split is used to construct few-shot tasks, apply DAs, and perform model finetuning. The validation split is used for early stopping to determine the optimal number of finetuning epochs. Final performance metrics are computed on the test split, ensuring a consistent and realistic evaluation of each model's effectiveness and efficiency.

As for the utilized energy metrics, we use $\mathbf{AP_{50}}$ to measure detection performance, which is the Average Precision score obtained when a predicted bounding box is considered a true positive only if its Intersection-over-Union with the corresponding ground-truth box is at least 0.5. We also measure energy consumption (EC) during augmentation and finetuning to assess efficiency, and the **EF** and **mEF** metrics to capture the trade-off between performance and energy use.

Regarding energy consumption, we measure energy consumed during the DA and finetuning phases. While model training involves three main stages, pretraining on a base dataset, augmentation of the novel dataset, and finetuning, we exclude pretraining from our measurements, as it is performed once and reused across tasks. This ensures our energy measurements reflect only the cost of adapting models to new, limited-data scenarios. Energy usage is tracked using the CodeCarbon Python library.

Also note that although absolute values may vary across hardware, the relative comparisons between methods remain valid and informative.

For all experiments, we use the **YOLOv8n** variant, pretrained on the MS COCO dataset, as the base model. This lightweight architecture has approximately 3.2 million parameters, but during finetuning only the three detection modules are updated, resulting in about 750K trainable parameters.

The model is finetuned for up to 1000 epochs using early stopping with a patience of 100 epochs. Training is performed with the AdamW optimizer (learning rate: 0.01), and a batch size of 32. The resulting finetuned model is referred to as ***FT***. All input images are resized to 640 × 640 pixels.

Custom DAs are implemented using the Albumentations [16] library with default settings. Note here that in the following reported results, ***DA(1)*** refers to the augmentations from Set 1, ***DA(2)*** corresponds to Set 2, and ***DA(1&2)*** denotes their combination. For automated DA strategies, we use the default settings from the Ultralytics [5]library, applying augmentations such as translation, scaling, horizontal flip, hue, saturation and brightness.

Object-Based FSL Results

Following the object-based FSL formulation, we finetune the models employing N-way K-shot tasks, for K ∈ {1, 2, 3, 5, 10, 30} indicating the number of annotated object instances per class in the support set and N corresponing to the number of classes in the dataset. The query set for evaluation is the entire test set of each dataset.

Table 1: Test set AP50 for different training strategies and number of shots in the object-based FSL scenario

Dataset	Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
PPE	FT(baseline)	12.23	10.24	12.42	12.00	17.70	23.99
	FT+DA(1)	11.80	10.51	12.79	12.60	17.69	23.17
	FT+DA(2)	**12.96**	**11.12**	14.15	13.77	**25.89**	27.47
	FT+DA(1&2)	12.63	11.07	**15.71**	15.16	25.56	28.11
	FT+AutoAugment	12.17	10.41	15.11	**15.71**	23.22	**33.59**
	FT+RandAugment	12.17	10.41	15.11	**15.71**	23.22	**33.59**
	FT+AugMix	12.17	10.41	15.11	**15.71**	23.22	**33.59**
CS	FT(baseline)	11.58	14.51	12.99	14.06	18.49	32.25
	FT+DA(1)	11.67	14.28	12.72	14.49	17.19	25.17
	FT+DA(2)	11.78	**16.29**	**14.50**	19.14	26.96	37.84
	FT+DA(1&2)	10.31	14.30	13.30	**19.34**	27.97	36.00
	FT+AutoAugment	**12.07**	15.27	14.15	15.18	**33.33**	**45.68**
	FT+RandAugment	**12.07**	15.27	14.15	15.18	**33.33**	**45.68**
	FT+AugMix	**12.07**	15.27	14.15	15.18	**33.33**	**45.68**

[5] *https://pypi.org/project/ultralytics*

Table 1 summarises model performance using different DA strategies across varying shot settings for the CS and PPE datasets. As expected, increasing the number of shots consistently improves performance due to the availability of more training samples during finetuning.

In the PPE dataset, performance across DA strategies is similar in few-shot settings. However, as the number of shots increases, automated DA methods (FT+AutoAugment, FT+RandAugment, FT+AugMix) outperform both handcrafted DAs (e.g., FT+DA(1&2) achieving 28.11% and FT with 23.99%), reaching up to 33.59%.

In the CS dataset, automated DA strategies similarly show stronger performance, especially in the 10- and 30-shot scenarios (45.68%), surpassing both custom DA (37.84% for FT+DA(2)) and baseline finetuning (32.25%). Interestingly, in extremely low-shot conditions (2-, 3-, and 5-shot), custom DA approaches perform better, suggesting they may be more effective at mitigating overfitting when data is extremely limited.

Overall, automated DA methods consistently deliver competitive results, particularly as more shots are available, indicating their advantage over manual DA selection in these settings.

Table 2: Total energy consumption in Wh during training for different training strategies and number of shots in the object-based FSL scenario

Dataset	Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
PPE	FT(baseline)	**3.79**	**3.93**	**4.41**	**4.94**	**7.00**	17.00
	FT+DA(1)	4.62	4.67	5.11	5.76	11.62	29.39
	FT+DA(2)	5.84	9.60	12.15	18.30	53.68	231.66
	FT+DA(1&2)	12.28	46.43	60.02	128.46	244.56	685.06
	FT+AutoAugment	4.26	6.46	9.16	8.11	9.21	11.18
	FT+RandAugment	4.31	6.43	9.32	8.23	9.10	**10.84**
	FT+AugMix	4.23	6.31	9.22	8.00	9.10	10.90
CS	FT(baseline)	**6.40**	**6.88**	**7.00**	**7.12**	13.99	16.97
	FT+DA(1)	6.87	7.26	7.42	8.52	**11.91**	42.55
	FT+DA(2)	7.93	10.18	12.31	24.97	74.76	177.15
	FT+DA(1&2)	15.53	37.12	58.23	146.52	247.87	654.12
	FT+AutoAugment	6.44	7.09	7.11	7.44	15.12	16.03
	FT+RandAugment	6.46	7.14	7.17	7.58	14.94	15.97
	FT+AugMix	6.43	7.05	6.99	7.43	14.95	**15.83**

Table 2 reports the energy consumption during DA and finetuning for the CS and PPE datasets across different shot counts. As expected, energy usage increases with the number of shots, due to the larger number of training samples and longer finetuning required for convergence. Minor deviations from this trend, such as in the 5- and 10-shot settings for automated DA methods in PPE, and FT+AugMix in the 3-shot CS setting, are attributed to the use of early stopping.

Applying DA methods generally results in higher energy consumption compared to baseline finetuning. However, in the 30-shot setting, certain automated methods like FT+RandAugment (PPE) and FT+AugMix (CS) can lead to more efficient convergence, partially offsetting their additional computational cost.

Notably, the combined DA strategy FT+DA(1&2) consumes significantly more energy than the sum of its individual components, highlighting the overhead of combining multiple augmentations during training.

The energy efficiency of DA strategies in the CS and PPE datasets is influenced more by their effect on model convergence than by their computational complexity. Most augmentation techniques add minimal direct overhead since their main impact comes from how they alter training dynamics. For instance, methods like FT+DA(1) and FT+DA(1&2) increase the number of training samples, leading to higher per-epoch energy use.

In contrast, automated DA strategies, despite being algorithmically more complex, can improve convergence in higher-shot settings, thereby lowering overall energy consumption. On the other hand, simpler strategies like FT+DA(2) may degrade convergence under severe data scarcity, ultimately increasing energy usage.

Table 3: Efficiency factor (EF) metric values for different training strategies and number of shots in the object-based FSL scenario

Dataset	Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
PPE	FT(baseline)	**2.51**	**2.09**	**2.32**	**2.07**	**2.43**	1.55
	FT+DA(1)	2.07	1.88	2.12	1.87	1.43	0.78
	FT+DA(2)	1.87	1.11	1.11	0.76	0.47	0.13
	FT+DA(1&2)	0.94	0.28	0.26	0.12	0.10	0.04
	FT+AutoAugment	2.28	1.54	1.78	1.92	2.34	2.81
	FT+RandAugment	2.26	1.54	1.76	1.93	2.33	**2.89**
	FT+AugMix	2.30	1.57	1.78	1.94	2.34	2.88
CS	FT(baseline)	1.57	1.85	1.62	1.73	1.47	1.81
	FT+DA(1)	1.49	1.74	1.51	1.52	1.32	0.63
	FT+DA(2)	1.32	1.46	1.09	0.73	0.37	0.21
	FT+DA(1&2)	0.62	0.38	0.23	0.13	0.11	0.06
	FT+AutoAugment	**1.63**	1.89	1.75	**1.80**	2.11	2.73
	FT+RandAugment	1.62	1.88	1.74	1.78	**2.13**	2.74
	FT+AugMix	**1.63**	**1.90**	**1.78**	**1.80**	2.12	**2.75**

Table 3shows the EF scores for each DA strategy across varying shot counts. EF increases with higher detection performance (AP_{50}) and lower energy consumption. In the PPE dataset, the baseline FT achieves the highest EF scores, driven by its low energy requirements and adequate AP_{50}. Here, energy consumption plays a dominant role in EF, with the most energy-efficient models also scoring highest in EF. In contrast, for the CS dataset, automated DA methods deliver the best EF scores.

Despite consuming more energy than FT or FT+DA(1), they provide a substantial performance boost, which offsets their energy cost and improves EF.

Overall, EF tends to decrease as the number of shots increases, since energy consumption grows faster than model performance. However, automated DA methods are an exception, maintaining the highest EF in the 30-shot setting due to their favorable trade-off between energy use and performance.

Table 4: Modified efficiency factor (mEF) metric values for different training strategies and number of shots in the object-based FSL scenario

Dataset	Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
PPE	FT(baseline)	**4.77**	**3.95**	**4.62**	4.31	5.75	6.17
	FT+DA(1)	4.33	3.84	4.55	4.33	5.00	5.25
	FT+DA(2)	4.44	3.31	3.96	3.48	5.18	4.26
	FT+DA(1&2)	3.52	2.28	3.07	2.59	3.90	3.73
	FT+AutoAugment	4.58	3.46	4.55	4.90	6.99	9.60
	FT+RandAugment	4.56	3.46	4.53	4.88	**7.01**	**9.68**
	FT+AugMix	4.58	3.48	4.56	**4.91**	**7.01**	9.66
CS	FT(baseline)	3.86	4.74	4.22	4.54	4.99	8.29
	FT+DA(1)	3.81	4.59	4.06	4.46	4.83	5.27
	FT+DA(2)	3.69	4.77	4.04	4.50	5.06	6.12
	FT+DA(1&2)	2.71	3.08	2.62	3.23	4.29	4.81
	FT+AutoAugment	4.01	4.94	4.57	**4.85**	8.82	11.91
	FT+RandAugment	4.01	4.93	4.56	4.82	**8.84**	11.92
	FT+AugMix	**4.02**	**4.95**	**4.60**	**4.85**	**8.84**	**11.95**

Table 4 presents the mEF values across all shot settings for the evaluated models. Compared to EF, mEF provides a clearer view of the trade-off between performance and efficiency, as it penalises energy consumption less by using the natural logarithm of energy in its formulation.

For both the PPE and CS datasets, EF and mEF rankings align closely in extremely few-shot scenarios due to minimal energy usage. However, in the 10- and 30-shot settings, automated DA methods achieve the highest mEF scores, even in cases where their EF is not optimal. This highlights their ability to balance improved accuracy with moderate energy demands under more data-rich conditions. Overall, mEF is more informative when model performance is a priority, while EF remains preferable when minimizing energy consumption is the primary objective

Image-Based FSL Results

The second experimental setting we examine is image-based FSL, in which models are finetuned using N-way K-shot tasks, where N is the number of dataset classes and K $\in$ {1, 2, 3, 5, 10, 30} represents the number of images per class in the support set. Unlike the object-based setting, all objects within each image are used during finetuning.

Since real-world industrial images typically contain multiple objects, this setting is expected to result in higher energy consumption due to the increased annotation and processing workload. However, it may also lead to better performance by alleviating data scarcity.

Table 5: Test set AP50 for different training strategies and number of shots in the image-based FSL scenario

Datasets	Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
PPE	FT(baseline)	21.36	29.43	29.69	35.85	42.84	48.46
	FT+DA(1)	20.57	30.60	31.85	37.01	40.85	47.88
	FT+DA(2)	29.10	35.73	36.25	39.74	**47.81**	52.26
	FT+DA(1&2)	27.50	35.29	36.91	**43.63**	46.98	**53.58**
	FT+AutoAugment	**32.19**	**38.32**	**37.80**	42.59	46.04	53.44
	FT+RandAugment	**32.19**	**38.32**	**37.80**	42.59	46.04	53.44
	FT+AugMix	**32.19**	**38.32**	**37.80**	42.59	46.04	53.44
CS	FT(baseline)	27.78	34.26	37.41	40.15	45.49	53.88
	FT+DA(1)	26.59	33.55	34.92	40.84	45.04	53.27
	FT+DA(2)	32.92	42.99	42.02	48.53	54.04	62.54
	FT+DA(1&2)	32.07	40.22	41.40	47.27	56.29	60.66
	FT+AutoAugment	**40.66**	**49.08**	**47.90**	**53.99**	**59.76**	**65.81**
	FT+RandAugment	**40.66**	**49.08**	**47.90**	**53.99**	**59.76**	**65.80**
	FT+AugMix	**40.66**	**49.08**	**47.90**	**53.99**	**59.76**	**65.81**

Table 5 presents model performance (AP_{50}) using various DA strategies across different shot counts for the CS and PPE datasets in the image-based FSL setting. As in the object-based setting, performance improves consistently with more shots, driven by the increased availability of labeled samples during finetuning, which allows for better adaptation to novel classes.

In the PPE dataset, automated DA methods perform best in the 1-, 2-, and 3-shot scenarios. As the number of shots increases, the gap narrows, with custom strategies like FT+DA(1&2) outperforming others in the 5- and 30-shot settings, and FT+DA(2) leading at 10 shots. In the CS dataset, automated DA methods consistently outperform all other approaches across all shot counts, making them a reliable baseline in most cases.

Finally, as expected, image-based models generally outperform their object-based counterparts (Table 1), benefiting from the larger number of annotated objects available during finetuning

Table 6: Total energy consumption in Wh during training for different training strategies and number of shots in the image-based FSL scenario

Dataset	Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30

PPE	FT(baseline)	11.93	14.19	14.34	**10.49**	**12.54**	**13.08**
	FT+DA(1)	12.22	12.73	**9.97**	13.20	14.70	35.42
	FT+DA(2)	**11.74**	**12.24**	20.21	19.91	54.44	222.26
	FT+DA(1&2)	14.20	28.18	55.00	68.01	232.55	434.00
	FT+AutoAugment	13.04	14.14	12.20	19.30	16.03	31.33
	FT+RandAugment	13.19	13.77	12.18	19.64	15.66	31.66
	FT+AugMix	13.19	14.44	12.37	19.70	16.07	31.26
CS	FT(baseline)	20.89	20.63	**16.99**	20.58	17.59	**13.45**
	FT+DA(1)	21.01	**17.38**	18.19	**12.80**	**14.13**	23.26
	FT+DA(2)	22.18	30.40	28.60	26.32	50.47	157.34
	FT+DA(1&2)	**19.30**	35.17	48.89	75.69	187.56	437.91
	FT+AutoAugment	25.80	22.89	25.57	30.55	26.96	38.54
	FT+RandAugment	25.54	23.22	25.10	30.36	27.15	38.97
	FT+AugMix	25.75	23.40	25.56	30.72	26.80	39.60

Table 6 shows energy consumption during DA and finetuning for the CS and PPE datasets across different shot counts. As expected, baseline FT and custom DA methods generally consume less energy than automated DA approaches. For automated DA methods, energy use increases with the number of shots due to longer training times. In contrast, this trend is less consistent for custom DA strategies. Notably, in both datasets, methods like FT+DA(1) occasionally use less energy than vanilla FT, likely due to faster convergence when addressing limited data diversity in few-shot scenarios. Across all settings, FT+DA(1&2) is the least energy-efficient approach, primarily due to the additional computational cost introduced by combining multiple augmentation strategies.

These results reinforce that energy consumption is primarily driven by the training dynamics introduced by DA strategies, rather than their algorithmic complexity. While DAs add minimal computational overhead, their influence on sample count and convergence behavior significantly impacts energy use. Thus, the efficiency of a DA method depends more on how effectively it aligns with dataset characteristics, such as size, class distribution, and data scarcity, than on its internal complexity.

Table 7: Efficiency factor (EF) metric values for different training strategies and number of shots in the image-based FSL scenario

Dataset	Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
PPE	FT(baseline)	1.79	2.04	1.96	**3.13**	**3.22**	**3.58**
	FT+DA(1)	1.67	2.33	**3.02**	2.67	2.60	1.48
	FT+DA(2)	2.29	**2.71**	1.87	1.96	0.91	0.31
	FT+DA(1&2)	1.81	1.23	0.70	0.64	0.20	0.12
	FT+AutoAugment	**2.32**	2.63	2.92	2.20	2.79	2.01

	FT+RandAugment	2.28	2.69	2.91	2.14	2.84	2.03
	FT+AugMix	2.28	2.62	2.85	2.14	2.76	2.03
CS	FT(baseline)	1.33	1.64	**2.09**	1.87	2.55	**3.74**
	FT+DA(1)	1.23	1.91	2.06	**2.96**	**2.98**	2.20
	FT+DA(2)	1.43	1.46	1.56	1.79	1.08	0.44
	FT+DA(1&2)	**1.60**	1.16	0.86	0.67	0.30	0.14
	FT+AutoAugment	1.55	**2.07**	1.87	1.73	2.18	1.69
	FT+RandAugment	1.57	2.05	1.91	1.73	2.15	1.66
	FT+AugMix	1.55	2.03	1.89	1.72	2.18	1.64

Table 7 presents EF scores for each DA strategy across shot counts in the image-based setting for the CS and PPE datasets. While automated DA methods yield the highest AP_{50}, their increased energy demands at higher shot counts result in lower EF values beyond the 2-shot setting. In contrast, custom DA strategies and baseline FT show stronger EF performance at higher shot counts due to their lower energy usage.

However, when comparing to mEF scores (Table 8), there is less consistency. Since mEF penalizes energy consumption more mildly, it favors automated DA strategies in most scenarios across both datasets, reflecting their superior accuracy.

Table 8: Modified efficiency factor (mEF) metric values for different training strategies and number of shots in the image-based FSL scenario

Dataset	Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
PPE	FT(baseline)	6.12	8.00	7.98	10.42	11.93	**13.40**
	FT+DA(1)	5.78	8.54	9.47	10.18	10.88	10.61
	FT+DA(2)	8.21	9.98	9.10	9.89	9.61	8.42
	FT+DA(1&2)	7.39	8.09	7.41	8.35	7.28	7.57
	FT+AutoAugment	**8.86**	10.39	**10.60**	**10.72**	12.08	12.36
	FT+RandAugment	8.82	**10.45**	**10.60**	10.64	**12.14**	12.37
	FT+AugMix	8.82	10.37	10.54	10.64	12.06	12.38
CS	FT(baseline)	6.86	8.47	9.63	9.87	11.70	**14.69**
	FT+DA(1)	6.52	8.65	9.05	11.27	12.12	12.72
	FT+DA(2)	7.95	9.78	9.73	11.28	10.98	10.42
	FT+DA(1&2)	8.02	8.83	8.48	8.95	9.02	8.56
	FT+AutoAugment	9.51	**11.78**	11.26	12.15	13.84	14.10
	FT+RandAugment	**9.54**	11.74	**11.32**	**12.16**	13.81	14.06
	FT+AugMix	9.51	11.73	11.29	12.13	**13.85**	14.02

Discussion

From the above study in the two examined settings, the key takeaway is that while some general recommendations can be made regarding data augmentation strategies across different scenarios, their effectiveness is highly dataset-dependent and does not always guarantee improved performance. As a result, in the future it would be valuable to explore how other recent methods for automated and efficiency-driven augmentation selection perform in these settings. Also, applying these insights to develop data augmentation strategies that jointly optimize model performance and energy efficiency would be a promising direction for future work.

3.1.3 Instantiation with pilot data

A key consideration regarding the results obtained from the PPE and CS datasets is the nature of their image sources. These datasets comprise images scraped from various online platforms. Although the images are associated with practical use cases, such as detecting personal protective equipment on construction sites (CS dataset) and on firefighters (PPE dataset), not all were captured in authentic real-world environments. This introduces a disparity between these datasets and others that are strictly collected under real-world conditions and protocols. Consequently, the performance of few-shot object detection models trained and evaluated on these datasets may not accurately reflect their performance in real-world applications, due to differences in data characteristics.

To address this issue, we evaluate the performance and energy efficiency of few-shot object detectors on a real-world dataset, following the methodology outlined in section 3.1.2. Specifically, within the context of TALON, we focus on UC4, which involves detecting PPE worn by workers in industrial environments.

TALON's UC4 dataset focuses on automating safety monitoring in industrial settings through human-robot collaboration. Collected via drone patrols in a real warehouse environment, the dataset captures workers wearing or not wearing PPE, specifically helmets and vests, across diverse scenarios. The data collection took place in five distinct areas (indoor and outdoor), with variations in drone altitude, camera angles, lighting conditions, and worker activities to simulate realistic conditions. All images are in full HD (1920x1080) resolution to ensure adequate detail for detecting PPE and environmental factors. The dataset aims to reflect real-world complexity, including occlusions, group scenes, and varying background clutter.

Annotation was conducted using the CVAT tool, with a rigorous validation process involving three annotators. The final dataset contains significantly more instances of helmets (~1750) than vests (~500), highlighting a class imbalance. To support model training, the dataset was split into training (75%), validation (15%), and test (10%) sets, while maintaining balanced distributions across worker PPE usage and environmental conditions. Despite challenges such as partial occlusions and variable image quality, this dataset provides a robust foundation for developing and evaluating PPE detection systems in safety-critical industrial applications.

Similar to the CS and PPE datasets, the training split is used to construct few-shot tasks, apply DAs, and perform model finetuning. The validation split is used for early stopping to determine the optimal number of finetuning epochs. Final performance metrics are computed on the test split, ensuring a consistent and realistic evaluation of each model's effectiveness and efficiency.

Object-Based FSL results

Table 9 summarizes model performance on TALON's UC4 dataset across various shot counts. As expected, performance improves with more shots due to the availability of additional training samples during finetuning.

Automated DA strategies (FT+AutoAugment, FT+RandAugment, FT+AugMix) consistently outperform custom approaches at all shot levels. The performance gap is substantial, for example, FT+AutoAugment reaches 37.58%, compared to 28.68% with FT+DA(1&2). This highlights the effectiveness of learned augmentation policies over manually designed ones in this real-world dataset.

Table 9: Test set AP50 for different training strategies and number of shots in the object based FSL scenario using TALON's UC4 data

Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
FT(baseline)	0.15	2.08	7.11	10.59	15.72	37.34
FT+DA(1)	0.05	2.25	8.27	10.61	20.80	35.44
FT+DA(2)	0.17	12.11	19.73	21.98	28.14	46.14
FT+DA(1&2)	0.05	12.16	20.43	16.77	28.68	43.34
FT+AutoAugment	**0.24**	**14.40**	**24.04**	**27.00**	**37.58**	**51.55**
FT+RandAugment	**0.24**	**14.40**	**24.04**	**27.00**	**37.58**	**51.55**
FT+AugMix	**0.24**	**14.40**	**24.04**	**27.00**	**37.58**	**51.55**

Table 10 reports energy usage during DAs and finetuning across different shot counts for TALON's UC4 dataset. Unlike the trend observed in datasets such as CS and PPE, where energy consumption generally increases with more shots, TALON's UC4 dataset shows a less consistent pattern. This is likely due to its extremely limited data setting, with only two available classes.

For example, with the FT+DA(2) strategy, energy usage in the 1-shot scenario (9.74) exceeds that of the 10-shot scenario (8.88), indicating inefficient convergence under severe data scarcity. Such irregularities highlight how the small sample size and variability introduced during task sampling can significantly impact training stability and energy consumption in TALON's UC4 dataset.

Table 10: Total energy consumption in Wh during training for different training strategies and number of shots in the object-based FSL scenario using TALON's UC4 data

Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
FT(baseline)	3.61	7.47	**5.35**	6.96	6.37	12.10
FT+DA(1)	**3.11**	6.46	13.35	5.51	**5.82**	**8.67**
FT+DA(2)	9.74	**5.79**	8.14	**4.93**	8.88	30.85
FT+DA(1&2)	5.54	7.32	10.67	6.83	70.16	178.18
FT+AutoAugment	7.71	7.01	6.96	12.31	10.77	12.31
FT+RandAugment	7.75	7.09	7.21	12.41	10.86	12.30
FT+AugMix	7.77	7.14	7.60	12.29	10.74	12.30

Table 11 presents EF scores for each DA strategy across varying shot counts in TALON's UC4 dataset. Since EF reflects both performance (AP_{50}) and energy consumption, strategies that optimize either or both tend to score higher.

In this real-world dataset, automated DA methods achieve the highest EF values despite their greater energy demands, as their improvements in AP_{50} outweigh the added energy consumption. As with other datasets, EF generally declines with more shots due to a steeper increase in energy usage relative to performance gains. However, this decline is less pronounced for automated strategies, which maintain the top EF scores even in the 30-shot scenario.

Table 11: Efficiency factor (EF) metric values for different training strategies and number of shots in the object-based FSL scenario using TALON's UC4 data

Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
FT(baseline)	**0.03**	0.25	1.12	1.33	2.13	2.85
FT+DA(1)	0.01	0.30	0.58	1.63	3.05	3.66
FT+DA(2)	0.02	1.78	2.16	**3.71**	2.85	1.45
FT+DA(1&2)	0.01	1.46	1.75	2.14	0.40	0.24
FT+AutoAugment	**0.03**	**1.80**	**3.02**	2.03	3.19	3.87
FT+RandAugment	**0.03**	1.78	2.93	2.01	3.17	**3.88**
FT+AugMix	**0.03**	1.77	2.80	2.03	**3.20**	**3.88**

Table 12 shows the mEF scores for each DA strategy across shot counts in TALON's UC4 dataset. Compared to EF, mEF values are generally higher and provide a clearer differentiation between methods, as mEF penalizes energy consumption less severely by applying a logarithmic scaling.

In extremely few-shot scenarios, EF and mEF rankings align closely due to the limited impact of energy consumption. However, in the 10- and 30-shot settings, automated DA methods achieve the highest mEF values, even when their EF scores are not the top-ranked. This underscores the value of mEF in scenarios where performance gains outweigh modest increases in energy usage.

Overall, we observe that, as with PPE and CS, mEF is more appropriate when model accuracy is a primary concern, while EF is better suited for scenarios prioritizing minimal energy consumption.

Table 12: Modified efficiency factor (mEF) metric values for different training strategies and number of shots in the object-based FSL scenario using TALON's UC4 data.

Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
FT(baseline)	0.06	0.66	2.50	3.44	5.24	10.45
FT+DA(1)	0.02	0.75	2.26	3.69	7.12	10.84
FT+DA(2)	0.05	4.15	6.14	**7.91**	8.55	10.34
FT+DA(1&2)	0.02	3.90	5.91	5.48	5.45	7.00
FT+AutoAugment	**0.08**	**4.67**	**7.82**	7.52	10.84	14.36
FT+RandAugment	**0.08**	4.66	7.74	7.51	10.82	**14.37**
FT+AugMix	**0.08**	4.65	7.63	7.53	**10.85**	**14.37**

Figure 3 plots AP_{50} performance against energy consumption for vanilla FT and the three custom DA strategies in TALON's UC4 dataset, along with the corresponding Pareto front that illustrates the

tradeoff between performance and efficiency. As observed elsewhere, extremely few-shot configurations (e.g., 1-, 2-, 3-shot) appear on the left of the curve with lower energy usage, while higher-shot settings (10- and 30-shot) are positioned on the right due to increased energy demands.

In this dataset, FT and FT+DA(1) offer the best trade-off in extremely few-shot settings due to their minimal energy consumption. As the number of shots increases, FT+DA(2) emerges as the most balanced option in the 10- and 30-shot cases, combining higher performance with acceptable energy usage. Additionally, 30-shot FT+DA(1&2) lies on the Pareto front, offering the best AP_{50} in TALON's UC4 dataset but at the cost of significantly higher energy consumption.

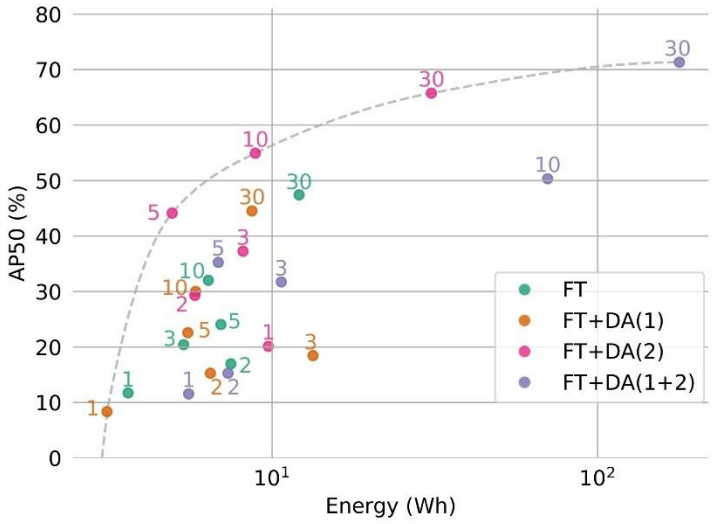

Figure 3: AP50 with respect to energy consumption for different DA strategies and numbers of shots in the object-based FSL setting using TALON's UC4 data

Figure 4 shows the difference in energy consumption between automated DA methods and the baseline FT for TALON's UC4 dataset. Since all automated strategies yield identical AP_{50} performance, this comparison isolates their energy efficiency.

In this dataset, automated DA approaches generally consume more energy than FT, except in the 2-shot setting, where a slight reduction is observed. This anomaly may result from randomness in task sampling under severe data scarcity, allowing faster convergence via early stopping. Despite the higher energy usage, Table 11 and Table 12 show that the gains in AP_{50} more than compensate for the cost, as reflected in superior EF and mEF scores. These findings underscore the importance of using composite metrics like EF and mEF to guide DA strategy selection, rather than relying solely on performance or energy metrics in isolation.

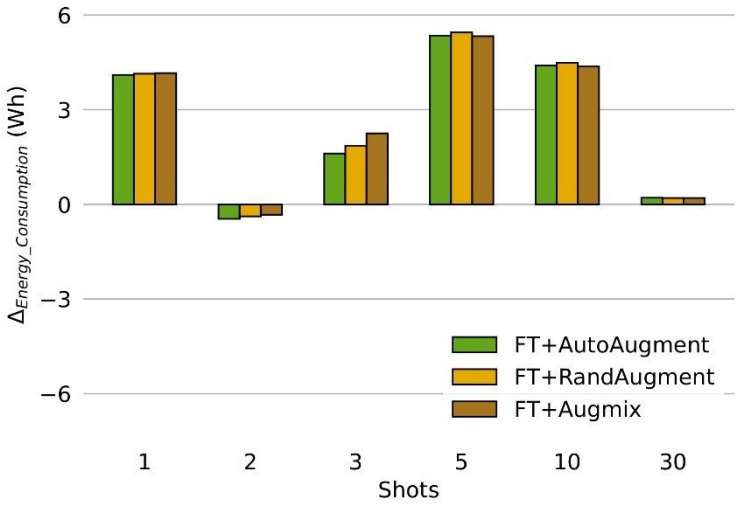

Figure 4: Energy consumption difference between different DA strategies and vanilla fine tuning for a varying number of shots in the object-based FSL setting using TALON's UC4 data

Image-Based FSL Results

Table 13 presents AP_{50} performance across various shot counts in the image-based FSL setting for TALON's UC4 dataset. As expected, increasing the number of shots leads to improved model performance due to the greater availability of labeled objects during finetuning.

Automated DA strategies consistently deliver strong results across most shot settings in the dataset. However, in the 30-shot scenario, FT+DA(1&2) outperforms all other methods, indicating that a well-designed custom augmentation strategy can remain competitive or even superior when more data is available.

Table 13: Test set AP50 for different training strategies and number of shots in the image-based FSL scenario using TALON's UC4 data

Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
FT(baseline)	11.73	16.96	20.42	24.06	32.03	47.43
FT+DA(1)	8.35	15.27	18.45	22.58	30.02	44.53
FT+DA(2)	20.12	29.30	37.26	44.14	54.93	65.76
FT+DA(1&2)	11.55	15.28	31.73	35.25	50.33	**71.33**
FT+AutoAugment	**24.77**	**35.16**	**38.81**	**47.62**	**55.73**	66.00
FT+RandAugment	**24.77**	**35.16**	**38.81**	**47.62**	**55.73**	66.00
FT+AugMix	**24.77**	**35.16**	**38.81**	**47.62**	**55.73**	66.00

Table 14 reports energy consumption during the DA and finetuning phases across different shot counts for TALON's UC4 dataset. As with other datasets, vanilla FT and custom DA strategies generally consume less energy than automated methods.

In this dataset, energy usage increases with more shots for automated DA approaches, while the trend is less consistent for custom methods. Notably, FT+DA(1) sometimes leads to lower energy consumption than vanilla FT, likely due to faster convergence enabled by improved sample diversity in data-scarce conditions. FT+DA(1&2) is consistently the least energy-efficient strategy in this dataset, owing to the added cost of combining multiple augmentation operations during training.

Table 14: Total energy consumption in Wh during training for different training strategies and number of shots in the image-based FSL scenario using TALON's UC4 data

Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
FT(baseline)	22.64	**24.87**	28.25	27.78	27.96	46.49
FT+DA(1)	**22.09**	25.32	37.76	**15.62**	**14.43**	**23.00**
FT+DA(2)	27.60	26.15	**25.94**	34.10	41.10	130.27
FT+DA(1&2)	49.24	85.89	111.09	149.71	540.91	866.57
FT+AutoAugment	31.88	32.88	33.27	46.91	47.72	44.61
FT+RandAugment	31.75	32.65	32.73	47.38	47.60	43.89
FT+AugMix	32.20	32.83	32.79	47.83	46.93	44.42

Table 15 presents EF values for each DA strategy across shot counts in TALON's UC4 dataset. While automated DA methods achieve the highest AP_{50} scores, their increased energy usage, particularly in higher-shot settings, leads to only moderate increases in the EF values beyond the 2-shot case. In contrast, custom strategies and vanilla FT maintain better EF performance at higher shot counts, primarily due to their lower energy demands.

However, when compared to the mEF values in Table 16, a different pattern emerges. Because mEF down-weights energy consumption using a logarithmic scale, it favors automated DA strategies in most scenarios, reflecting their superior accuracy despite higher energy costs.

Table 15: Efficiency factor (EF) metric values for different training strategies and number of shots in the image-based FSL scenario using TALON's UC4 data

Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
FT(baseline)	0.50	0.65	0.72	0.90	1.11	1.07
FT+DA(1)	0.43	0.57	0.53	**1.36**	**1.94**	**1.88**
FT+DA(2)	0.70	1.07	**1.54**	1.26	1.32	0.55
FT+DA(1&2)	0.23	0.18	0.28	0.23	0.09	0.08
FT+AutoAugment	0.75	**1.08**	1.15	1.02	1.16	1.45
FT+RandAugment	**0.76**	**1.08**	1.17	1.01	1.17	1.47
FT+AugMix	0.75	1.07	1.16	1.00	1.18	1.45

Table 16: Modified efficiency factor (mEF) metric values for different training strategies and number of shots in the image-based FSL scenario using TALON's UC4 data

Model Variant	Shot=1	Shot=2	Shot=3	Shot=5	Shot=10	Shot=30
FT(baseline)	2.82	3.98	4.67	5.58	7.34	9.85
FT+DA(1)	2.10	3.56	4.01	5.92	8.03	10.68
FT+DA(2)	4.61	6.79	**8.85**	9.69	**11.61**	11.30
FT+DA(1&2)	2.35	2.80	5.55	5.86	6.90	9.19
FT+AutoAugment	5.49	7.83	8.58	**9.81**	11.43	13.69
FT+RandAugment	**5.50**	**7.84**	8.60	9.79	11.44	**13.74**
FT+AugMix	5.48	7.82	8.60	9.77	11.47	13.71

Following the approach used in the object-based FSL analysis, Figure 5 illustrates the trade-off between AP_{50} performance and energy consumption for custom DA strategies and vanilla FT in TALON's UC4 dataset, along with the corresponding Pareto front.

In lower energy regimes, FT and FT+DA(1) offer the best performance-efficiency balance, particularly in mid- to high-shot scenarios, unlike in the object-based setting, where similar trade-offs were achieved with fewer shots. At higher energy levels, FT+DA(2) and FT+DA(1&2) dominate the Pareto front due to their ability to deliver stronger AP_{50} despite greater energy consumption.

These results emphasise that, in the image-based FSL setting for TALON's UC4 dataset, increasing the number of shots improves the overall balance between performance and efficiency. This underscores the difficulty of optimising both objectives under extreme data scarcity.

