MANOLO

CLOUD-EDGE EFFICIENT & TRUSTWORTHY AI

D5.1 – Trustworthy Efficiency-Performance Assessment v.1

ATOS IT

30/06/2025

Funded by the European Union

MANOLO

Document Information

Issued by:	**ATOS IT**
Issue date:	01/05/2025
Due date:	30/06/2025
Work package leader:	ATOS IT
Dissemination level:	PU - Public

Document History

Version	Date	Modifications made by
0.1	01/05/2025	ATOS IT, ToC and draft of the chapter 4
0.2	08/05/2025	NCSR "D" draft of the chapter 3
0.3	15/05/2025	FDI, ATOS, CeADAR, draft of the chapter 5
0.4	22/05/2025	UPC, draft of the chapter 6;
0.5	16/05/2025	ARCADA, draft of the chapter 7
0.6	17/05/2024	Fraunhofer IIS, draft of the chapter 4
0.7	23/06/2025	Internal Review Process
1.0	30/06/2025	Updated version
1.2	30/06/2025	Updated version with comments addresses
1.3	01/07/2025	ARCADA new contributions
1.4	04/07/2025	Final version
1.5	30/08/2025	Review and modifications by NUIDUCD-CeADAR on benchmarking & testing

Authors

Name(s)	Beneficiary
Francesco D'Andria, Jose Ramon Manjavacas, Adrià Estevan, Alejandro García Bedoya	ATOS IT
Muhammad Imran, Eva Marín Tordera	UPC
Panagiotis Zazos	FDI
Magnus Westerlund, Roberto V. Zicari, Elisabeth Hildt	ARCADA
Natalia Koliou, Stasinos Konstantopoulos	NCSR "D"
Simon Geis, Bijoy Kundu	Fraunhofer IIS
Ricardo Simon Carbajo, Andrés L. Suárez-Cetrulo, Cristian Bosch Serrano, Eric Arazo, Hristo Stoev, Hsin-Hung Chen	NUIDUCD-CeADAR

Quality Reviewers

First Name	Beneficiary
Paulinus Ofem	LAU
Jean De Meyere, Daniela Spajić	KU Leuven
Ricardo Simon Carbajo	NUIDUCD-CeADAR

Disclaimer
Funded by the European Union under GA no. 101135782. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or CNECT. Neither the European Union nor the granting authority can be held responsible for them.

Executive Summary

Deliverable D5.1 named “Trustworthy Efficiency-Performance Assessment v.1 presents the first in-depth evaluation of the MANOLO benchmarking and system’s performance with a focus on energy efficiency, model robustness, trustworthiness and performance. This report outlines the initial benchmarking framework and methodological tools designed to assess AI models operating within the MANOLO architecture (see Deliverable 1.3), which spans both cloud and edge environments.

The work begins by detailing the design and implementation of the MANOLO benchmarking infrastructure, which allows the MANOLO consortium to carry out experiments in a controlled environment on a diverse set of models and hardware configurations. A key part of this system is the KOBE engine, which has been specially adapted and extended for AI workload orchestration supporting benchmarking.

The benchmarking framework will enable performance tracking in realistic and repeatable conditions. This architectural approach helps to collect important information about energy usage, performance of the system and how well computers are working. It also makes sure that the results can be understood and that the system is robust and reliable.

The deliverable also describes how several metrics tracking tools, such as Kepler, Alumet, and CodeCarbon, are integrated within the MANOLO monitoring and observability layer. These tools provide comprehensive visibility and detailed insights into resource utilisation. They also provide assistance in identifying areas for enhancement. The benchmarking efforts have been aligned with the project's goal to provide low-footprint; energy-efficient AI models suitable for use in restricted environments.

Furthermore, D5.1 addresses the subjects of model explainability and robustness, introducing metrics and tools to assess model transparency and resilience to adversarial conditions. The deliverable provides further elaboration on the detection of data drift in operational settings, offering methods to ensure that model behaviour remains consistent and trustworthy over time.

The document also establishes the framework for forthcoming integration and stress testing activities, detailing the utilisation of benchmarking tools to validate system behaviour under load and in real-world deployment scenarios. This iterative, component-driven approach will ensure the MANOLO system evolves with reliability and trust at its core.

Finally, the approach taken to achieve human oversight and trustworthy AI monitoring and validation of MANOLO’s system design is presented along with a detailed socio-technical scenario with a use case which has been evaluated using the Z-Inspection methodology for Trustworthy AI.

Overall, this deliverable marks a significant step towards establishing MANOLO's position as a platform for deploying responsible, efficient, and transparent AI in real-world, distributed environments.

MANOLO

Table of Contents

List of Figures

List of Tables

List of Terms and Definitions

Abbreviation	Definition
AI	Artificial Intelligence – the simulation of human intelligence processes by machines, especially computer systems.
KOBE	A benchmarking engine adapted to evaluate the performance, energy consumption, and robustness of AI workloads in MANOLO.
WP	Work Package – a defined section of a project plan that groups related tasks and objectives.
D5.1	Deliverable 5.1 – the first official report for Work Package 5, focused on trustworthy efficiency and performance assessment.
LIME	Local Interpretable Model-agnostic Explanations – a post-hoc method for explaining the predictions of black-box models locally.
Grad-CAM	Gradient-weighted Class Activation Mapping – a visualization technique used to highlight important regions in an input image for CNNs.
CNN	Convolutional Neural Networks – A set of deep learning models, particularly effective for image analysis.
Kepler	Kubernetes-based Efficient Power Level Exporter – a tool for estimating energy consumption at the container/process level.
Alumet	Adaptive, Lightweight, Unified Metrics – an open-source framework for measuring eco-efficiency and carbon footprint of software workloads.
CodeCarbon	A Python package used to estimate carbon emissions from computing tasks, especially machine learning experiments.
Drift	A change in the statistical properties of input data or model performance over time, which may lead to degraded prediction accuracy.
Z-Inspection®	A methodology for assessing the trustworthiness of AI systems through ethical and socio-technical analysis.
CAM	Class Activation Mapping – a set of techniques for visualizing the parts of input data that are relevant for a model's prediction.

ROAD	Remove and Debias – a metric used to evaluate the robustness of visual explanations by measuring model performance after masking key regions.
Surrogate Model	A simpler, interpretable model trained to approximate the behavior of a more complex (often opaque) AI model.
YAML	Yet Another Markup Language – a human-readable data-serialization format used for configuration files (e.g., in KOBE experiments).
Edge Computing	A computing paradigm where processing is performed closer to the data source (e.g., IoT devices), rather than in the cloud.
EUPL	European Union Public Licence – a free software licence used by EU institutions, including for tools like Alumet.
IDP	Infrastructure and Policy Definition – a YAML-based specification used in KOBE to define benchmarking workloads, models, datasets, and execution policies.
YAML	(from WIKIPEDIA) Yet Another Markup Language - is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted[1].

Table 1: Terms and Definitions

[1] https://en.wikipedia.org/wiki/YAML

1. Introduction

The MANOLO project is driven by the ambition to develop and validate a trustworthy, efficient AI (Artificial Intelligence) framework for deployment across the cloud-edge continuum. As AI applications increasingly migrate from centralized data centres to distributed and resource-constrained environments, there is a growing need to ensure that these models maintain high performance, transparency, and energy efficiency regardless of the deployment setting.

Deliverable D5.1 is the first version of the Trustworthy Efficiency-Performance Assessment report. It outlines the methodologies, tools, and initial results that support the evaluation of MANOLO's AI components in terms of their computational performance, energy consumption, explainability, and robustness. The work reported here is situated within Work Package 5 (WP5), which focuses on the evaluation, monitoring, and validation of trustworthy AI.

This document sets the foundation for evaluating AI workloads not only based on accuracy or latency, but also through a multi-dimensional lens that includes carbon footprint, interpretability, and the ability to detect and adapt to data drift. Such dimensions are becoming increasingly vital for the real-world adoption of AI, especially in sensitive or regulated domains.

The introduction also frames the rationale for integrating performance benchmarking with observability and monitoring subsystems, as these are essential for characterizing the behaviour of AI models in complex, real-time environments. The approach adopted in MANOLO promotes modularity and transparency, allowing developers and stakeholders to understand better how system decisions are made, and to trust the models deployed at both the cloud and the edge.

1.1 Structure of the Document

The document is organized as follows:

- Section 2 introduces the system architecture, previously developed in Deliverable 1.3), with an emphasis on the MANOLO's benchmarking and monitoring capabilities.
- Section 3 presents the benchmarking framework, including the KOBE engine, as the MANOLO AI Workload Orchestrator, for experiment orchestration.
- Section 4 describes the tools used for monitoring energy efficiency and system performance.
- Section 5 details the methods and metrics used to evaluate explainability and robustness.
- Section 6 outlines the use of the benchmarking framework in MANOLO for testing procedures.
- Section 7 focuses on human oversight, ethical validation, and alignment with Trustworthy AI principles.
- Finally, Section 8 concludes with a summary of the work and an outlook on future developments.

2. System Architecture

WP5 directly contributes to several core elements of the overall MANOLO system architecture, as shown in Figure 1 (see Deliverable 1.3 for an overall description of the MANOLO system and network architecture). It focuses on the foundational infrastructure for benchmarking, monitoring, explainability, and robustness, key pillars for ensuring the trustworthiness and operational reliability of AI workloads executed within MANOLO.

From an architectural point of view, the work described in D5.1 sits mainly within the "Benchmark & Central Storage" and "AI Workload Orchestrator" components of the MANOLO Controller, interacting tightly with both MANOLO Node components and external observability tools.

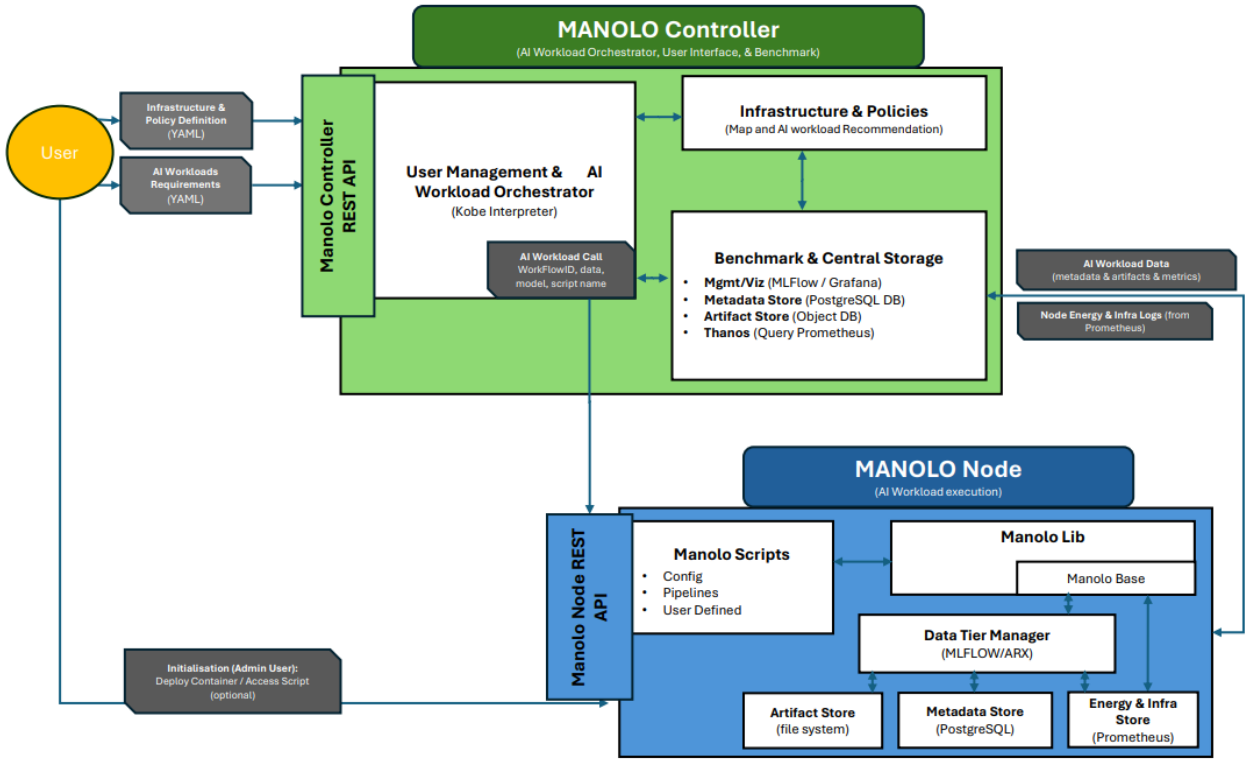


Figure 1: MANOLO Overall Architecture (from Deliverable 1.3)

- **Benchmarking Framework**

The benchmarking infrastructure developed in Task 5.1 is operated through the AI Workload Orchestrator (based on KOBE engine), which parses user-submitted YAML (Yet Another Markup Language) specifications (defining workload type, model, script, and data). Once the workload is launched, benchmarking is executed on the MANOLO Node side, via user-defined scripts and supported Machine Learning (ML) pipelines. Performance and efficiency metrics collected at the node level (e.g., training time, accuracy, resource consumption) are then returned to the central Metadata Store and visualised through tools such as MLflow or Grafana, part of the Data Management/Viz layer.