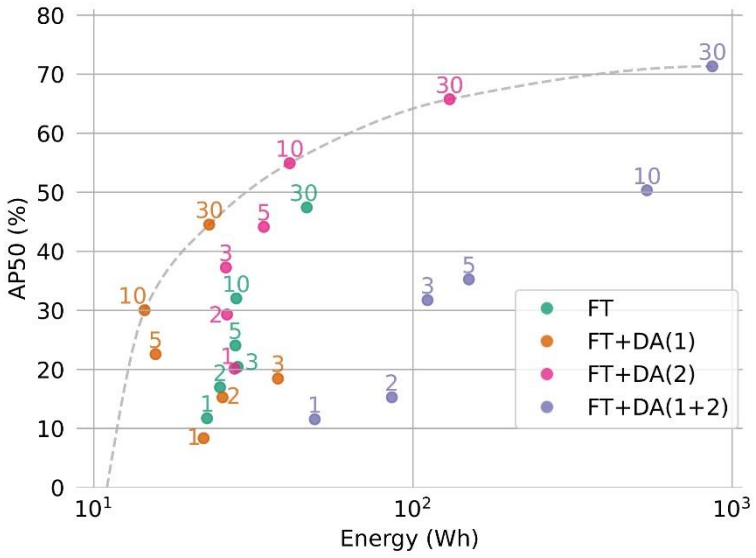

Figure 5: AP50 with respect to energy consumption for different DA strategies and numbers of shots in the image-based FSL setting using TALON's UC4 data

To better understand the energy consumption behavior of custom DA strategies in the image-based FSL setting, Figure 6 shows the percentage change in energy usage for FT+DA(1), FT+DA(2), and FT+AutoAugment across different shot counts in the dataset, relative to vanilla FT. While the primary

focus is on custom methods, FT+AutoAugment is included for comparison. The analysis highlights the variability in energy usage across different shots, with no consistent correlation between the number of shots and the energy consumption for custom DAs. FT+DA(1&2) is excluded due to its disproportionately high energy usage, which obscured meaningful interpretation in this comparative context.

As shown in Figure 6, FT+DA(2) leads to a substantial increase in energy consumption in the dataset, reaching up to 1600% in the 10- and 30-shot scenarios. This inefficiency is notable given that FT+DA(2) does not increase the number of training samples, yet negatively impacts convergence, requiring more training iterations. In contrast, FT+DA(1), despite generating more samples, often results in lower energy consumption. This counterintuitive outcome is especially apparent in this dataset, where severe data scarcity makes the additional synthetic samples valuable for enabling faster convergence. This reinforces the benefit of using FT+DA(1) in extremely data-scarce regimes. Finally, FT+AutoAugment typically consumes more energy than FT, with the exception of the 30-shot setting, where it results in a marginal reduction, likely due to early convergence effects.

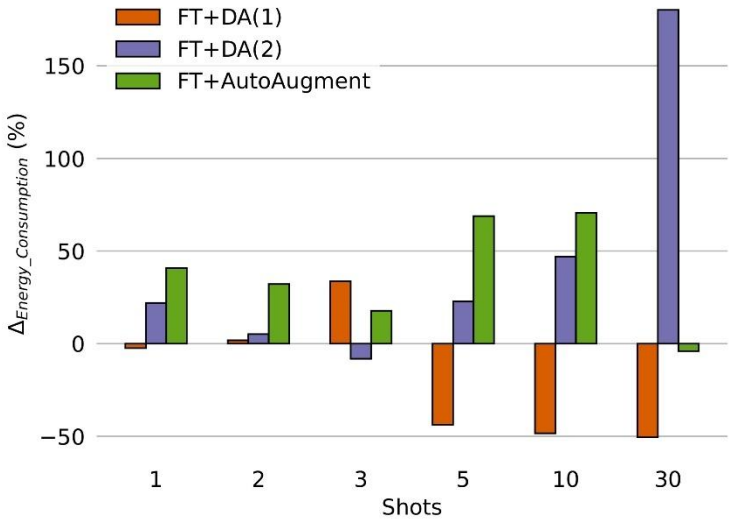

Figure 6: Energy consumption difference between different DA strategies and vanilla fine tuning for a varying number of shots in the image-based FSL setting using TALON's UC4 data

3.2 Semantic Enrichment and Alignment (UL)

3.2.1 Key Technologies & Technical Updates

The Data Enrichment module was developed to target the improvement of both structured and unstructured datasets across TALON pilots, with a focus on enhancing the contextual, temporal and environmental relevance of input features. It supports multiple data modalities, including time series from predictive maintenance applications and image-based inputs for visual safety compliance. While initial exploarions of the time series component have been reported in D4.1, significant enhancements concerning images in relation with pilot data have been made. The enrichment process operates in two parallel components:

1. **Time Series Enrichment for Failure Prediction**: Using the MetroPT[6] dataset which is collected in 2022 as part of a predictive maintenance initiative for an urban metro system in Porto, Portugal, the module focuses on enriching temporal data to improve failure prediction accuracy. After selecting the top 10 most relevant features using Recursive Feature Elimination (RFE), enrichment was performed via:
   - Initially, PCA was used to reduce dimensionality and highlight principal trends in time series data, providing a baseline for predictive modelling.
   - Feature selection via Recursive Feature Elimination (RFE): reduction to the 10 most informative features.
   - Timestamp decomposition: generation of granular features (year, month, week, day_of_week, hour, minute, second).
   - Environmental context integration: automatic retrieval of temperature and humidity via open APIs, linked to latitude/longitude.
   - Cyclical encoding: sine/cosine transformations for time-related features (hour, month), capturing periodicity.

2. **Image enrichment for visual safety use cases** (UC#4 – MINDS images):
   - Augmentation: geometric and photometric transformations (rotation, flipping, color shifts).
   - Caption generation: generative AI model producing descriptive captions of enriched fire-related images.
   - TALON UI integration: Flask-based service exposing user-friendly upload and augmentation options, with enriched outputs (augmented images and captions) accessible through the TALON interface.

This modular design ensures complementarity with other TALON preprocessing tools (such as curation) and enhances interoperability across use cases by providing an efficient enrichment framework.

3.2.2 Scientific and Technical Results

Two sets of benchmarks were conducted, each one related with the each task (tabular and image enrichments):

Failure Prediction: Using the enriched MetroPT dataset (RFE + enrichment features), predictive models achieved improved performance compared to baseline inputs as shown in Table 17 and Table 18:

Table 17: Classification performance before the enrichment

Model	Accuracy	F1-score
DT	98.14	79.06
RF	98.79	82.45
Adaboost	97.78	36.81
XGBoost	98.26	67.92
Catboost	98.29	66.71

[6] *https://zenodo.org/records/6854240*

LGBM	98.14	79.06

Table 18: Classification performance after the enrichment

Model	**Accuracy**	**F1-score**
DT	98.32	92.87
RF	98.5	95.01
Adaboost	98.27	75.58
XGBoost	98.75	98.21
Catboost	99.03	98.91
LGBM	98.99	98.97

While model accuracies were already high before enrichment (≈97–99%), the corresponding F1-scores were much lower, especially for Adaboost (36.81%) and LGBM (57.51%), revealing difficulties in handling class imbalances and rare failure events. After applying the enrcihment (PCA+RFE+new features) to isolate the most relevant features and enriching them with temporal, environmental, and cyclical metadata, all models achieved a substantial increase in F1-scores, in some cases by more than +40% (e.g., LGBM reaching 98.97%). Accuracy remained consistently high, but the alignment with improved F1-scores demonstrates that the enriched features significantly enhanced the models' ability to generalize and detect failures reliably. This confirms that enrichment not only complements PCA but also provides critical contextual information that boosts predictive maintenance performance within TALON.

Fire Detection:

For the visual domain, a dedicated Flask-based enrichment application was developed to handle:

- Image augmentation: Applying transformations such as flipping, rotation, and color adjustments to improve model generalization.
- Caption generation: Leveraging a pre-trained generative model to automatically describe enriched images, with outputs stored in a JSON format. To note, the application uses structured folders (uploads, augmented) and exposes a web interface for usability and pipeline management (Figure 7).

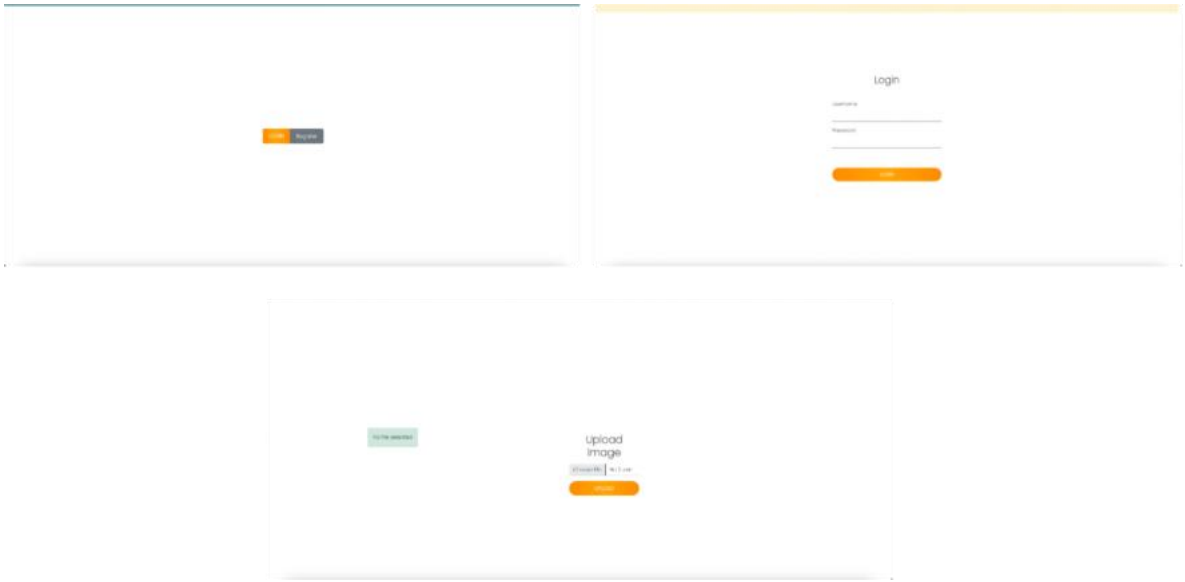

Figure 7: Image Enrichment application

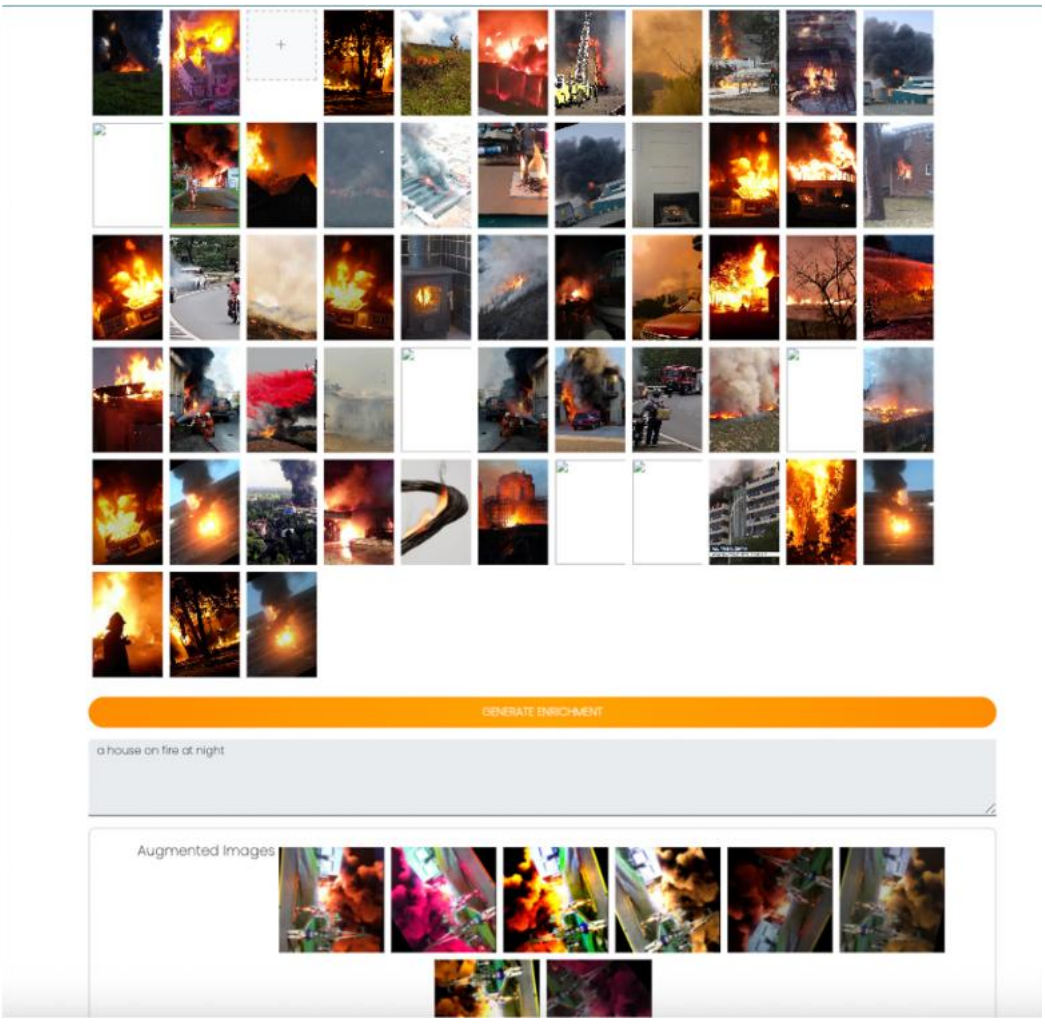

Figure 8: Augmented image instances with captions

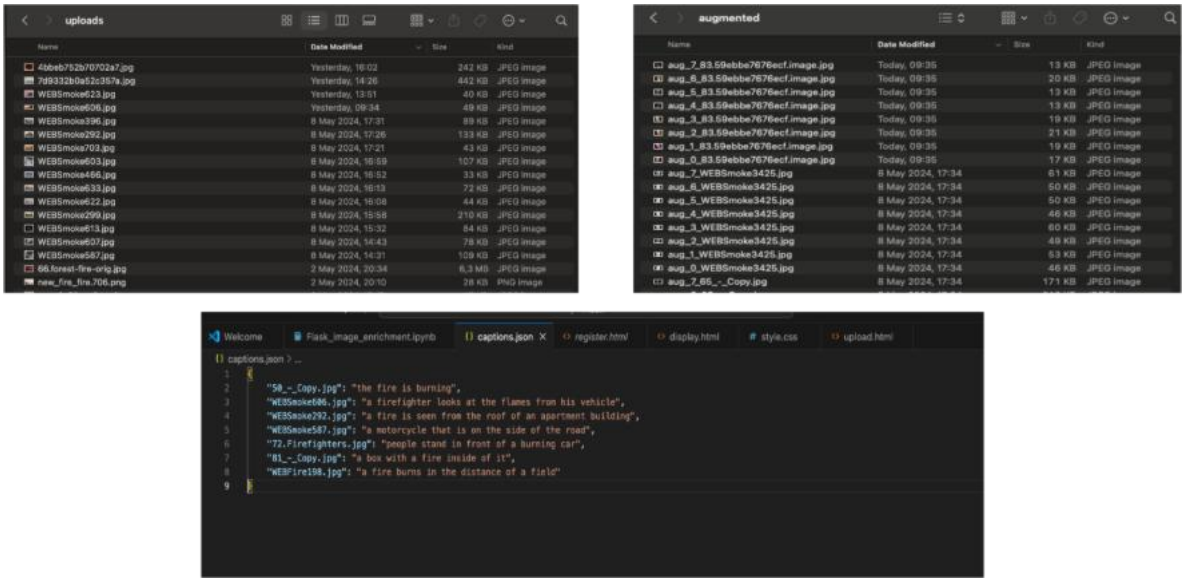

Figure 9: File management & Data Storage system representation

3.2.3 *Instantiation with pilot data*

For the UC#4 which provides the Fire Images dataset, the enrichment module was validated through a combination of image augmentation and caption generation. As shown in the figure, the pipeline processes raw fire/smoke images uploaded into the system (left), automatically generates multiple augmented variants (right) and stores the outputs in a dedicated directory structure for downstream training. This augmentation step diversifies the dataset by applying transformations such as flipping, rotation and color adjustments, thereby improving the robustness and generalisation of fire detection models. In parallel, the module produces descriptive captions for each image, stored in a JSON file (bottom). These captions enhance dataset interpretability by linking visual patterns with semantic descriptions (e.g., “the fire is burning”, “a firefighter looks at the flames from his vehicle”, “a fire burns in the distance of a field”). While some captions may still be simplistic or contextually limited, the overall approach provides both quantitative (augmented image diversity) and qualitative (semantic annotations) enrichment. This dual functionality makes the tool valuable not only for training computer vision models but also for building multimodal AI systems that leverage both visual and textual cues for fire detection and safety monitoring.

3.3 Data Curation (ENG)

3.3.1 *Key Technologies & Technical Updates*

The data curation module has been purposefully developed to tackle a wide range of preprocessing challenges within the TALON ecosystem. At its core, the module features a robust and extensible catalogue of curation algorithms, systematically organised in a hierarchical, tree-like structure. This design enables intuitive and dynamic navigation through the available algorithms, starting from a root node or entry point typically recommended by the Explainable AI (XAI) module. From there, users are guided through a branching path of algorithmic options, each tailored to specific data issues or objectives.

User inputs play a pivotal role in this process, allowing for interactive refinement and configuration of algorithm parameters. This ensures that the selected preprocessing steps are not only contextually relevant but also optimized to meet the specific requirements of the downstream tasks. Ultimately, the curation module streamlines and enhances the preprocessing pipeline, making it more intelligent, adaptable, and responsive to the evolving needs of data-centric workflows within TALON.

The data curation module includes a suite of algorithms designed to enhance data quality across various dataset types. These algorithms are grouped based on the nature of the dataset they operate on.

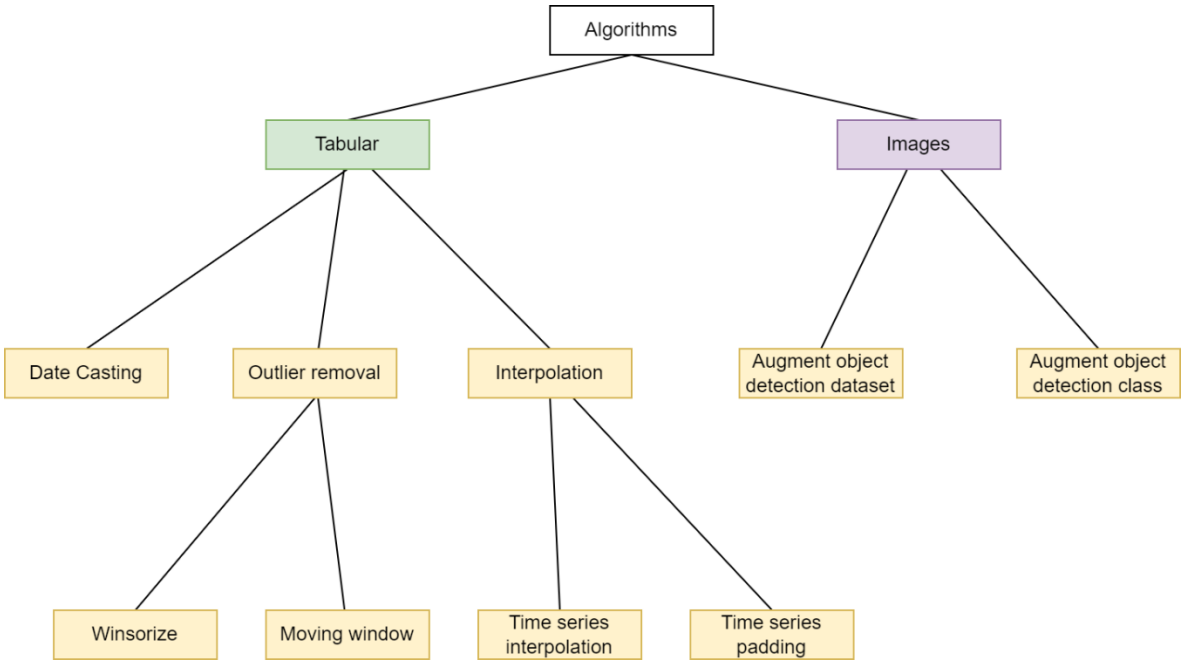

Figure 10: Hierarchical structure of Data Curation Core

Applicable on **tabular dataset**:

Date casting algorithm: This algorithm standardises date formats by converting date fields from an initial format to a specified target format. It simplifies downstream processing by ensuring consistent date representations across the dataset.

Winsorize: A robust outlier mitigation technique that limits extreme values in the data while preserving distribution characteristics. It is implemented using the Spicy Stats[7] function and operates by analyzing the signal distribution to determine appropriate clipping thresholds.

Moving window: This algorithm removes statistical outliers in time series data using a rolling Z-score approach. After normalizing the data, Z-scores are computed within a sliding window, and any values exceeding a user-defined threshold are flagged and removed. This method is well-suited for detecting context-aware anomalies over time.

[7] scipy.stats.mstats.winsorize

Time series interpolation: Implemented as a wrapper around the Pandas DataFrame[8] method, this algorithm enables flexible interpolation of missing or irregular time series values. Users can choose from various interpolation strategies provided by Pandas, such as linear, nearest-neighbour, or polynomial interpolation.

Applicable on **object detection datasets**:

Data augmentation: This algorithm applies a range of image augmentation Transformations[9] to entire folders or subfolders while preserving (or transforming) the associated labeling information. It leverages the Albumentations[10] library to perform geometric and photometric transformations, such as flipping, rotation, scaling, and more.

Balance classes: An extension of the data augmentation capability, this algorithm selectively augments images belonging to a specific class. Users can specify a target class and define either the desired number of augmented samples or the percentage of augmentation relative to the original class size. This helps address class imbalance in training data without over-augmenting already dominant categories.

3.3.2 Scientific and Technical Results

Here the results from two algorithms will be presented. The first example shows the application of the class augmentation algorithm on a single image (for demo purposes). From one image four new ones are generated, applying a series of transforms associated with a probability. For an extensive list of transforms check the Albumentations docs[11]. Thanks to a series of configurations, it is possible to augment classes, by a certain number of images. Performing an augmentation on a dataset that is going to be used to train an object detection model helps in increasing robustness of the model and making it properly generalized, avoiding overfitting.

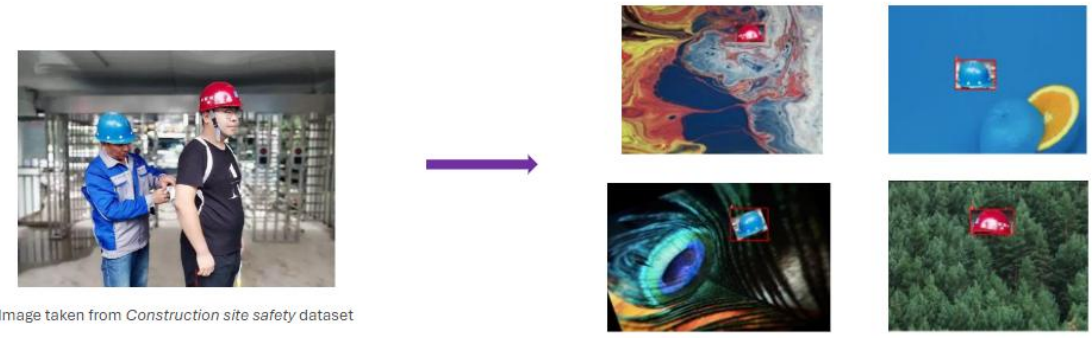

Image taken from *Construction site safety* dataset

Figure 11: Example of class augmentation

The second example shows how a curation algorithm can be used for recovering missing values in a time series. More specifically the plots below show the application of a linear regressor curation

[8] pandas.DataFrame.interpolate
[9] transformations

[10] Albumentations
[11] https://albumentations.ai/docs/

algorithm to replace null values. The Delhi climate dataset has been modified, removing values that have been restored using curation module. As before a series of parameters can be used by the advanced user to fine tune the algorithm; otherwise default values can be used to simplify the process and still get results.

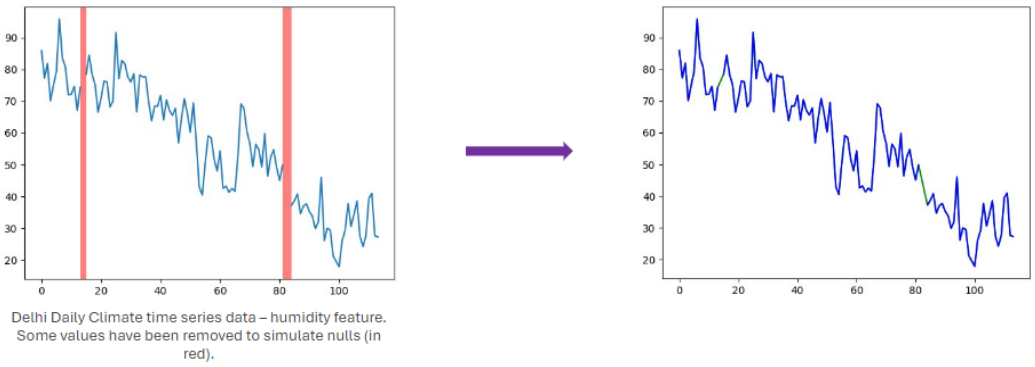

Figure 12: Linear interpolation on time series

As can be seen from the plot on the left, some values are missing (columns in red). On the right plot those values are restored using curation algorithms. Specifically, a linear interpolation algorithm has been used.

In order to provide metrics to the TALON smart orchestrator, some metrics have also been collected regarding the two services. All the metrics have been calculated using a server with a 13th Gen Intel(R) Core(TM) i5-13500 CPU. Measures have been taken 5 times and averaged, to mitigate noise in the results.

Table 19: As we can clearly see, padding is 3 times faster than linear interpolation, but its mean absolute error is greater.

Task	Name	Dataset	Mae	Execution time (s)
Time series interpolation	Linear interpolation	Delhi time series	0.021	0.001240
Time series interpolation	Padding	Delhi time series	0.028	0.000410

3.3.3 Instantiation with pilot data

The curation algorithms have been evaluated using pilot datasets, with a particular focus on Factor's time series data derived from Nakamura Machines. These datasets provided a valuable benchmark for testing and validating time series–specific algorithms within the module.

The curation process for Factor's data follows a two-step pipeline:

- **Date Formatting**: Raw date values are standardized using the date casting algorithm, ensuring consistent formatting across the dataset.

- **Missing Value Handling**: Null or missing values in the time series are then addressed using the time series interpolation algorithm. This step is essential to prepare the data for downstream tasks, such as modelling or anomaly detection.

The image below illustrates the results after applying the curation process. It also highlights how different interpolation strategies can yield distinct outcomes.

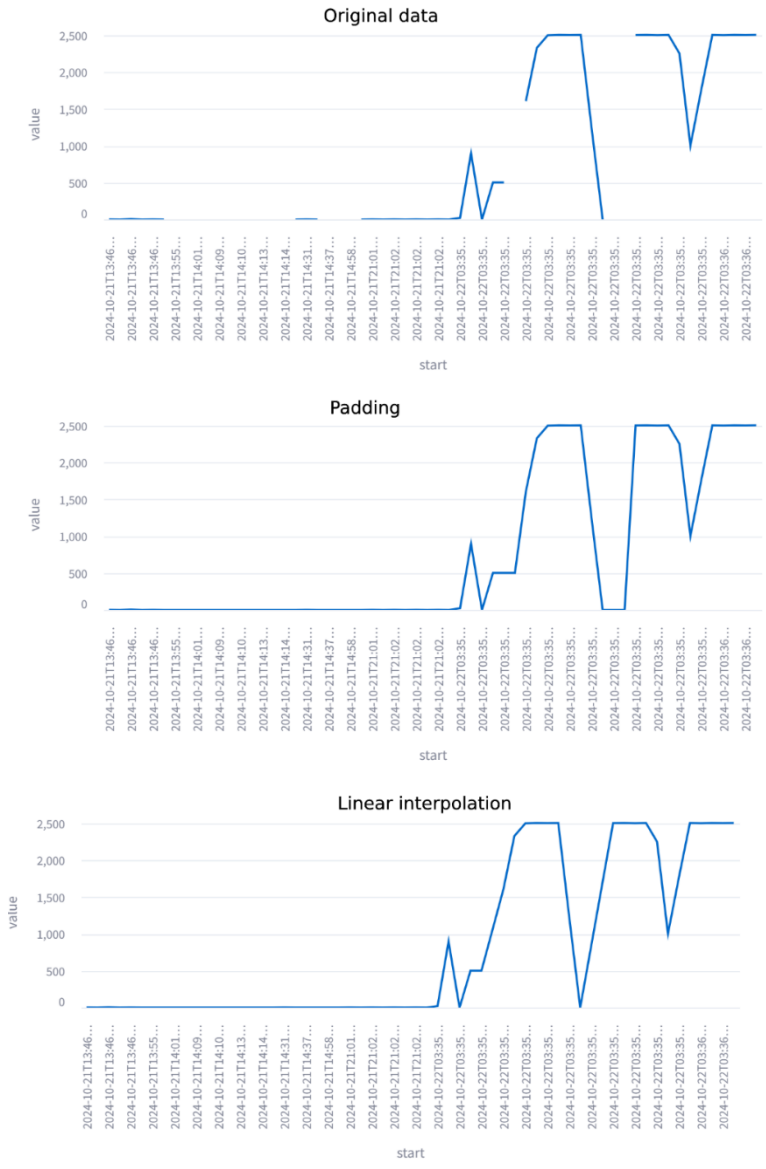

Figure 13: Curation results

While simple techniques like padding offer faster performance, they tend to produce lower-quality results compared to more sophisticated interpolation methods. Fortunately, the interpolation algorithm supports a wide range of strategies - such as linear, polynomial, and nearest-neighbor interpolation - all of which are already implemented and can be configured based on the specific needs of the dataset.

3.4 Self-healing and Self-correcting

3.4.1 Key Technologies & Technical Updates

In precision machining environments, tool degradation has a direct impact on process reliability and productivity. CNC lathes utilise multiple cutting tools that wear out over time and must be replaced to ensure dimensional accuracy and prevent unexpected failures or defects. Currently, the timing of tool replacement is based on fixed usage thresholds or operator experience, resulting in suboptimal usage, increased downtime and cost overruns.

To improve this process, a data-driven approach has been developed to estimate the Remaining Useful Life (RUL) of each tool using time series forecasting techniques. This approach supports informed decision-making on when a tool is likely to reach its wear limit, allowing planned replacements that maximise usage without compromising quality. The solution explores few-shot and hybrid learning strategies to build models that operate effectively with minimal training data and reduce computational demand.

Machine data collected from CNC systems, including servo load signals, override parameters, and tool change records, is processed to construct multivariate time series. The objective is to forecast the future evolution of sensor signals related to tool usage to estimate the lifetime remaining before tool degradation reaches a critical level. This constitutes a survival-type analysis, where the goal is not to detect failure after it occurs, but to anticipate it based on how key indicators are expected to evolve. By predicting this evolution, it is possible to estimate when a tool is likely to require replacement, allowing better planning and avoiding premature or reactive interventions.

3.4.2 Scientific and Technical Results

Figure 14 summarises the complete process carried out, structured in three main phases: the migration of the database, the pre-processing of the data and the development of the predictive model.

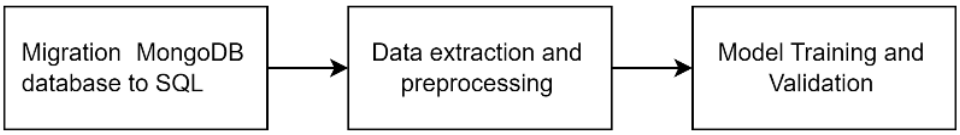

Figure 14: End-to-end workflow.

Data collecting

FACTOR shop floor has many CNC lathes, which can have different signals and equipment. As a starting point, the pilot had a *MongoDB* system in which data coming from the lathe machines was stored.

To define a scalable and organised data structure, a relational database following the ISA 95[12] standard was modelled and deployed in Factor. The technology used for this was PostgreSQL[13], utilising TIMESCALEDB [14]for time series data generated by sensors.

[12] *https://www.isa.org/products/ansi-isa-95-00-01-2025-iec-62264-1-mod-enterprise*
[13] *https://www.postgresql.org*
[14] *https://github.com/timescale/timescaledb*

Aside from the benefits mentioned above, using TIMESCALEDB and a relational schema allows more efficient and complex data queries, being able to use joins between entities or group by in a more robust and standardised manner.

Moreover, *PostgreSQL* has a mature ecosystem to perform reporting or data analysis, using tools like *Grafana* [15] or *Power BI*, which Factor can use to do these types of tasks.

The data model used has the following structure in Figure 15 (showing the most relevant entities):

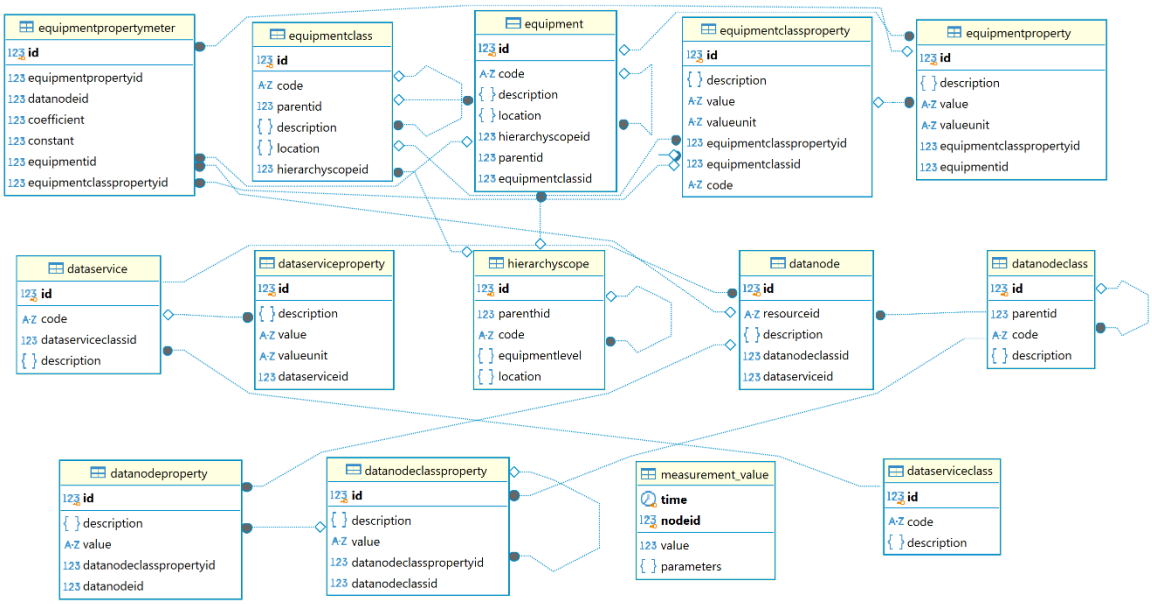

Figure 15. Relational schema of the SQL database resulting from the MongoDB migration. Source: Own elaboration.

Three different segments of data can be distinguished: data related to specific shop floor machines from which signals are being measured, data related to the signals and services that provide these data, and the actual signal measurements that the algorithm will use.

Now, the entities related to each of the groups mentioned above and what type of data they manage will be explained:

- **Shop floor machines data:** Based on ISA-95 (IEC 62264, 2013), manufacturing operations are organised around roles and responsibilities to manage resources, coordinate activities, and ensure performance. The entity hierarchy scope facilitates defining those roles. Four levels are distinguished: Enterprise, Site, Area, Work centre and Work unit.

In the *equipmentclass* entity, all the different classes of equipment are established, specifying "parent" relationships between them and their hierarchy scope level.

The equipment entity allows for defining the actual pieces of equipment in the shop floor, associating to each of them an equipment class and determining hierarchical relationships between them.

[15] *https://grafana.com*

The *equipmentclassproperty* entity defines the properties of each equipmentclass, specifying their measurement units and code. In Factor case, some of these are: Servo load, Cut time, Tool code or Servo current.

The *equipmentpropertymeter* entity links each *equipmentclassproperty* with a data node (signal) and a specific equipment.

- **Signals and services data:** These entities define what signals will be measured and which services will provide the information.

The entity *dataservice* defines the different services that will be used to get signals from the equipment. These dataservices can also have a class, which are defined in the *dataserviceclass* entity.

Datanodeclass defines the types of data nodes that will be measured, for example: short integer type, floating point type, byte type, etc.

Datanodeclassproperty allows to define properties to the different datanodeclass, for example, for a MongoDB node class, properties collection data, signal value type and node description were defined.

The *datanode* entity contains the different signals coming from machines, identifying them with a resource id and specifying their data node class and data service.

The *datanodeproperty* entity maps the data node class properties of each data node and specifies their value.

- **Signal measurements:** This data is stored in the measurement value entity. It consists of time series data coming from sensors. The different node id values are stored with their corresponding time stamp.

This data model that follows the ISA-95 standard allows for a flexible, scalable and consistent modelling of the floor shop, being able to include as many different machines as possible and their signals. Data analysis and further possible processing and use of the data is facilitated by the relational structure of the model.

After the database structure was determined and deployed, the different types of equipment, equipment classes, data services and signals were collected for the Factor case. Each signal needed to be linked with specific equipment. After this process was completed, all this information was inserted into the database.

The service that provided the data coming from the studied machine was MongoDB.

This technology stores data in a BSON (binary JSON) format. These collections have several key-value pairs (Figure 16).

```
_id: ObjectId('67dd1f9f3c426b4218edc0a7')
L1Name : "Nakamura2"
updatedate : 2025-03-21T08:12:15.500+00:00
enddate : 2025-03-21T08:12:16.500+00:00
timespan : 1
signalname : "ServoLoad_0_path1_Nakamura2"
value : 31
filter : null
TypeID : null
Judge : null
Error : null
Warning : null
```


Figure 16. Example of a single time-series record from the original MongoDB dataset. Source: Own elaboration.

The *Node-red* service filters the data, only inserting the signals that were useful for the algorithm and performs some small data transformation when it is needed for inserting data into the measurement value entity of the PostgreSQL database.

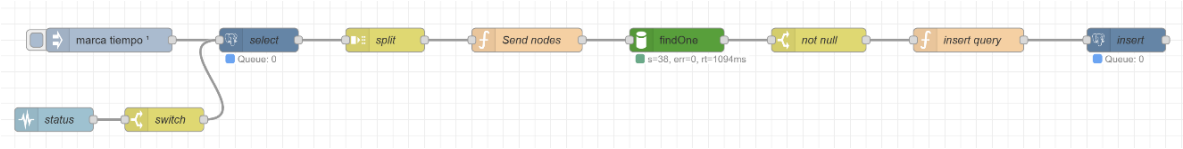

Figure 17: Node-RED flow for automating data retrieval, filtering and insertion between databases. Source: Own elaboration.

Figure 17Figure 17 shows the flow used to transfer data from MongoDB to PostgreSQL. It will be explained in detail:

- First the status node sends the status of the insert node to the switch node. The switch node allows the flow to continue only when the insert node has no queue. This is done to avoid saturation and errors.
- Then the select node checks the last values inserted of each signal in the PostgreSQL database and sends all the data needed to query the *MongoDB* database with the information needed (last update time, signal names, collection name and manufacturing resource name)
- The send node's function builds the query and sends it to the findOne node, which is a *MongoDB* node that returns a message for the input query.
- Then, if the query result is not null, the output is minimally pre-processed for some specific signals and then the insert query for the *PostgreSQL* is built.
- This query is passed to the insert node, which executes the insert into with the given parameters to the measurement value entity.

The main technologies used for preparing the data that the model would use are MongoDB, Node-RED and PostgreSQL, also, the ISA-95 standard was used. These were mentioned above. In this section, a brief explanation of them and the specific functionalities used in this case will be provided:

- **MongoDB:** It is a NoSQL database, document-oriented, that stores data in BSON format (a binary extension of JSON). It is useful to store great quantities of data in a fast but unstructured way. In this case, MongoDB was initially used to store signal data coming from CNC lathes. Its flexibility allowed a fast data collection, but it presented limitations for more complex analysis and modelling of shop floor manufacturing resources.
- **Node-RED:** It is a flow-based development tool that allows connectivity between hardware devices, databases, web services and other APIs simply and visually. It is based on Node.js, which allows running JavaScript code efficiently. Its event-based architecture makes it useful for real time applications and industrial automation systems. In this case, it was used in data migration as an intermediary, connecting MongoDB with PostgreSQL. Node-RED allows to:
    - Query current and historical data from MongoDB.
    - Carry a part of the data processing needed for the algorithm.
    - Inserting processed data into a relational database.
- **PostgreSQL and TimescaleDB:** PostgreSQL is an open source relational database management system, known for its sturdiness and SQL standard compliance.

TimescaleDB is a PostgreSQL extension specifically designed to manage time series data such as sensor reads. It accomplishes this by optimising insertion, compression and query of high volumes of data, which are associated with time stamps. Some of its advantages are:

- Hypertables, which divide data into different time partitions.
- Data compression, which reduces required storage significantly.
- Advanced time queries, such as down-sampling or time-based aggregations.

- **ISA-95:** It is an international standard developed by the International Society of Automation for enterprise control and information systems integration. Its main objective is to define a formal structure of functional levels within an industrial plant, from management to the physical devices level. It defines a hierarchy divided into five levels:
    - Level 0-1: Physical equipment and sensors.
    - Level 2: Process control (PLC, SCADA)
    - Level 3: Manufacturing operations management
    - Level 4: Enterprise management

Besides, this standard defines an object model for resources like equipment, materials, operations, personnel and the relationships between these. It allows clear, standardized and scalable data model design, which is essential in multiple machines, signals and complex industrial processes environments.

3.4.3 Instantiation with pilot data

Data Processing

Besides, this standard defines an object model for resources like equipment, materials, operations, personnel and the relationships between these. It allows clear, standardized and scalable data model design, which is essential in multiple machines, signals and complex industrial processes environments.

Introduction

This section provides an overview of the data processing workflow applied to the dataset collected for tool wear prediction in Pilot 2. The dataset contains time-series signals recorded during Nakamura 2 (CNC machine) operations and supports predictive maintenance through AI models. The raw data originates from Factor's monitoring infrastructure and captures relevant operational variables. The section describes the data source, extraction method and preprocessing steps used to convert raw signals into structured input for model training and evaluation. It also highlights key data preparation challenges and the measures implemented to ensure data quality and consistency.

Overview of the Collected Dataset

The dataset collected from the Nakamura 2 contains time-series data representing the machine's operational status during machining cycles. It is used to support predictive maintenance by providing a detailed view of the tool's behaviour and process parameters.

The main concepts used in the dataset are as follows:

- **time**: Timestamp of the recorded data entry.
- **Override_path_ {1,2} _Nakamura2**: Percentage of the programmed feed rate being applied in each path (1 or 2). Values above 100% indicate faster operation than nominal; values below 100% indicate slower speeds.
- **ServoLoad_ {0-3} _path {1,2} _Nakamura2**: Load on each servo motor (0,1,2 or 3), expressed as a percentage of rated current, for each direction and path(1 or 2).
- **Reference**: Product identifier.
- **t_ciclo**: Duration of the process cycle in seconds.
- **(0-44) → Tool**: Tool life expressed as a percentage.

Timespan

The timespan of available data is 5 months.

Purpose and Scope

The dataset collected contains time-series data describing machining operations in a high-precision manufacturing context. The purpose is to provide AI models with relevant process information to enable accurate prediction of tool wear. This data supports the development of predictive maintenance strategies aimed at optimising tool usage and minimising unplanned downtime. Accurate estimation of tool wear allows postponing replacement until the optimal point, avoiding premature changes of high-cost tools and reducing the frequency of machine stops. This contributes to lowering operational costs and increasing Overall Equipment Effectiveness (OEE) by maximising availability and resource utilisation.

Sources of Data

The data used in this study originates from the operational monitoring system of the Nakamura. Initially stored in a NoSQL database (MongoDB), the data was migrated to a relational SQL database to align with ISA-95 standards and to facilitate integration with the structured data model used in the project. The migration and data capture process were carried out using Node-RED and storage in Timescale database, optimal for the native hyperfunctions and hyperparameters it provides for working with time series. The dataset used for model training and analysis was extracted directly from this SQL database.

Data Extraction

The data is extracted from the TIMESCALE database using structured SQL queries. These queries are executed on the relational database that was populated during the migration from MongoDB, and which follows a schema aligned with ISA-95 standards. The extraction process retrieves fields such as timestamp, override percentages, servo loads, product reference, cycle time, and tool wear level.

The query returns the following fields:

Column Name	Description	Data Type	Example
time	Timestamp of the recorded data entry.	datetime	2024-11-04 06:00:00+00:00
Override_path {1,2} _Nakamura2	Feed rate override percentage applied to each path.	float	100
ServoLoad_ {0–3} _path{1,2} _Nakamura2	Servo motor load as % of rated current, for axes 0 to 3 in both paths.	float	24.8
Reference	Identifier assigned to each product.	string	1360350132A
t_ciclo	Duration of the machining cycle.	float	212
(0–44) → Tool	Remaining tool life, expressed as a percentage (0 to 1 scale).	float	0.73

The extracted data is used in:

- **Training**: For model training, a limited amount of historical data is used, following a few-shot learning approach. The datasets are obtained from exports of the SQL database and processed in an offline environment to avoid interference with the production system.

The queries used are custom designed for the project's structure and are executed within a controlled development environment.

Data Transformation

The raw time-series data was initially stored in two different SQL databases and exported as separate CSV files. These files were merged using Python scripts. The dataset includes sensor values and tool wear indicators from the Nakamura 2 CNC machine.

The transformation process included the following steps:

- **Merging and cleaning**: Two raw input files were concatenated and cleaned by dropping irrelevant columns. The first was a CSV export from TimescaleDB, containing all sensor values recorded from the Nakamura 2 machine. The second was a dataset extracted from a SQL database, which included manually entered records by operators indicating tool changes, along with associated product references and launch timestamps. Irrelevant columns were removed to retain only the variables required for modelling.
- **Pivoting and resampling**: The raw sensor and target data were pivoted to structure the dataset with one column per signal and tool identifier. This allowed each servo load or override measurement to be clearly distinguished by axis and path, and each tool life signal (labelled 0 to 44) to occupy its column. The time column was used as an index to enable temporal alignment. Once pivoted, the dataset was resampled to hourly intervals to obtain a

consistent and unified time index across all signals. This step ensures proper temporal alignment between sensor values and tool wear labels, facilitating the subsequent merging and interpolation processes.

- **Target preprocessing**: In the preprocessing of the targets, the data relating to tool changes were initially represented by a categorical variable with three values: 0 for scheduled changes, 1 for wear changes and 2 for breakage changes, which indicated the specific reason for replacement. To simplify the problem and facilitate the development of more effective predictive models, feature engineering techniques were applied. The first transformation consisted of converting this variable into a binary: 0 for scheduled changes and 1 for unscheduled changes (grouping wear and breakage). Subsequently, the problem was reformulated as a survival analysis, assuming that at each change, whether planned or unplanned, the tool has completely exhausted its useful life. Thus, each change was labelled as 1 (indicating 0 % remaining life), and the instant after replacement was labelled as 0 (corresponding to 100 % remaining life). This binary approach allowed accurate segmentation of wear cycles and served as the basis for interpolating a continuous degradation curve from 1 to 0 over the life cycle of each tool. In addition, it is important to be aware of batches of machining parts where each is referenced by a launch date, which translates into a change of all tools, regardless of their life. So, in each launch date, the lifetime of each tool is reset to 0.
- **Interpolation**: Missing values were filled using different interpolation strategies depending on the nature of the signal. For tool wear indicators, time-based interpolation was used to generate a smooth progression from 1 (new tool) to 0 (end of life), enabling the estimation of remaining useful life over time. For sensor variables such as Override and ServoLoad, a forward-fill strategy was applied to propagate the last observed value until the next update, preserving the physical meaning of the signals in periods of sparse data.
- **Export**: The final pre-processed dataset was saved as a CSV file and used for model training and visual evaluation to verify the correctness of the preprocessing pipeline.

This process ensured the generation of a coherent, structured dataset and ready for downstream modelling tasks.

Data Issues

The dataset used for predictive modelling consists of time-series signals collected from the Nakamura 2, as we discussed before. While sensor data was recorded automatically at regular intervals, the tool change information was manually entered by production managers, introducing certain inconsistencies and limitations into the dataset.

The figures below illustrate the **degradation curves** for four selected tools, showing how the tool life evolves for different product references (5,8,10,14). Each curve represents the interpolated life cycle of a specific tool, with a value of 1 indicating a new tool and 0 corresponding to complete wear or breakage. The coloured lines distinguish between the product references being machined.

Several issues were identified during exploratory analysis:

- **Class imbalance and data sparsity**: A significant portion of the recorded time series contains missing values, particularly in the tool wear labels. In many cases, the number of actual tool change events is very low, leading to highly imbalanced sequences that are difficult to use for training.

- **Reference heterogeneity**: As observed in the plots, some product references appear very infrequently or not at all for certain tools. In those cases, either no wear was recorded or the tool was not used, resulting in flat or incomplete degradation curves.
- **Tool lifecycle variability**: The tool usage patterns differ substantially across product references and periods. This heterogeneity complicates the learning process, as the model must generalise across non-uniform usage conditions.

The following plots illustrate the interpolated life cycle of four representative tools across different product references.

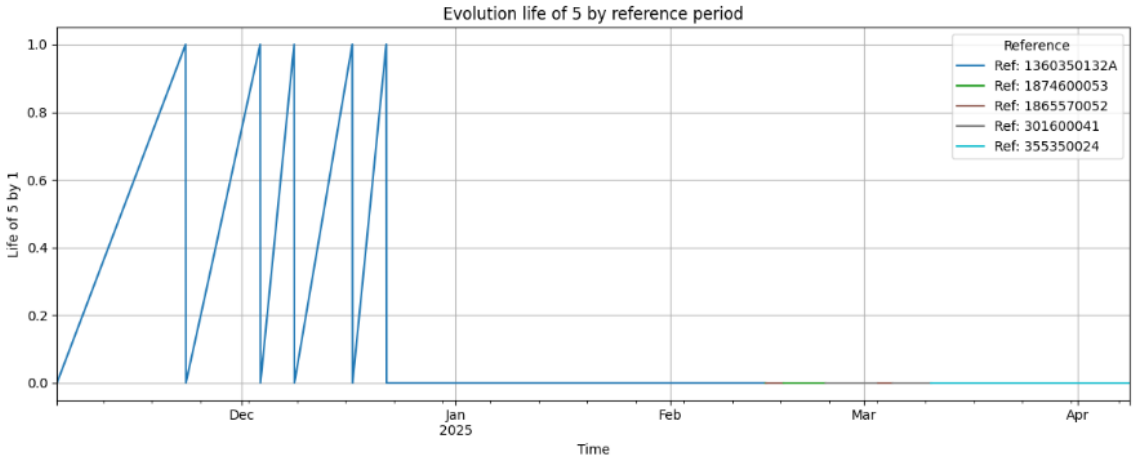

Figure 18: Life evolution of component '5' over time for different reference IDs. Source: Own elaboration.

Figure 18 (Tool 5) shows a high number of complete wear cycles, particularly associated with reference 1360350132A, which dominates the usage. After January 2025, tool usage declines, and no new degradation patterns are observed. The frequent cycling indicates regular replacement and makes this signal suitable for training. However, the period in which no changes occur must be eliminated for the model to function correctly.

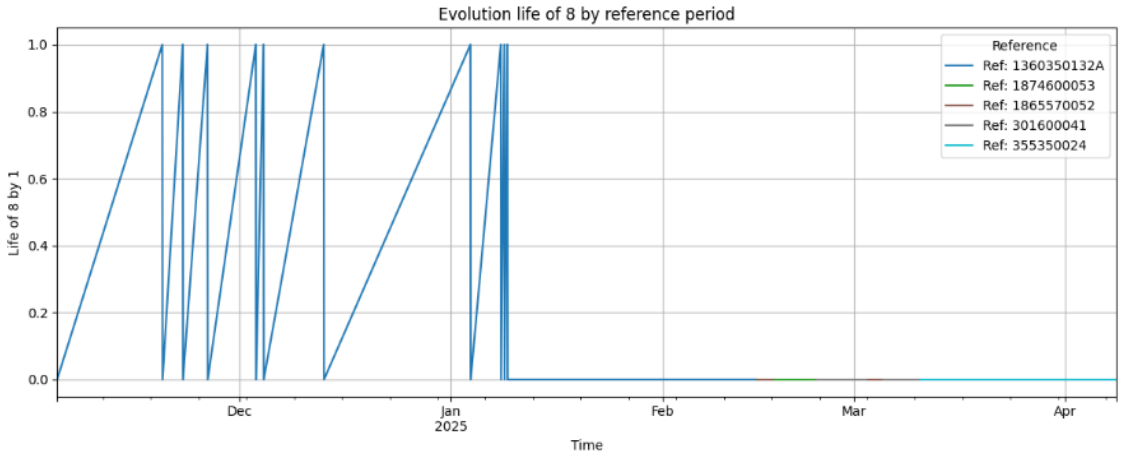

Figure 19: Life evolution of component '8' over time for different reference IDs. Source: Own elaboration.

Figure 19 (Tool 8) also presents multiple clear degradation cycles, mainly for the same reference as Tool 5. However, the spacing between replacements is more variable, which could reflect differences

in machining time or process conditions. Minimal activity is observed from February onwards, so it is excluded.

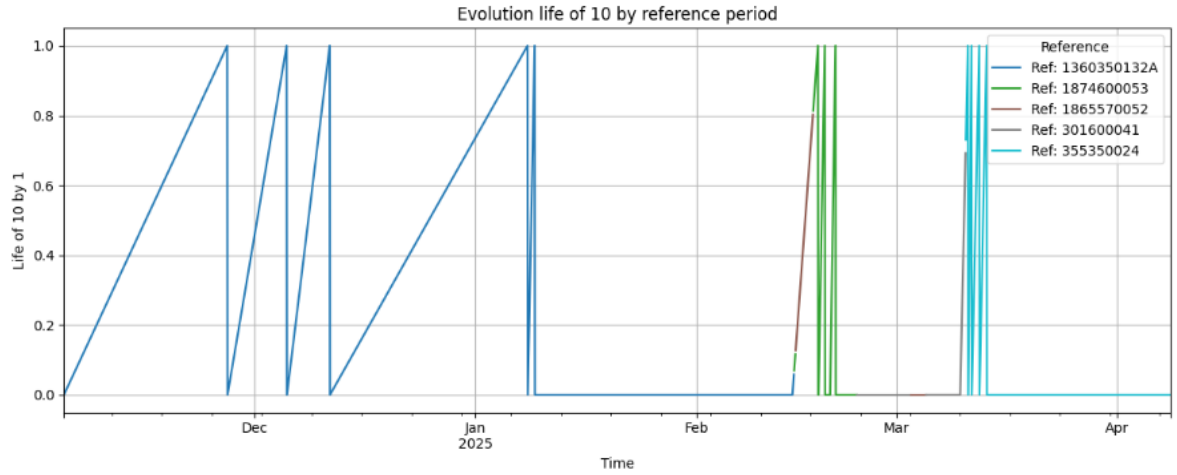

Figure 20: Life evolution of component '10' over time for different reference IDs. Source: Own elaboration.

Unlike previous cases, Tool 10 shows heterogeneous behaviour across references (Figure 20) Several degradation cycles appear for 1360350132A, but additional activity is also visible for other references such as 1874600053 and 355350024. These variations suggest cross-product usage and less consistent wear dynamics.

Finally, Tool 14 is dominated by 1360350132A, with many sharp degradation cycles early in the time window (Figure 21). From February onwards, there is no further wear recorded. This tool shows a clean, repetitive pattern, making it well-suited for survival modelling, but limited generalisation due to lack of reference diversity.

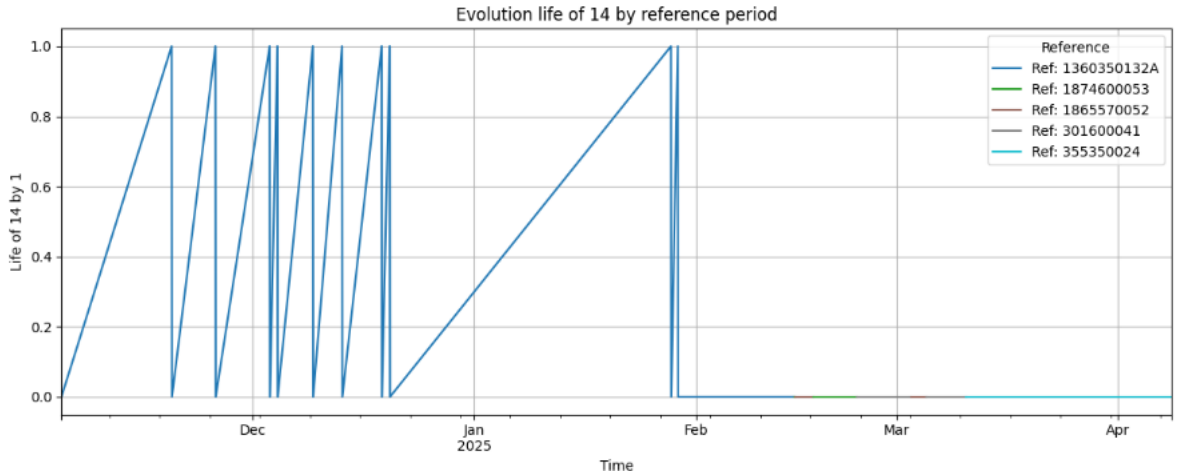

Figure 21: Life evolution of component '14' over time for different reference IDs. Source: Own elaboration.

To address these issues, the decision was made to focus the training data on a single product reference, 1360350132A, which contains a richer history of tool usage and multiple complete degradation cycles. This allowed the creation of a more balanced and informative dataset, improving the model's ability to learn meaningful survival patterns. Moreover, the analysis window was limited to periods where actual tool replacements occurred, ensuring that the model is exposed to relevant wear dynamics.

By refining the dataset in this way, the quality and consistency of the training data were significantly improved, enhancing the reliability of the predictive results.

Use Case: Predictive Maintenance Based on Tool Wear

Introduction

In manufacturing environments, unexpected tool failures during CNC machining operations can severely impact productivity, increase production costs, and reduce the overall equipment effectiveness (OEE). Despite the robustness of the machines, the limited predictability of tool life, especially under variable load and operational conditions, often results in either premature tool replacement or unplanned machine stoppages. Both situations negatively affect production continuity and resource efficiency. This use case addresses this challenge by applying predictive maintenance strategies focused on tool wear, using time-series data collected from the Nakamura 2 CNC machine to anticipate and optimise tool replacement events.

Approach

This use case leverages artificial intelligence to enable a shift from reactive or scheduled tool replacement to predictive maintenance in CNC machining. The objective is to anticipate tool wear and optimise replacement timing, thereby reducing unnecessary tool changes and avoiding unexpected failures. The approach begins by processing time-series data collected from the Nakamura 2.

Given the limited number of recorded tool changes, a few-shot learning strategy is adopted to maximise model performance with minimal data. The degradation curve of each tool is inferred by interpolating from labelled replacement events, enabling the model to learn the progressive wear behaviour over time. Feature engineering focuses on selecting relevant signals that capture operational stress on the tool, while model training targets the prediction of tool life as a continuous variable or classification of imminent replacement needs.

This predictive approach delivers significant benefits: by accurately estimating the remaining useful life of tools, companies can extend tool usage up to its actual wear limit, avoiding premature replacements. This translates into lower tooling costs, especially relevant for high-value tools, and reduced production downtime associated with manual interventions. As a result, the system contributes to improved equipment availability and overall equipment effectiveness (OEE), making production more cost-efficient and resilient.

Impact

This case directly contributes to Pilot 2's business strategy by reducing operational costs and increasing production efficiency. By avoiding premature tool replacements and minimising unexpected stoppages, the company reduces expenditure on high-cost tooling and improves machine availability. These gains translate into higher productivity with the same resources, enabling better delivery reliability and increased competitiveness. Furthermore, aligning maintenance operations with actual tool condition supports leaner planning and inventory strategies, reinforcing the company's shift towards data-driven manufacturing.

Model description

Model Selection

Deep learning models are known for their ability to capture complex patterns in large-scale datasets. However, in industrial contexts such as CNC tool wear prediction, collecting and labelling sufficient data is often costly, time-consuming, and sometimes unfeasible. This use case faces exactly that limitation: only a small number of annotated tool changes are available, making traditional supervised

learning approaches impractical. To address this challenge, a Few-Shot Learning (FSL) strategy was adopted to enable accurate predictions with minimal training data.

FSL offers an effective paradigm for generalising from limited examples, reducing both data acquisition and computational requirements. According to Wang et al. (2020) [17, 18], this can be achieved by exploiting prior knowledge through three complementary perspectives: data, by enriching supervision with contextual knowledge; model, by embedding priors that constrain the solution space; and algorithm, by optimising the learning process to converge more efficiently under data-scarce conditions.

In the context of time-series classification and prediction, few-shot models have shown promising results in industrial applications where signals are noisy, irregular, and multidimensional. Studies such as Pan et al.(2024) [18] and Mo et al., (2025) [19] highlight the relevance of FSL in Remaining Useful Life (RUL) prediction, where models must capture complex degradation trends from short sequences.

Approaches explored in [19] the literature include:

- Meta-learning models, such as Prototypical Networks or Memory-Augmented Neural Networks (MANNs), which aim to adapt quickly to new classes or operational conditions.
- Transformer-based models, which excel at modelling long-range temporal dependencies and perform well under few-shot regimes due to their high representational capacity.
- Transfer learning and pretraining, in which large-scale models are adapted to specific tasks using limited local data.

Given the characteristics of the available dataset in Pilot 2, limited labelled examples, temporal dependencies, and multivariate signals, the **Granite Time Series TTM-R2** model was selected as the main candidate. This transformer-based architecture is optimised for few-shot scenarios and supports multivariate forecasting tasks, aligning well with the predictive maintenance goals of this use case.

Selected Model: Granite Time Series TTM-R2

The Granite Time Series TTM-R2 model [20] is a transformer-based architecture designed specifically for multivariate time series forecasting under low-data regimes, making it well-suited for few-shot learning scenarios such as tool wear prediction in CNC machining.

At its core, TTM-R2 is a variant of the TinyTimeMixer (TTM) architecture, which combines efficiency and performance by leveraging channel mixing and temporal mixing operations to learn both intra-variable and temporal dependencies. Unlike traditional transformers, which rely on attention mechanisms with quadratic complexity, TTM achieves comparable performance with significantly lower computational cost-making it ideal for industrial edge applications where resources may be limited.

Key Features:

- Multivariate input support: TTM-R2 is optimised to process multiple sensor signals in parallel, such as servo loads and feed overrides, enabling the model to learn complex correlations across variables.
- Few-shot adaptation: It is designed to operate effectively even when only a few labelled sequences are available, thanks to its strong generalisation capability and efficient training strategy.

- Time-aware mixing: Instead of relying on position encodings, TTM uses a mixing approach that directly incorporates temporal structure, enhancing its robustness to missing data and irregular sampling.
- Zero-shot and fine-tuning modes: The model can be used either in zero-shot mode, where it is applied directly without retraining, or in few-shot fine-tuning mode, which allows adaptation to specific machine or process conditions with minimal data.

For this use case, the model was fine-tuned using historical time-series data from the Nakamura 2, with features including override paths, servo loads and interpolated tool wear degradation signals. The target variable represents the progressive wear of each tool, interpolated from real change events.

TTM-R2 was selected for its predictive performance and for its operational efficiency and ease of integration into real-time pipelines. Its low inference latency and ability to generalise from limited examples make it particularly valuable in environments where frequent retraining is not viable.

In the following sub-sections the evaluation with real and synthetic data follows.

Evaluation with Real Data

Introduction

This section presents selected results obtained for the tool wear prediction task, focusing on two representative tools: **Tool 5** and **Tool 8**, both associated with the product reference 1360350132A. The evaluation was carried out using two distinct approaches:

- **Zero-shot inference**, where the pre-trained Granite Time Series TTM-R2 model was applied directly to the dataset without additional training.
- **Few-shot fine-tuning**, where the model was trained using historical degradation data specific to each tool to better capture its behaviour.

Predictions were performed using a context window of **512-time steps** and a **forecast horizon** of **96** future time steps. All input variables were normalised to facilitate the learning process and model convergence. After inference, outputs were denormalised to assess the accuracy and realism of the predicted tool wear evolution curves.

Results

The figures below show the model's predictions for three evaluation windows. In each plot:

- The blue line represents the true degradation curve (interpolated from tool change events).
- The orange dashed line shows the model's predicted values.
- The vertical red line marks the starting point of the prediction window.

Figure 22 shows the output for Tool 4, which has **no recorded tool changes** during the available time window. As a result, the model is unable to infer any meaningful degradation pattern and produces unstable predictions across all time intervals:

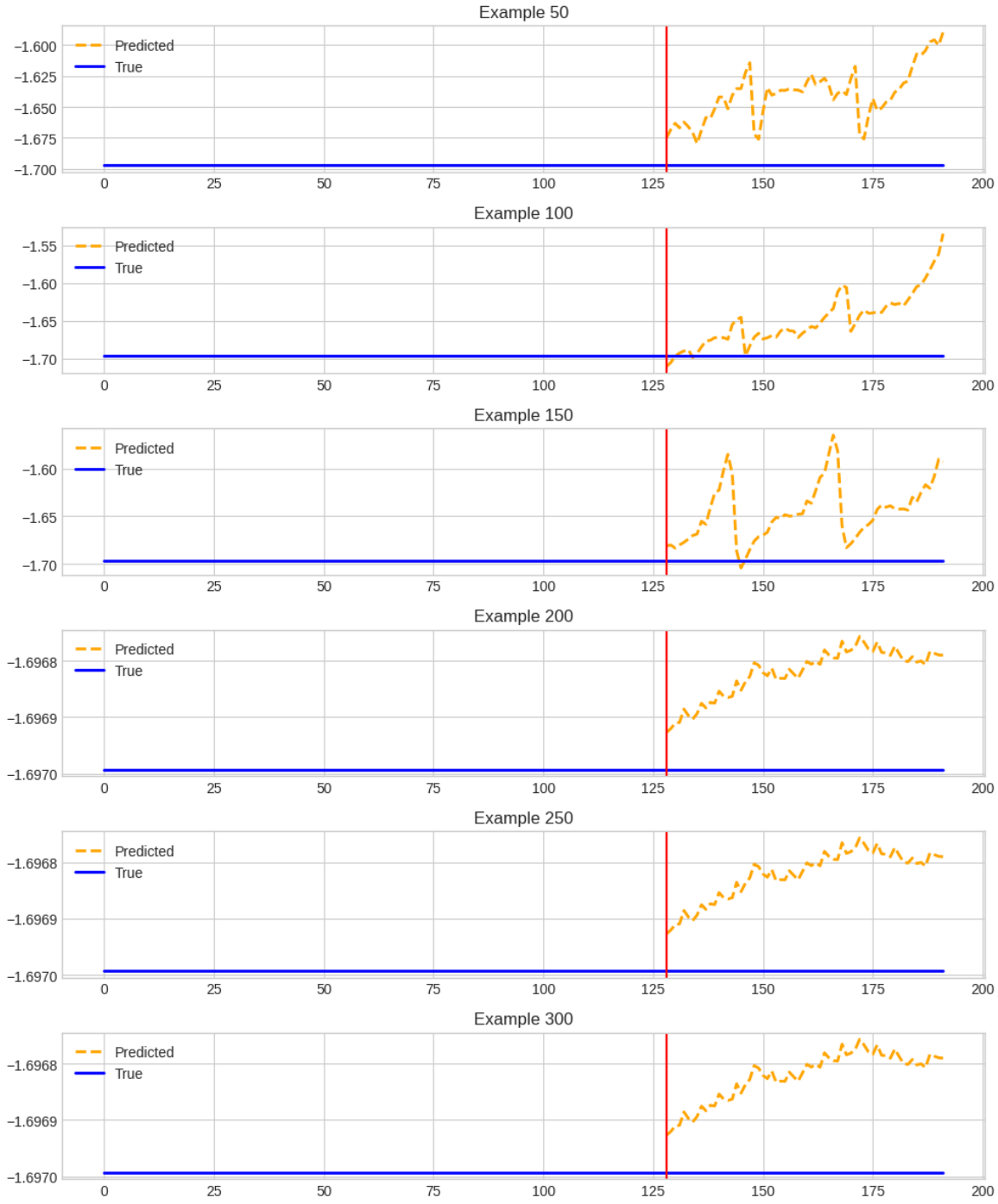

Figure 22: Predictions for Tool 4 with no degradation events

Even when Tool 5 (Figure 23) includes some operational values, its degradation curve is **monotonic** (increasing) and lacks drops that typically indicate tool replacement. When the model is applied in zero-shot mode, the lack of domain adaptation leads to **erratic predictions** disconnected from the actual tool wear trend.

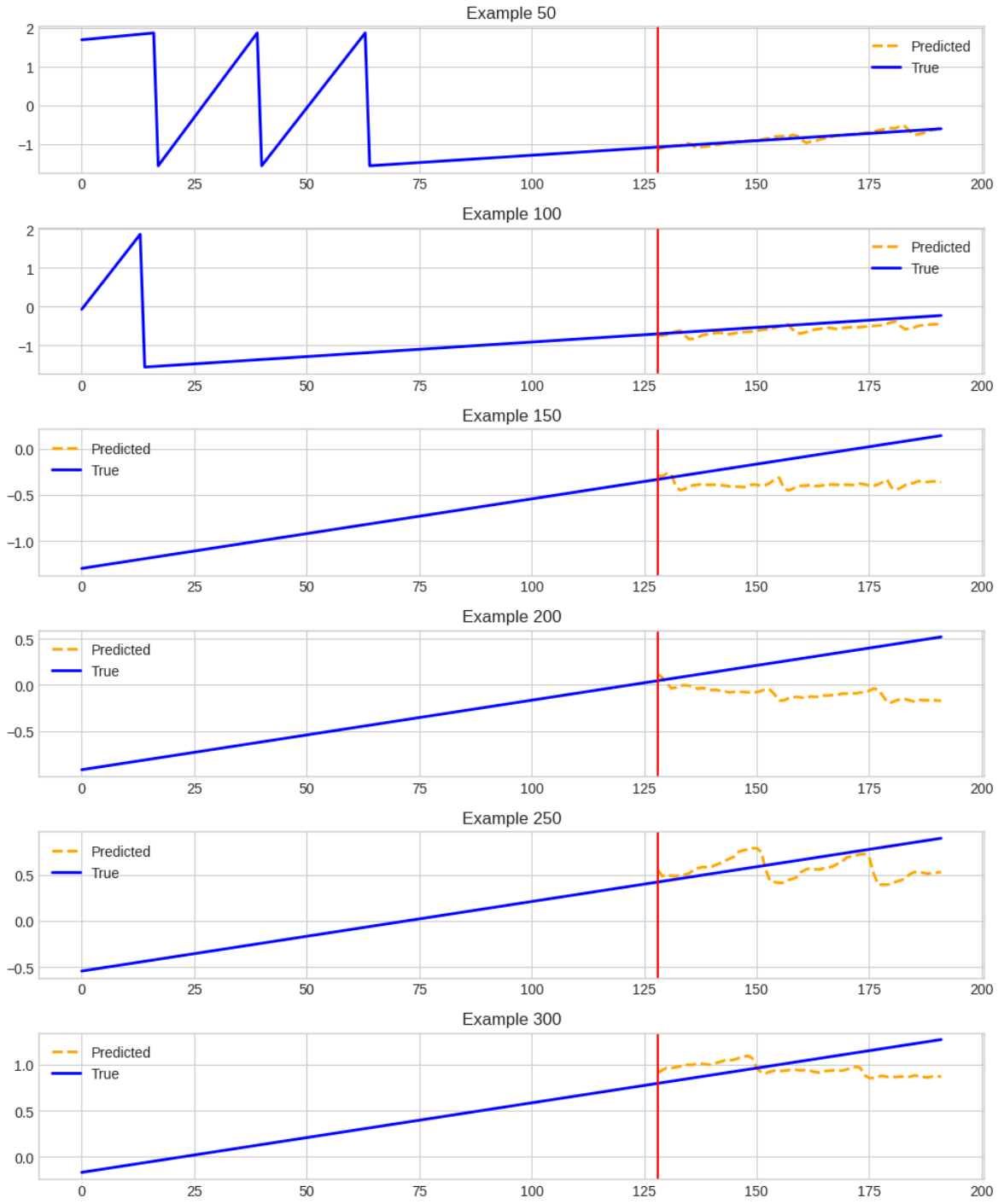

Figure 23: Predictions for Tool 5 using zero-shot inference

Figure 24 (Tool 8) presents **multiple degradation cycles**, with clearly recorded replacement events. Once fine-tuned, the model can **accurately reproduce the degradation slope**, closely following the real survival trend over all evaluation windows.

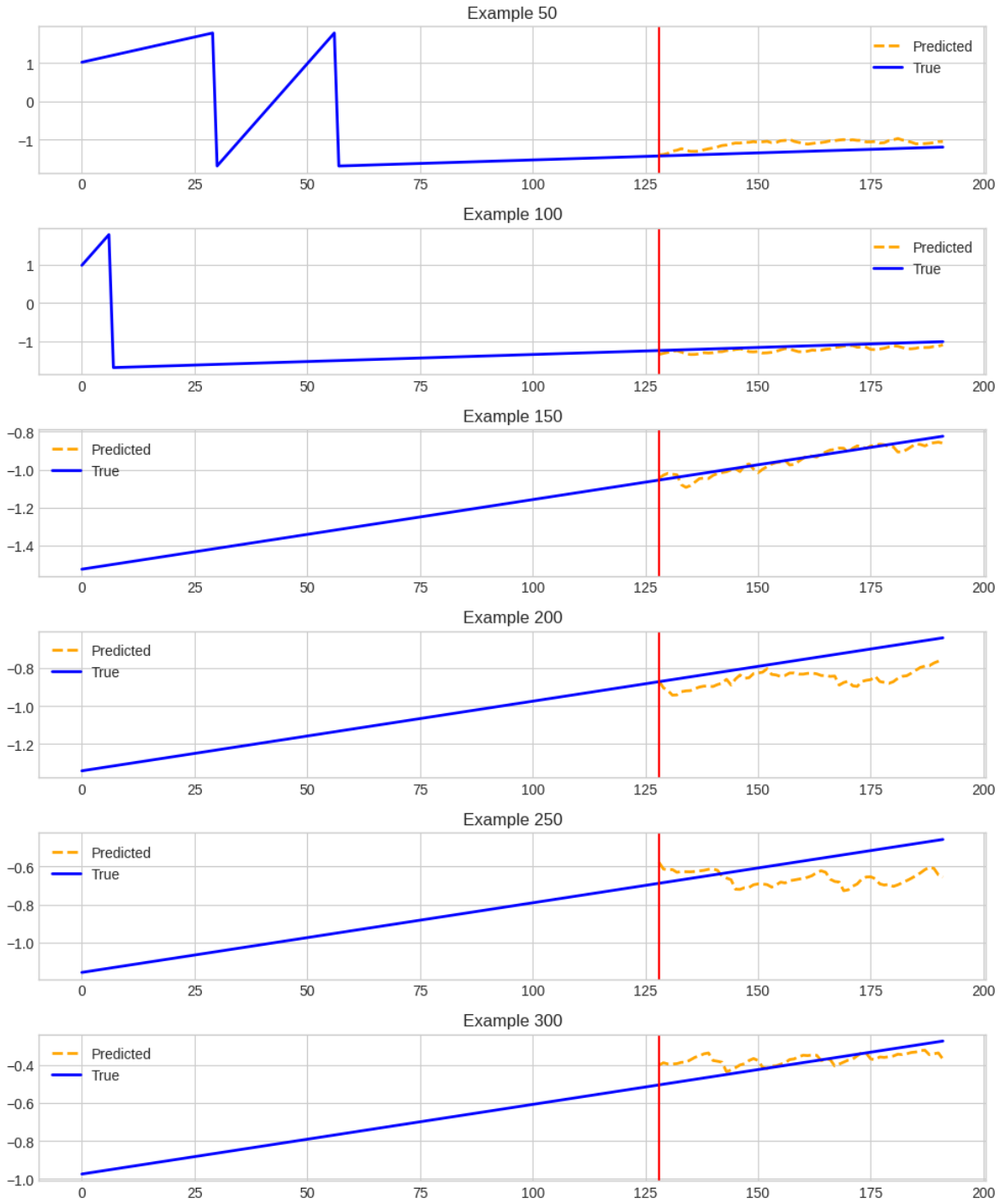

Figure 24: Predictions for Tool 8 using fine-tuning.