- **Energy and Infrastructure Monitoring**

D5.1 also introduces the eco-efficiency monitoring subsystem, which involves integrating tools like Kepler, CodeCarbon, and Alumet at the MANOLO Node level. These tools gather runtime data on energy consumption

and carbon impact. Logs and metrics from these tools are exported to Prometheus Monitoring Subsystem, and then forwarded to the Energy & Infra Store, both locally and centrally (via Thanos queries), allowing the MANOLO Controller to access detailed sustainability-related insights.

- **Link with Explainability and Drift Detection**

In Task 5.3, D5.1 introduces initial components for model explainability and data drift monitoring. These functionalities, while running locally on the MANOLO Nodes, are coordinated via the workload orchestrator and feed relevant outputs (e.g., attribution maps, drift scores, alerts) into the same central metadata and observability infrastructure. This allows future components in MANOLO to correlate performance and robustness indicators in real time and over time.

3. Benchmarking Framework

3.1 Benchmarking Concept

Benchmarking in the context of ML is the process of evaluating and comparing the performance of ML workloads under different infrastructure and conditions. It involves running AI Workloads in a range of nodes and collecting relevant metrics to support comparison and optimization.

3.1.1 What is KOBE?

In the MANOLO system architecture, benchmarking orchestration is leveraging KOBE, a benchmarking engine originally developed for evaluating SPARQL query federators. KOBE provides a structured way to run predefined workloads and capturing relevant metrics. In MANOLO, KOBE adapts the user requirements to ML workloads through a YAML configuration file (infrastructure & policy definition). The main adjustments for ML benchmarking include:

Databases → ML algorithms and models
SPARQL queries → ML performance metrics definitions
Original parameters → Training and inference parameters

In the MANOLO system, the KOBE benchmarking engine is responsible for executing ML benchmarking workloads. Each KOBE component is mapped into the specific parts of the MANOLO architecture as follows:

- The user defines benchmarking AI workloads through a YAML specification. This YAML describes the workload type, models, datasets, metrics, and execution details. It aligns with the Infrastructure and Policy Definition (IPD) YAML shown in the MANOLO architecture (see Figure 1). The user submits the YAML to the MANOLO Controller using its REST API.
- The AI Workload Orchestrator, also referred to as the KOBE Interpreter/Orchestrator, in the controller parses the IPD YAML and invokes the appropriate workload. This component manages the end-to-end benchmarking flow by interpreting the specification and coordinating execution.
- Benchmarking workloads are executed on the MANOLO Node, where the actual datasets and ML systems will be located. The MANOLO Controller communicates with the MANOLO Nodes via REST API calls.
- Once benchmarking is complete, the collected metrics are sent from the MANOLO Nodes back to the MANOLO Controller. The controller stores these metrics in the central metadata storage system, using the MANOLO data-tier which uses libraries and tools such as MLFlow and Thanos, and exposes them for visualization through Grafana. These components are part of the Benchmarking and Central Storage layer in the MANOLO architecture.

3.1.2 Infrastructure and Policy Definition YAML Configuration

The benchmarking experiment is defined using one or more IPD YAML files. Each file represents an AI Workload/run, and multiple runs together form an experiment. Each run contains metadata, implementation info, and a sequence of steps (prepare/work) (see Error! Reference source not found. **and**

Figure 2).

Key	Description
Metadata	IDs, version, name, and description of the run.
Implementation	Specifies how to access the system.
Steps	Each step specifies the execution node, implementation parameters, data, and metrics to evaluate.

Table 2: Structure of the IPD YAML Configuration File.

```yaml
metadata:
  parent_id: kn7fej4o
  id: kn7fej4o_01
  version: v1
  name: BitBrain-LSTM-Autoencoder
  description: This run applies a LSTM autoencoder architecture that learns to reconstruct the BitBrain time series dataset.

steps:
  - id: kn7fej4o_01_01
    type: prepare

    node:
      id: edge_server_1
      type: edge

      communication:
        vpn:
          enabled: True
          config_path: s3://manolo-data/wireguard-peer-profile.conf
        ssh:
          host: edge.pal.local
          user: natalia
          port: 22

      deployment:
        docker_image: s3://manolo-data/lstm_ae.tar
        run_mode: http-server
        stage: lstm_ae
        port: 8000
        network_mode: host

    implementation:
      model:
        num_feats: 2
        latent_seq_len: 1
        latent_num_feats: 8
        hidden_size: 4
        num_layers: 1
        dropout: 0.05

      process:
        seq_len: 240
        loss: BlendedLoss
        epochs: 3
        patience: 30
        lr: 0.0001
        optimizer: Adam
        scheduler:
          name: ReduceLROnPlateau
          params:
            factor: 0.99
            patience: 3

      weights: s3://manolo-data/models/lstm_ae.pth

    data:
      loader: bitbrain_torch_loader
      location: s3://manolo-data/datasets/bitbrain-ds/

      parameters:
        process: prepare
        batch_size: 512
        train_size: 2
        val_size: 1
        test_size: 1
        seq_len: 240
        exist: True

    metrics:
      - epochs
      - train_time
      - best_train_loss
      - best_val_loss
      - mae
      - mse

  - id: kn7fej4o_01_02
    type: work

    node:
      id: edge_server_1
      type: edge

      communication:
        vpn:
          enabled: True
          config_path: s3://manolo-data/wireguard-peer-profile.conf
        ssh:
          host: edge.pal.local
          user: natalia
          port: 22

      deployment:
        docker_image: s3://manolo-data/lstm_ae.tar
        run_mode: http-server
        stage: lstm_ae
        port: 8000
        network_mode: host

    implementation:
      model:
        num_feats: 2
        latent_seq_len: 1
        latent_num_feats: 8
        hidden_size: 4
        num_layers: 1
        dropout: 0.05

      process:
        seq_len: 240
        loss: BlendedLoss

      weights: s3://manolo-data/models/lstm_ae.pth

    data:
      loader: bitbrain_torch_loader
      location: s3://manolo-data/datasets/bitbrain-ds/

      parameters:
        process: work
        batch_size: 512
        train_size: 2
        val_size: 1
        test_size: 1
        seq_len: 240
        exist: True

    metrics:
      - mae
      - mse
```

Figure 2: IPD YAML Configuration File Example.

3.2 Benchmarking AI Workloads

- **KOBE Installation:** The core benchmarking AI Workload orchestrator is installed on the MANOLO controller.
- **IPD Configuration File:** A YAML file that specifies all AI Workloads and their models, datasets, and ML systems will be benchmarked.
- **Benchmarked Systems:** The actual ML systems and functionality on the MANOLO nodes.
- **KOBE Connectors:** Adapters that link KOBE on the controller to the ML systems on the nodes via the MANOLO Node API.
- **Data:** Datasets stored on the MANOLO controller and subsequently on the nodes where the models run. Data is stored in MANOLO dataset formats and accessed through data loaders compatible with PyTorch managed via the MANOLO data-tier.

3.2.1 MANOLO Datasets

Different datasets come in other file structures and formats, and other AI systems expect data in different formats. The data-tier component in MANOLO will define the set of Python-based formats for all AI components integrated in the MANOLO library which will be exposed via the MANOLO API for the user to conform.

A MANOLO dataset comprises two files:

- **data file:** This file contains the actual dataset values
- **.json metadata file:** This file provides semantic information about what each column/structure represents, allowing MANOLO to access only the relevant inputs and outputs.

Figure 3 shows an example of a .json metadata file for the Bitbrain dataset.


```
{
  "columns": ["time", "HB_1", "HB_2", "onset", "duration", "begsample", "endsample", "offset", "majority",
  "ai_psg", "night"],
  "features": ["HB_1", "HB_2"],
  "time": ["time"],
  "labels": ["majority"],
  "other": ["night", "onset", "duration", "begsample", "endsample", "offset", "ai_psg"]
}
```


Figure 3: MANOLO JSON Metadata File Example.

The. json metadata file for instance defines the role of each column in the dataset. Below are the fields it must contain, along with their definitions:

- **columns:** Column names in the dataset, in the exact order they appear in the data file. This defines the full schema of the dataset.
- **features:** Column names that represent the input features used by ML models. These columns are passed as inputs during training and inference.
- **labels:** Column names that represent the target outputs the models should predict. These are used as ground truth during evaluation and loss computation.

- **time:** Column names (typically just one) that contain time-related information. This is useful for time series tasks where samples are temporally ordered.
- **other:** Column names that contain additional annotations. These columns are preserved in the dataset but are not directly used as inputs or targets. Examples include experiment settings, auxiliary scores, or segmentation information.

Every column listed in features, labels, time, or other must also appear in the columns list. A column can belong to only one role-specific category; it cannot be both a feature and a label, for instance. It is acceptable for any of the role-specific lists (*features*, *labels*, *time*, *other*) to be empty, depending on the nature of the task. The dataset provider is responsible for preparing the dataset in the required CSV format (for instance see Figure 4).

		time	HB_1	HB_2	onset	duration	begsample	endsample	offset	majority	ai_psg	night
0	0	0.000000	-7.003891	0.869764	0	30	1	7680	0	3	3	1
1	1	0.003906	-5.447471	1.144427	0	30	1	7680	0	3	3	1
2	2	0.007812	-3.158618	2.609293	0	30	1	7680	0	3	3	1
3	3	0.011719	-1.876860	3.707942	0	30	1	7680	0	3	3	1
4	4	0.015625	-0.137331	5.722133	0	30	1	7680	0	3	3	1

Figure 4: Example of a tabular CSV dataset before conversion into MANOLO dataset format.

3.2.2 Data Loaders

KOBE uses dataloaders to prepare that data for training and evaluation: These loaders are implemented as KOBE connectors, since they act as adapters that link the raw datasets (along with their metadata) to the benchmarked ML systems.

Each loader takes as input the data file (which holds the numerical data) and the. json metadata file (which defines what columns are features, labels, time, or other). The loader returns a dataset in the appropriate format (PyTorch Dataset) that can be passed directly into training and evaluation pipelines. Thanks to the metadata, the same dataset can be used for different tasks, such as classification, regression, prediction, or reconstruction, depending on how the loader is configured (e.g., which labels to use, whether time columns are involved, etc.).