The results confirm the following:

- In the absence of historical events (Tool 5, no changes), the model cannot learn meaningful degradation patterns, and predictions are unstable.
- In zero-shot mode, the model does not adapt to the specific structure of the target tool and fails to follow the wear trend.
- It is important to give the model a time range where there are sufficient changes to train, validate and test.

- Once fine-tuned with even a small number of annotated events, the model significantly improves in accuracy and stability.
- Tools with multiple degradation cycles (e.g., Tool 8) offer better training signals, enabling the model to generalise and reproduce the actual tool life trajectory.

This analysis confirms that data quality and the event number are more important than data volume in the success of predictive models under the few-shot learning paradigm. Fine-tuning on even a modest number of cycles yields substantial gains in stability and realism. On the contrary, zero-shot inference in the absence of degradation dynamics is ineffective and may lead to misleading predictions.

These findings reinforce the critical role of data engineering and minimal labelling efforts in industrial AI deployments, especially in high-cost, low-frequency failure contexts such as predictive maintenance of CNC tools.

Evaluation with Synthetic Data

Introduction

As an additional analysis, the model's behaviour was evaluated using a set of synthetic data. This evaluation aims to explore whether the model can extract meaningful patterns from signals that do not originate directly from the industrial environment. Furthermore, it assesses the potential of synthetic data to support preliminary training stages or to serve as a proxy in scenarios where labelled real-world data is not available.

Results

Figure 25 shows the predictions obtained for Tool 4 when applying the model in zero-shot mode on synthetic time series. The graphs show four randomly selected prediction windows, where the actual evolution of the synthetic signal (blue line) is compared with the prediction generated by the model (orange dashed line). The red vertical line indicates the start of the inference window. This graphical representation maintains the same format used in the experiments with real data, allowing a direct comparison of the model's performance in both contexts.

The results show that the model fails to adequately capture the underlying degradation dynamics, generating flat or weakly correlated outputs, even in the presence of significant variations in the signal.

This behaviour is due to the absence of fine-tuning and a lack of alignment between the structural characteristics of real and synthetic data. While the real data show a typical sawtooth evolution (characterised by a progressive rise associated with wear, followed by a restart after each tool change), the synthetic signals evaluated here do not replicate this morphology. As a result, the model does not have familiar patterns that it can recognise or extrapolate, which severely limits its generalisation ability.

In short, zero-shot inference on synthetic signals with structures different from those observed during training leads to inaccurate and uninformative predictions, highlighting the need for input data that maintains a certain structural consistency with the real data.

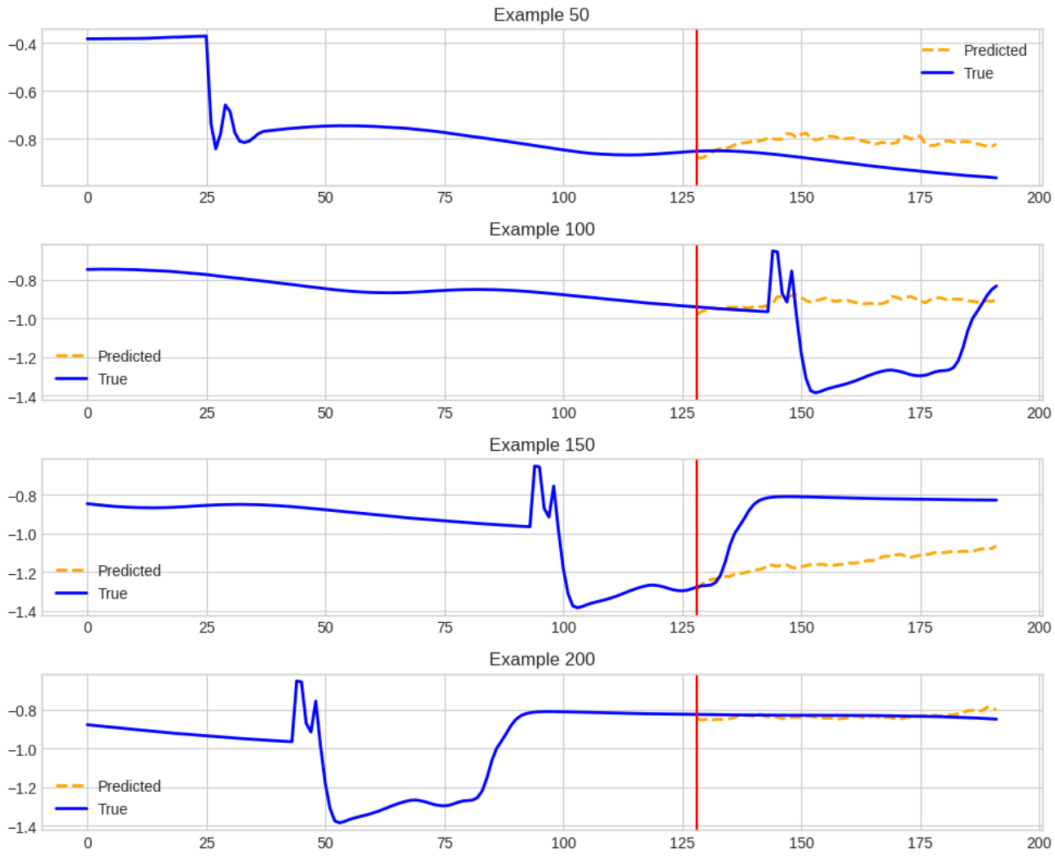

Figure 25: Tool 4, zero-shot inference on synthetic data. Source: Own elaboration

Figure 26 shows the predictions generated by the model for Tool 4, after applying a fine-tuning process on synthetic data. Unlike the zero-shot scenario, the results show a notable improvement in the quality of the predictions. The model manages to partially reproduce the slope and general direction of the degradation, especially in cases where the synthetic signal includes several cycles of wear and restart. This effect is particularly evident in examples 150 and 200, which have a sawtooth-like morphology, making it easier for the model to identify the progression of wear and anticipate its evolution.

However, in examples where the signal lacks clear changes or has smoother trajectories (as in examples 50 and 100), the model tends to produce flatter or misaligned predictions. This suggests that, although fine-tuning improves adaptability, the model's effectiveness still depends largely on the input data having a structure consistent with the patterns observed during initial training.

It should be noted that the model responds better when the synthetic signals present structured and cyclical degradation, like the actual evolution of tool wear. For the model to generalise correctly, it is not enough to apply fine-tuning to any synthetic data; it is necessary for such data to maintain a clear morphology, with well-defined linear ascents and restarts, which faithfully reflect the behaviour to be predicted.

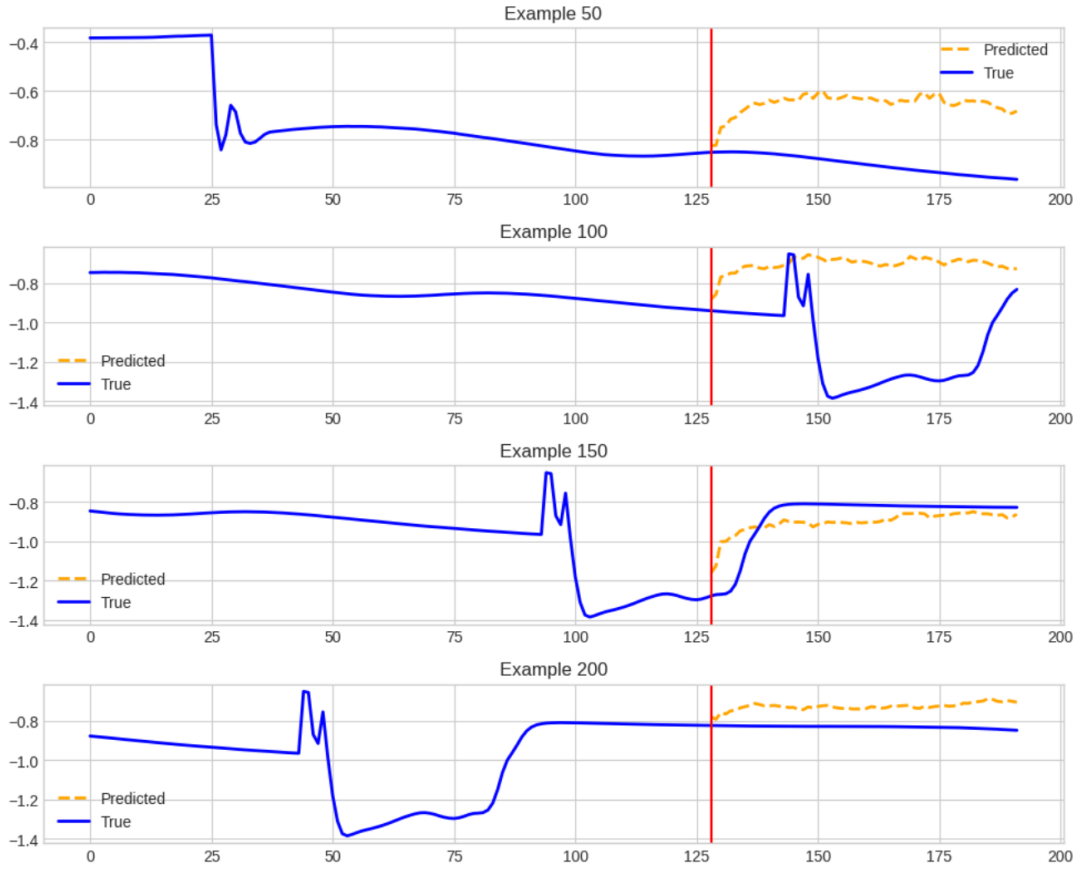

Figure 26: Results Tool 4 using Fine-tuning on synthetic data. Source: Own elaboration.

In the case of Tool 5 (Figure 27), the model shows similar behaviour to that observed previously: it manages to adjust partially when the signal shows more structured degradation (example 200), but it maintains a tendency to smooth the predictions and does not capture the actual shape of the wear with sufficient accuracy. In these examples, the model manages to follow the general direction of the curve, albeit with errors.

In contrast, in windows where the signal does not show clear variations, the prediction stabilises around an average value, without adapting to the actual evolution of the signal. This reconfirms that the quality of the predictions depends not only on fine-tuning but also on the synthetic data having a clear linear progression with transitions that mimic the wear-restart pattern present in the real data.

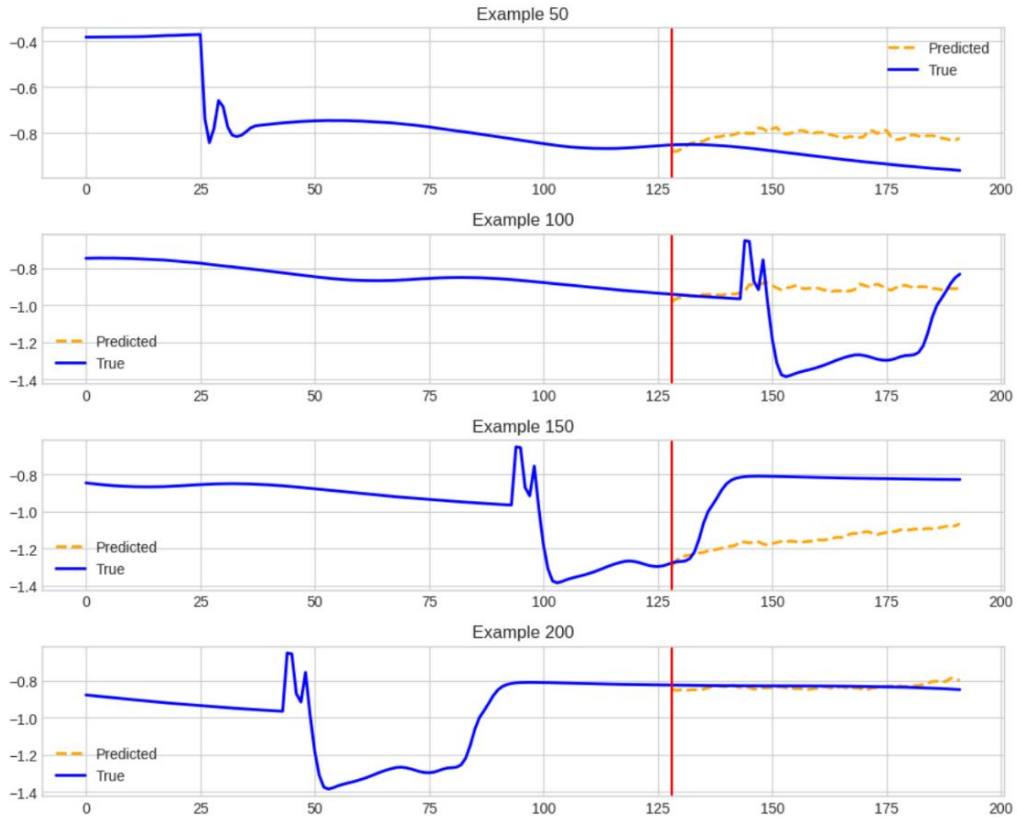

Figure 27: Results Tool 5 using fine-tuning on synthetic data. Source: Own elaboration.

In the case of Tool 8, the model shows clear limitations when operating in zero-shot mode (Figure 28). In all examples, the prediction is flat and remains far from the actual signal, which presents abrupt transitions and pronounced variations in slope. Although some of the examples show a structure with multiple changes and drops, the model fails to identify or anticipate such dynamics.

This result reinforces the idea that the model, without prior adjustment, is unable to adapt to unknown structures. In particular, the shape of the actual signal in these cases does not match the wear patterns typically seen in the training data (such as the sawtooth-like linear downward progression), which limits its generalisation ability. Thus, direct inference on synthetic data with different morphology leads to a significant loss of predictive accuracy.

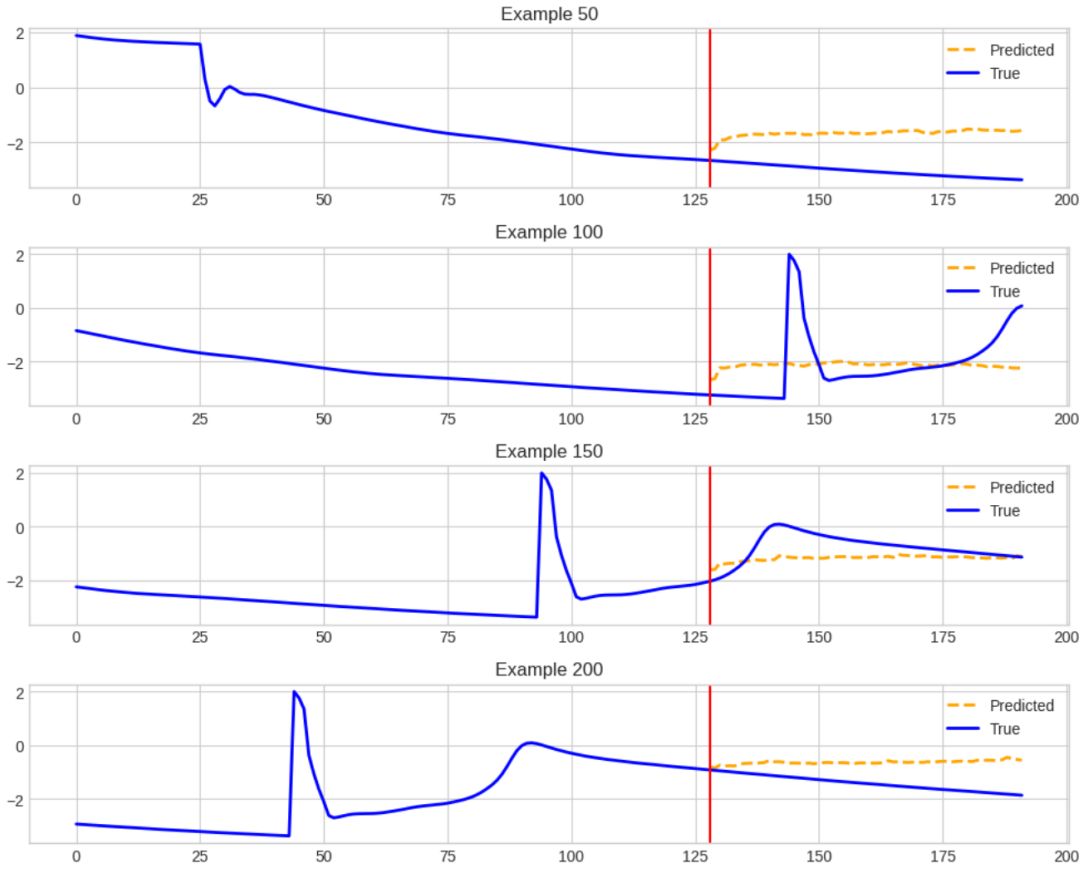

Figure 28: Results Tool 8 using zero-shot on synthetic data. Source: Own elaboration.

In this case, the model predictions for Tool 8 (Figure 29) have been denormalised after fine-tuning, allowing the results to be represented on an interpretable scale (0 = new tool, 1 = end of useful life). However, this transformation does not imply any performance improvement but only facilitates the visual reading of the values.

The model's behaviour remains like that observed in the normalised version: in examples with a certain structure (such as cases 150 and 200), the prediction partially follows the general wear trend, although with errors in magnitude and without adequately capturing peaks or cycle restarts. In examples where the signal is smoother or more linear (cases 50 and 100), the model's response tends to stabilise around an average value, without following the actual evolution.

These results reinforce the idea that the model needs signals with a clear cyclical structure consistent with the training patterns (such as sawtooth wear curves) to generate reliable predictions, even after fine-tuning.

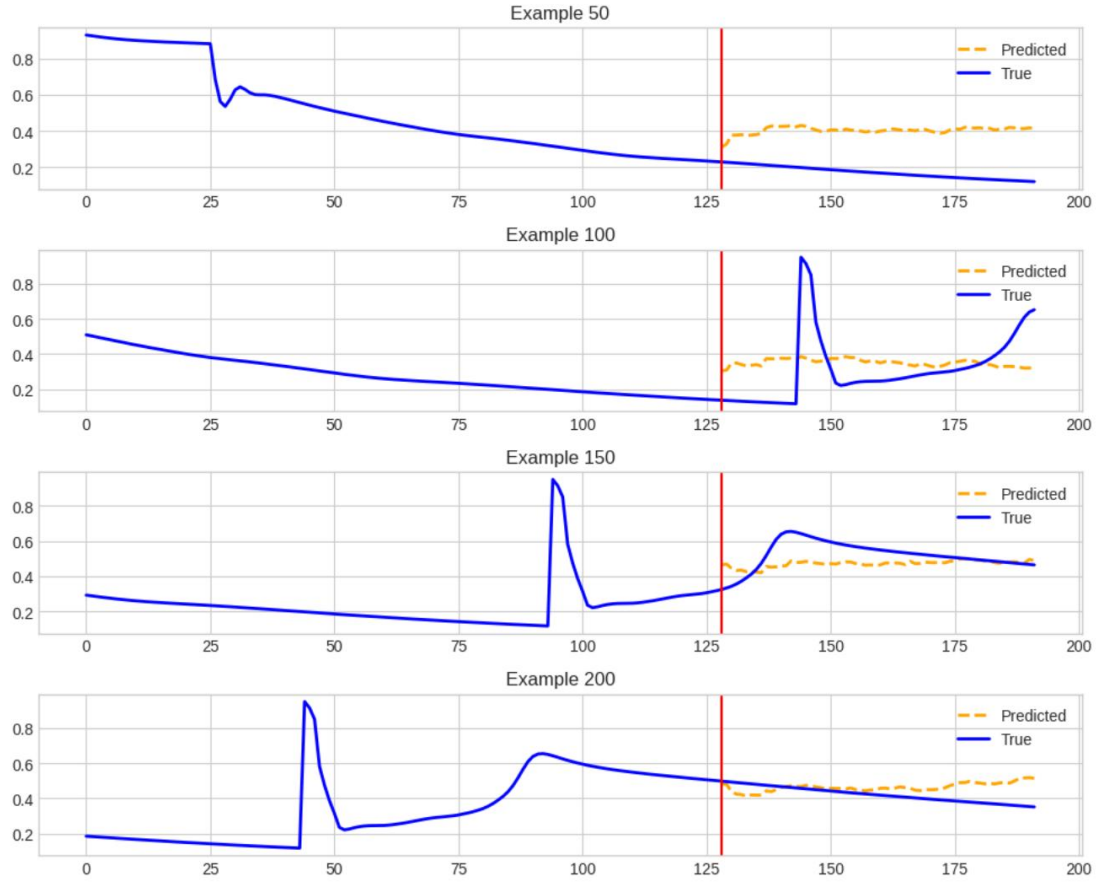

Figure 29: Predictions for Tool 8 using synthetic data after fine-tuning with denormalised outputs. Source: Own elaboration.

In general, these experiments highlight the model's dependence on structured and cyclical degradation patterns, particularly those resembling the sawtooth morphology observed in the real data. Fine-tuning allows for some improvement when synthetic signals align with this structure, but it is insufficient when signal dynamics diverge. The results suggest that, for synthetic data to be useful in predictive maintenance applications, its design must reflect the operational characteristics of the real environment. Otherwise, the model's generalisation ability remains limited, regardless of the training strategy.

Overall Impact to the Pilot

In 2024, the pilot reported an annual Overall Equipment Effectiveness (OEE) of 73.4%, with Availability standing out as the main limiting factor at 76.17%, being the only component below 90%. A focused root-cause analysis determined that most of the unplanned downtime stemmed from tool-change events: replacements were scheduled based on operator experience rather than objective tool-health metrics, resulting in either premature swaps (and higher tooling costs) or delayed changes (and defective parts).

Within the framework of TALON, a predictive algorithm was developed to estimate the Remaining Useful Life (RUL) of lathe tools, enabling more informed planning of tool changes. The first step was the identification of key operational variables closely linked to tool wear. The analysis showed that the effort exerted by the machine's servomotors, specifically the ServoLoad, correlates with the degree of tool degradation. As tools wear out, servomotors must apply more force to complete machining tasks. Additionally, the Override parameter, indicating the percentage of programmed feed rate used, was also found to impact wear, as higher override values reduce tool lifespan.

Data was collected for all relevant axes. Each turret includes five servomotors, of which three (X, Z and Y axes) were selected for the study due to their direct involvement in cutting actions. The dataset combined these sensor values with recorded tool change events. However, due to the limited number of annotated degradation cycles, a Few-Shot Learning (FSL) strategy was applied. This enabled the development of an accurate model with minimal labelled data, minimising data acquisition effort and computational requirements.

The predictive model has been successfully developed, trained, and validated using real operational data from Factor's CNC machines, combining high-frequency sensor readings and historical records of tool changes. The experimentation was carried out on actual shop-floor datasets, ensuring full alignment with real industrial conditions.

The model demonstrated a high capacity to reproduce tool degradation dynamics and to accurately estimate the remaining useful life of each tool, even in scenarios with very limited labelled data. Finally, the study and investigation conducted enabled the pilot to extend lathe-tool lifespan and increase machine availability to 80.44%, which in turn raised OEE to 78.35% as of 2025.

4 Explainable AI Framework

Explainable AI (XAI) plays a critical role in the TALON project, aiming at revolutionising and increasing the transparency of the next-generation industrial systems that incorporate advanced technologies like AI, edge-to-cloud (E2C) computing, blockchain, and visualization. XAI is essential for attaining a high degree of explainability, trustworthiness, and transparency. It renders the operations, outcomes, and decisions of AI models to be transparent and comprehensive to human users, a fundamental requirement within industrial systems where AI decisions have a considerable impact. Understanding AI's decision-making processes significantly bolsters trust and reliability, a crucial aspect, particularly within industrial applications [21].

In TALON, XAI is pivotal in advancing intelligence to the network's edge, validating, and ensuring data sources reliability, understandability of AI decisions, and overall transparency throughout the full TALON system pipeline and operations. In particular, it is comprehensively structured across different Trust Levels (TrLs), each covering different phases of TALON's system pipeline, from analysis and validation of raw data through the explainability of predictions from AI models. These TrLs are categorized into two broad areas: explanations of data (TrL1 and TrL2) and explanations of model results (TrL3 and TrL4), dealing with two primary data types – images and time-series data [22].

In the following sections the advancements of TALON's framework, compared to D4.1 "Initial Reusability, Explainability, Trustworthiness Report" are presented. Advancements that are taking place across all TrLs (TrL1, TrL2, TrL3, TrL4) and across all modalities (images, timeseries). Firstly, the advancements in TrL1 and TrL2 are presented which are used to analyze the raw data from the dataset that was used (either the time series or the image datasets). Then TrL3 is used in model level to produce more interpretable results either we are talking for images or timeseries. Finally, in TrL4, a sophisticated adversarial way of evaluating the models is presented.

4.1 Explanations of Data

4.1.1 Key Technologies & Technical Updates

XAI plays a critical role in assisting experts reach meaningful and impactful decisions and in very constrained time frames. It is imperative to know "why" an AI model has reached its' decision as well as attaining highly accurate results. So far, 4 key technologies have been implemented during TALON. They are numbered as TrL1 through TrL4 and each of these components has a very specific task to accomplish. Additionally, a REST API has been implemented to enable these components to communicate with external end users and other components. Also, a web application that includes a dashboard has been implemented and is accessible through this URL: https://talon-ui.euprojects.net/dashboard. The XAI interface can be reached through navigating from the left side panel to "AI cognition". From the menu that pops up choose "Explanations". Finally, a menu materialises through which the specific TrL, the template type as well as the dataset can be chosen.

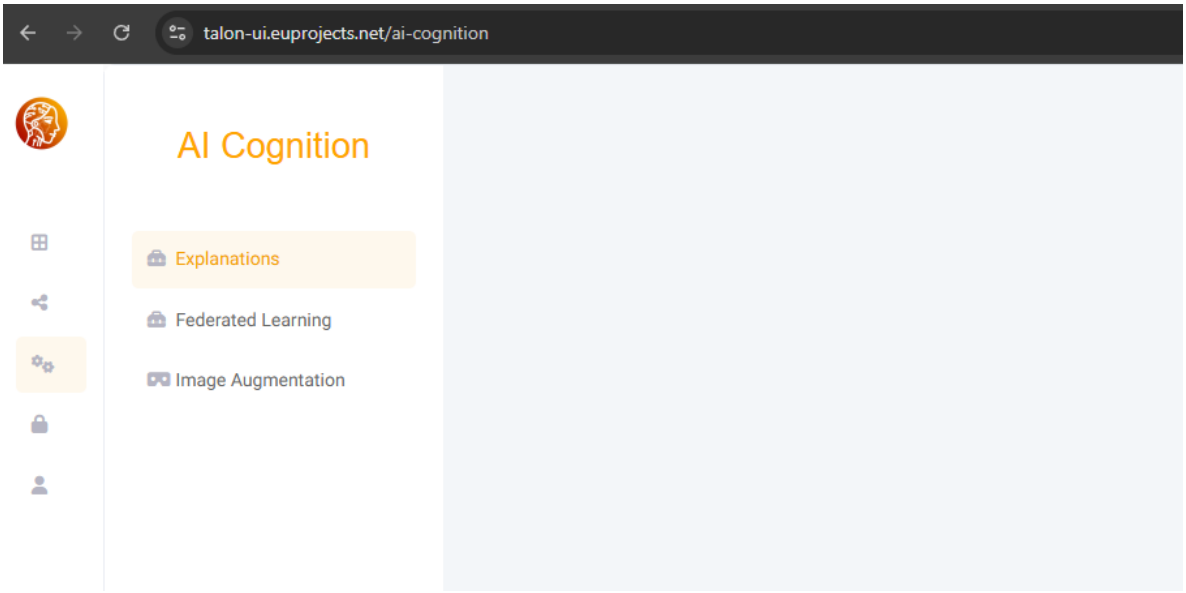

Figure 30: XAI Dashboard navigation

Under TrL1, in the context of images, the image shape, the image format and whether the image is corrupted are checked. In the context of the time series data, TrL1 detects whether there are any null or missing values in the dataset. In this way, correlations between specific points in time and data values can become apparent. This approach enables the end user to identify possible “gaps” in the data which may lead to conclusions about the specific sensor’s availability and / or integrity through time. TrL2 shows insights into a next-stage classification task. TrL2 receives as input the data and outputs the number of class instances for each categorical class that exists in the dataset. In the context of the time series data, TrL2 applies the Local Outlier Factor (LOF) algorithm which detects possible outliers in the dataset. A very important note is that no preprocessing of the dataset occurs at this step, as the outliers are detected but not removed.

4.1.2 Scientific and Technical Results

The scientific and technical results of the current deliverable are aligned with the initial goals of the TALON project. Specifically, with the help of the second pilot Use Case 2 (UC2) - *I5.0 Automation & Planning*- a connection between the manufacturing line equipment (a machining lathe, see figure 3) and a kind of a Digital Twin has been established, as this goal was set in the initial Grant Agreement. The equipment offers the near real-time dataset for the Digital Twin to operate on. In its’ turn, the Digital Twin applies the specific AI model for the use case as it was previously defined, along with the appropriate TrL layers that have been developed so far, that analyse the input dataset, preprocess it, feed the cleaned dataset to the AI models and finally help explain the inference performance of the AI models.

Through the utilisation of the AI models, which were fed the timeseries data streams that were generated by the machining lathe sensors, the manufactured parts’ quality can be sampled and controlled accordingly. The task of quality control for the manufactured parts is currently being done manually and with a very low sampling frequency -out of the thirty parts that the machine can complete in one hour only one of them is put under a quality check. If the specific part is found to be faulty then, all the other twenty-nine parts will be assumed to be faulty as well and as a result the machining lathe would have been running pointlessly for an hour. Having generated faulty parts in this timeframe, precious raw materials are wasted in the process. This fact, that many raw materials were being wasted, became one of the reasons behind the conception of this specific pilot. On top of these

problems, the data generated manually for each report are currently being stored in excel files which also creates a huge management problem for the operators.

The UC2 pilot comes in to solve all the above problems, as the sampling frequency is increased from about one hour to between five-minute and fifteen-minute intervals, the data is logged automatically so the operator does not need to manually generate, and fill excel files. Finally, the TrL layers along with the AI models that have been implemented help determine whether there is a fault with the machining lathe or not, much faster – in a matter of seconds-. Specifically, TrL1 along with TrL2 helps the UC2 pilot through detecting a series of corruptions in the data, providing more capable data to train an efficient Ai model. Additionally, the system can be continuously fed near real time data from the machining lathe and as a result the AI models can be further enhanced with these new data through re-training of the AI models

4.1.3 Instantiation with pilot data

During the TALON's second pilot demonstrator UC2 -*I5.0 Automation & Planning* pilot Use Case 2- a dataset has been generated through the utilization of the twelve (12) signals as they are defined below for the Nakamura machine (a machining lathe). Specifically, the twelve signals are:

Table 20: Signals from Nakamura machine

No	Name of data stream	Description of data stream
1	Speed S P1	Speed (rpm) of the spindle motor in turret 1 (P1)
2	Speed S P2	Speed (rpm) of the spindle motor in turret 2 (P2)
3	Feed rate P1	The velocity (mm/min) at which the cutter is advanced against the workpiece. In the turret 1, P1.
4	Feed rate P2	The velocity (mm/min) at which the cutter is advanced against the workpiece. In the turret 2, P2.
5	Spindle motor load P1 S0	Spindle motor load (%) of the turret 1 that is, percentage of load at which the spindle motor is working.
6	Spindle motor load P2 S0	Spindle motor load (%) of the turret 2 that is, percentage of load at which the spindle motor is working.
7	Servo load current (a) P1A1	Servo load (%). Percentage of a rated current (consumed electricity) of turret 1.
8	Servo load current (a) P2A1	Servo load current (a) P2A1: servo load (%). Percentage of a rated current (consumed electricity) of turret 2.
9	Cutting time P1	Time (s) that the turret 1 is cutting material
10	Cutting time P2	Time (s) that the turret 2 is cutting material
11	Temp APC P1A0	Temperature in a motor in the turret 1
12	Temp APC P2A0	Temperature in a motor in the turret 2

This dataset was then incorporated in TrL 1 under "time series" template type and by choosing the "FACTOR Timeseries" dataset, all the collected data is made available through the TALON web application.

The "gaps" (whitespace) between the datapoints in the following figure, depict the missing (null) values in the dataset. Such values probably correlate with some kind of fault or unavailability of the sensor that records them. Also, there are some gaps in the measurements that span a few hours in length. This might mean that the machining lathe was taken off-line for some reason (e.g. for maintenance). Another reason this event might have occurred is that due to the relatively low

sampling frequency (about one measurement per fifteen minutes) that has been programmed into the sensors and by extension to the generated data streams. Theoretically the sampling time could be further increased in the future to supply even higher fidelity data streams with little to no additional effort put into changing the AI models' architectural properties.

Figure 31: TrL1 UC2 application

TrL2 utilises the Local Outlier Factor (LOF) algorithm -as explained previously- and any outliers in the modified dataset are detected. In the following figure, the outliers are highlighted with a red dot and are discriminated from the rest of the -regular- data points. This procedure helps in the UC2 pilot by enabling the appropriate actors to detect and correlate the datapoints that cause outliers with the results of the previous layer and ultimately help in the decision-making process of the actors.

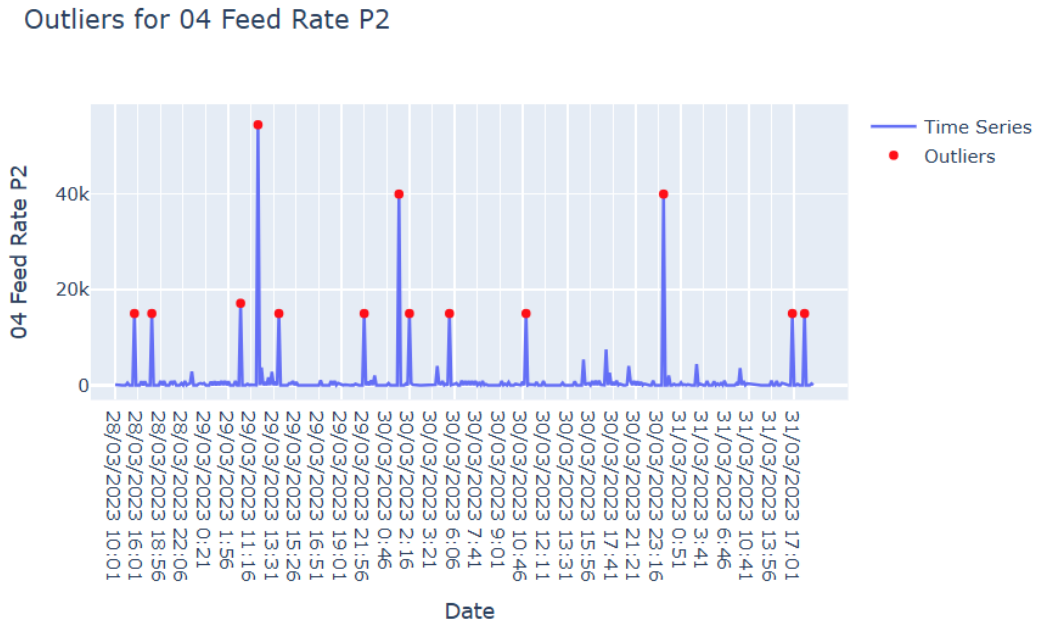

Figure 32: TrL2 UC2 application

4.2 Explanations of AI Model Results

4.2.1 Key Technologies & Technical Updates

TrL3 and TrL4 explainability layers have been designed towards completing a different set of tasks than the previous two layers, which were focused on preprocessing the dataset. These two layers focus on implementing techniques which will help explain the output of the AI models that are responsible for making the decisions. The explainable artificial intelligence (XAI) models that have been developed are demonstrated in the field through UC4 pilot ("Human Robot Collaboration").

As was previously explained a dashboard has been developed, was made available and was integrated to the main web application. A representative snapshot of the TrL3 graphical user interface that has been developed is shown in the figure below.

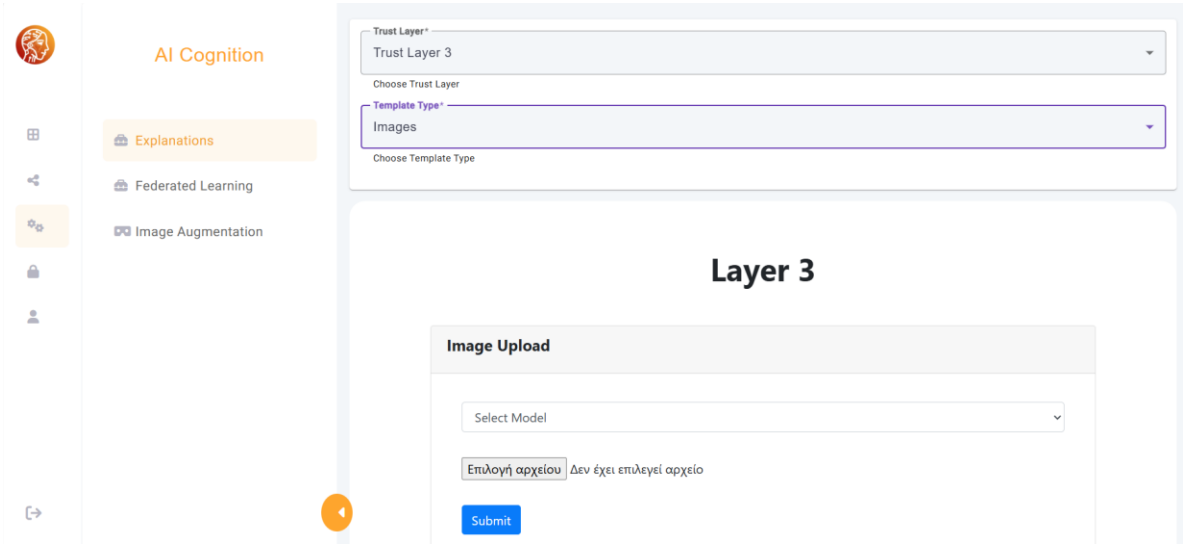

Figure 33: Snapshot of XAI screen of TALON dashboard

TrL3 when supplied with image data and the corresponding model is chosen, outputs a few bounding boxes with the corresponding labels of the identified objects as shown in the figure below. In the context of the TrL3, 3 AI models based on the YOLOv8 neural network architecture have been developed, namely:

1. Personal Protective Equipment,
2. Fire
3. Construction Safety Measures

The original YOLOv8 model was fine-tuned on a different dataset for each one of these models. The resulting model of this fine-tuning was exported and linked to the web application.

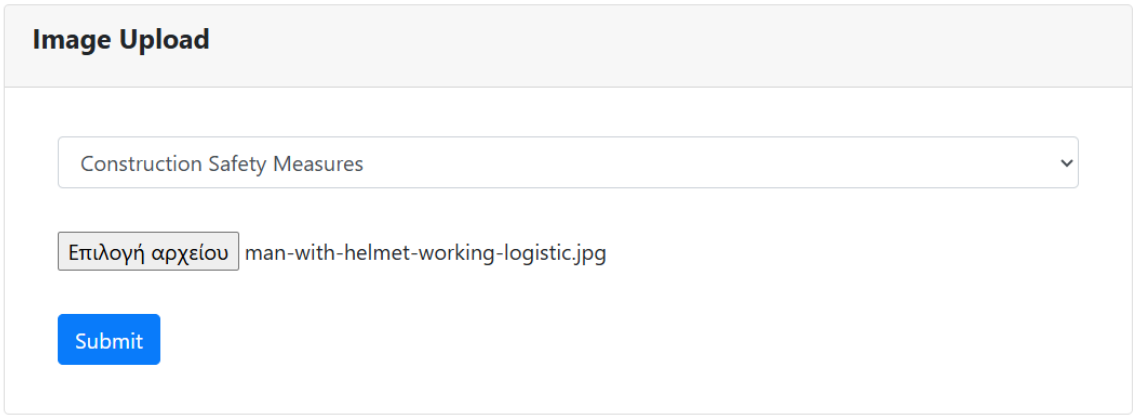

Figure 34: Image upload and model selection to TrL 3

In the above example the “Construction Safety Measures” model is chosen along with an uploaded image that contains instances of relative objects. Underneath the hood, the artificial intelligence model that was utilized is YOLO (You Only Look Once) architecture which can be found in this URL: https://yolov8.com/. Specifically, the YOLOv8 instance was utilized in order to detect the objects in the input images. YOLOv8 is a one-stage detector, which means that through the neural network architecture that is utilized under the hood, it can detect the bounding boxes for objects (that exist in the training dataset) and at the same time classify the object in one of the categories that it learns during training. Furthermore Grad-Cam (Gradient-weighted Class Activation Mapping) [23] was used in order to produce the heatmaps that are visible (to right hand side of the figures) below. Grad-Cam is a technique which aides in the visualization of the specific location where the model “looks at” when assigning a class to a detected object effectively showing which features of the input image the AI model is “paying attention to” through the use of a heatmap. In the case of TALON (development and pilot phase), no real-time requirement for the application is necessary and the results are informative, used by the developer/user at his/her own pace for decision making regarding data and models.

In the next figure (Figure 35) the results of an inference example are visible. On the left images, the bounding boxes with the corresponding labels are depicted. On the right, the explanation regarding “why” were the specific bounding boxes chosen, are depicted utilizing a heatmap overlay, in which the pixels that are coloured red represent the most important pixels, as classified during the inference of the model, and the blue pixels are the least important.

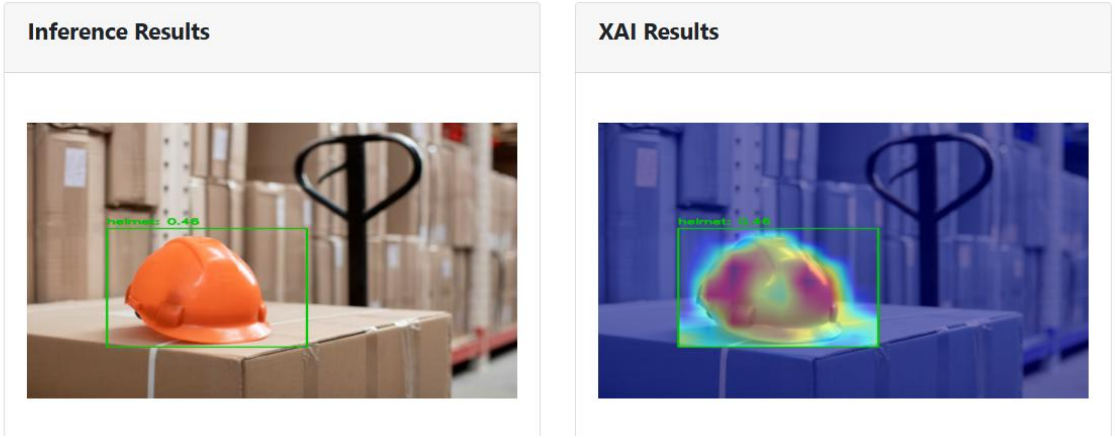

Figure 35 - (Left) bounding boxes of model in inference, (Right) TALON's TrL3 XAI heatmap overlay

In the right side of figure 6, the areas that play a significant role in detecting the helmet are depicted in the heatmap. Specifically, the most unique parts of the helmet are coloured with deep red and yellow such as the top and the sides of the helmet, while the rest of the image is coloured blue which signifies low correlation with the activation of the output pixels. Therefore on the right side we can see the pixels responsible for the decision that the model has taken regarding the location of the bounding box.

Another model that has been created and is evaluated with the XAI layers, is called "Personal Protective Equipment" (PPE). In the following figure the Graphical User Interface is depicted.

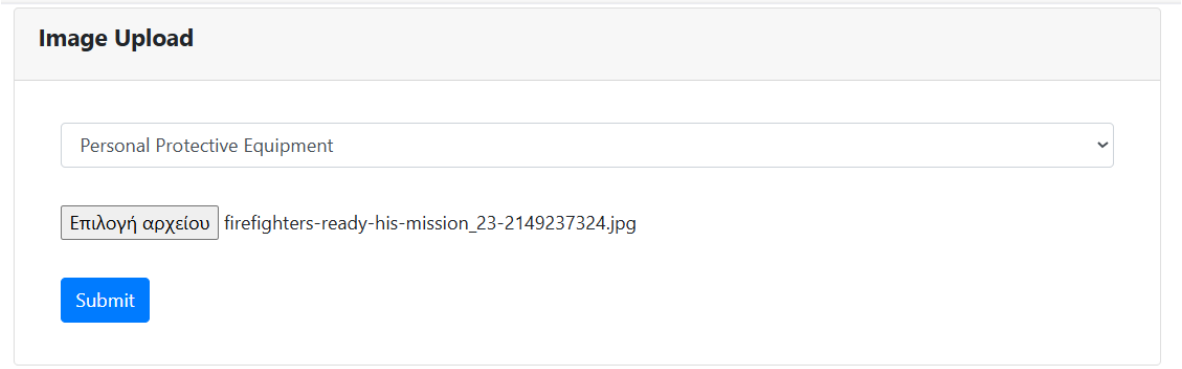

Figure 36 - GUI for PPE use case

Directly below the above GUI options in the TALON web app (Figure 37), two images are displayed similarly to Figure 35.

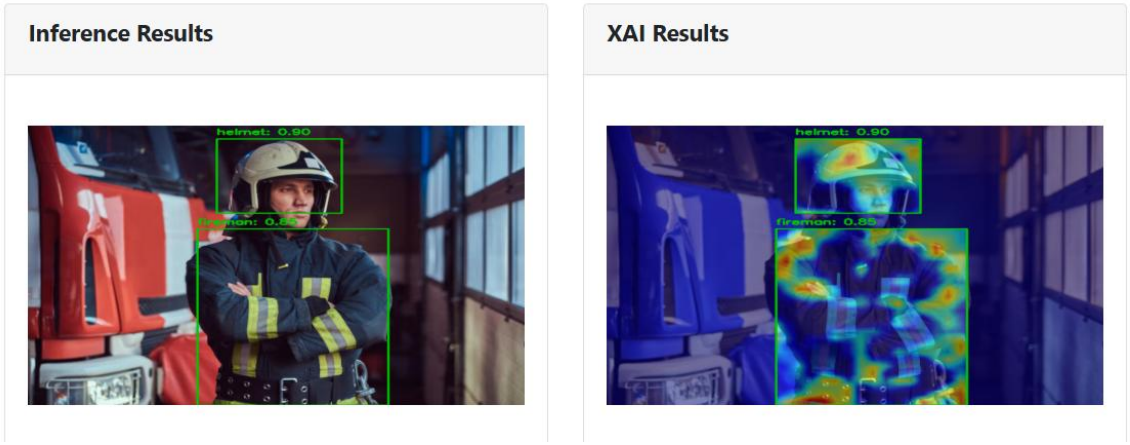

Figure 37 - (Left) Bounding box with the inference results, (Right) the corresponding XAI heatmap is overlayed on the input picture

As was the case with the previous example, on the left is depicted the original image and on the right the correlated input pixels are coloured with red and yellow. Consistently, it is observed that the model “sees” -at least in a loosely defined manner- the correct features of the fireman object -which mainly consist of the various distinct features of the fireman’s uniform- and the corresponding features of the helmet.

The third and final model that was used is called “Fire”.

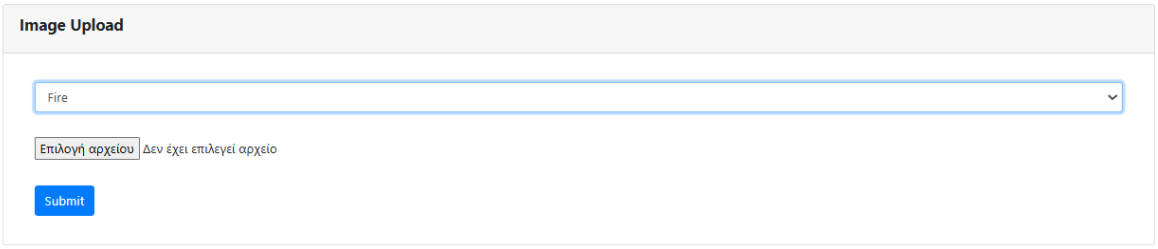

Figure 38 - "Fire" model inference

In the following figure the results of XAI are depicted similarly to the previous ones.

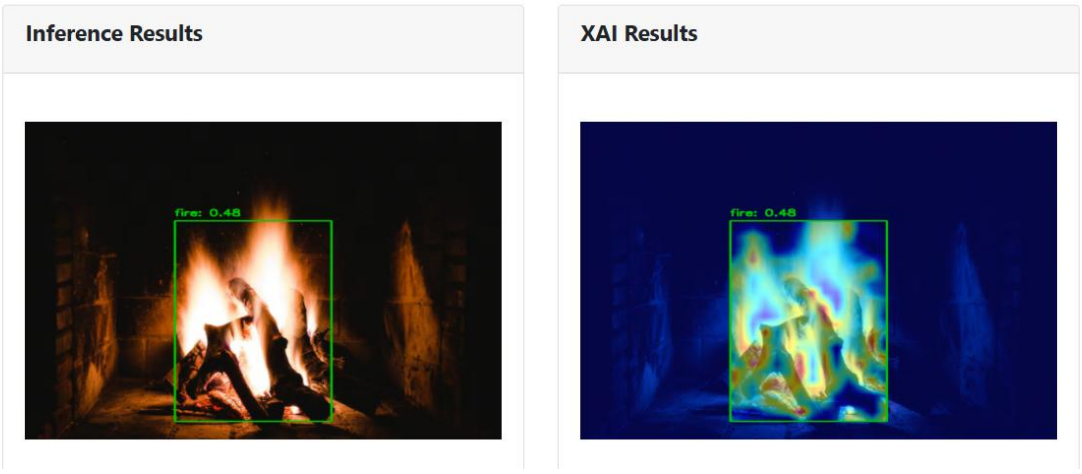

Figure 39 - (Left) inference result of the "Fire" model, (Right) XAI heatmap overlay – the bounding box correctly identifies an ongoing fire in the image

Again, like the two previous cases, on the left is depicted the input image and on the right is the heatmap that corresponds to the input pixels that correlate to the outputs of the model. For example, similarly with the previous two cases, the brightest regions of the fire object are coloured red and yellow -the heart of the fire is in the red and yellow colour spectrum- while the rest of the pixels are coloured blue, meaning that as we move away from the heart of the fire -in pixel coordinates-, we get more blue coloured pixels.

Regarding TrL4, two key metrics regarding the image classification models have been so far defined, have been implemented and are now made available through the TALON web application. The first metric is called “Image-Fidelity”. This metric determines which features are chosen by the model and contribute to the objects detected in the image. One definition for the image-fidelity metric refers to the accuracy and quality of the output that an AI model is able to create. By artificially obstructing the most critical parts of the object, it is anticipated that the model will no longer be able to correctly identify the obstructed object -as a human cannot do either. In the following figure the TALON web app’s GUI is showcased. In the boxes on the left side of the image the user can input the coordinates of a “black box” artifact which is exactly that, a black box which is strategically placed by the user on the object that should be obstructed in order to perform the explainability test. The test is, as explained in the present section, a test to check whether the AI model is “guessing” instead of actually detecting important image features in order to classify the object in question.

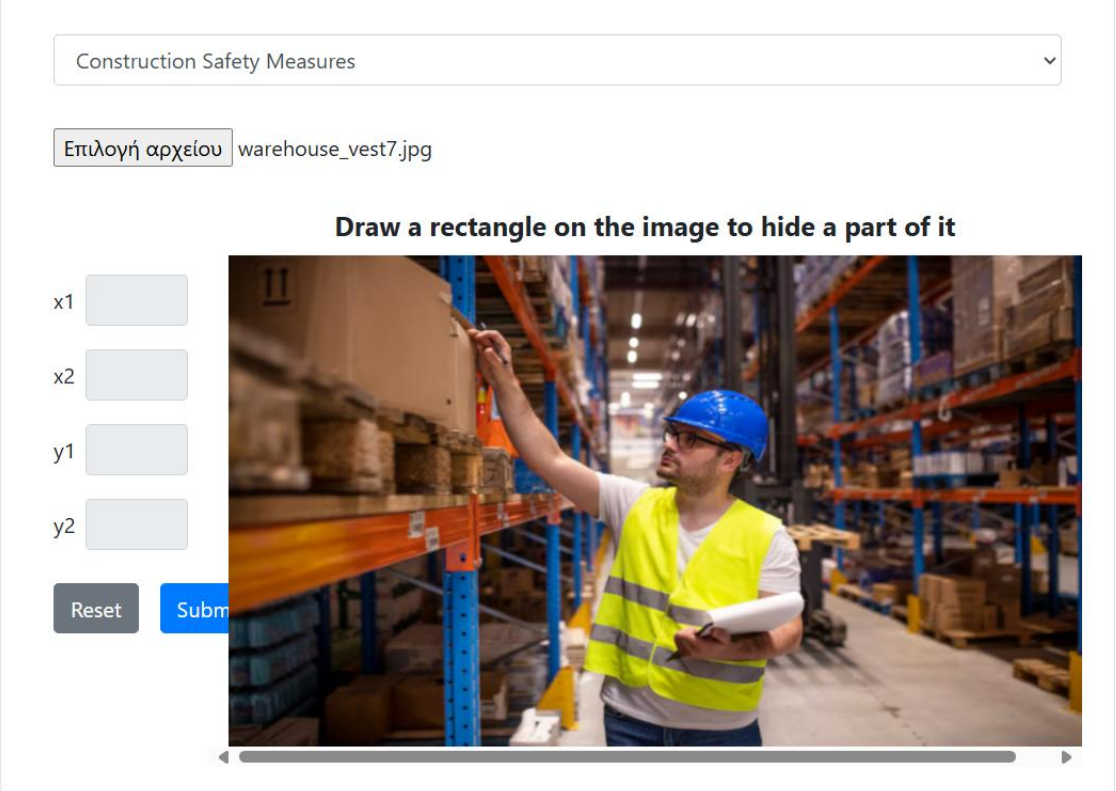

Figure 40 - The GUI of the TrL 4 Image fidelity metric

An obstructing bounding box was added to the input image (via the utilization of the graphical user interface – effectively by clicking at a point on the image and then dragging it to create a black rectangle as shown above) so that it covers most of the vest object. The model “successfully’ does not detect the vest object.

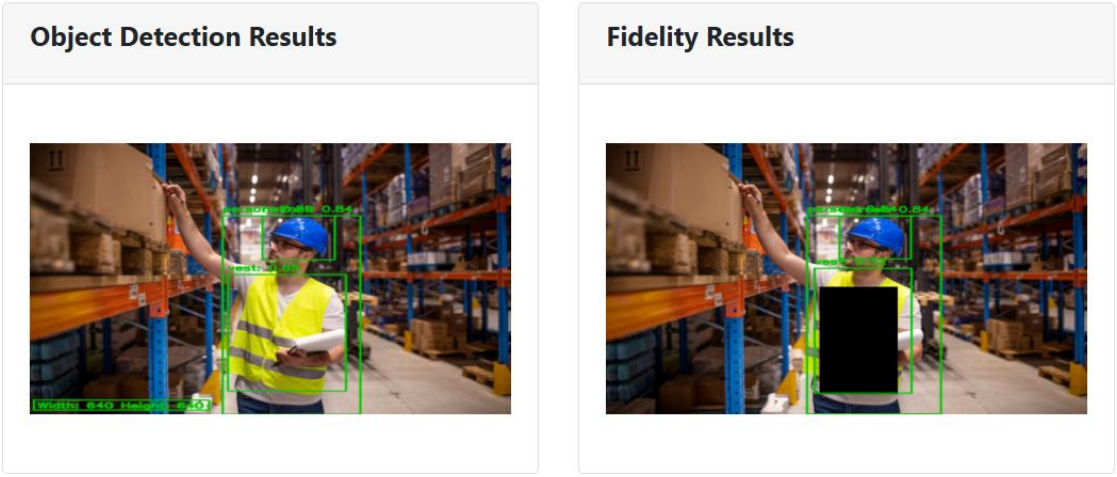

Figure 41 -Added an obstructing bounding box and the result is that the AI model can no longer detect the vest.

On the above figure, on the left hand side the input image is visible while on the right side an obstructing black box is added on the main features of the vest. The concept of image-fidelity is that as more of the key features of an object are obstructed, artificially or in any other way, the less will be the confidence of the model's output. This is the exact observed behaviour, as it is shown in the above figure. Specifically, the model's confidence without the obstruction is 0.85 while with the artifact in place, it drops down to 0.32. This observed behaviour indicates that the model bases its' decision on the important features of the object -the vest in this case-. Therefore, the image-fidelity metric's criteria are fulfilled.

The second metric that was defined in respect to object detection in still images is the "Image-Consistency" metric. This metric showcases the model's ability to recognize the same object under different conditions -such as different photometric phenomena and environmental conditions- proving the model's ability to retain the same accuracy under different scenarios. In its' core the Image-Consistency metric is about an AI model's ability to create robust and reliable results across different inputs and their states.

Through the graphical user interface that has been developed by MINDS, three parameters have been added towards artificially distorting the input image. These parameters are the following:

1. Brightness,

   This parameter helps the user of the web application to change the overall brightness of the image, either increasing a slider or decreasing it, with maximum value being two and minimum value being zero.

2. Contrast,

   This parameter helps the user change the overall contrast of the image, through the use of a slider either by increasing -up to a value of two- or by decreasing it -down to a value of zero.

3. Orientation,

   This parameter helps the user add rotation to the image in question. The available range is turning the image 360^{o} to the left or right, effectively the user can complete one full rotation of the input image in either direction.

In the following figure a specific (seemingly random) configuration of parameters was chosen for the “Construction Safety Measures”.

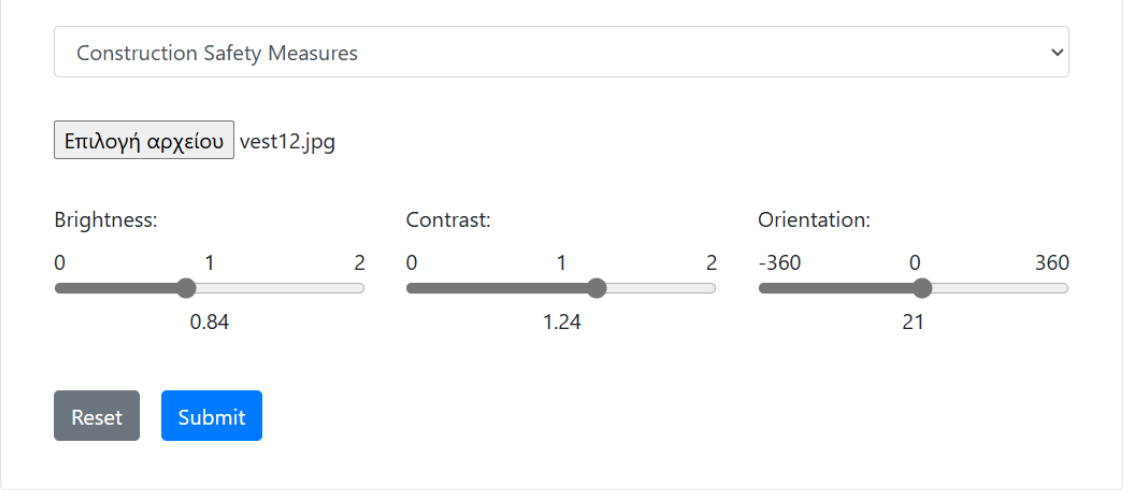

Figure 42 - Parameter configuration for "Image-Consistency" metric

The results of the object detection model and the image-consistency metric’s results are depicted in the next figure.

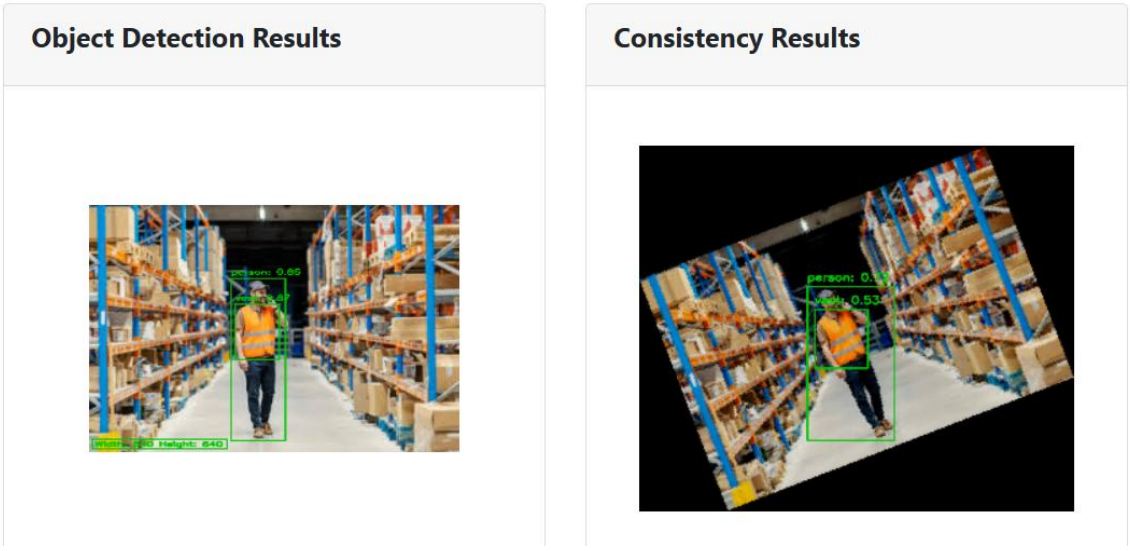

Figure 43- (Left) Normal inference output, (Right) inference results after applying the transforms

In the above figure a person wearing a reflective vest in a warehouse is visible. The model correctly recognizes these objects. After having applied the transformations as they are shown above it is apparent that the model’s confidence in its’ decisions has been reduced. This fact is due to the photometric and rotational transformations that were applied to the input image prior to showing the image to the model. This behaviour is expected, but despite it the model correctly detects the aforementioned objects therefore the assumptions of image-consistency metric stand.

Next, the configuration is shown similarly for the “Personal Protective Equipment” model and the same results are produced.

Figure 44 - The parameter configuration is depicted for PPE model

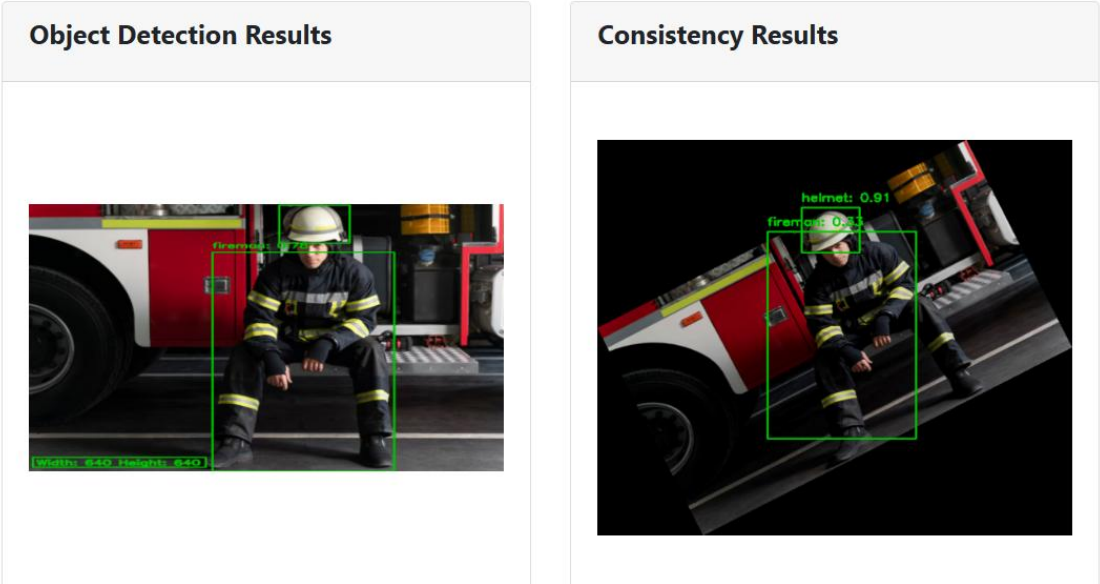

Figure 45 - Consistency results for PPE model after having applied the transformations

Same as with the construction safety measures model, in Figure 45 it is observed that the model's confidence falls as photometric and rotational transformations occur. This means that despite all the noise that is usually present in image inference data samples like these, the model can still detect the objects in the image but if the object is of small dimensions the model might miss the object all together. Despite these facts, the model still performs well as can be seen above, because despite the small disturbances that occurred in the input image, the model successfully detects the objects albeit with smaller confidence.

Finally, the same results were shown in the "Fire" object detection model as shown in the next two figures.

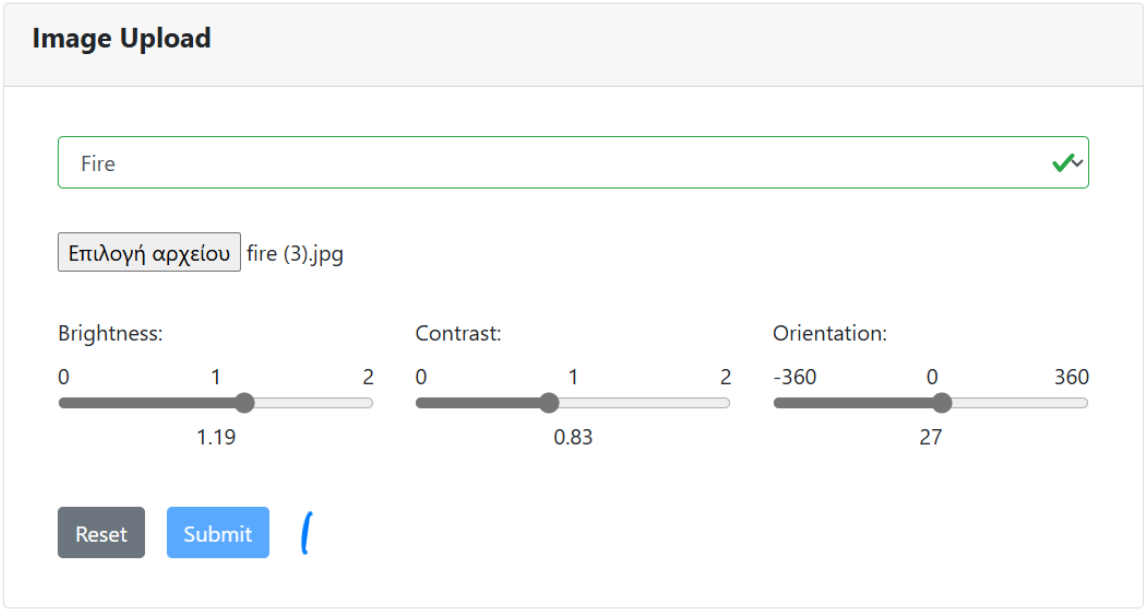

Figure 46 - Parameter configuration for "Fire" model

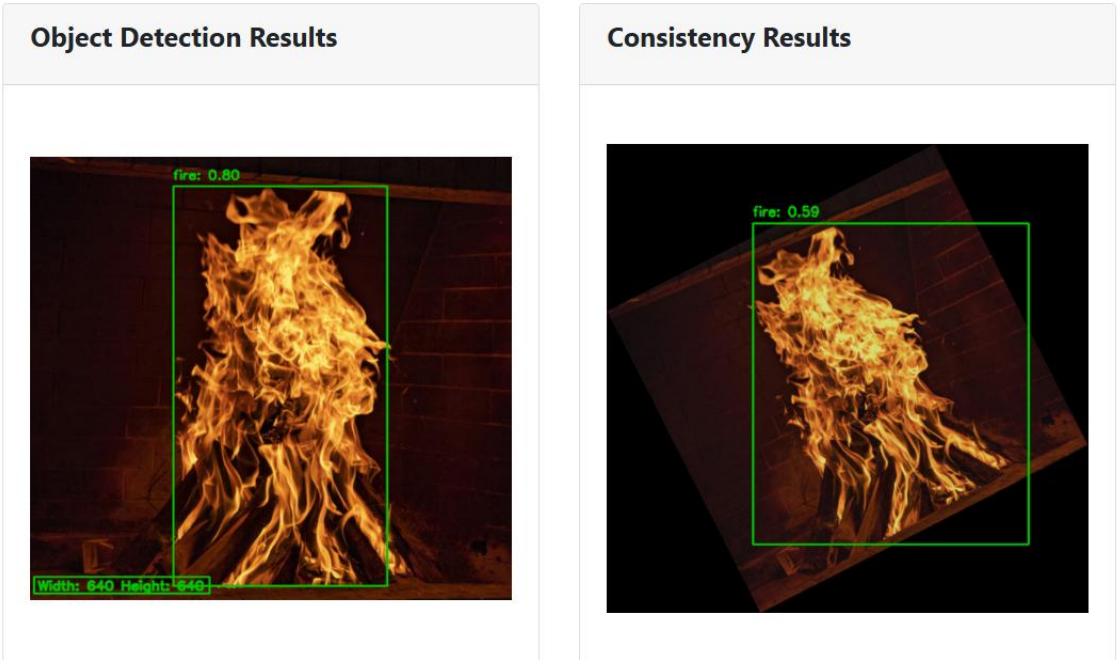

Figure 47 - (Left) normal inference results, (Right) inference results after applying the transforms

Again, like in the image-fidelity metric, in Figure 47 it is observed that due to the rotation of the input image as well as the photometric distortion the model's confidence in the prediction drops, but despite these aforementioned factors, the object is still successfully detected, probably due to its' large size. It has been also observed that for smaller objects – for example a fire object that is far in the distance – sometimes on the right hand side the model stops detecting them altogether.

Regarding UC1 on the task of wild fire detection from drones and in relation to TrL4, we introduced for fidelity an Explanation-Driven Adversarial approach. We harvested multimedia data collected from

drones at the edge and augmenting it with diverse and massive adversarial examples, we transparently trained explainers and robustified Neural Networks (NNs) to improve the AI model's fidelity. The robustified AI model performs exceptionally well against novel and unseen attack types and concept drifts. Also, through benchmarking across diverse adversarial attacks, we extended research in sensitive application domains and promote the adoption of more responsible and informed AI integration. The conceptual architecture of Adversarial Explanations is depicted in Figure 48.

This architecture was presented in the EEITE International Conference5 and differentiates to the methodology of UC4 by launching explanation-driven adversarial attacks using Adversarial Robustness Toolbox (ART) on the images. The AI models for explanations (i.e., YOLOv8 and Grad-Cam) remain the same. This approach merges the strengths of XAI and adversarial robustness and fosters trust between AI systems and human operators.

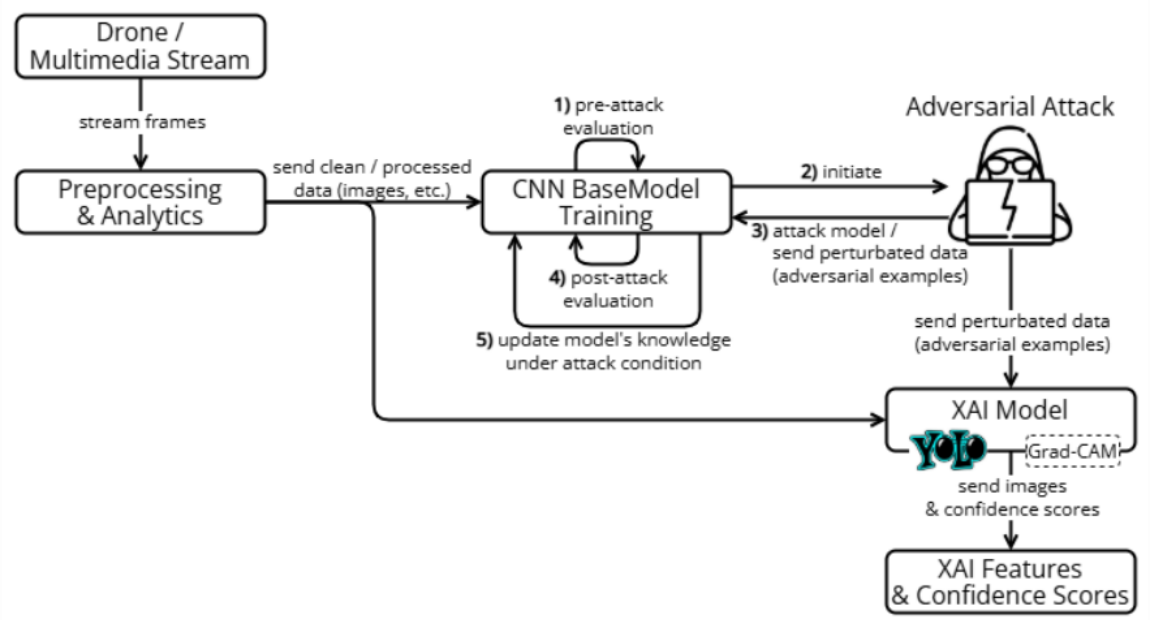

Figure 48: Adversarial Explanations Architecture

In Figure 49, we illustrate how additional adversarial examples emerge when the original image undergoes meticulous preprocessing and vigorous attacks. It becomes evident that while DeepFool and NewtonFool may initially appear to leave the data untouched, the reality is different regarding their impact on the model's performance.