These loaders make it easy to switch between different ML systems during the benchmarking process, without rewriting the dataset preparation logic. Since all column roles (features, labels, time, etc.) are declared in the metadata file, the loaders can correctly parse and shape the data regardless of its original layout. Custom libraries can be supported by extending the existing loaders or creating new ones that follow the same design pattern.

3.2.3 AI Workload Orchestration

This example demonstrates how MANOLO's KOBE Interpreter benchmarks multiple ML systems by training their models first, then evaluating them on the same dataset.

- The user passes the **IPD YAML** file via the MANOLO Controller API that lists the ML systems, nodes, datasets, and metrics to collect.
- The kobectl executable invokes the **KOBE Orchestrator** to carry out the experiment.

- The AI Workload Orchestrator dynamically loads the **KOBE Connectors** specified in the IPD YAML and performs the appropriate policy and requirements checks against the Policies & Infrastructure component before progressing, otherwise returns a message/recommendation to the user.
- The AI Workload Orchestrator reads the node configuration from the IPD YAML, then connects to each node to set up the execution environment.
- The Orchestrator orders each connector to start training their respective ML system:
    - The Orchestrator communicates the locations of the **training data** to the connectors in the MANOLO node. KOBE Connectors are responsible for any pre-processing necessary to format the data in manner suitable for the system they support. The connectors provided as part of the KOBE system have a specification for the data format, they can read. If the data does not follow the specification, the user must either pre-process the data or extend the connector.
    - The experiment specification includes **metrics** to collect during training (such as computation time, loss, and accuracy). The AI Workload Orchestrator reads this list and passes it to each connector to perform the appropriate API call to the MANOLO Node API.
- After training ends, the Orchestrator triggers the inferencing phase through the connector API:
    - The inference workload runs on the same MANOLO nodes, using the trained models produced in the training phase.
    - The connectors are in charge of selecting the trained models and run evaluation on the specified **test/inference dataset**, again performing any necessary preprocessing.
- AI Workloads may be composed of **multi-step pipelines** for data processing, training models and inferencing.

4. Energy Efficiency and Model Performance Analysis and Manolo Observability Layer

4.1 Manolo Monitoring

The Manolo Monitoring Subsystem is designed to provide deep, continuous insight into the performance and eco-efficiency of the IT infrastructure. It combines mature, open-source monitoring solutions from the Cloud Native Computing Foundation (CNCF)—such as Prometheus and Thanos—with specialized tools for energy and sustainability analysis, including Kepler, Alumet, CodeCarbon and Scaphandre. Together, these tools enable real-time observability, historical trend analysis, and sustainability tracking at scale across the MANOLO architecture.

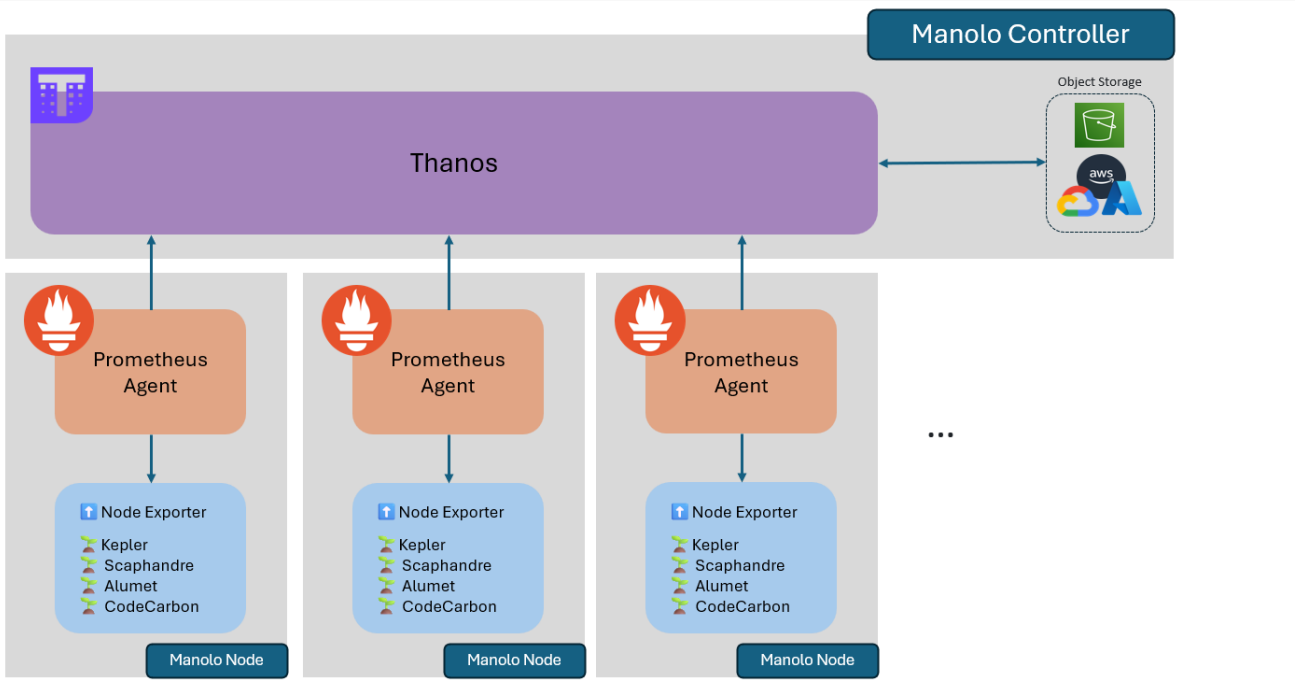


Figure 5: Monitoring solution proposal for MANOLO Nodes and Controller

For performance monitoring, Node Exporter is deployed on all MANOLO nodes to collect low-overhead, system-level metrics. These include CPU and memory usage, disk operations, and network throughput. Running as a lightweight service, Node Exporter ensures minimal impact on system performance while delivering detailed telemetry to detect processing bottlenecks, resource saturation, or hardware imbalances.

At the node level, Prometheus aggregates these metrics in real time, periodically scraping data from Node Exporter and other sources. Its robust time-series database and PromQL query language allow operators to monitor key performance indicators (KPIs), define alerts based on configurable thresholds, and visualize short-term system behaviour. To support long-term observability and scalability, Thanos is deployed at the MANOLO controller level. Thanos extends Prometheus by enabling long-term storage (via S3-compatible object storage). This setup is essential to integrate monitoring data into the MANOLO Data Storage.

To complement performance monitoring with energy and environmental metrics, the subsystem integrates Kepler and leverages hardware counters, Linux kernel telemetry, and machine learning models to estimate energy consumption at the pod, container, or process level. It enables fine-grained monitoring of energy use and carbon footprint across workloads—especially valuable in containerized and orchestrated environments.

Additionally, Alumet provides an application-centric view of sustainability by correlating resource usage with energy metrics, enabling developers and operators to assess the eco-efficiency of specific services or pipelines. Alumet also supports automated recommendations to reduce energy waste, helping teams make informed decisions about workload placement, scaling, and optimization.

Finally, CodeCarbon is an open-source Python package which was designed to track and estimate carbon emissions produced by computational tasks at runtime, especially ML experiments. Since AI models grow in complexity, they normally require increasing amounts of computer power, and this translates into more energy consumption and carbon emissions. CodeCarbon's goal is to encourage sustainable AI practices by increasing awareness of the energy necessary to run the experiments.

By integrating these tools, the Manolo Monitoring Subsystem offers a dual perspective:

- Performance monitoring ensures high system availability, responsiveness, and reliability.
- Eco-efficiency monitoring supports energy-aware decision-making and contributes to carbon footprint reduction.

This comprehensive approach allows Manolo to align operational excellence with sustainability goals, ensuring that IT operations are both performant and environmentally responsible.

4.2 Eco-efficiency Tracking

4.2.1 Kepler

Kepler (Kubernetes-based Efficient Power Level Exporter) is a tool designed for containerized environments that uses a combination of telemetry from Linux kernels (e.g., eBPF and cgroups), hardware performance counters and ML models trained to estimate the power consumption of the processes. Its approach combines hardware precision (RAPL) with eBPF efficiency for scalable and accurate energy monitoring in containerized environments. It can be deployed as a DaemonSet on Kubernetes, ensuring that power metrics are collected across all nodes in the cluster. Additionally, it supports configuration through CRDs (Custom Resource Definitions), enabling fine-grained control over telemetry sources, model selection, and metric granularity.

Kepler reads energy consumption data from hardware-provided RAPL interfaces for CPUs, DRAM and other components, giving precise energy usage at the node level. It uses eBPF to monitor CPU time slices allocated to PIDs, which then allows mapping PIDs to containerized workloads via cgroups, attributing energy usage to specific containers or processes. The RAPL data is then scaled proportionally to CPU usage (e.g., 30% CPU time ~ 30% energy). For components without direct telemetry (e.g. GPUs), Kepler uses trained power models based on metrics such as CPU cycles and instructions.

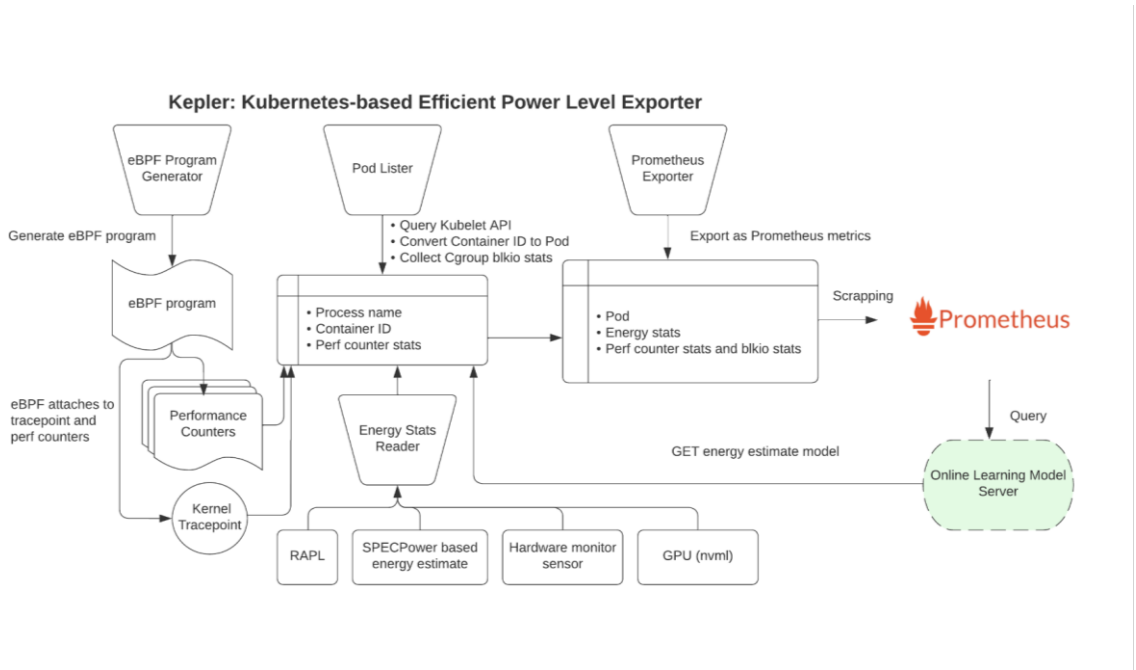


Figure 6: Kepler energy measurement and estimation diagram

Kepler integrates seamlessly with Prometheus by exposing its metrics in a format compatible for scrapping, querying and visualizing, making use of the existing monitoring stack. Metrics exported include container-level power consumption, node-level aggregates, and even custom labels (e.g., workload type, namespace). Some examples are provided in Figure 7.