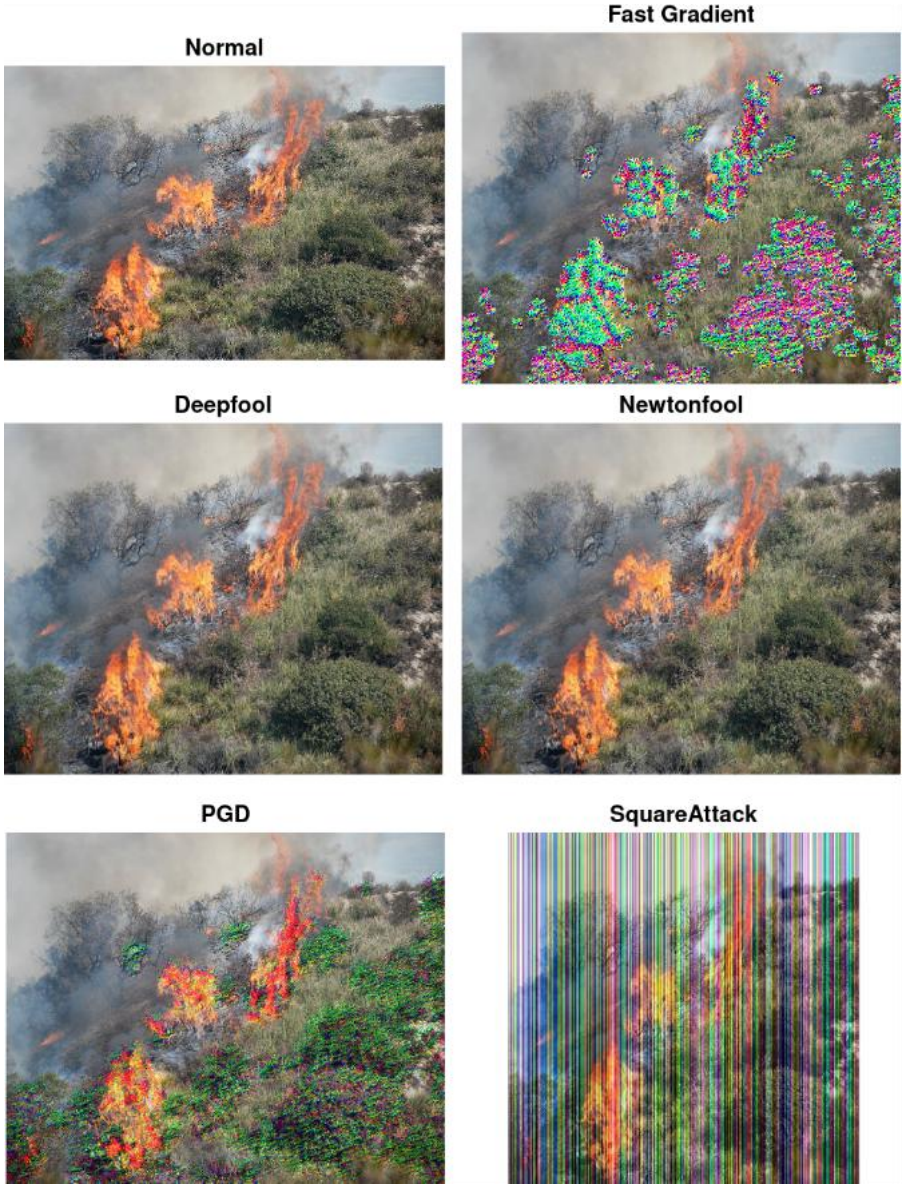

Figure 49: Attacked Image Instance

Adversarial Evasion (AE) attacks were first created for the imagery dataset on fire, where the main goal was to hide any alterations from human sight6. The adversarial attacks are categorized into white-box, where attackers have full model access for accurate gradient computation, and black-box, which rely solely on input-output queries. In our methodology, we employed three (3) different types of adversarial attacks, detailed below:

- **Optimization Attacks**: These attacks systematically distort an image and deceive a model while minimizing noticeable changes. They use mathematical techniques, such as gradient descent, to detect the least identifiable way of fooling the model.
- **Gradient-Based Attacks**: Gradient-Based Attackers exploit different levels of model access to manipulate its outputs using gradient-based methods to generate adversarial inputs. By taking advantage of model vulnerabilities during training and inference, they manage to create misclassifications or steal sensitive information.

Query-Based Attacks: These attacks interact with the target model to generate adversarial images with subtle perturbations, misleading the model in the process. They optimize these perturbations to

make them harder to detect. Decision-based attacks are more common than score-based attacks because they are more realistic.

In UC1, we experimented with several types of adversarial attacks. Table 21 summarises the main differences between DeepFool *[24]*, FGM[16], NewtonFool *[25]*, PGD *[26]*, and Square Attack, which we used to demonstrate and benchmark the robustness and fidelity aspects of the AI models. These attacks involve various trade-offs between speed, effectiveness, and computational cost, making them suitable for different scenarios based on the objectives and model defenses.

Table 21: Comparison of Attack Methods.

Method	Type	Access	Feature	Limitations
DeepFool	Optimisation-based	White-box	Iteratively pushes input toward decision boundaries	Less effective on deep, highly non-linear models.
FGM	Gradient-based	White-box	Single-step and fast pertubations	Weaker than multi-step attacks.
NewtonFool	Optimisation-based	White-box	Newton's method optimisation for minimal pertubations	Computationally heavy due to second order calculations.
PGD	Gradient-based	White-box	Iteratively applies gradients with projection for robustness	High computational cost from multiple iterations
Square Attack	Query-based	Black-box	Applies random square-shaped perturbations	Needs many queries; larger pertubations

For UC1, we employed specific configuration parameters to launch the attacks, as shown in Table 22.

Table 22: Configuration Params for Adversarial Attacks

Method	norm	eps	eps_step	decay	max_iter	targeted	Num_random	batch_size	nb_grads	eta	p_init	nb_restards
DeepFool	-	10^{-6}	-	-	10	-	-	1	10	-	-	-
FGM	np.inf	1	0.1	-	-	FALSE	5	-	-	-	-	-
Newton Fool	-	0	-	-	10	-	-	1	-	0.01	-	-

[16] *https://medium.com/@zachariaharungeorge/a-deep-dive-into-the-fast-gradient-sign-method-611826e34865*

PGD	np.inf	0.3	0.1	None	10	FALSE	0	5	-	-	-	-
Square Attack	np.inf	0.3	-	-	100	-	-	128	-	-	0.8	1

We now illustrate in detail the steps involved in explaining how adversarial attacks are initiated.

1. At first, we perform a pre-attack evaluation on the baseline model, which is then used to initiate our attack and generate the adversarial examples.
2. Secondly, a post-attack evaluation follows to assess the performance of the model on the perturbated data.
3. The resulting adversarial images (perturbated data) are then used to augment the dataset that includes both benign and attacked images and update the model's knowledge under attack conditions.
4. After generating the perturbated data, the images are sent to the XAI Model Module.

This enriched dataset, which includes two data classes, the "fire" (which represents typical fire occurrences with the attacked fire photographs) and the "non-fire" (which represents backdrop images devoid of fire) is used to build a CNN classifier as part of the XAI process. Using Grad-Cam to create a significance map of prominent pixels, the CNN classifier is built to withstand adversarial assaults and provide explanations even in the presence of attacked data examples.

Five classes of data were used to train YOLOv8: ``fire'' (which represents regular cases) and ``fgm'', ``deepfool'', ``newtonfool'', ``pgd'' and ``square assault'' (which represent various adversarial events). YOLO's bounding boxes and confidence scores give explainability and because it can identify and categorise the particular adversarial assault used, it also acts as an attack type detector.

4.2.2 Scientific and Technical Results

The technical and scientific results of this deliverable align with the initial goals of the Grant Agreement (GA). Specifically, in the pilot HRC -or Use Case 4 (UC4)- automated assessments of human-safety requirements such as detecting successfully whether personal protective equipment is worn by the humans in the area that is being surveyed, was put in-place. Colour cameras were positioned in various locations at an industrial setting, and were used to capture images of various scenes that sometimes include humans wearing (or not) protective equipment as well as other hazardous situations like when a specific item is on fire. Object detection models embedded on the system are capable to detect the use or not of the protective equipment from the technical stuff. TrI3 and TrI4 implemented under TALON dashboard is there to continue validate the model’s decisions by providing a transparent, interpretable and adversarial evaluation. In this way the Digital Twin has been connected to the real-world and can now act or help humans take better actions to improve the production and assembly lines’ efficiency. Also, through the use of constantly retrained AI-models, intuitive robot programming can occur to reduce or even avoid all together manual programming efforts.

Regarding UC1, as shown in Table 23, the NewtonFool ranks as the least effective among the tested attacks. The CNN Model initially has an accuracy of 85%, which drops to 42% after the attack. Although adversarial training does not fully restore robustness, NewtonFool consistently demonstrates lower efficacy, as the adversarial training does not restore the accuracy of the model (accuracy after adversarial training - 44%). Conversely, while DeepFool shows promising effectiveness with a decrease to 10% accuracy, the anticipated improvements from the robustification

process are not fully realized as the accuracy shows a light increase to 65% accuracy. Both the Fast Gradient Method (FGM) and PGD significantly impact CNN model performance, with a decreased to 10% of the accuracy of the base model 90% and 87% respectively, while the robustification mechanism achieves strong results in these cases as it manages to regain the accuracy of the models from 10% to 88% for FGM and to 86% for PGD.

Table 23. Accuracy of CNN Training, Attacked, and Robustified Models

Attack Category	CNN Model	Attacked Model	Robust CNN
DeepFool	0.85	0.10	0.65
FGM	0.90	0.10	0.88
NewtonFool	0.82	0.42	0.44
PGD	0.87	0.10	0.86
Square Attack	0.87	0.14	0.63

In the following paragraphs, we present the YOLO results alongside the explanation outcomes with confidence scores. The results are summarized in three (3) tables. Table 24 establishes the baseline performance evaluation on the TALON datasets and NN models. ResNet50 achieves an accuracy of 67%, while YOLOv8 (small and large) yield mAP50 scores of 0.446 and 0.478.

Table 24. Baselines YOLOv8 & ResNet50

	Accuracy	mAP50
ResNet50	0.67	-
YOLOv8 small	-	0.446
YOLOv8 large	-	0.478

Table 25 compares the adversarial explanation results on the UC1 data. Notably, the CNN-Res50-Robust-Classifier attains high performance with 98% accuracy and 97% precision, outperforming the YOLOv8 models—which register lower accuracy (58–62%), precision (72–79%), and mAP50 (0.25–0.30).

Table 25. Adversarial Explanation Results

Model	Accuracy	Precision	mAP50
CNN-Res50-Robust-Classifier	0.98	0.97	-
YOLOv8 small	0.58	0.72	0.25
YOLOv8 large	0.62	0.79	0.30

Finally, Table 26 details how YOLO's confidence scores vary under different adversarial attacks. While the model maintains relatively high performance under normal conditions (with a confidence score of 0.76), its performance drops significantly for certain attacks (e.g., DeepFool, with a score of 0.39), illustrating the YOLO framework's vulnerability to adversarial perturbations.

Table 26. YOLOv8 Confidence Score along Different Attacks

Class	Precision	Recall	Conf.Score
Normal Fire	0.89	0.90	0.76
Deepfool	0.43	0.10	0.39
Fast Gradient	1.0	0.55	0.52
Newtonfool	0.44	0.51	0.40
PGD	1.0	0.86	0.74
Square Attack	0.98	0.83	0.72

4.2.3 Instantiation with pilot data

Use Case 4 -the HRC pilot- image datasets were captured in a real-world environment, specifically in an industrial setting. These datasets contain around four hundred (400) images captured at an industrial workspace under various photometric conditions and various physical locations. The images also contain various types of objects, some of which include protective equipment for industrial workers such as vests and helmets. Additionally, the AI models utilized have been fine-tuned using open-source datasets to contain no-vests & no-helmets labels in the case where a person is not wearing the respective equipment. The goal for this dataset was to create AI models that are consistent -meaning the models can detect the same objects in a specific scene regardless of the photometric and environmental conditions that influence parameters such as brightness, contrast and rotation of the image in question-. Also, these models should have high fidelity -meaning they should be able to detect an object only whenever the most important features of the object are being shown in the image. For example, if the surface area of an object is obstructed by another object or some other artifact by a large percentage and to the point where the object is not recognizable by a human, then the model should also fail to detect the object. This expectation is valid and the expected behavior is correctly exhibited by the models as it was depicted in the previous section. In order to fulfil the initial goal of creating a mixture of humans and robots working in tandem and in close proximity at an industrial setting, the robots must learn to carefully handle themselves around humans. This has been realized during TALON's UC4 through effectively teaching the robots via a centralized computer vision task utilizing AI models to detect certain objects of interest in captured images and at the same time fulfilling the constraints of high image fidelity and consistency.

Through the XAI models and techniques which were realised in this deliverable, production strategies such as successfully detecting humans nearby and (for example) whether they wear protective equipment or not, can now be employed in the field. This new type of insight can in turn help make the industrial workplaces smarter and safer both for humans as well as for robots. More specifically, YOLOv8 can be utilized in the context of TALON project and the UC4 pilot as a way to detect potentially hazardous situations, notify the appropriate actors about these situations and take measures to mitigate them as well as provide explainable results to back the decisions taken. Especially useful is the utilization of TrL3 in the UC4 pilot as a way to explain these very same decisions.

TrL4 also contributes to this very same outcome through its' use, when affine transformations (rotations) or photometric phenomena are applied to the input images for some reason – for example because one of the camera-sensors is mechanically flipped over or rotated before capturing an input image or because the image is taken at sundown or in the early morning- TrL4 can be utilized to explain the decisions that were taken by the trained AI models, and thus enable the actors produce reliable (consistent and with high-fidelity) as well as explainable results.

TrL3 and TrL4 help shed some light on whether the AI model used in each case give appropriate attention to the critical areas. TrL3 showcases this through heatmaps shown in the previous sections (the right hand side of the images in question). Another aspect of the AI model that is scrutinized using TrL4, is whether the model can still perform as expected while having access to less info (for example when something obstructs partially the object that the model is trying to detect). TrL4 in addition to TrL3 can show whether there are declines in the performance of the model in which case the actor should revisit the training of the AI model to improve its' inference performance.

Next, three examples are shown drawn from the UC4 pilot dataset (CERTH dataset) showcasing the above observations. Namely:

1. Explainability of the model's decision (TrL3):

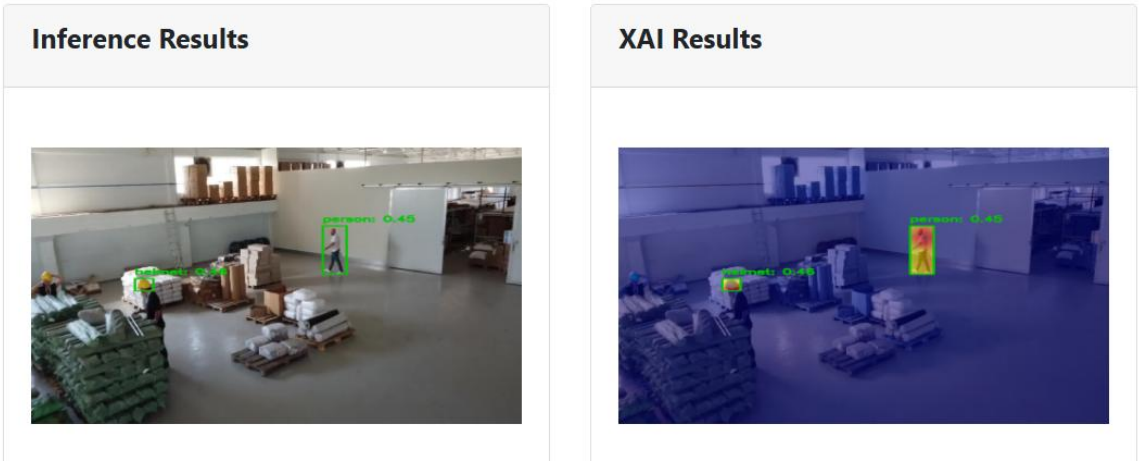

Figure 50 - An example of TrL3 to the model's inference results

Above is a quite difficult image for the model to detect the objects that lie within the image, because the three people that are depicted are obstructed or are too far away in the distance. Nevertheless, the Grad-Cam algorithm successfully shows the important input pixels that correlate to the model's inference decisions. As it is depicted above, the detected helmet is coloured red on its' most important features. Also, the person in the back is coloured red at the feet, torso and head, while yellow in the vicinity of these features. Finally, it is worth mentioning that although this example is a difficult case for the AI model to successfully detect the corresponding objects that exist in it, the correlation between the input pixels, the output of the algorithm and the XAI results are consistent.

2. Image-fidelity metric (TrL4):

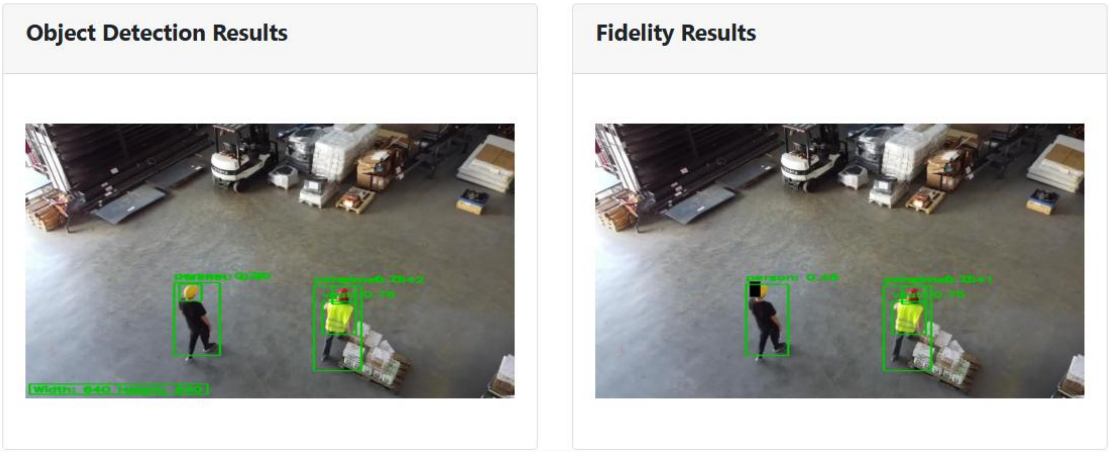

Figure 51 - (Left) Input image to TrL4 without obstructions, (Right) input image fed to TrL4 with an obstructing artifact that makes the helmet on the person on the left undetectable

Above is a very good example of the use case. A black rectangle was placed on top of the left person's helmet, which led to the important features of the helmet to be obstructed. This cased the model to not detect the object due to the fact that the important features of it were missing. Again, we can see that the results of the XAI and the model's results are consistent.

3. Image-consistency metric (TrL4):

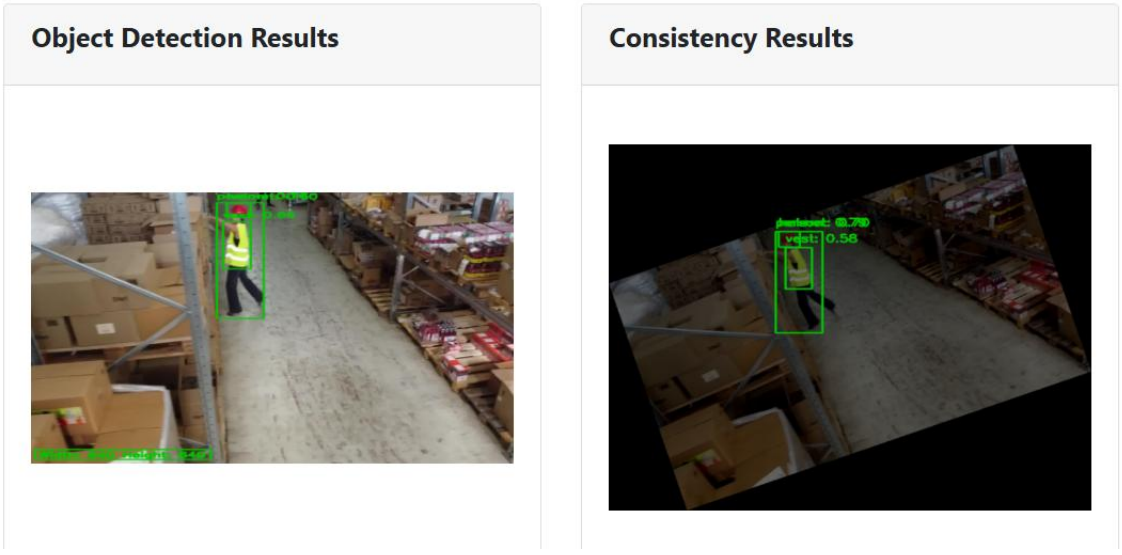

Figure 52- image-consistency results

In the above image it is observed that after applying a seemingly random mixture of rotation, contrast and brightness parameters to the input image, while the model is capable of successfully detecting the same objects in both cases, due to the transformation of the input image in the second case the model's confidence drops as it becomes more difficult to recognize these objects. This is the required and assumed behaviour.

Regarding UC1, we can see the explanations of both YOLO and the Resnet-classifier with Grad-CAM. In Figure 53: YOLOv8 Adversarial Explanations on UC1 Data Instances., YOLO successfully highlights the fire-affected area even under attack while also identifying the type of attack along with a confidence score.

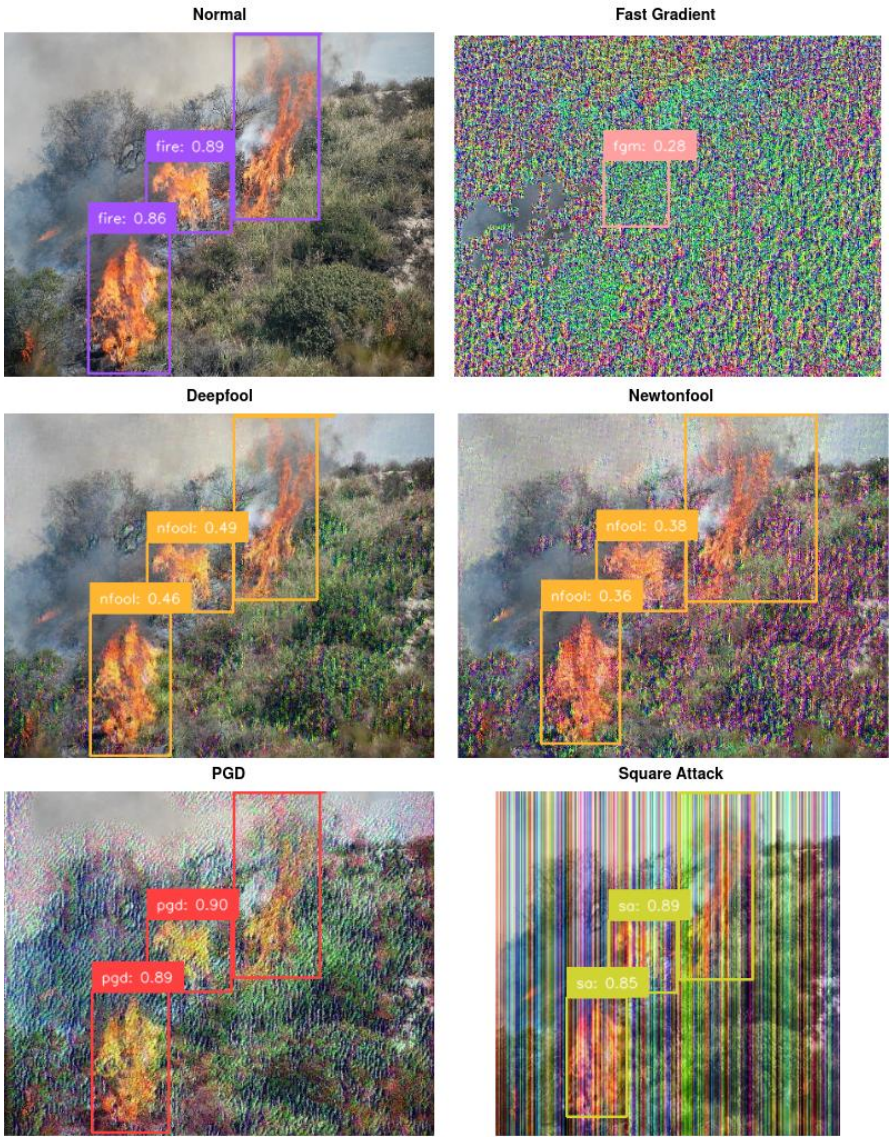

Figure 53: YOLOv8 Adversarial Explanations on UC1 Data Instances.

Figure 54: Grad-Cam Adversarial Explanations on UC1 Data Instances.showcases the explanation robustness of the ResNet classifier, where the fire-related explanations remain valid even under adversarial attacked data instances. This robustifies the models, as even under attack the model and the explanations still work well, and help us understand also how different attacks behave on the models.

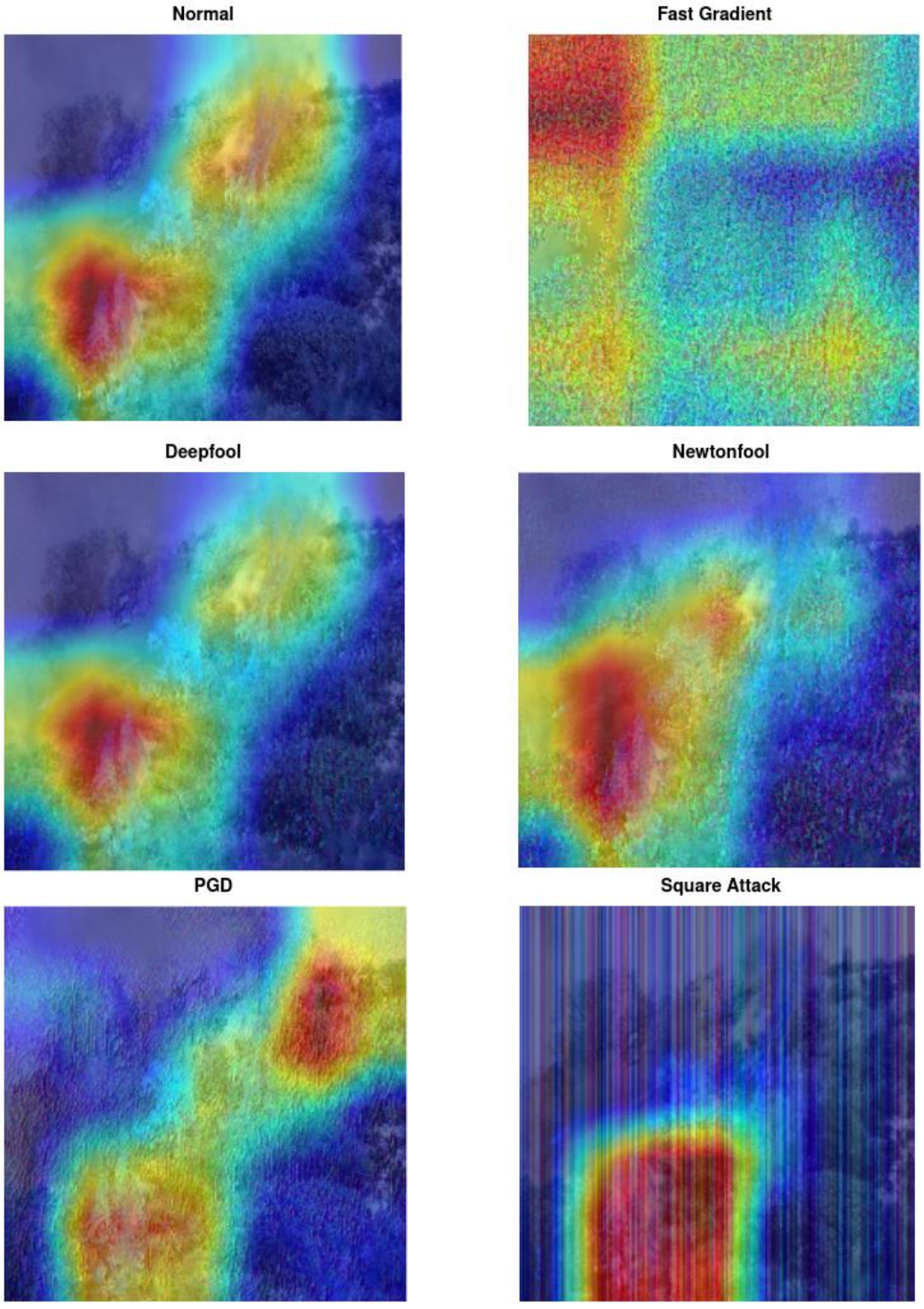

Figure 54: Grad-Cam Adversarial Explanations on UC1 Data Instances.

These figures depict the varying impacts of different attacks on the images. The FGM attack stands out as it creates significant confusion to the AI models, resulting in difficulties in accurately identifying the presence of fire. In comparison, the DeepFool and NewtonFool attacks exhibit similar effects, leading to a less pronounced but still notable challenge in image interpretation. This emphasizes the distinct ways in which each attack affects the models' ability to process visual information.

The adversarial explanations from the YOLOv8 offer valuable insights on potential attacks contributing to greater fidelity and trust. By combining the adversarial explanations with Confidence

Scores, this approach provides a promising solution for applications demanding exceptional reliability in object detection and image classification. Furthermore, this research work demonstrates that explanation-driven adversarial attacks are an effective approach to enhancing the security of computer vision applications. By integrating clear and interpretable adversarial insights with robust confidence scoring, the robustified models not only identify existing vulnerabilities but also provide actionable awareness to improve model resilience.

5 Data and AI Models Trustworthiness

5.1 Textual, Tabular & Numerical Anonymisation

Regarding the anonymization of textual, tabular, and numerical data, no additional developments or updates are reported in this deliverable beyond what was already presented in D4.1. This is due to the fact that none of the use cases within TALON involved data of these types that required anonymisation. As such, this corresponding module described in D4.1 remains as they were finalised at that stage, with no further modifications or refinements. Given the absence of applicability across the TALON pilots for these specific data modalities no additional implementation, evaluation and adjustment activities were conducted in WP4 in this area.

It is important to note that while textual, tabular and numerical anonymisation are critical components in many contexts, in TALON's case, the focus of anonymization efforts shifted toward image and facial data addressed in the following section where ethical, privacy and technical considerations were more directly relevant to the project's real-world pilots and use case needs.

5.2 Image Anonymisation

5.2.1 Key Technologies & Technical Updates

The Image Anonymization Module (ImAM), one of the mechanisms within the Talon Access & Security layer, was previously described in Deliverable D4.1, Section 5.2.

5.2.2 Scientific and Technical Results

Similiraly, the scientific excellence and initial technical competence of this module has been presentend in D4.1

5.2.3 Instantiation with pilot data

This final WP4 deliverable presents the results of the module's integration into two use cases: UC3 – AR Maintenance Application and UC4 – Human-Robot Collaboration.

UC3 - AR Maintenance Application

In the AR Maintenance use case, a communication solution was developed for maintenance and support personnel operating in an industrial environment. The system is enhanced with AI models for real-time scene analysis.

To ensure compliance with current and upcoming regulations such as the EU AI Act and GDPR, the application integrates the Anonymization Layer of the TALON platform (see Figure 55). This layer enables the detection and removal of personally identifiable information from visual data, while preserving key contextual elements essential for AI processing — such as object detection.

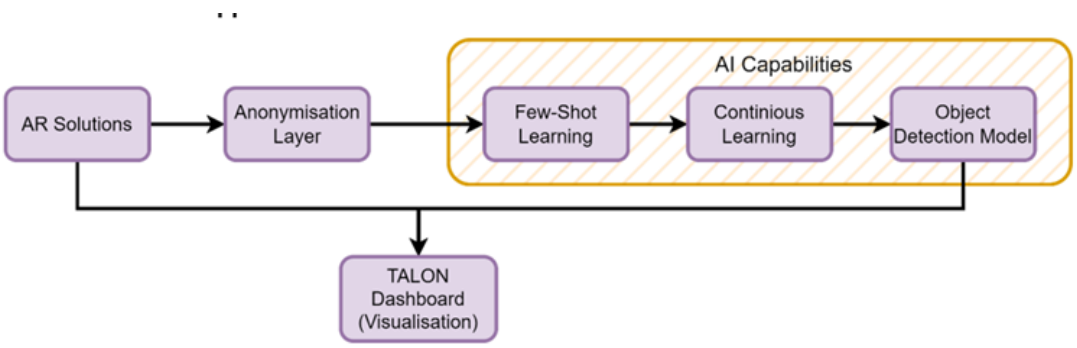

Figure 55: Overview of AR-based Maintenance Application

In this scenario a maintenance operator wears augmented reality glasses (Figure 56) to receive information on visors about complex machines to maintain, as Nakamura machine.

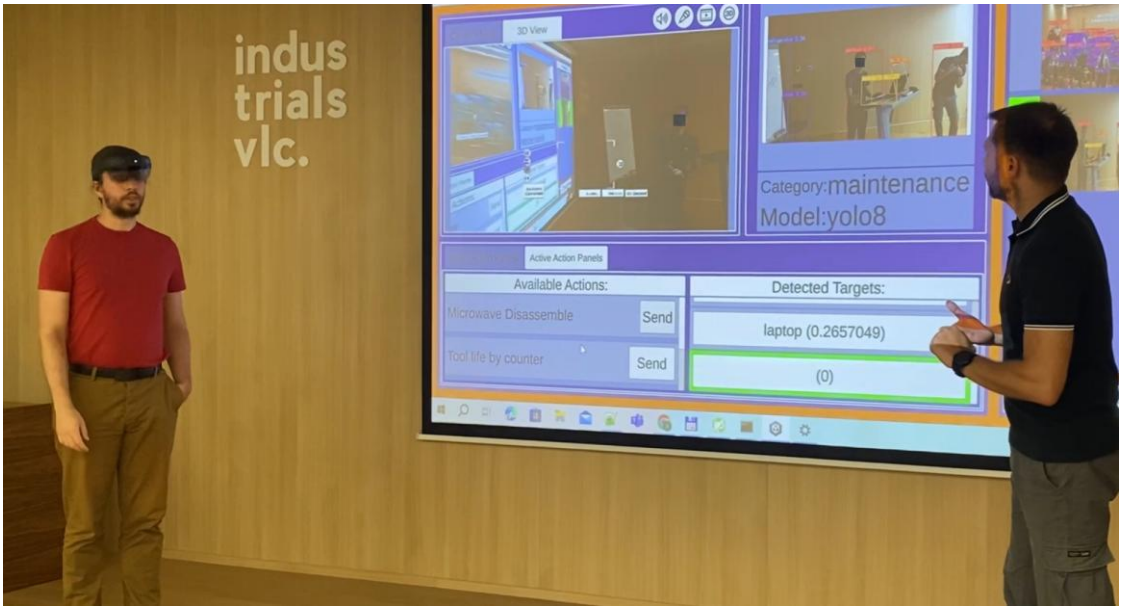

Figure 56: AR glasses worn by an operator

During the operation, several images of the environment are captured. To protect the privacy of any operators appearing in these images, the Image Anonymization Module automatically removes visual sensitive information. Examples of anonymised frames are shown in Figure 57.

Figure 57: Privacy granting of operators present in the footage

The AR Maintenance use case has also been conceived for other operational scenarios, such as firefighting, where AR glasses could assist in detecting the presence (or absence) of Personal Protective Equipment (PPE) on field personnel.

This scenario was simulated using static images, in order to validate that the application is still capable of performing accurate PPE detection even when working with previously anonymized images (see data flow in Figure 55).

The results demonstrated that the anonymization process did not impair the AI's object detection capabilities. Example of an anonymised frame is shown in Figure 58.

Figure 58: Anonymized scene

UC4 – Human-Robot Collaboration

In the Human-Robot Collaboration use case, drones stream video from designated warehouse zones to verify whether workers are wearing the required Personal Protective Equipment (PPE). If PPE compliance is not detected, an alert is triggered and displayed on the TALON dashboard (UI).

In this scenario, the Image Anonymization Module ensures the removal of sensitive information from the visual data captured by the drones.

A simplified representation of the data flow is provided in Figure 59.

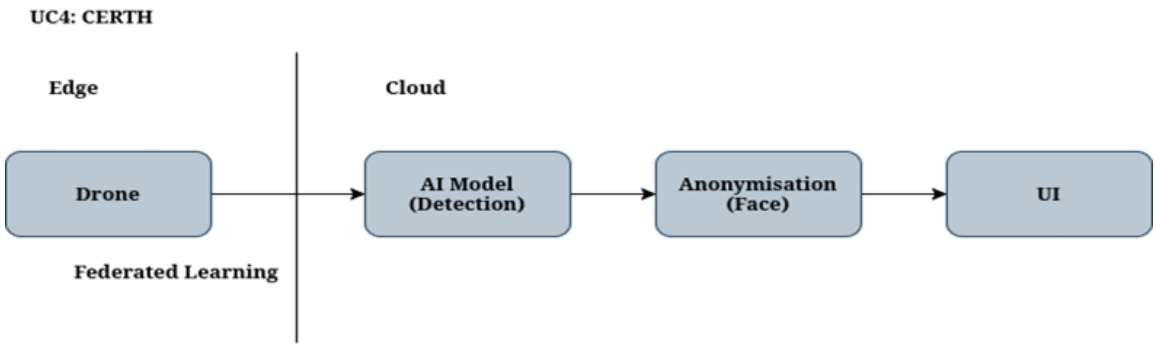

Figure 59: scenario pipeline

Figure 60, illustrating the AI model's detection output, highlights the critical need for image anonymization. Drones operate continuously within their assigned zones, recording workers without default privacy-preserving mechanisms.