CPU Metrics (bpf-based and derived)

- `kepler_container_bpf_cpu_time_ms_total`: Aggregated CPU time in milliseconds.
- `kepler_container_cpu_cycles_total`: Total CPU cycles.
- `kepler_container_cpu_instructions_total`: Total executed instructions.
- `kepler_container_cpu_ref_cycles_total`: Reference CPU cycles.

Energy Consumption Metrics

- Container-Level
  - `kepler_container_joules_total`: Total energy consumed by containers.
  - `kepler_container_gpu_joules_total`: GPU energy consumption.
  - `kepler_container_package_joules_total`, `kepler_container_platform_joules_total`, `kepler_container_other_joules_total`, `kepler_container_uncore_joules_total`: Energy usage breakdown by hardware components.
- Node-Level
  - `kepler_node_core_joules_total`: Core-level energy usage.
  - `kepler_node_dram_joules_total`: DRAM energy usage.
  - `kepler_node_package_joules_total`, `kepler_node_platform_joules_total`, `kepler_node_uncore_joules_total`: Node energy breakdown by hardware.

Figure 7: KEPLER Examples

Additionally, out-of-the-box Grafana dashboards are provided for visualizing power usage trends across workloads, namespaces, and clusters over time (see Figure 8, Figure 9 and Figure 10).

Figure 8: Power consumption dashboard showcasing pods running in monitoring namespace

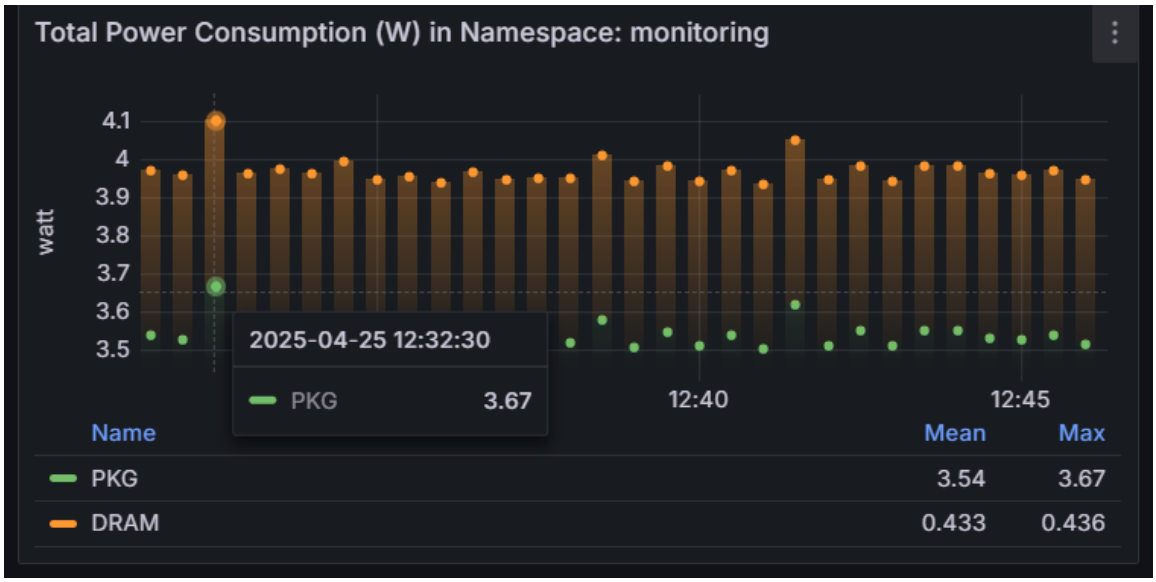


Figure 9: Dashboard with total power consumption for monitoring namespace

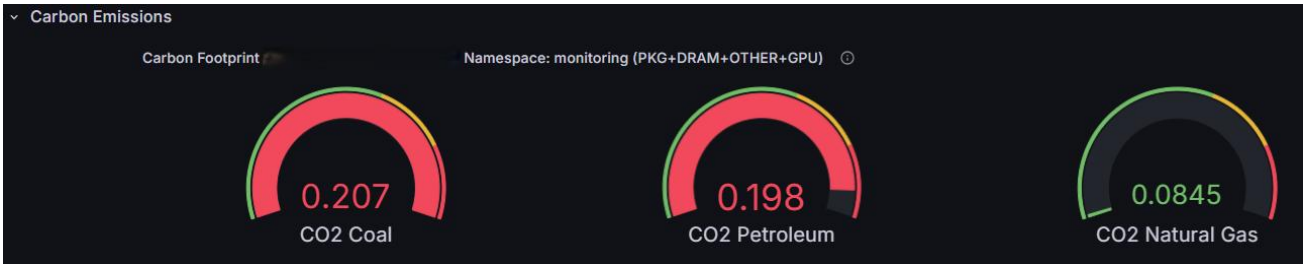


Figure 10: Total consumption of different types of CO2 emissions for monitoring namespace

Regarding use cases, in terms of sustainability and optimization we can highlight the following:

- Green Scheduling: provides input to custom schedulers in Kubernetes aiming to reduce energy usage or carbon emissions by prioritizing energy-efficient nodes or workloads for resource deployment.

- Carbon Footprint Estimation: when combined with cloud provider emission data or region-specific energy conversion rates, these metrics can help estimate the carbon footprint for each workload.

- Cost Optimization: power usage data can be correlated with billing information to gain better insights into energy-to-cost efficiency for each workload.

All in all, Kepler provides the foundational layer for a full stack sustainability perspective, providing additional insights at the infrastructure level with real-time power metrics. The synergy, when used with other tools, allows for a monitoring system that can track sustainability from hardware to software.

4.2.2 Alumet

Alumet (Adaptive, Lightweight, Unified Metrics) is a modular framework for local and distributed eco-efficiency measurement. It is an open-source software tool (released under EUPL license) implemented by ATOS IT (https://alumet.dev/) that focuses on capturing and analysing the carbon footprint of software workloads by bridging runtime information with infrastructure-level emissions data. It aims to provide accurate, fine-grained insights into the sustainability impact of digital processes by correlating execution data from software systems with telemetry from the underlying infrastructure (Figure 11).

Alumet was specifically designed for scenarios where sustainability metrics need to be attributed not only to systems or nodes, but also to individual services, processes, or transactions. At its core, Alumet utilises lightweight agents and/or logging mechanisms to gather metadata on workload execution, such as timestamps, identifiers, and resource requests. These data points are then mapped to energy usage and emissions via two main approaches:

1. **Direct telemetry integration**, when used alongside tools such as Kepler or power-aware cloud APIs, allows Alumet to ingest real-time power metrics at the node or container level.
2. **Emissions modeling**, where in the absence of direct energy measurements, Alumet estimates the energy usage based on utilization metrics (CPU, memory, IO) and matches them with regional carbon intensity data (e.g. using sources such as Electricity Maps or the Carbon Intensity API).

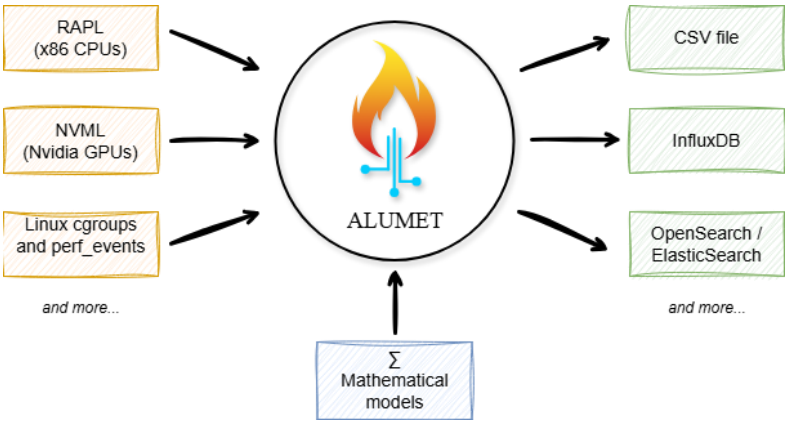


Figure 11: Alumet Monitoring Model

We've recently been working on integrating Alumet with monitoring and observability tools such as Prometheus and OpenTelemetry, making it easier to incorporate sustainability metrics into existing monitoring setups and dashboards. This would be the way to easily integrate Alumet technology with Manolo.

Finally, at the present, Alumet is not compatible with ARM-based systems, which is increasingly important given the growing use of ARM in edge computing and energy-efficient data centers. To address this, ATOS IT

is currently working to extend Alumet's support to ARM platforms, allowing it to deliver energy and sustainability metrics on a wider range of hardware architectures.

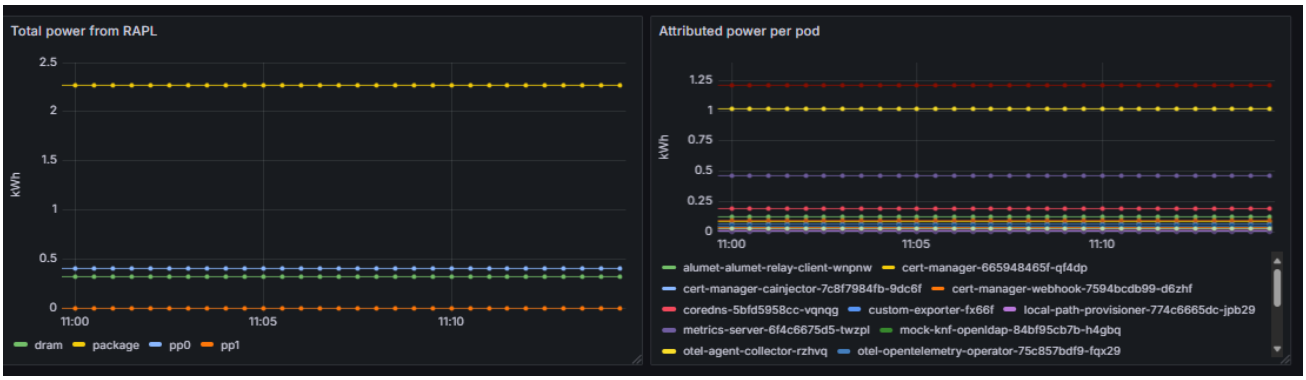


Figure 12: Alumet metrics monitoring

4.2.3 CodeCarbon

As mentioned earlier, this tool aims to encourage sustainable AI practices by monitoring the energy and emissions involved in ML experiments. It works as a Python library that provides high-level CO2 emissions tracking for scripts and ML jobs, giving macro-level environmental impact per ML workload.

First, it tracks energy consumption of CPUs, GPUs and RAM during code execution, supporting both local machines and cloud environments. Then, based on the hardware specifications, utilization over time and the execution time, it estimates the total energy used in kWh (kilowatt-hours) units. In addition, if a location is specified, the tool can use region-specific carbon intensity, e.g. grams of CO2/kWh, from Electricity Maps sources. If a location is not defined, the global average usage of the local or the cloud provider is used as a default.

The expected output can be a JSON or CSV log containing, among other variables, the duration of the execution, the energy consumption, the estimated CO2 emissions and the location-based emission factors. This way, CodeCarbon makes the invisible cost of AI tangible, contributing to more responsible usage, reducing unnecessary emissions and helping developers become aware of the cost of different models and training strategies, thereby optimizing them to achieve higher efficiency.

This tool provides an approximation, as it doesn't actually measure hardware-level energy usage (e.g., using sensors) and its accuracy depends on the availability and granularity of regional intensity data. It is also strongly focused on Python environments, working well in Jupyter notebooks, Python scripts and cloud environments, and is often integrated into ML lifecycle tools such as MLFlow, to provide environmental metrics alongside performance ones.

We have recently started working with this tool and are currently integrating it into our development workflows. A more detailed evaluation and in-depth analysis of its capabilities, limitations, and practical impact will be conducted in the second phase of the project, once we have collected more usage data across diverse scenarios.