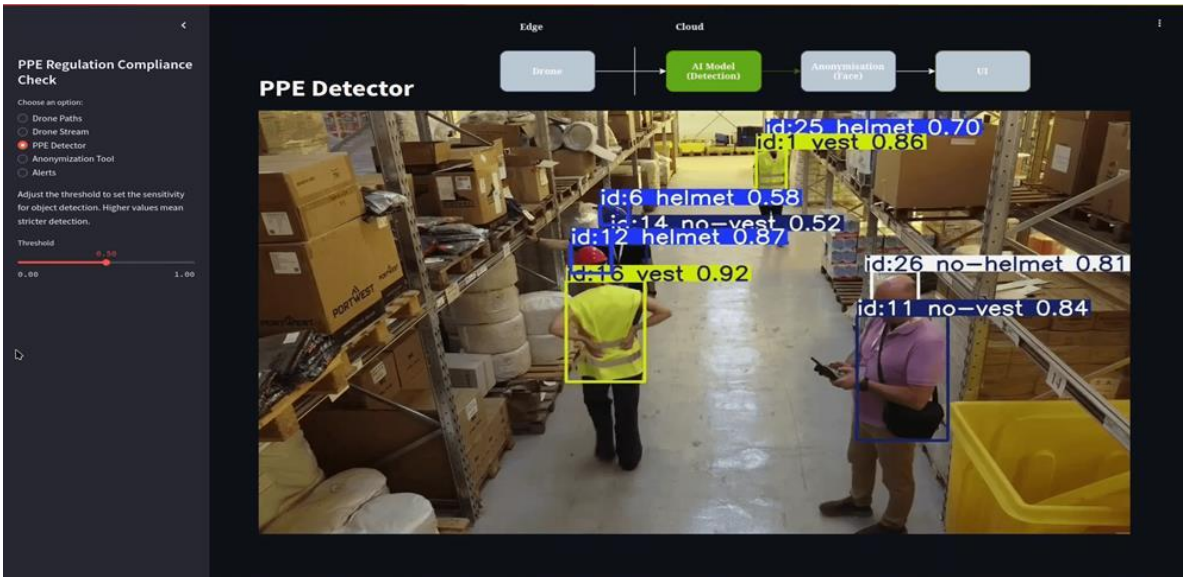

Figure 60: data stream from drones

The Image Anonymization Module intercepts and processes the video stream, applying privacy-preserving techniques to anonymise any identified individuals (e.g., by masking faces or blurring personal identifiers), Figure 61.

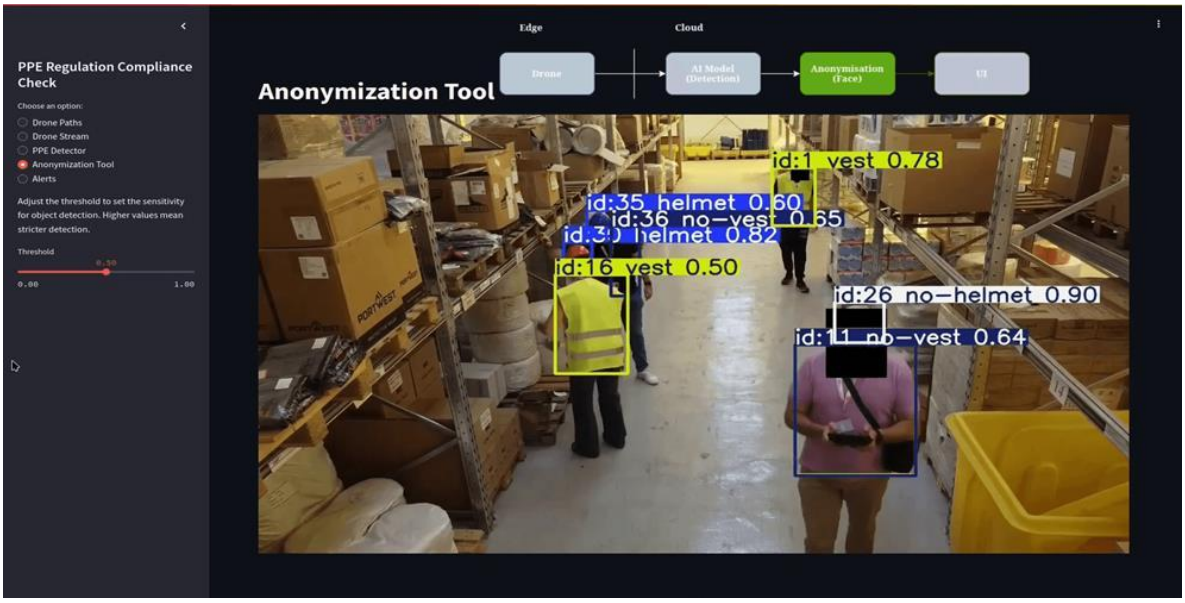

Figure 61: Anonymisation tool obscures identities

The anonymised stream is then transmitted to a centralised safety dashboard, Figure 62. Since all personally identifiable information has been removed, the video feed can now be safely monitored by authorised personnel (e.g., safety supervisors), in full compliance with the GDPR — particularly Article 5(1)(c), which mandates that personal data be limited to what is necessary for the intended purpose.

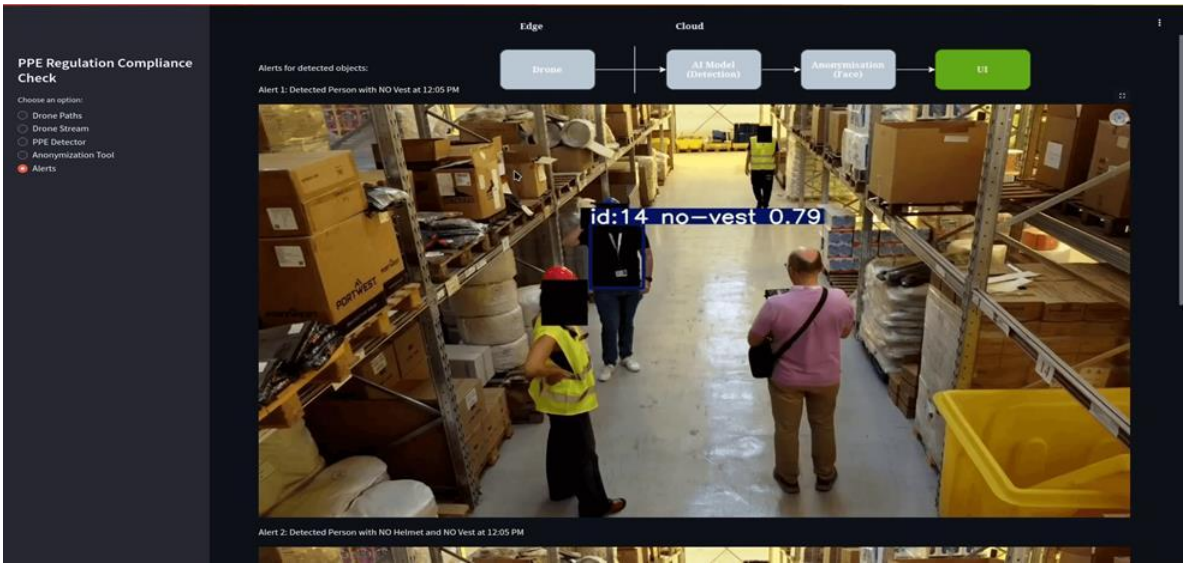

Figure 62: Dashboard generates alerts on non-compliance, preserving privacy

5.3 Federated Learning Framework

5.3.1 Key Technologies & Technical Updates

The TALON project's Federated Learning (FL) framework represents a distributed machine learning approach that enables model training across various devices while maintaining data localization. Building upon the theoretical foundation and architectural concepts established in TALON Deliverable D4.1, this work demonstrates significant technical progress from initial concept to fully operational implementation. D4.1 introduced the FL framework's internal architecture and key aggregation technologies, along with communication interfaces using the Flower framework. The initial deliverable presented preliminary demonstrations using simulated scenarios, establishing baseline performance metrics across distributed nodes. The current implementation in D4.3 advances beyond these foundational concepts by delivering a production-ready containerized system with comprehensive real-world validation, blockchain integration for audit trails, advanced dashboard monitoring capabilities, and seamless deployment across distributed infrastructure. Where D4.1 established the theoretical framework and initial proof-of-concept demonstrations, D4.3 presents a mature, scalable platform validated through industrial pilot deployments with real-world datasets from CERTH and enhanced safety monitoring applications. The framework addresses critical challenges in privacy preservation, data security, and bandwidth optimization through its core principle of model aggregation without requiring the exchange of local data samples. The technical advancements from D4.1 to D4.3 represent a substantial evolution from theoretical design to practical industrial deployment capabilities.

The FL framework utilises Federated Averaging (FedAvg) [27] as the primary aggregation technique. FedAvg aggregates model updates from multiple clients to update a global model by computing the weighted average of local model updates based on the number of data points on each device:

$$w_{global}^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^{t+1}$$

where w_{global}^{t+1} represents the global model weights at each iteration, K is the total number of devices, and n_k is the total number of data points on device K. This aggregation strategy ensures that devices with larger datasets have proportionally greater influence on the global model while maintaining the distributed nature of the learning process.

The selection of *FedAvg* over alternative federated aggregation methods was driven by several practical considerations specific to the TALON platform's requirements. While more sophisticated algorithms such as *FedProx* or *FedAdam*, offer theoretical advantages for handling system heterogeneity and adaptive optimization, *FedAvg* provides the optimal balance of computational efficiency, implementation simplicity, and proven effectiveness for the project's industrial use cases. The algorithm's straightforward weighted averaging approach minimizes computational overhead on edge devices, which is crucial for deployment in resource-constrained industrial environments. Additionally, FedAvg's extensive validation across diverse machine learning applications provides confidence in its reliability for the TALON platform's object detection and time series prediction tasks. The decision prioritizes robustness and practical deployment considerations over theoretical optimization, ensuring that the FL framework can operate effectively across the heterogeneous industrial infrastructure that characterizes real-world TALON deployments.

The implementation leverages Flower as the primary FL framework, providing standardized public and accessible APIs and communication protocols for distributed machine learning. Flower handles client-server orchestration, asynchronous updates from nodes, and model parameter aggregation through its robust communication infrastructure. The server implementation uses Flower's FedAvg strategy with initial parameters derived from a pre-trained YOLOv8 model, establishing a solid foundation for the federated training process. The framework is specifically implemented for YOLOv8n object detection models, utilizing the Ultralytics library for model training and inference. This integration supports both CPU and GPU training environments with automatic device selection, ensuring optimal performance across heterogeneous hardware configurations. The system implements efficient parameter serialization and deserialization mechanisms where model state dictionaries are converted to NumPy arrays for transmission between clients and server, with proper parameter restoration using *PyTorch's load_state_dict* functionality.

Communication between clients and the central server operates through Flower's built-in protocols, enabling transmission of model updates from distributed clients to the aggregation server. Clients transmit model parameters to the server, which aggregates these updates using the FedAvg algorithm and redistributes the enhanced global model back to participating clients. This bidirectional communication ensures that all clients benefit from the collective learning across the entire network. The implementation includes integration with the TALON Dashboard through REST API endpoints, enabling real-time monitoring of the FL process. Clients automatically post training metrics including precision, recall, mean Average Precision (mAP) scores, and various loss values after each training round. The dashboard receives structured JSON data containing detailed client performance metrics, providing stakeholders with visibility into the training progress and model performance across all participating nodes. The relevant data sent from the clients to the TALON Dashboard are *precision, recall, mAP50, mAP50-95, box_loss (both val and train), cls_loss (both val and train)*. These data are crucial in determining successful from failing experiments, and incorporating the appropriate models.

The framework also incorporates blockchain mechanisms for audit trails and model update verification through integration with SIDROCO's blockchain asset management system. This integration enables secure tracking of model evolution across training rounds, providing immutable

records of the FL process and ensuring transparency in model development. The core identifying features communicated between clients and the asset management system are: *asset_id, weights,*

accuracy and *trainingRound*. These data are important in maintaining trustworthiness across experiments by utilizing the immutability and transparency of the IPFS system.

The TALON FL framework is implemented using *Docker* containers for both server and client components, providing a containerised deployment approach that enables practitioners to execute FL experiments with minimal setup requirements while ensuring scalable deployment across truly distributed infrastructure. This containerised approach supports dynamic client participation, resource isolation, and simplified deployment across heterogeneous computing environments, with the capability to deploy FL nodes in Barcelona, Rome, and Thessaloniki, while coordinating with a central server located in Amsterdam, for example. The geographical distribution flexibility allows industrial partners across different European countries to participate in collaborative model training without data leaving their local premises, regardless of where individual nodes are located. This distributed architecture enables real-world scenarios where a drone surveillance system in one country can contribute to safety model training alongside edge devices from multiple other European locations, all coordinated through a single FL experiment while maintaining complete data locality and privacy compliance with regional regulations. The system supports configurable minimum client requirements with a default of two clients and allows for flexible federated round configuration based on per-case needs. The current implementation uses YOLO-compatible object detection datasets supporting train, validation and test splits, and has been demonstrated using industrial environment safety dataset provided by CERTH (classes: *person, vest, no-vest, helmet, no-helmet*.

Integrated metrics collection captures comprehensive model performance indicators throughout the training process. The system automatically logs mean Average Precision(mAP), recall, precision, training loss, validation loss, and classification loss metrics. These performance indicators are systematically transmitted to the TALON Dashboard, enabling continuous monitoring and evaluation of the FL process. Prior to FL experiment execution, the TALON FL Dashboard must be operational to enable real-time monitoring and metrics collection. The dashboard provides essential visualization capabilities for tracking experiment progress and model performance across distributed clients.

The FL process follows a server-first initialisation pattern, with clients connecting to the established server instance. The minimum deployment configuration requires **one server** and **two clients** for effective federated training. The server accepts multiple configuration parameters to execute the experiment and proceed with the TALON Dashboard updates.

Table 27: Server Initialization Parameters

Parameter	Description
-s (str)	Server network address. Change according to the actual server IP address.
-p (str)	Network port for client communications. Adjusted according to the real server port.
-m (str)	Initial model file path (YOLOv8 format).
-r (int)	Total number of federated training rounds for this experiment.
-i (int)	Unique experiment identifier for dashboard tracking.

An example **server** initialization command according to Table 27 would look like:

```
docker run --network=host flower_server -s "<server_ip>" -p "<server_port>" -m "<model_path>" -r 3 -i 3
```

Client deployment follows server initialization, with clients deployed in parallel across distributed nodes. Each client requires unique identification and dataset access, accepting parameters such as local dataset configuration file path and FL server network address within the current experiment.

An example **client1** initialization command according to Table 28 would look like:

```
docker run --network=host --rm -v ./cl1-data:/data --name client1
flower_client -d "<client1_dataset_yaml_path>" -s "<server_ip>" -p
"<server_port>" -m "<model_path>" -e 2 -u
"<dashboard_api_url>/store_round" -i 1 -c 1
```

For subsequent client nodes, *CLIENT_ID* must be incremented while maintaining the same *EXPERIMENT_ID*. Example for **client2*:***

```
docker run --network=host --rm -v ./cl2-data:/data --name client2
flower_client -d ""<client1_dataset_yaml_path>" -s "<server_ip>" -p
"<server_port>" -m "<model_path>" -e 2 -u
"<dashboard_api_url>/store_round" -i 1 -c 2
```

Each subsequent client (client3…clientN) can join the federated process by incrementing the integers ***N*** in the parameters *-v ./cl**N**-data:/data, and -c **N*** in the above command.

Table 28: Client Initialization Parameters

Parameter	Description
-d (str)	Local dataset configuration file path.
-s (str)	FL server network address. Change according to real server IP address. E.g.: *127.0.0.1*
-p (str)	FL server communication port.
-m (int)	Local model file path.
-e (int)	Local training epochs per federated round.
-u (str)	Dashboard API endpoint for metrics transmission. E.g.: *http://localhost:8086/store_round*
-i (int)	Experiment identifier for tracking coordination.
-c (int)	Unique client identifier within the experiment.

The containerised deployment automatically manages training artifacts through volume mounting. Each client generates training results in dedicated directories *(./cl[1..N]-data)* that persist on each client machine beyond container lifecycle. Additionally, performance metrics are automatically transmitted to the TALON Dashboard system upon completion of each training round, enabling real-time monitoring of the FL process across all participating nodes. The final aggregated and trained model is distributed and saved inside each client with the name *best.pt* and can be used as a checkpoint for any future experiments. This deployment approach ensures scalable FL experiment execution while maintaining data locality and enabling comprehensive monitoring of distributed training performance.

5.3.2 Scientific and Technical Results

A comprehensive demonstration has been implemented to showcase the TALON platform's FL capabilities for object detection applications. The demonstration provides empirical validation of the framework's effectiveness in distributed learning scenarios, demonstrating the platform's adaptability to industrial computer vision applications while maintaining consistent performance across heterogeneous client environments. The experimental setup involved multiple federated clients operating in a distributed environment, with each client processing distinct portions of the dataset while contributing to a shared global model. The demonstration was designed to validate the platform's ability to facilitate effective FL rather than to engineer the most superior individual models, emphasizing the collaborative learning capabilities inherent in the federated approach.

Figure 63 illustrates the comprehensive FL architecture implemented in the TALON demonstration. The system encompasses multiple interconnected components including distributed edge devices such as drones, FL nodes for local model training, a central FL server for model aggregation, blockchain integration for secure audit trails, and comprehensive user interfaces for monitoring and human-in-the-loop annotation processes. The numbered workflow demonstrates the complete FL cycle from data collection at edge devices through local training at distributed nodes, central aggregation at the server, blockchain-secured model versioning, and real-time monitoring through dedicated dashboards. This architecture provides the foundation for validation conducted in the demonstration, showcasing how industrial edge devices can participate in collaborative machine learning while maintaining data privacy and system transparency.

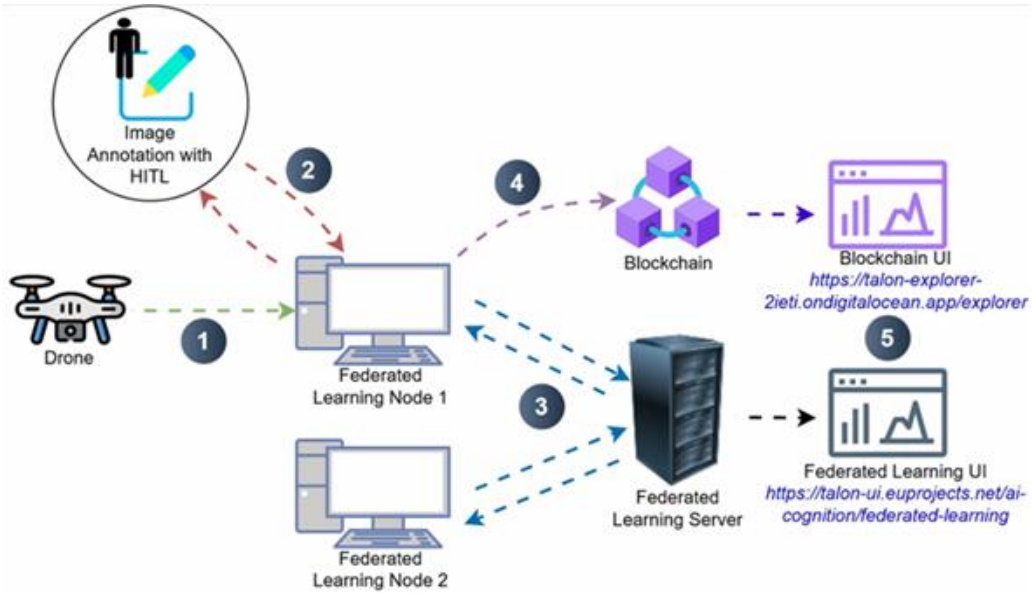

Figure 63: Representation of FL Architecture

The TALON FL framework provides comprehensive real-time monitoring capabilities through an integrated dashboard interface that enables stakeholders to observe the FL process as it unfolds. The dashboard serves as the primary visualization tool for tracking experiment progression, client performance metrics, and overall system health throughout distributed training operations. Figure 64 demonstrates the FL dashboard interface during an active FL experiment. The dashboard displays critical system information including FL configuration parameters, client participation status, and real-

time performance analytics. Users can monitor the total training rounds, minimum client requirements, and the selected aggregation strategy (FedAvg[17]) through the configuration panel.

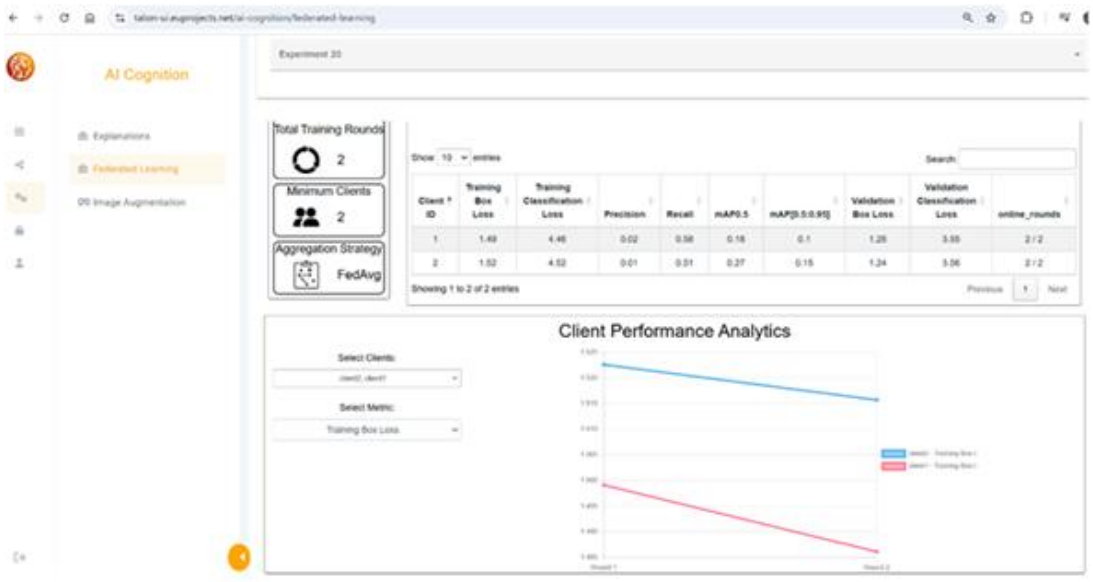

Figure 64: An example run for an FL experiment, after 2 federated rounds as can be seen through the TALON Dashboard

The dashboard supports comparative analysis between multiple clients, and across experiments, enabling identification of performance variations and convergence patterns across the federated network. The client information section provides detailed visibility into individual node performance, displaying metrics such as training box loss, training classification loss, precision, recall, mAP scores, and validation metrics for each participating client. This granular view enables administrators to identify potential issues with specific nodes and assess the overall health of the distributed training process. This monitoring interface is essential for maintaining oversight of complex distributed training operations, providing the transparency and control necessary for successful FL deployments in industrial environments. The dashboard updates automatically as new training rounds complete, ensuring that stakeholders have access to the most current information about the FL experiment's progress and performance.

The pilot implementation includes comprehensive monitoring capabilities through the integrated TALON Dashboard system which can be reached through https://talon-ui.euprojects.net/ai-cognition/federated-learning and contains all executed experiments. The dashboard provides real-time visualization of the FL progress, client performance metrics, and training analytics. During experimental runs, the system successfully tracked multiple training rounds, displaying precision, recall, mAP50, loss and other metrics for each participating client as can be seen in Figure 64**Errore. L'origine riferimento non è stata trovata.** and later in Figure 69 and Figure 70 (real results). The monitoring interface enables stakeholders to observe FL experiment progression in near real-time, with experiment tracking capabilities that allow for efficient analysis of model performance across distributed nodes. The dashboard integration demonstrates the system's readiness for production deployment in industrial monitoring scenarios. Additionally, the framework integrates with the

[17] *https://flower.ai/docs/framework/ref-api/flwr.server.strategy.FedAvg.html*

blockchain mechanism developed by SIDROCO, ensuring that FL experiment results cannot be mutated and providing transparent audit trails of the training process.

The empirical results from this demonstration provide crucial validation of the TALON platform's FL capabilities for computer vision applications rather than simply showcasing individual model performance. The varying performance levels across different nodes reflect realistic deployment scenarios where participating devices may have different computational capabilities, data quality, or network conditions. The demonstration results underscore the platform's inherent ability to conduct federated training effectively for computer vision tasks. The success of the object detection implementation validates the TALON platform's readiness to handle complex distributed learning scenarios where data privacy, system efficiency, and collaborative learning are paramount considerations. The experimental validation confirms that the FL framework can successfully coordinate distributed training across heterogeneous nodes while maintaining model performance standards suitable for industrial applications. This capability positions the TALON platform as a viable solution for real-world deployments requiring distributed machine learning with privacy preservation and collaborative model development. Results of the training process and various useful metrics are discussed in section 5.3.3, where actual pilot demonstration takes place.

Figure 65: Example of annotated input images

5.3.3 Instantiation with pilot data

The TALON FL framework has been successfully instantiated and validated for Demonstrator #4: HRC using real-world pilot data from industrial environments, demonstrating its practical applicability for construction safety monitoring and worker protection scenarios. The implementation leverages drone-captured imagery and ground-based surveillance systems to create a comprehensive safety monitoring ecosystem, with applications including but not exclusive to contributing to alerting systems capable of detecting missing personal protective equipment such as no-vest and no-helmet scenarios, later in the project's pipeline.

The pilot implementation utilised multiple data sources to validate the FL framework's capabilities in real industrial environments, as shown in Figure 65. The primary dataset consisted of construction safety imagery captured through drone patrols, with the help of CERTH, focusing on Personal Protective Equipment (PPE) detection and compliance monitoring. The system was configured with distributed FL nodes (Node 1 and Node 2) connected to a central FL server, as illustrated in the deployment architecture in Figure 63. A considerable fact is that although the pilot setup involved two nodes, the datasets were sufficiently heterogeneous, with the open-source dataset having been annotated with a substantially larger number of classes than the CERTH dataset. This difference enabled fine-tuning of the model to detect in CERTH imagery, not only the PPE-related classes from

the CERTH dataset but also the additional classes from the open-source dataset, enhancing the overall representational diversity and robustness of the framework

The demonstration employed an industrial environment safety dataset provided by CERTH, specifically designed to identify critical safety equipment and personnel in industrial environments. These images were taken from patrolling drones in an industrial setting and were human-in-the-loop annotated. To enhance the training diversity and robustness, the FL setup utilized two distinct but complementary datasets: Node 1 was configured with the CERTH dataset containing real industrial drone imagery, while Node 2 operated exclusively with the Roboflow Construction Safety Dataset[18], which contains the same 5 classes (person, vest, no-vest, helmet, no-helmet) but from different sources and perspectives. This deliberate dataset separation between the two federated clients created a realistic scenario where different industrial partners contribute their own local datasets while collaboratively training a unified safety detection model. The complete federated setup provides a comprehensive testing ground for computer vision object detection applications in industrial safety monitoring, encompassing scenarios commonly encountered in construction and manufacturing environments while demonstrating the framework's ability to leverage heterogeneous data sources.

Figure 66: From left to right: Node1, Node2, FL Server

The experimental setup involved a containerized client-server application running on distributed computing nodes, each processing local distinct datasets while maintaining data privacy through the FL approach. The individual node setups demonstrate the distributed nature of the implementation, with Node 1 and Node 2 each configured with dedicated computing resources. The FL server, hosted on dedicated remote hardware infrastructure, coordinated the training process across multiple clients using the Flower framework with the FedAvg aggregation strategy. The overall FL schematic with Node1, Node2 and the FL server can be seen in Figure 66. In Figure 67 we see a comprehensive view of the live FL training process, showcasing the real-time execution across all system components. The top panel of Figure 67 displays the FL training coordination interface with server logs on the left demonstrating the federated rounds progression, client aggregation, and model parameter updates, while the right side shows the server usage dashboard providing real-time visualization of multiple useful statistics across the training epochs and rounds. The bottom panels illustrate the individual Node1 and Node2 training processes (also referred to as client1 and client2), displaying detailed YOLO model training logs with epoch-by-epoch performance metrics, dataset processing information, and local model optimisation results. This comprehensive visualisation demonstrates the seamless coordination between distributed clients and the central server, validating the framework's capability to maintain synchronised FL across heterogeneous computing environments while preserving data locality and privacy.

[18] *https://universe.roboflow.com/roboflow-100/construction-safety-gsnvb/dataset/2*

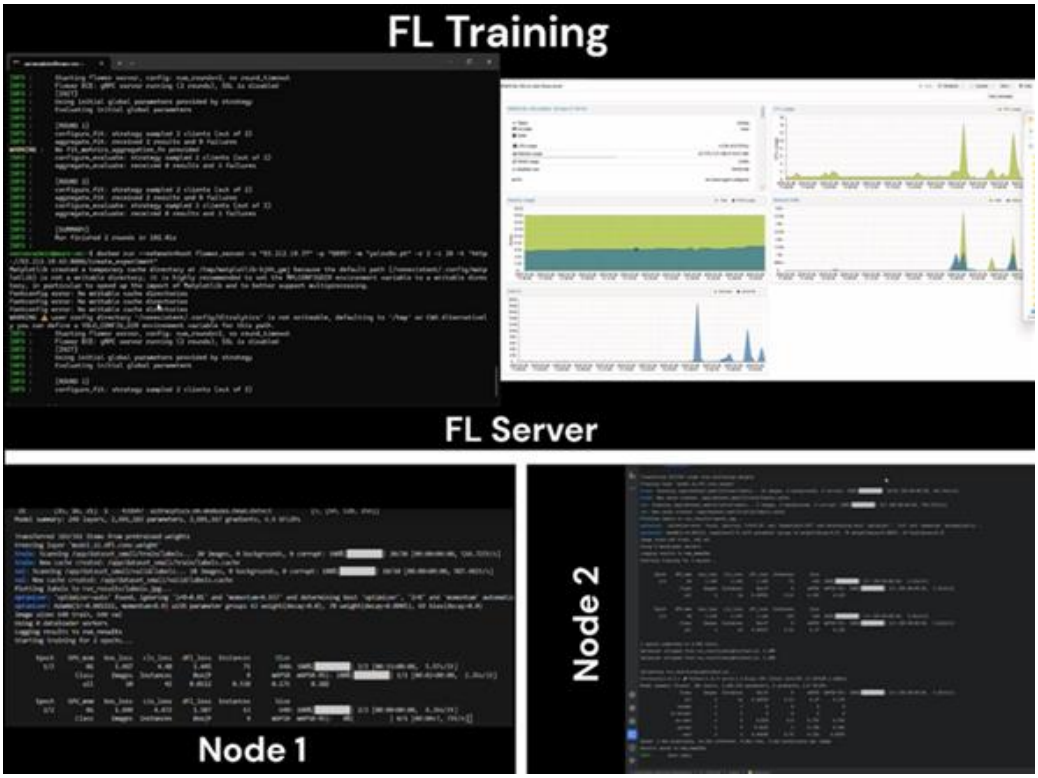

Figure 67: Overview of the FL process. Top panel: the FL Server terminal along with its usage metrics and statistics. Bottom panel: Node 1 and Node 2 perform training on their local data and the server aggregates the results after each round.

The pilot deployment focused on construction safety use cases, implementing object detection models trained to identify five critical safety classes: **person, helmet, no-helmet, vest, and no-vest**. The system demonstrates real-time detection capabilities with confidence scoring, enabling automated safety compliance monitoring in industrial environments.

Figure 68: Snapshot of real-time inference in an industrial setting

The detection results show the system's ability to accurately identify workers and their PPE status in various scenarios, including warehouse environments with forklifts and construction areas with multiple personnel. Detection confidence scores typically range from 0.72 to 0.91 on average, indicating robust model performance across different lighting conditions and viewing angles. The result of this process can be seen in the Figure 68 snapshot, where the trained system is able to detect the classes mentioned above in an industrial workplace environment.

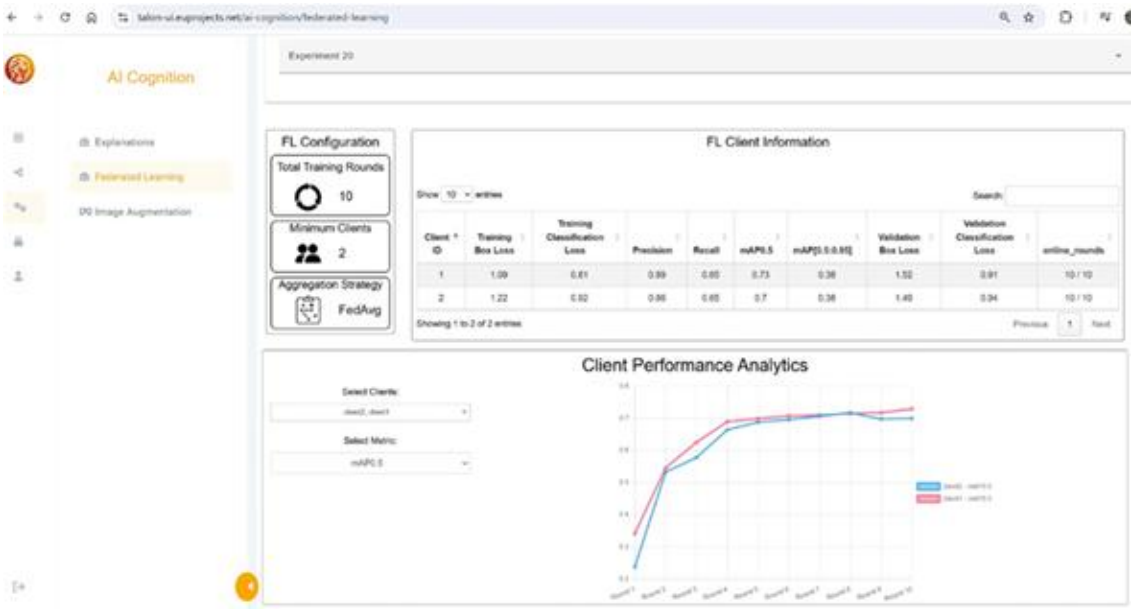

Figure 69: FL Experiment mAP metrics after 10 federated rounds

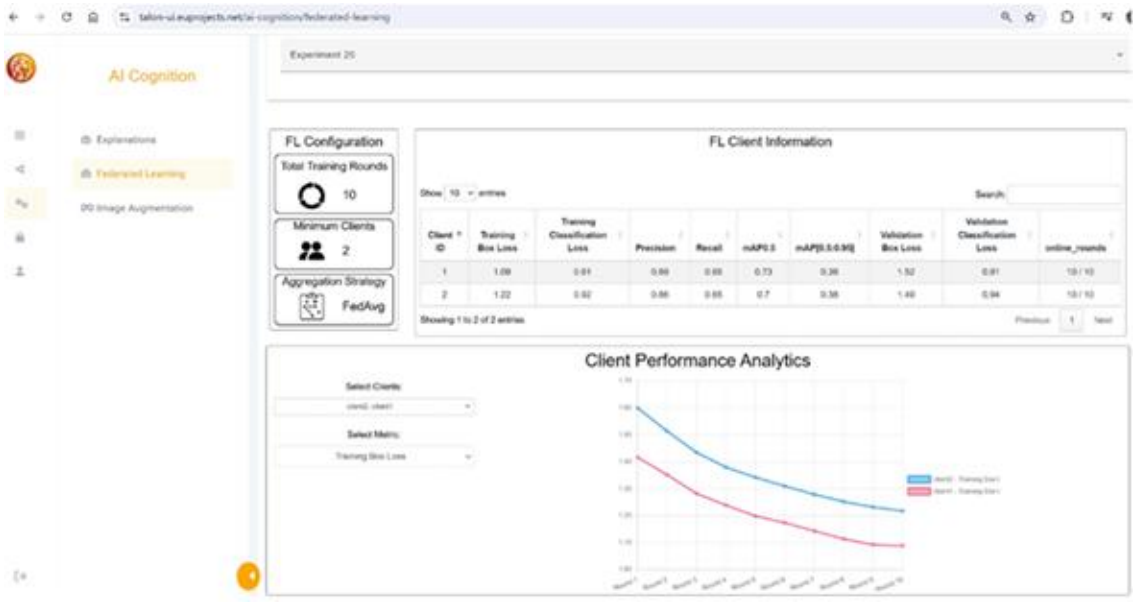

Figure 70: FL Experiment Training Box Loss metrics after 10 federated rounds

The federated deployment involved **2 clients**, that trained on their local data for **2 epochs** per federated round, each processing distinct portions of the image data for object detection tasks. Performance evaluation focused on recall, mean Average Precision (mAP) and various other metrics after completion of **10 federated training rounds.** This specific experiment ran for about 52 minutes on an AMD Ryzen 5 7600X CPU, proving that even hardware that is not used specifically for machine

learning tasks can produce respectable results, and can be improved by an order of magnitude (10x faster) with the use of a GPU [28]. The distributed learning process demonstrated varying performance levels across participating nodes, reflecting the heterogeneous nature of real-world federated deployments.

Table 29. Performance metrics of the 10 round FL experiment

	Training Box Loss	Precision	Recall	mAP
Node 1	0.81	0.89	0.65	0.73
Node 2	0.92	0.86	0.65	0.7
Average	0.865	0.875	0.65	0.715

Additionally, as can be deduced from, Figure 69, Figure 70, and Table 29, the loss and mAP curves have not yet asymptoted, meaning that the model's performance can further be improved by simply letting it train for more federated rounds, or more epochs per round per client. Nevertheless, current model performance proved satisfactory as can be seen from Figure 68's real-time results.