4.2.4 Full-stack view

In theory, it is known that CodeCarbon provides information about the environmental impact of ML workloads at a high level, while other tools such as Kepler provide real-time energy metrics for infrastructure at a micro-level. Together, they could provide a full-stack overview, spanning from the energy consumed per script executed to the power usage per container or service running in the specific infrastructure. The **Error! Reference source not found.** provides further details on how the previously described tools can complement each other.

Use Case	CodeCarbon	Kepler	Alumet
Estimating emissions from an ML model	Yes	No	No
Monitoring energy use in Kubernetes	No	Yes	Yes
Reporting CO2 per experiment	Yes	No	No
Building an energy-aware cluster dashboard	No	Yes	Yes
Real-time energy monitoring	No	Yes	Yes

Table 3: Full-stack view

4.3 Benchmarking of embedded devices

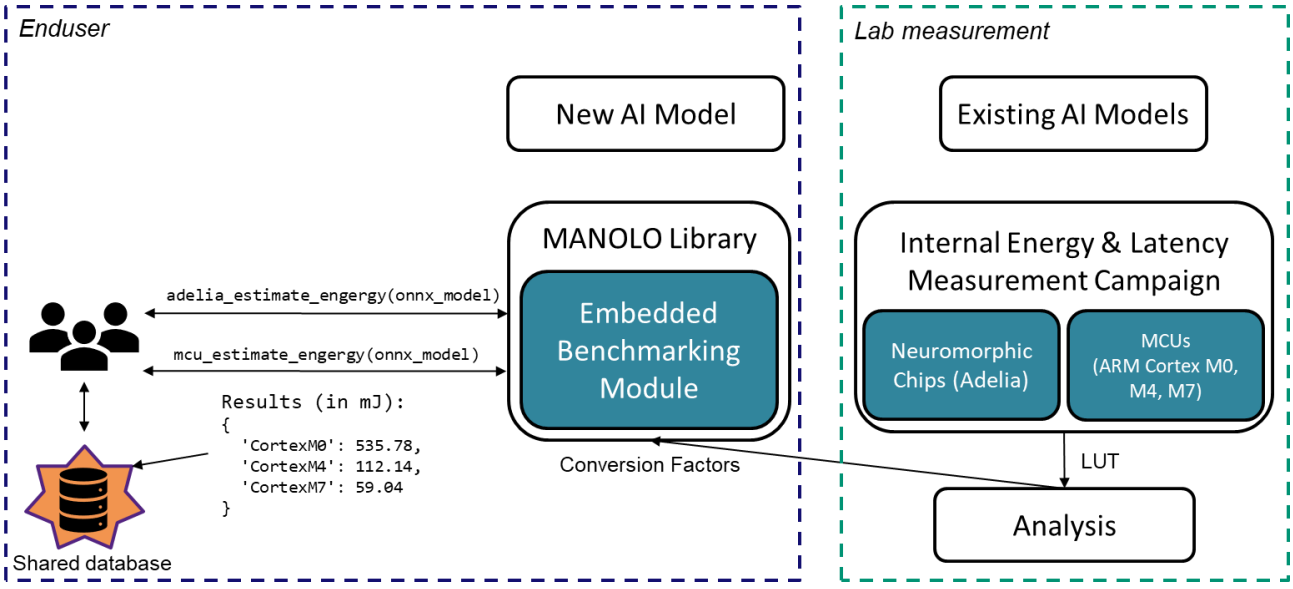


Figure 13: Integration of embedded benchmarking into the MANOLO library

As the tiny, embedded edge devices used inside MANOLO are unable to run Docker containers and therefore do not fit into the overall benchmarking architecture, an additional approach is required. Instead of

performing the energy and latency benchmarking on the device with a software-based solution, an estimation approach is implemented for these devices. Initially, a several existing AI models are benchmarked during an internal measurement campaign performed in a laboratory environment. The resulting database can then be used to derive conversion factors that map the model KPIs (such as number of operations) to the desired hardware KPIs (energy + latency). To provide access to the estimation engine, the embedded benchmarking module will be integrated into the MANOLO library by means of Python APIs to query and report the energy and latency of the different microcontroller platforms (ARM Cortex M0, M4, M7) and the dedicated hardware accelerator (ADELIA).

4.3.1 Tool architecture

The KPI estimator tool comprises two broad components:

Benchmarking Data Collection (LUT Generation):

In this step, controlled laboratory measurements are carried out to create a structured database that links normalized, fine-grained, or coarse-grained hardware operational parameters or primitives to their empirically measured performance on specific classes of edge devices (Cortex M0, M4, M7 and the dedicated accelerator - ADELIA). The LUT stores data for fundamental computational primitives (e.g., cycles per MAC for specific precision, energy per MAC, or memory access of a certain size, etc.). To date, 70% of the benchmarking data collection has been completed in this project.

Estimation Engine:

This component maps the extracted features and operations from the user's model to the characterized hardware parameters/primitives in the LUT. It retrieves relevant KPI data points from the LUT and interpolates these values (if an exact match is not found) to compose final KPI’s like latency, energy, etc.

5. Explainability & Robustness Validation

5.1 Intrinsic & Post-hoc Interpretability: Methods & Metrics

5.1.1 Introduction

As modern AI systems, particularly deep neural networks, become more sophisticated, their internal reasoning processes remain largely opaque ("black-box" models). While these models often achieve state-of-the-art performance across a variety of tasks, the lack of transparency poses significant challenges in safety and regulated environments, where understanding why a particular prediction or decision was made is as important as the prediction itself. In MANOLO, we ensure that AI components are both trustworthy and interpretable by focusing on two core techniques: **intrinsic** and **post-hoc** explainability. Intrinsic explainability refers to models that are transparent by design, while post-hoc methods are applied after training to explain the behaviour of more complex and opaque models. Applications and validations of these are being considered in two representative AI domains: **Time-Series Analysis** and **Image Classification.** By tackling the aforementioned two domains, we aim to demonstrate the generality and limitations of various interpretability approaches. This pipeline consists of: **intrinsic interpretability** (feature (alpha) importance) and **post-hoc interpretability**.

Additionally, we complement these methods with quantitative **metrics**, including fidelity via surrogate models, confidence-drop scores and ROAD perturbation protocol, to assess how well explanations reflect the actual model behaviour.

5.1.2 Data & Model Architecture

Time-Series Domain (Bitbrain)

Dataset

The dataset used for this task is the publicly available Bitbrain Sleep Dataset hosted on OpenNeuro[2]. It contains EEG recordings from multiple subjects undergoing overnight sleep monitoring. The dataset includes five discrete sleep stages, labelled according to standard clinical annotation protocols. In order to tackle the inherent class imbalance, we introduced a sparse categorical focal loss. This loss enhances the model's focus on hard to class/underrepresented sleep stages by down-weighting the contribution of well-classified examples.

Model Architecture

The model consists of two main components:

A Base CNN Model that acts as a feature extractor and a Sequential CNN Model that applies the base model to each of the five consecutive epochs and outputs a sequence of feature vectors, one for each epoch. Predictions can incorporate both past and future context, and the 5th epoch in each sequence is selected for final classification. This leverages a full receptive field when predicting.

[2] Eduardo López-Larraz and María Sierra-Torralba and Sergio Clemente and Galit Fierro and David Oriol and Javier Minguez and Luis Montesano and Jens G. Klinzing (2024). Bitbrain Open Access Sleep Dataset. OpenNeuro. [Dataset] doi: doi:10.18112/openneuro.ds005555.v1.0.0

Image Classification Domain (Caltech101 ~ ArxNET)

Dataset

To evaluate interpretability techniques in the visual domain, we leverage the Caltech-101 dataset[3], which approximates ArxNET's usecase, a standard benchmark in object recognition tasks. The dataset consists of 102 classes (101 object categories plus one background class), featuring images of varying sizes and complexities.

Model Architecture

The CNN used for this task is adapted from a high-performing baseline, re-implemented using PyTorch. The quite simple, yet expressive model, is well-suited for demonstrating the interpretability of CNNs across the different classes, serving as the basis for post-hoc explainability analyses using activation based techniques, particularly class activation mapping (CAM) variants that will be explored later in detail.

5.1.3 Interpretability Methods

Post-hoc Local Methods

To enhance the interpretability of our sequential CNN model for sleep stage classification, we adapted the **LIME**[4] (**Local Interpretable Model-agnostic Explanations**) framework to the structure of multichannel EEG time-series data. Our implementation builds upon the open-source LIME-for-Time-Series repository[5], originally designed for ECG signal analysis, and reconfigures it to accommodate the temporal and sequential dependencies inherent in EEG-based sleep staging.

Objective

This method provides local explanations for predictions made on the **last epoch** of each EEG sequence, highlighting which preceding time windows (epochs) most influenced the model's decision.

Input Format

- Channels: 2 EEG channels (HB_1 and HB_2).
- Input Format: A sequence of 5 epochs per sample (each epoch = 30 seconds, 3840 time points).
- Model Output: A prediction vector per epoch; explanation targets the last epoch.

Explanation Procedure

1. The input sequence is perturbed using random binary masks, which essentially determine which epochs are "kept" or perturbed, selectively replacing entire epochs with either zero values or channel-wise mean values.

2. The perturbed sequences are passed through the trained CNN. Only the prediction for the final epoch is collected (same decision used in inference).

[3] Li, F.-F., Andreeto, M., Ranzato, M., & Perona, P. (2022). Caltech 101 (1.0) [Data set]. CaltechDATA. https://doi.org/10.22002/D1.20086

[4] https://github.com/marcotcr/lime

[5] https://github.com/mdhabibi/LIME-for-Time-Series

3. A linear model is fitted over the generated perturbation-prediction pairs, with sample weights computed using cosine distance from the original input to estimate local importance.

4. The regression coefficients of the local model quantify the **influence of each epoch** in the input sequence **on the final prediction**.

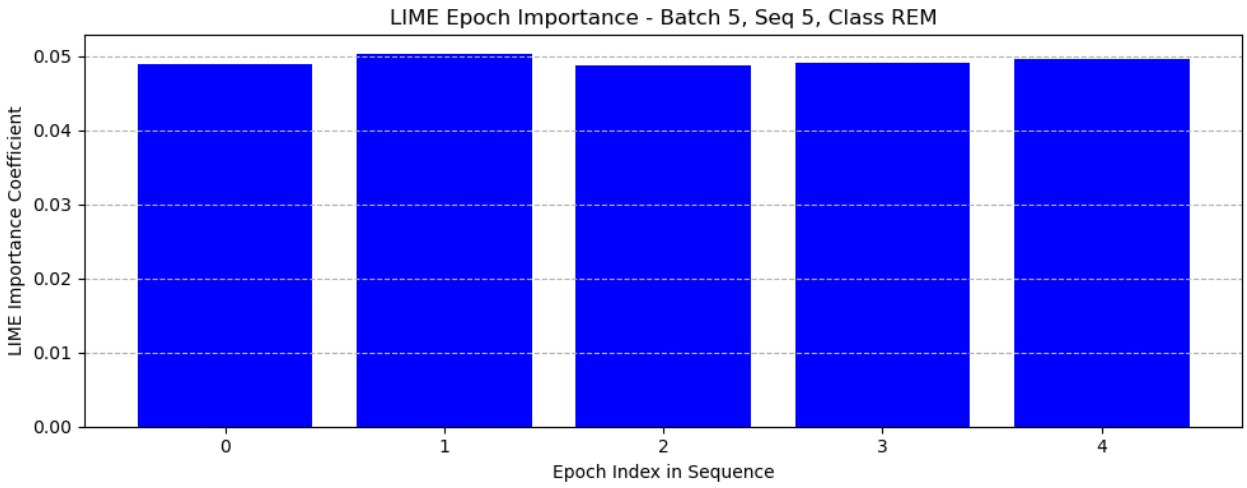
LIME-for-Time-Series Hyperparameters

Two key hyperparameters control the behavior and quality of LIME-based explanations in the time-series setting:

- **Number of perturbations** (*num_perturbations*): Determines how many synthetic variations of the input sequence are generated by masking or replacing epochs. Increasing the number of perturbations generally improves the stability and granularity of the explanation by giving the surrogate model a richer dataset to fit. The cost is higher due to the increased computational overhead, particularly for larger models or long sequences.

- **Kernel width** (*kernel_width*): Showcases the size of the local neighborhood around the original input by controlling how the proximity weights are assigned to perturbed samples. It affects the sensitivity of the surrogate model to variations in the input.
    - Smaller kernel width yields highly localized explanations, where only perturbations very similar to the original input are emphasized. Enhances specificity but amplifies noise.
    - Larger kennel width assigns non-negligible weight to more distant perturbations, resulting in smoother but less specific attributions.
    - In practical terms, it sets how many epochs can differ before a perturbation stops being considered "relevant."

Visualization & Interpretation



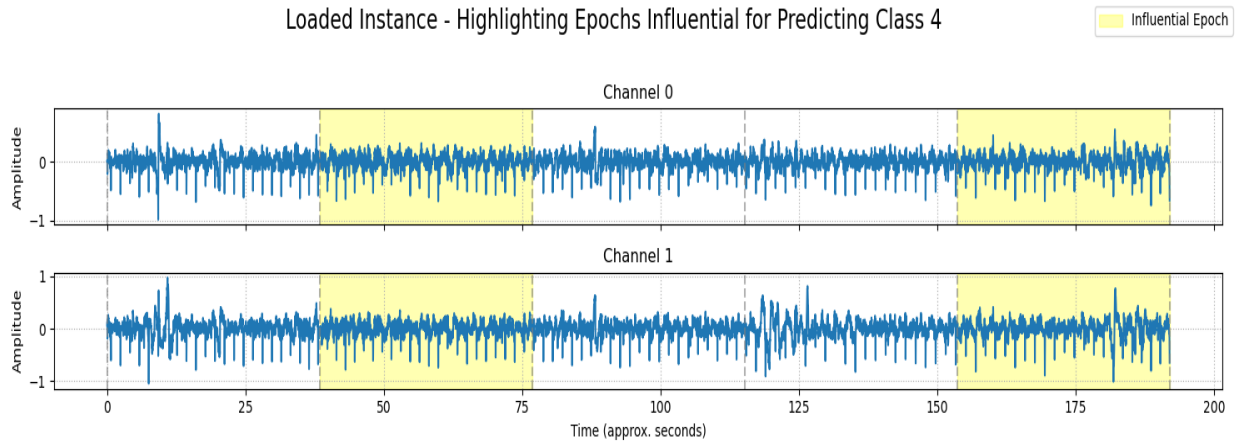
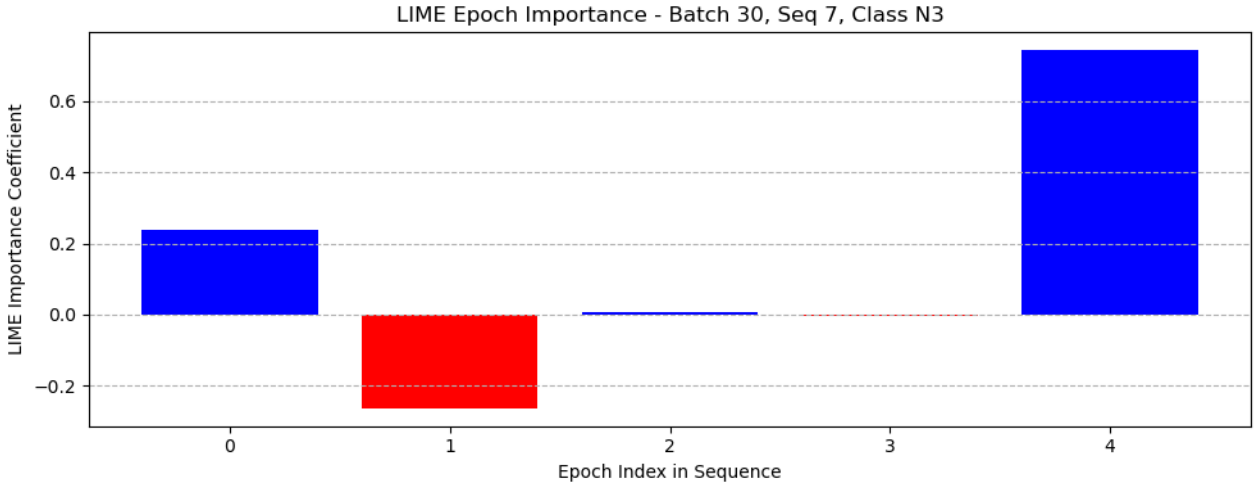


Figure 14: LIME Epoch and Segment Importance for REM Sleep Stage

In Figure 14, LIME assigns nearly equal positive weights to each of the five preceding epochs, showing the CNN draws on a broad temporal context to flag REM. The channel-wise segmentation below highlights faint, sustained high-frequency spikes on both HB_1 and HB_2 throughout the sequence, consistent with the low-amplitude mixed-frequency signature of REM sleep that the model leverages evenly across time.

On the other hand, the bar plot in Figure 15 shows a dominant positive coefficient on the most recent epoch and a negative dip at epoch 1). Correspondingly, the time-series shading marks large, slow delta deflections (±20 µV) in those two windows and minimal activity in the neutral middle epochs. These intervals capture clear, high-amplitude slow waves on both HB_1 and HB_2, tightly aligning the surrogate's attributions with the physiologic delta activity driving the N3 decision.



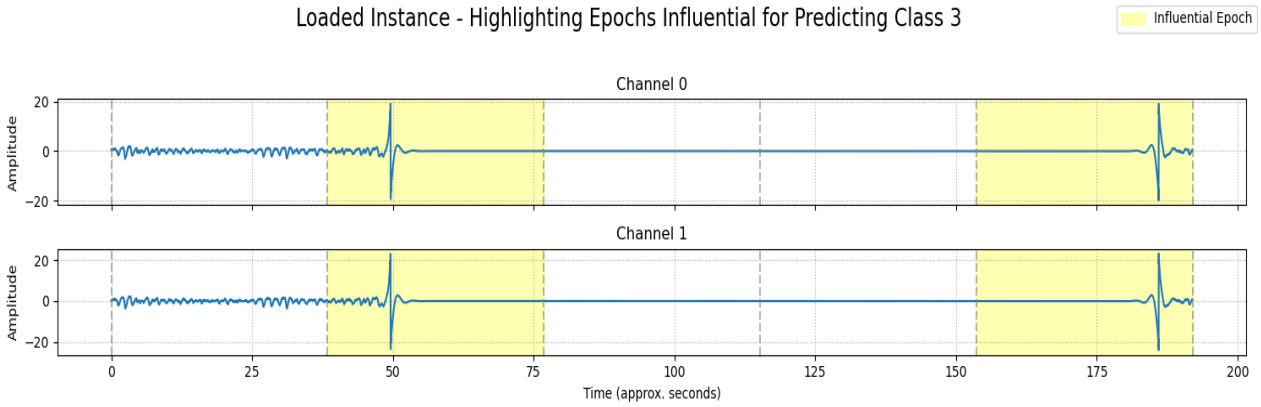


Figure 15: LIME Epoch and Segment Importance for N3 Sleep Stage

Post-hoc Activation-Based Methods

To better understand and visualize the decision-making process of the CNN trained to classify images (Caltech-101), we employed a suite of **class activation map (CAM)** techniques. These methods generate class-specific localization maps, highlighting the spatial regions within an input image that most strongly influence the model's predicted class.

All CAM techniques were implemented using the PyTorch Grad-CAM library[6][7], which provides standardized and extensible support for a wide range of CAM variants, as well as computer vision tasks.

Overview of CAM Techniques

CAM methods fall under gradient/activation-based **post-hoc explainability** techniques. They operate during inference, leveraging intermediate convolutional activations and gradients to backtrack the source of a model's confidence.

The following techniques were used in the analysis:

- **Grad-CAM:** The most widely used method, it utilizes the gradients of the target class flowing into the final convolutional layer to produce a coarse localization map. It highlights semantic regions in the input by weighting feature maps with the gradients of the target class.

- **Score-CAM:** It's a gradient free variant that perturbs feature maps and observes the effect on class scores. Generally, this technique produces smoother and more robust explanations, especially in noisy or high-resolution inputs. Score-CAM gives pixel-space saliency maps by perturbing the input with activation sampled as a mask.

[6] https://github.com/jacobgil/pytorch-grad-cam

[7] @misc{jacobgilpytorchcam,
title={PyTorch library for CAM methods},
author={Jacob Gildenblat and contributors},
year={2021},
publisher={GitHub},
howpublished={\url{https://github.com/jacobgil/pytorch-grad-cam}},
}

- **Eigen-CAM:** A technique based on the principal components of the activation space. It visualizes dominant activation patterns without requiring gradients. It produces class-independent heatmaps by projecting activations onto their dominant eigenvector, revealing the model's global attention.

- **Ablation-CAM:** Systematically ablates individual channels in the convolutional feature map and observes the drop in output confidence. Ablation-CAM gives channel-level importance by selectively disabling internal feature maps and observing output degradation.

- **XGrad-CAM:** Extends Grad-CAM by weighting activation maps not only by gradients but also by the magnitude of activations themselves, improving spatial focus.

Each method provides a slightly different perspective on the model's internal representations and attention, enabling cross-validation of explanations for robustness.

Application & Visualization

To evaluate the effectiveness of different CAM techniques in highlighting relevant visual features, we selected a representative image from the Caltech-101 test set (an airplane) on which the model produced perfect classification confidence. This choice ensures that the visualizations reflect a high certainty prediction, where attribution maps are more likely to align with the object of interest rather than peripheral noise or ambiguous features.

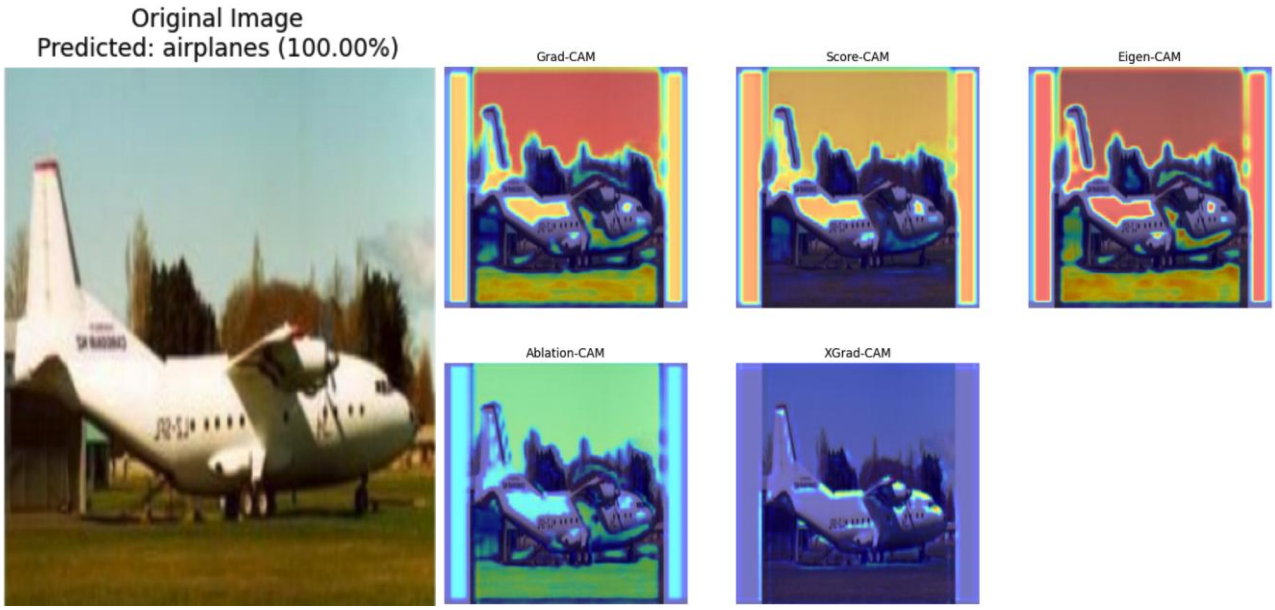


Figure 16: Original Image and various CAM Techniques

Each CAM variant was applied to the same image, as shown in Figure 16, and the model consistently focuses on the fuselage, engines, and wings across methods, confirming that it attends to semantically meaningful features of aeroplane class.