The pilot implementation successfully validated the FL framework's performance across multiple distributed nodes, achieving competitive detection accuracy while maintaining data privacy. The system demonstrated stable training convergence across federated rounds, with consistent model improvement observed through the integrated monitoring dashboards. The framework's readiness for production use in industrial environments is confirmed through this practical deployment, which demonstrated capabilities for handling real-time object detection, distributed model training, and comprehensive monitoring across multiple industrial use cases. The trained models effectively contribute to the targeted alerting system capabilities for detecting safety violations including missing protective equipment in industrial environments.

5.4 Digital Twins

In precision CNC machining environments, the development of robust AI models for tasks such as predictive maintenance, anomaly detection and system diagnostics requires large volumes of high-quality, labeled sensor data. However, real-world data collection in industrial settings is often limited by several practical constraints. Anomalies and failures occur infrequently, making it difficult to capture enough representative samples for supervised machine learning. Moreover, operational requirements rarely permit deliberate disruption of machinery to produce fault data and extensive sensor retrofitting is not always economically viable. To address these challenges, synthetic sensor data generation has become an essential method for supplementing real datasets and enabling the training of effective machine learning models in data-sparse domains.

CERTH has developed a synthetic data generation tool tailored to the operational context of CNC machines. The tool aims to produce realistic, high-dimensional, multivariate time-series data that replicate the dynamics of machine behavior across varying conditions and operational modes. By mimicking patterns seen in real sensor signals, including gradual degradation, fluctuating loads and tool wear cycles, the synthetic data enables experimentation, model pre-training and validation in scenarios where real-world data may be incomplete or entirely unavailable. This capability is particularly important in precision machining, where equipment operates under strict tolerances and the cost of undetected anomalies can be substantial. The synthetic data generator thus offers a

scalable and safe alternative for enriching datasets used to develop AI-driven maintenance strategies in modern industrial environments.

5.4.1 Key Technologies & Technical Updates

The synthetic data generation module is based on the **TimeGAN** *[29]* architecture (Figure 71*)* selected for its proven ability to capture both temporal dynamics and feature-wise dependencies inherent in multivariate time-series data. TimeGAN uniquely integrates autoencoding structures, recurrent neural networks (RNNs)[19], and adversarial training, enabling it to generate synthetic sequences that retain the statistical characteristics, sequential coherence, and cross-variable correlations observed in real-world datasets. This hybrid design makes TimeGAN particularly well-suited for modeling industrial sensor data, where preserving temporal continuity, contextual dependencies, and multi-sensor interactions is essential for producing realistic and reliable synthetic samples.

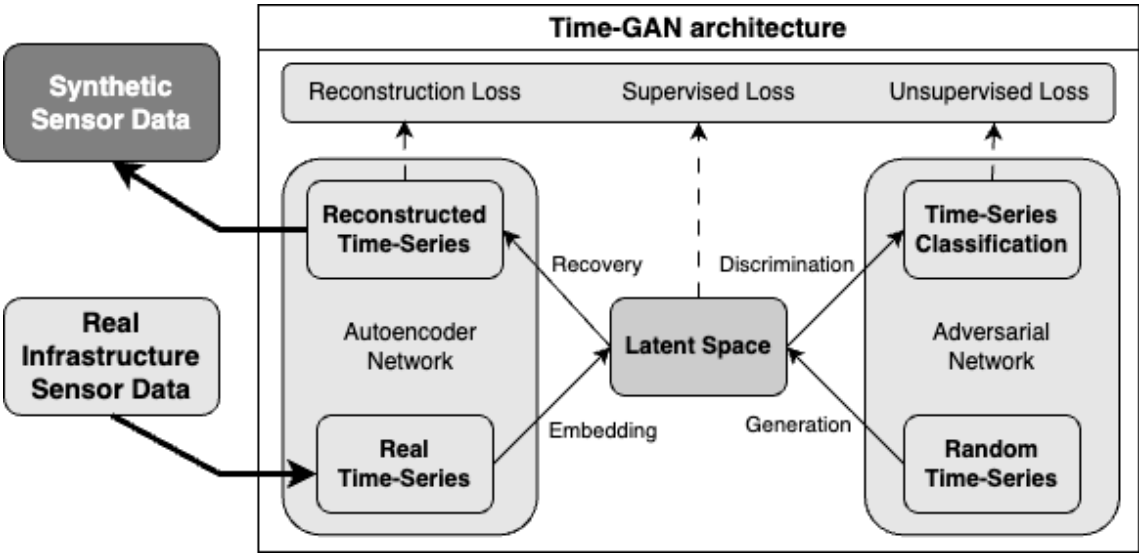

Figure 71: Architecture of TimeGAN

The initial development of the tool utilized a dataset provided by the FACTOR end-user, consisting primarily of temperature readings from various components of the Nakamura2 CNC machine, coupled with an indicator of the overall machine health status. This allowed for early prototyping and evaluation using standard techniques such as PCA[20], t-SNE[21] and the Train-Synthetic-Test-Real (TSTR) *[30]* framework.

As the project progressed, a more detailed and structured dataset became available, curated and preprocessed by the UPV team. This final dataset represented a significant step forward in terms of granularity and relevance. It included time-stamped operational data with **30 variables**, encompassing feed-rate override values, servo motor loads across multiple axes and paths and detailed tool wear metrics for up to **20 tools**. Each data instance corresponded to approximately one hour of CNC machine operation, resulting in a total of **3137 timesteps**.

Adapting the model to this dataset required a series of architectural and pre-processing updates. The higher dimensionality and increased temporal complexity called for data standardization and

[19] *https://www.ibm.com/think/topics/recurrent-neural-networks*
[20] *https://www.ibm.com/think/topics/principal-component-analysis*
[21] *https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html*

extensive hyperparameter optimization, conducted through a structured grid-search methodology. Due to the computational load associated with long sequences, the sequence length for generation was set to **168 timesteps**, corresponding to **one week of machine operation**.

In its final implementation, the tool has been deployed as a Flask-based REST API, allowing synthetic samples to be generated on demand. Users can specify parameters such as sequence length and the number of samples to be generated. The system is containerized using Docker, facilitating scalable and portable deployment. The API is fully documented via Swagger, ensuring ease of use and interoperability with other components in the TALON architecture.

5.4.2 Scientific and Technical Results

The evaluation of the initial version of the tool, trained on the legacy dataset provided by FACTOR, suggested that the synthetic data captured the general trends and relationships present in the original dataset, though with some degradation in predictive power based on the TSTR assessment.

Table 30: Initial tool results, based on FACTO legacy dataset

	R2 score	MAE	MRLE
Real	0.709582	0.018577	0.001434
Synthetic	0.622268	0.033815	0.002878

To further assess the structural similarity between synthetic and real data, PCA and t-SNE were applied. The PCA plots showed that while the synthetic samples roughly followed the variance patterns of the real data, the clusters were less compact and exhibited slight shifts in centroid alignment. Similarly, the t-SNE embeddings highlighted that real and synthetic data points partially overlapped in low-dimensional space but remained distinguishable, indicating that the model reproduced some, but not all of the nonlinear dependencies and temporal dynamics. Overall, the first version of the tool demonstrated promising results, providing a foundational capability for synthetic data generation, while also highlighting opportunities for refinement in data quality and temporal fidelity.

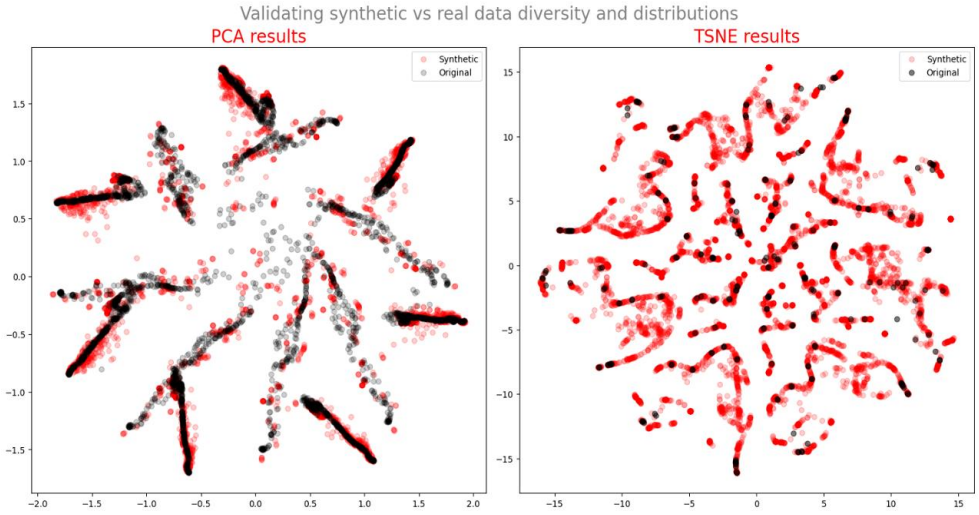

Figure 72: Projection of Synthetic & Real data (Initial Dataset)

The final version of the synthetic data generation tool demonstrated a marked improvement in statistical fidelity and model utility, driven by the transition to a richer, more representative dataset containing detailed signals related to machine paths, servo loads, feedrate overrides, and tool wear metrics. Quantitative evaluation using the Train-on-Synthetic-Test-on-Real (TSTR) framework confirmed this enhancement, with regression models trained on synthetic sequences achieving an R^2 and MAE values that closely approach the baseline performance of models trained directly on real data. These results suggest that the synthetic sequences captured much of the temporal and contextual structure present in the real-world signals.

Table 31: Final tool results

	R2 score	MAE
Real	0.769	0.041
Synthetic	0.669	0.064

However, deeper analysis using dimensionality reduction techniques revealed nuanced limitations. PCA and t-SNE projections indicated that, despite overall tighter clustering, synthetic data distributions still exhibited areas of misalignment with the real data manifold. This visual drift suggests that while the synthetic sequences approximate real trends at a global level, they may lack some of the fine-grained morphological features critical for domain-specific inference tasks.

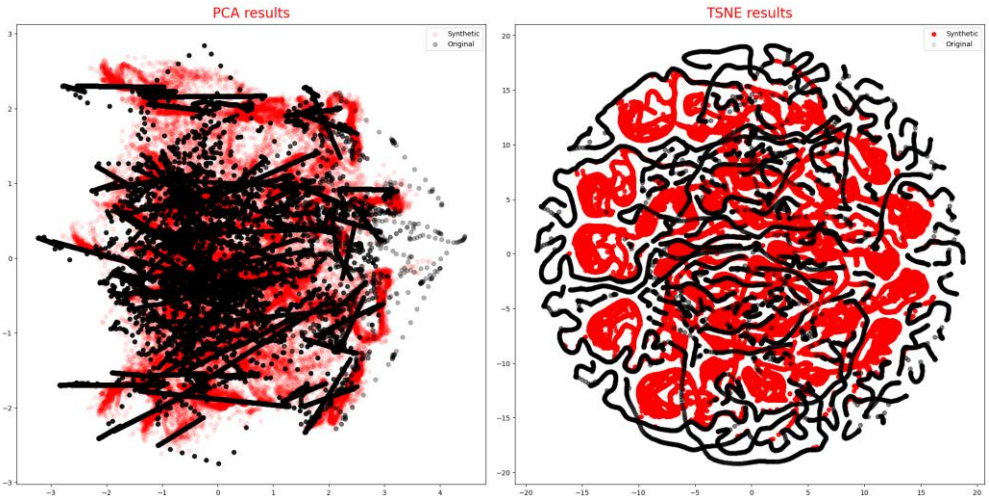

Figure 73: Projection of Synthetic & Real data (Final Dataset).

This insight was reinforced by an independent evaluation carried out by UPV as part of Task T3.4, using their Self-Healing and Self-Correcting framework. In zero-shot inference mode, where models trained on real data were applied directly to synthetic inputs, the system consistently failed to reproduce accurate degradation predictions, often generating flat or weakly correlated outputs. This behavior highlighted a lack of structural compatibility between synthetic and real signals, especially regarding the cyclic sawtooth wear profiles typical of real tool life data.

In contrast, when the same models underwent fine-tuning on structurally consistent synthetic sequences, prediction quality significantly improved. The system was able to partially recover the slope and shape of the degradation curve, particularly in examples that included multiple wear cycles and clear restart transitions. These findings confirm that synthetic data is most effective when it

reflects not only statistical similarity but also the underlying morphological and temporal dynamics inherent to the industrial process being modeled. Without such structural alignment, even high-dimensional synthetic data may fail to support reliable generalization in downstream predictive maintenance applications.

5.4.3 Instantiation with pilot data

The instantiation of the synthetic data generation tool with pilot data provided a valuable opportunity to evaluate its real-world applicability and limitations within an industrial AI pipeline. The final version of the tool, containerised via Docker and exposed through a Flask-based API, allowed UPV and other partners to generate synthetic time-series data on demand for predictive maintenance tasks, using the actual operational context of the Nakamura2 CNC machine as a reference.

Through this deployment, several critical insights emerged. First, the success of synthetic data in supporting AI model development is not solely determined by statistical metrics such as R^2 or MAE, but by the extent to which the generated sequences preserve the structural and morphological traits of real degradation processes. While the synthetic data could replicate some global patterns, initial attempts revealed a misalignment in temporal structure, particularly the absence of characteristic sawtooth wear patterns, leading to poor zero-shot model performance. Only after fine-tuning the predictive models on synthetic sequences that closely mimicked the cyclical behavior of actual tool wear did the models begin to generalize effectively.

The lesson learned from the conducted experiments suggests that the usefulness of synthetic data hinges on its realism, not just in values but in behavior. Domain-specific signal characteristics must be preserved for synthetic data to serve as a reliable proxy in predictive maintenance. The pilot deployment confirmed the value of synthetic data as a complement, not a substitute to real-world data, especially when it is carefully shaped to reflect operational realities. It also demonstrated the importance of close collaboration between data generation, domain expertise, and model evaluation to ensure synthetic signals meet the needs of downstream applications.

6 Blockchain and Authentication modules for Security & Privacy

6.1 Authentication and Authorisation

6.1.1 Key Technologies & Technical Updates

Within the TALON project, an advanced Authentication and Authorization engine was designed and implemented to effectively manage and control user access through the assignment of dedicated roles and permissions as well as securely handle and aggregate the APIs exposed by the technical partners' components. This entailed the development and deployment of a Role-Based Access Control (RBAC) system, achieved by configuring the Keycloak framework, which facilitated the allocation of permissions in accordance with predetermined user roles and categories, such as Administrator, Pilot User, and Technical User. Additionally, a comprehensive authentication and authorization protocol for APIs has been configured for each endpoint (e.g., POST, GET, PUT) supported by every technical component. This setup incorporates secure API key management via tokens in Authorization headers with regular rotation and the classification of appropriate token types, including JWTs for stateless authentication purposes. Stateless authentication was employed to incorporate claims concerning user APIs and previously defined permissions. Access tokens and user credentials are maintained within the centralized TALON cloud infrastructure and are provided through secure channels, namely HTTPS, ensuring they are securely stored in an encrypted form within a MySQL database.

Since the previous version, enhancements include the definition of access rights for a broader array of user roles, including all the pilot partners, as well as the integration of all technical component APIs into the Authentication and Authorization engine. Furthermore, the formulation of distinct roles for pilot partners has been accompanied by the development of specialized user interfaces designed to facilitate the illustration of their technical and research outcomes on the TALON Dashboard, thus permitting better control over their access to particular platform features and data pertinent to their specific use case.

6.1.2 Scientific and Technical Results

The Authentication and Authorization engine has not contributed to the scientific results of the TALON project. Instead, it has a horizontal technical role in the TALON solution and the integrated platform. The key technical contributions of this component are highlighted as follows:

- Unified Access Control across Users and Technical Endpoint API Security: This engine has established a single, unified access control layer for a diverse array of user roles, technical components and APIs developed by the TALON partners. This consolidated every interacting entity behind a single point and eliminated the need for insecure and redundant authentication/authorization mechanisms within each individual component.
- Having a consistent and robust mechanism for API authentication (via JWTs and secure API key management) and user authorization (via Keycloak RBAC), this engine facilitated the seamless and secure integration of diverse services and data flows across the TALON platform. This directly allowed to ensure the functional interoperability across different components.
- The definition and implementation of different roles (including the Administrator, Technical Users, and Pilot Users) within Keycloak has enabled precise control over who can access what computing resources and perform which actions. This fine-grained control is a direct technical outcome, ensuring that sensitive data and critical functionalities are only accessible to authorized entities.

6.1.3 Instantiation with pilot data

The Authentication and Authorisation engine has a central role on the TALON platform. It has been configured and extended to accommodate all the pilot users. However, it is a central mechanism targeting the TALON solution as a whole. Figure 74 presents the user prompt requested to provide her credentials when landing in the TALON Dashboard.

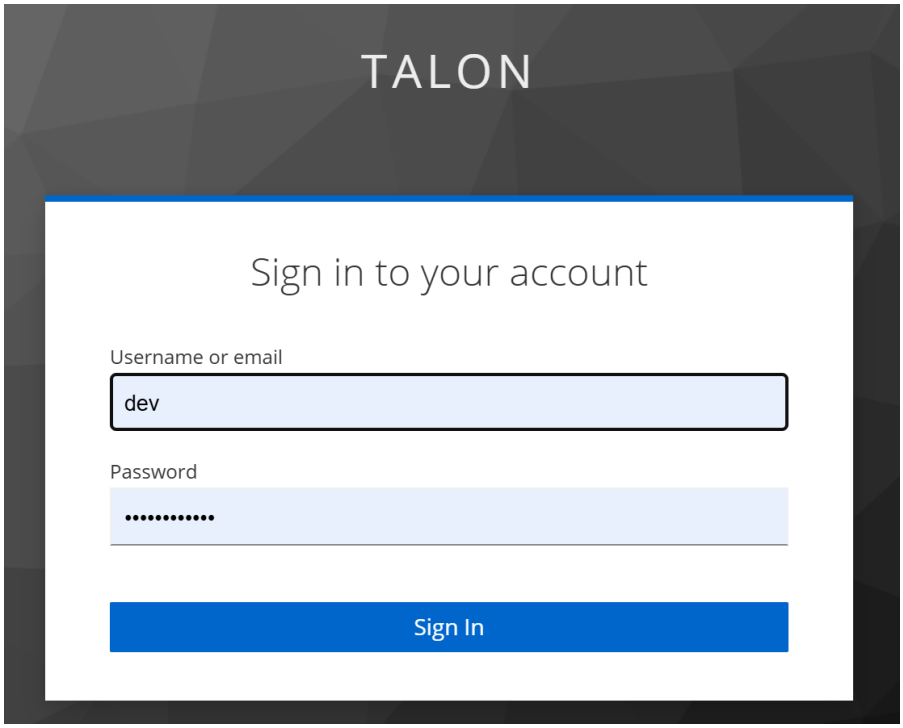

Figure 74: User Login and Authentication Mechanism.

We present in Figure 75 an example of integrating the Pilot 3 visual interfaces in the TALON Dashboard also providing access to the data supported by the end user application.

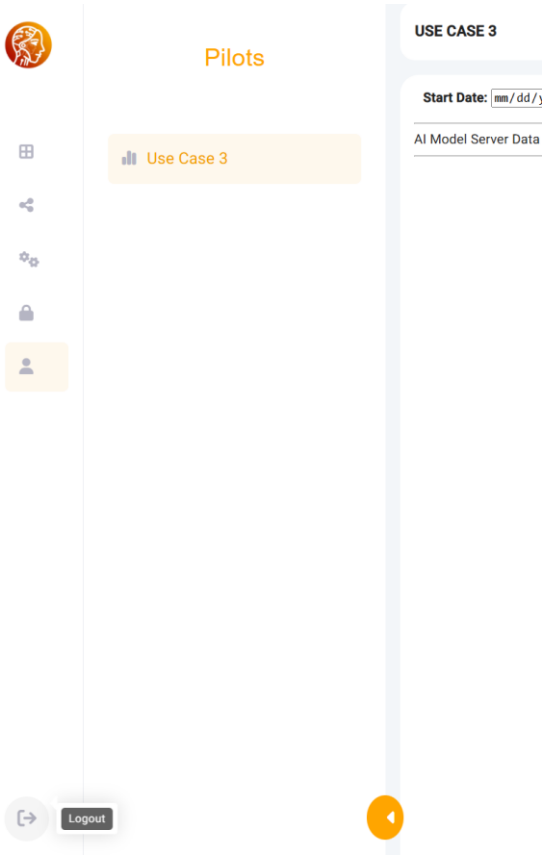

Figure 75: Authentication Mechanism for Pilot 3.

6.2 Blockchain Mechanism

6.2.1 Key Technologies & Technical Updates

Technology Stack

The core development and delivery of this module has been reported at D4.2 “Lightweight Decentralized Blockchain Toolbox”. Rationale, methodology, state-of-the art and core development approaches have been extensively reported in D4.2. In this deliverable we are reporting the technical enhancements and pilot usage of this tool.

Firstly, outline the technology stack used to develop and implement the TALON blockchain toolkit. Each component has been selected to support the toolkit’s goals of secure data management, scalability, and efficient processing in a blockchain-based infrastructure. Table 32 provides an overview of the core technologies, frameworks, and tools that form the foundation of the TALON solution, enabling decentralised storage, network monitoring, and seamless interaction between federated learning nodes and blockchain network.

Table 32: TALON's Blockchain Toolkit Technology Stack

Technology	**Library/Tools**

Blockchain	Hyperledger fabric v2.5.5 Golag v1.22 for smart contract
Explorer Api	Express.js v4.18.2 Fabric Gatway node SDK v1.4.0 Fabric network node sdk v2.2.0 PorsgreSQL DB
Client App	Express.js v4.18.2 Fabric Gatway node SDK v1.4.0 Pinata client SDK v2.1.0
Blockchain UI	Reactv18.2.0 Maaterial UI v5.15 DigitalOcean PaaS(Platform as a service) for deployment
Reverse proxy	Nginx

The whole underlying blockchain platform has no significant changes with some visual adjustments with the addition of history tab for each clientId for better auditability and traceability of the data and some changes to accommodate the UPV model weights.

Technical Updates

Part of the enhancements is the front end and back-end adjustments to accommodate properly the pilots.

UI/UX Refinements:

- Improved panel responsiveness and clearer transaction status indicators.
- “Blocks” and “Transactions” panes now display dynamic tooltips describing each field, drawn directly from chaincode metadata.
- **New “History” tab:** Allows users to pick a channel, select an asset ID, and retrieve the full on-chain history of that asset. Accordion-style entries show timestamp, transaction ID and active/inactive status with drill-down details for each update (Figure 76).

Client-Side (UPV pilot) Integrations:
The UPV model training client app was adapted to stream model-update hashes into the blockchain. A preliminary slicing scheme bundles large updates into digestible on-chain entries preserving confidentiality while ensuring traceability.

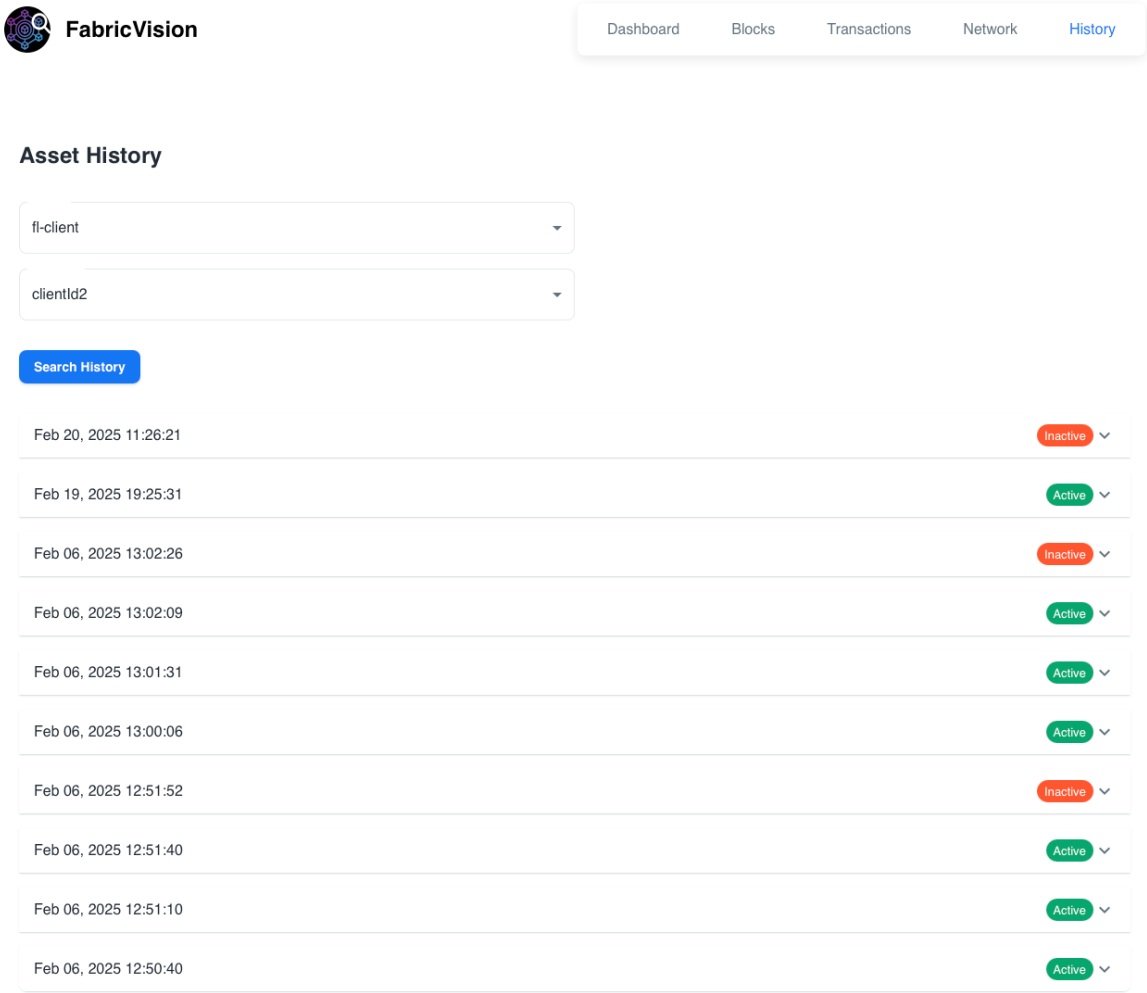

Figure 76: Newly added "History Tab"

6.2.2 Scientific and Technical Results

The pilot implementation of blockchain delivers end-to-end monitoring and auditability via the integrated TALON Dashboard and fully aligns with the Grant Agreement's objectives of reusability, explainability, trustworthiness, security and privacy.

We provide comprehensive monitoring through the TALON Dashboard, which queries on-chain records for each federated-learning update. For UC4, each client's model-update hash and accompanying metadata (trainingRound, accuracy, modelCID) are immutably logged on our Hyperledger Fabric network, while full model weights are pinned to IPFS. We acknowledge that IPFS availability depends on the broader peer network; to mitigate this external dependency, all CIDs are redundantly pinned via the Pinata service, ensuring high-availability gateways and rapid retrieval even under low peer counts. On the privacy front, model CIDs are stored only within Fabric channels scoped to the corresponding consortium participants, preventing unauthorized discovery. In the current deployment, raw weights are unencrypted prior to IPFS pinning; however, channel-level access control in Fabric effectively isolates CIDs from public queries. For heightened confidentiality in future releases, we plan to integrate client-side encryption of model artifacts before IPFS upload, or leverage Fabric's private data collections to keep sensitive payloads off the public ledger entirely.

UC#2 follows the same pattern: hashes of training data and weights are written on-chain, weights are sliced to respect IPFS size limits and pinned via Pinata under the same clientId, and stakeholders trace convergence through simple REST calls in the dashboard demonstrating a tamper-proof and privacy-preserving AI pipeline.

Model-training Pilot (UPV)

- On-chain Commitments: Each client's model update is hashed and recorded on the Hyperledger Fabric ledger.
- Off-chain Storage: Full model weights are pinned to IPFS. To respect IPFS pin-size limits, the UPV client app automatically slices large weight files into chunks, then uploads all chunks under a single clientId asset—preserving the update's atomic identity.

Monitoring & Traceability

- RESTful Queries: The dashboard's API lets auditors pull training round, accuracy and model CID values on demand, providing transparent insight into each federated round.
- **Immutable Provenance:** By combining blockchain hashes with IPFS-hosted weights, we achieve a tamper-proof audit trail from model creation through deployment.
- **Data Privacy:** Only cryptographic digests and metadata reside on-chain.

6.2.3 Instantiation with pilot data

In TALON, the lightweight hierarchical blockchain mechanism was instantiated with real pilot data in both Pilot #2 (I5.0 Automation & Planning) and Pilot #4 (HRC). The instatiation process is demonstrated the previous Scientific and Technical Results section due to the necessity to (i) showcase the technical/scientific progress with regard to the first deliverable and (ii) the fact that all technical updates and developments are correlated with the pilot requirements. The brief description of the pilot implementations in more inclusive and elaborative way is presented below.

In both TALON's pilots (#2 and #4) case, only cryptographic hashes and minimal metadata are committed on-chain via Hyperledger Fabric[22], while the full model-weight files remain off-chain in IPFS[23] and are referenced by content identifiers (CIDs). This approach minimises on-chain storage, preserves data privacy and creates an immutable provenance record that can be transparently audited .

During Pilot #4, the MINDS federated-learning client app was extended to hash each locally trained model update using SHA-256 and to submit this digest on-chain together with metadata fields, trainingRound (the round index), accuracy (the locally evaluated performance) and modelCID (the IPFS link to the full weights), all under the submitting client's unique identifier. The complete weight file is pinned to IPFS and the on-chain modelCID enables any stakeholder to retrieve the exact snapshot used in that round. Auditors and operators can then query the TALON Dashboard API to enumerate every federated-round hash, its reported accuracy and the associated IPFS link providing end-to-end transparency of model convergence without ever exposing raw data.

[22] *https://hyperledger-fabric.readthedocs.io/en/release-2.5*
[23] *https://ipfs.tech*

Pilot #2 applied a similar pattern for the UPV tool-wear prediction workflow. Here, the client app likewise emits hashes of each fine-tuning iteration, but because UPV weight files may exceed IPFS pin-size limits, it automatically slices each large file into multiple smaller chunks. Each chunk is independently uploaded to IPFS and their CIDs are atomically recorded in a single on-chain asset tied to the client's identifier. Reconstructing the full weight file simply requires retrieving and concatenating all recorded slices, while the combined on-chain hash and CID list guarantees that any tampering with even one slice invalidates the entire update.

Together, these instantiations demonstrate how TALON's blockchain mechanism delivers a scalable, privacy-preserving and verifiable architecture to support the AI pipeline: heavy payloads are kept off-chain in IPFS, audit-grade metadata reside on-chain and the integrated Dashboard unifies them into an immutable, tamper-proof audit trail that stakeholders can explore via RESTful queries for training rounds, performance metrics and model CIDs.

7 Conclusion & Future Outlook

This deliverable marks the formal conclusion of TALON's WP4 technical activities through comprehensive documentation and reporting. It stands as another in-depth technical demonstrator report that clearly outlines both the technical updates and the pilot implementations carried out throughout the project. While the length of the document may pose a challenge for quick reading, it also highlights (i) the wide spectrum of technologies employed to address the diverse TALON requirements, (ii) the depth of research and development invested by the project partners in the implemented tools and (iii) the continuous refinement and enhancement of these tools, which have evolved from standalone components into the final integrated TALON platform.

The deliverable begins with a brief yet focused section on ethics considerations, particularly around the core concepts of Reusability, Explainability, Trustworthiness and Security & Privacy also incorporating the recommendations of the project's external ethics advisor. This section offers an abstract but direct mapping between the technologies and the corresponding ethical dimensions, showcasing how TALON's technological innovation is not only grounded in scientific and technical excellence but also aligned with existing EU ethical frameworks.

Subsequent sections describe each technological module, clustered by conceptual coherence, detailing the technical progress made, illustrating results and highlighting the tangible impact of each module within the TALON pilots. This approach demonstrates the consortium's continuous commitment to the project's goals and underscores the real-world value created through these technological advancements.

In summary, we emphasise the necessity for technologies to operate as interconnected parts of a broader architecture, serving specific higher-level objectives (here are the Reusability, Explainability, Trustworthiness, and Security & Privacy).

TALON stands as a practical and forward-looking example of how to design, develop and implement such integrated technological endeavors with both technical robustness and ethical awareness at their core.

8 References

[1] *H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," in* International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.*

[2] *E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan and Q. V. Le, "AutoAugment: Learning Augmentation Strategies From Data," in* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, California, USA, 2019.*

[3] *E. D. Cubuk, B. Zoph, J. Shlens and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in* IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020.*

[4] *D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer and B. Lakshminarayanan, "AugMix: A Simple Method to Improve Robustness and Uncertainty under Data Shift," in* International Conference on Learning Representations*, 2020.*

[5] *S. Lim, I. Kim, T. Kim, C. Kim and S. Kim, "Fast AutoAugment," in* Advances in Neural Information Processing Systems*, Vancouver, Canada, 2019.*

[6] *A. Vedaldi, H. Bischof, . T. Brox and J.-M. Frahm, "Faster autoaugment: Learning augmentation strategies using backpropagation," in* Computer Vision--ECCV 2020*, Glasgow, UK, 2020.*

[7] *Y. Zheng, Z. Zhang, S. Yan and M. Zhang, "Deep AutoAugment," in* International Conference on Learning Representations*, 2022.*

[8] *J. Zhang, L. Zhang, D. Li and L. Wang, "A Unified Search Framework for Data Augmentation and Neural Architecture on Small-Scale Image Data Sets,"* IEEE Transactions on Cognitive and Developmental Systems, *vol. 16, no. 2, pp. 501-510, 2024.*

[9] *M. Yaseen, "What is YOLOv9: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector," arXiv preprint, 2024.*

[10] *A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint, 2020.*

[11] *C.-Y. Wang, A. Bochkovskiy and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling Cross Stage Partial Network," in* IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.*

[12] *T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," in* IEEE conference on computer vision and pattern recognition*, Honolulu, Hawaii, 2017.*

[13] *S. N. Gowda, X. Hao, G. Li, S. N. Gowda, X. Jin and L. Sevilla-Lara , "Watt For What: Rethinking Deep Learning's Energy-Performance Relationship," Computer Vision - ECCV 2024 Workshops, Milan, Italy, 2025.*

[14] *G. Tsoumplekas, V. Li, I. Siniosoglou, V. Argyriou, S. K. Goudos, I. D. Moscholios, R.-G. Panagiotis and P. Sarigiannidis, "Evaluating the Energy Efficiency of Few-Shot Learning for Object Detection in Industrial Settings," in* 2024 IEEE 3rd Real-Time and Intelligent Edge Computing Workshop (RAGE)*, 2024.*

[15] *V. Li, G. Tsoumplekas, I. Siniosoglou, V. Argyriou, A. Lytos, E. Fountoukidis and P. Sarigiannidis, "A closer look at data augmentation strategies for finetuning-based low/few-shot object detection," in* 2024 IEEE 14th International Symposium on Industrial Embedded Systems (SIES)*, 2024.*

[16] *A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov , M. Druzhinin and A. A. Kalinin, "Albumentations: fast and flexible image augmentations,"* Information, *vol. 11, no. 2, p. 125, 2020.*

[17] Y. Wang, Q. Yao, J. T. Kwok and L. M. Ni, "Generalizing from a Few Examples: A Survey on Few-shot Learning," ACM Computing Surveys, *vol. 53, p. 53, 2020.*

[18] S. Pan, H. Luo, M. C. H. Chua and e. al., "A Review of Similarity-Based Few-Shot Learning Methods for Time Series Classification in Manufacturing," *in* Proceedings of the 6th International Conference on Industrial Artificial Intelligence (IAI), *2024.*

[19] R. Mo, H. Zhou, H. Yin and X. Si, "A survey on few-shot learning for remaining useful life prediction," Reliability Engineering & System Safety, *p. 257, 2025.*

[20] V. Ekambaram, A. Jati, P. Dayama and e. al., "Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series," *2024.*

[21] C. Trivedi, P. Bhattacharya, V. K. Prasad, V. Patel and A. Singh, "Explainable AI for Industry 5.0: Vision, Architecture, and Potential Directions," IEEE Open Journal of Industry Applications, *vol. 5, pp. 177-208, 2024.*

[22] B. Goodman and S. Flaxman, "European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"," AIMag, *vol. 38, no. 3, pp. 50-57, 2017.*

[23] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," Venice, Italy, *2017.*

[24] S.-M. Moosavi-Dezfooli, A. Fawzi and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," *2016.*

[25] U. Jang, X. Wu and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," *in* Annual Computer Security Applications Conference (ACSAC), *2017.*

[26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," *2019.*

[27] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and A. y. A. Blaise , "Federated Learning of Deep Networks using Model Averaging," *in* arXiv, *2016.*

[28] S. Mukherjee, "Configure YOLOV8 for GPU: Accelerate object detection.," *2025. [Online]. Available: https://www.digitalocean.com/community/tutorials/yolov8-for-gpu-accelerate-object-detection.*

[29] J. Yoon, D. Jarrett and M. Van der Schaar, "Time-series generative adversarial networks," Advances in neural information processing systems, *vol. 32, 2019.*

[30] S. Sotudeh and N. Goharian, "TSTR: Too Short to Represent, Summarize with Details! Intro-Guided Extended Summary Generation," *2022.*

Funded by
the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070181