5.1.4 Evaluation Metrics

Feature-based MetricsAlpha Score via Channel-Wise Ablation

To quantify the number of input features needed to explain the model's decision, we employed an **Alpha-Score (a-score) analysis** inspired by the open-source HolisticAI library[8]. This method uses feature ablation to measure individual feature importance and determine the smallest subset required to explain a fixed proportion of the model's confidence

This technique is **model-agnostic** and applicable across various domains (e.g., tabular data, time-series, image patches), where input features are structured interpretable units.**Methodology**

1. Perturbing one feature at a time (zeroing out a channel, masking a patch, dropping a column)
2. Measuring the drop in confidence for the model's predicted class
3. Normalizing the confidence drops to yield a feature importance distribution per input
4. Computing the alpha score, defined as
    a. *The minimum fraction of input features whose cumulative normalized importance reaches a threshold alpha (typically 0.8)*

When no individual feature causes a change in confidence (i.e., all ablations yield zero impact), the resulting importance vector is all zeros. These silent samples are preserved in the analysis to reflect the model's robustness or reliance on feature interactions, as they may indicate strong joint patterns or over-confidence.

This metric offers simple but effective sparsity measure, as:

- A low a-score (0.2) indicates high reliance on a few decisive features.
- A high a-score (> 0.8) implies that the model requires a broader feature set for confident predictions.

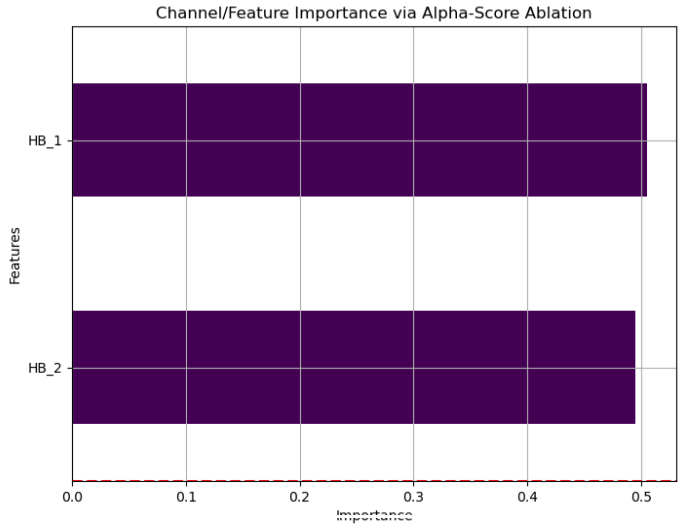
Application in EEG Sleep Classification

We applied this method to our sequential CNN for sleep stage classification on EEG recordings from the Bitbrain dataset. Each input consisted of multi-epoch EEG sequences with two channels (HB_1 and HB_2). The ablation was performed channel-wise by zeroing out each signal channel across time and observing howthe model's prediction confidence changed for the target epoch.

For 11,870 samples and an 80% alpha-score threshold, we got:

- Mean Alpha Score: 0.5572
- Average Importances: HB_1: 0.3601, HB_2: 0.3523

[8] https://holisticai.readthedocs.io/en/latest/



These results suggest the model's decisions are moderately sparse and the model uses features interchangeably (Figure 17), requiring slightly more than half of the available input space to reach 80% explanatory power.

Plot Importances exclude silent samples when normalizing (bars sum to 1).

Ranked values include silent samples in the computation.

Figure 17: multi-epoch EEG sequences

Fidelity via Surrogacy

As mentioned before, a key challenge in explainable AI lies in the accuracy-interpretability trade off. Deep learning models often provide superior performance but are inherently opaque, while simpler models (e.g., decision trees) offer transparent reasoning at the cost of predictive power. To bridge this gap, we apply the surrogate modelling approach that enables insight into the black-box behaviour by training an interpretable model to mimic the predictions of a complex model.

A surrogate model is trained on the predictions of a trained deep model and the goal is to approximate the decision boundaries of the deep model using a surrogate model that is inherently interpretable.

Application in EEG Sleep Classification

In our work, we applied this approach, again, to Bitbrain's EEG dataset. We constructed a surrogate function that maps engineered EEG features, extracted from the final epoch to the original model's predictions, like so:

$$g(\text{band power features}) \approx f(\text{raw EEG})$$

The surrogate model built was a decision tree classifier, trained on a set of biologically meaningful EEG features: delta, theta and low alpha band power.

Methodology

1. (Optional) Computation of frequency-domain features from the final epoch in each EEG sequence.

2. Used the original model to generate predicted labels. Then train a decision tree to map extracted features to these labels.

3. **Fidelity Evaluation (**Error! Reference source not found.**)**
    a. **Surrogate Fidelity Score**: This metric measures how well the decision tree replicates the deep model's predictions.

Confidence Drop Value	Value	Reference
Accuracy Degradation	0.0003	0
Surrogate Fidelity	0.84	1

Table 4: Fidelity Evaluation Metrics

    b. **Accuracy Degradation**: Another metric that compares the performance of the surrogate model to that of the deep model.

4. I**nterpretability Output (Figure 18)**: A visual representation of the decision tree provides a human-readable decision path for each predicted class. The depth and structure of the tree reflect the complexity of the original model's logic.
    a. Class order in 'value': ['W', 'N1', 'N2', 'N3', 'REM'] | Bands: ['Delta', 'Theta', 'LowAlpha']

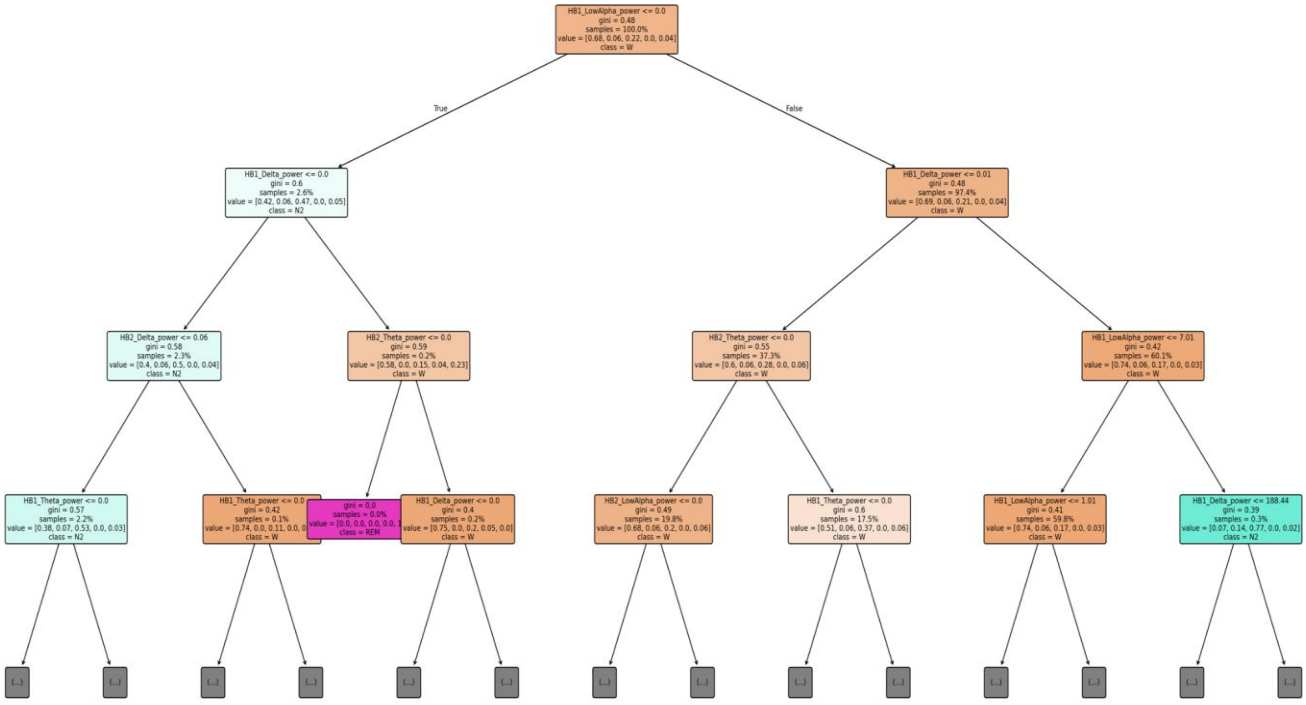


Figure 18: Surrogate Decision Tree (Max Depth: 3) - Features: Frequency Band Power

Activation-map Metrics

To evaluate the effectiveness and faithfulness of Class Activation Map (CAM) explanations, we applied two established metrics: **Confidence Drop** and **ROAD (Remove and Debias)**. These perturbation-based metrics assess how critical and identified salient regions are to the model's actual decision.

Confidence Drop

Confidence Drop quantifies how much the model's predicted probability for the correct class decreases when the most important input regions, as identified by a CAM method, are removed.

It accomplishes that by identifying these most important pixels and removing or masking them. Then it recomputes the model's class confidence, resulting in the drop in the predicted confidence.

If, for example, removing important pixels leads to a large confidence loss, then the explanation is likely faithful.

Confidence Drop Value	Meaning
-1.00	Model completely lost confidence
-0.75	75% confidence lost; CAM likely highlights crucial features.
-0.20	Partial loss; features are helpful but not decisive.
0.00	No change; CAM may be highlighting redundant or uninformative regions.
> 0.00	Model confidence increased; removed features may be misleading.

Table 5: Confidence Drop Values and Interpretations

This metric was used to evaluate Grad-CAM explanations on high-confidence predictions, revealing whether highlighted regions genuinely drive model behaviour.

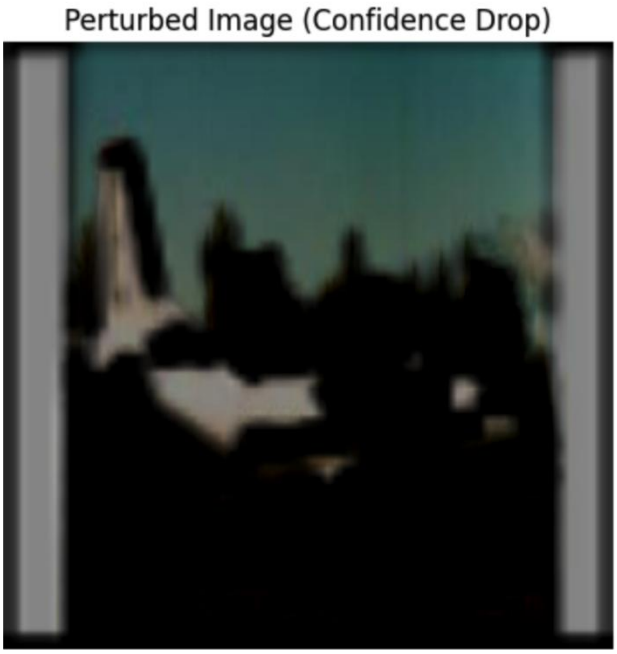


Figure 19: Perturbed Image with Confidence Drop -0.74

The Confidence Drop metric of approximately -0.75 indicates that removing the most salient regions, identified by Grad-CAM, resulted in a 75% decrease in the model's confidence, suggesting that these regions were crucial to the model's decision

ROAD Metric - Most Relevant First

ROAD (Remove and Debias) further explores explanation fidelity by removing a top percentile of most relevant pixels and evaluating the model's classification output. It tests whether the model still recognizes the object without the supposedly most important regions.

Similarly to the previous metric, it identifies the most important pixels provided by a CAM method for the original prediction and then removes, iteratively, the top X% most relevant pixels (we used 75%). Then it re-runs inference on the perturbed image and compares the classification confidence/accuracy.

ROAD Score	Meaning
0.00	No effect; model is redundant or CAM is inaccurate.
0.50	Partial failure; relevant features were important.
1.00	Model completely failed; CAM accurately identified critical features.
< 0.00	Performance improved; removed features were potentially misleading.

Table 6: ROAD Values and Interpretations

ROAD provides an important counterfactual validation. If removing highly scored regions degrades the performance of the model, the explanation is more trustworthy.

Figure 20: Perturbed Image with ROAD Score 0.00

The ROAD score of 0.00 suggests that even after stripping away the top 75% of important pixels, the model still predicts “airplane” just as confidently.

This means that the network probably has plenty of redundant cues spread across the image.

ROAD@75% removes a broader swath, including many lower-importance pixels, which here the model simply compensates for by leaning on the remaining important bits.

5.1.5 Conclusions & Next Steps

This work establishes a solid foundation for explainability validation within the MANOLO framework, leveraging both **intrinsic** and **post-hoc interpretability** techniques across two core AI, and UC-appropriate, domains: time-series and image recognition, which are also relevant to UC. By combining methods such as **LIME**, **CAM** variants, **Alpha Score analysis**, **surrogate modelling** and **perturbation-based evaluation** metrics, we have achieved a structured understanding of model behaviour, identifying not only what the model takes into account when learning but also how and why its decisions are formed.

All explainability functions are provided to the user via the MANOLO Controller UI and dedicated REST API endpoints, packaged as a Kubernetes pod. After each AI workload execution on the MANOLO Node, the interpretability modules automatically generate and store both quantitative metrics and visual artifacts in the central stores. Users can then retrieve and interact with these explanations through the Controller’s dashboard or external tooling (such as Grafana or MLFlow), ensuring full transparency.

From an interpretable standpoint, we plan to extend the surrogate modelling framework by introducing metrics such as **Infidelity** to more precisely quantify the alignment between surrogate models and deep learners. Additionally, we aim to investigate cross-**domain generalizability** of these techniques more thoroughly and explore the use of **Large Language Models (LLMs)** as explanatory agents, capable of not only translating raw importance outputs into human-readable insights but also contextualizing them based on domain-specific priors. However, while LLMs offer exciting opportunities for interpretable and accessible explanations, their deployment raises important ethical concerns, including the risk of generating plausible but misleading interpretations. Therefore, before integrating LLMs into critical pipelines, we will prioritize a thorough evaluation of their reliability, consistency and bias behaviour, ensuring that their use enhances transparency.

Together, these developments will move us closer to trustworthy, transparent AI systems that can be reliably deployed in real-world environments.

5.2 Data Drift Monitoring: Solutions for detecting divergences in serving data

The proliferation of AI models within dynamic cloud-edge computing environments requires robust mechanisms for ensuring their sustained performance and trustworthiness. A critical challenge in this regard is data and concept drifts[9], wherein the statistical properties of the data encountered by a model during

[9] https://www.sciencedirect.com/science/article/pii/S0957417422019522

inference diverge significantly from those upon which it was trained. Such divergences, if undetected and unaddressed, can lead to a silent degradation of model accuracy and reliability, undermining the utility and safety of AI applications.

Within the MANOLO project, addressing drifts is a cornerstone of its commitment to Trustworthy AI. This section outlines MANOLO's dedicated strategy for model and data monitoring, which forms an integral part of the broader framework for Trustworthy AI monitoring, validation, and lifecycle management. Our approach aims to provide a proactive system capable of automatically detecting various forms of shifts in serving data and facilitating timely interventions. This ensures that models deployed via the MANOLO framework maintain their integrity and continue to deliver value even as operational conditions and data landscapes evolve. We leverage the metric generation capabilities of WP3 and the functionalities of the open-source MANOLO library. The proposed framework draws from contemporary research, emphasising data-centric approaches to AI robustness[10].

While specific work in drift detection has already been performed as part of D2.1, this section will define the strategy that MANOLO will follow for drift monitoring and alarms, which development is expected to occur in IT-2. We will introduce the protocol that will be followed for drift detection, its interaction with MANOLO library modules from WP2 and WP3, as well as the monitoring, alarm generation, and reporting logic.

5.2.1 Data, Explainability, and Model Drift Detection

Effective model monitoring relies on understanding data characteristics, the ability to interpret model behaviour through explainability techniques, and the deployment of sophisticated drift detection algorithms.

5.2.1.1 Data Characterization and Monitoring

A key element is the continuous monitoring of input data distributions and output prediction distributions. The MANOLO library will instrument models to collect statistics describing these distributions. This involves tracking:

- **Reference Data:** The statistical properties of the dataset used to train or initially validate the model, serving as a stable baseline[11].
- **Inference Data:** The statistical properties of recent production data encountered by the model during inference.

Changes in key statistical measurements (e.g., mean, variance, percentile distributions) of individual features (univariate drift) or in the relationships between features (multivariate drift) will serve as primary indicators, as well as model explainability features and their outputs (e.g., loss values and model confidence).

5.2.1.2 Drift Detection Mechanisms

MANOLO will focus on detecting several types of drifts relevant to model performance and trustworthiness:

- **Data Drift:** This refers to changes in the probability distribution of the input features P(X). The MANOLO library will employ statistical tests and distance measures to compare the distribution of

[10] https://doi.org/10.1145/3711118
[11] https://nannyml.readthedocs.io/en/stable/

incoming inference data against the reference (training) data, as already used in D2.1 for noise detection.

- **Model Drift:** This involves changes in the underlying relationship between input features and the target variable, i.e., a change in the probability distribution P(Y|X). While direct detection of concept drift often requires labelled data (which may be delayed or unavailable in inf), MANOLO will monitor for its symptoms, such as a significant drop in model performance metrics (e.g., accuracy, F1-score) on any available labelled feedback data, or by using proxy metrics from unlabeled data (model confidence or changes in the loss). Training meta-models to predict model performance based on input data or to monitor model outputs directly can also serve as an early indicator of other forms of drift, even in the absence of ground-truth labels for inference data.

To implement these detection capabilities, MANOLO will integrate and leverage state-of-the-art drift detection algorithms from established open-source libraries, such as river[12], nannyML[13], and frouros[14]. The choice of specific algorithms will be configurable for the MANOLO users and adaptable to the nature of the data and the requirements of the AI application.

5.2.2 Drift-Detection Assisted Model Robustness

The detection of drifts is not an end in itself but a trigger for actions aimed at restoring or enhancing model robustness. MANOLO's strategy extends beyond mere detection to encompass mechanisms for adapting models to evolving data landscapes, with a focus on continual learning paradigms, which will be leveraged by T3.2. Upon detection of drifts, a primary response can be to trigger model adaptation. Instead of full retraining from scratch, which can be resource-intensive, MANOLO will facilitate continual fine-tuning:

1. Incrementally updating the registered model using new batches of (potentially labelled or pseudo-labelled) data that reflect the drifted distribution as a new MANOLO workload (under a new experiment ID).
2. Employing techniques to mitigate catastrophic forgetting, ensuring that the model retains knowledge from previous data distributions while adapting to the new one.

The MANOLO controller will orchestrate these fine-tuning workflows, potentially leveraging new data samples as representative of the current data regime if specified through the workload description YAMLs. The decision to fine-tune and the strategy employed will ultimately be decided by MANOLO users upon investigation of the alarms raised, the nature and severity of the detected drift, and the availability of relevant data.

For scenarios requiring even faster adaptation or where models need to perform well across a range of related tasks or slightly varying data distributions, MANOLO will explore the integration of meta-learning and model aggregation techniques to recommend a new model for a specific task. Unless the user explicitly defines this, the final decision will fall into the MANOLO user, and replacements or new workloads will need to be submitted as described in the MANOLO Architecture Deliverable v2.

[12] river: https://riverml.xyz/

[13] NannyML: https://nannyml.readthedocs.io/en/stable/

[14] Frouros: https://frouros.readthedocs.io/en/latest/

5.2.3 Monitoring and Alarms

The operationalisation of data and model monitoring, as well as drift detection within MANOLO, relies on a robust monitoring infrastructure and a clear system for generating and communicating alarms.

5.2.3.1 Continuous Monitoring Infrastructure

The MANOLO library components, deployed on distributed nodes, will be responsible for continuously collecting data metrics from AI workloads. This includes:

- Computing statistical properties of incoming inference data streams or batches.
- Comparing these properties against the reference data characteristics.
- Executing configured drift detection algorithms at predefined intervals or on specific triggers.

The MANOLO controller will trigger these distributed processes, being MANOLO users able to define which metrics are collected, which drift detection algorithms are applied, and the sensitivity thresholds for these, through the workloads' YAML files.

5.2.3.2 Alarm Generation and Management

Upon detection of statistically significant data drift that exceeds predefined thresholds or a notable degradation in estimated performance, the MANOLO monitoring backend will generate an alarm. These alarms will:

- Provide information about the type of drift detected (e.g., univariate drift in feature X, multivariate drift, output drift), the features involved, the magnitude of the drift, and the specific model/workload affected.
- Provide sufficient information for MANOLO users to initiate appropriate responses, such as triggering actions via the MANOLO controller, including model replacement, workload stoppage, or initiating a retraining/fine-tuning pipeline, as discussed earlier in this section.

All detected drifts and generated alarms will be logged in a distributed object storage system. This ensures traceability for auditing purposes, provides data for backtracking the root causes of model issues, and can aid in compliance with regulatory requirements. Alarms will be accessible to MANOLO users through the MANOLO controller as defined in the "architecture deliverable v2" presented in MANOLO Deliverable D1.3.

5.3 AI Model Protection through Data Drift Monitoring

Once an AI model is deployed into a production environment, it becomes exposed to operational risks that can undermine its security and reliability. An important defence is the implementation of a dedicated AI model protection tool, specifically engineered to monitor for data drift and anomalies. This system functions as a "firewall", safeguarding the model against a spectrum of threats that could otherwise degrade its performance or expose it to malicious exploitation.

This protection mechanism is particularly vital for mitigating risks from carefully orchestrated attacks. These include prompt-based attacks, such as prompt injection or jailbreaking, where malicious actors craft specific input queries to circumvent safety features. Another significant threat is the data poisoning attack, a more

insidious method where corrupted or malicious data is subtly introduced into the model's training or retraining pipeline, gradually skewing its decision-making processes and compromising its integrity over time.

To counter these threats, the AI model protection system is founded upon the principle of establishing a detailed and comprehensive reference baseline. This baseline is derived from an in-depth analysis of the model's original training data or a curated and trusted set of representative prompts. The creation of this baseline involves:

- Schema and Structural Profiling: The system first creates a detailed map of the data's architecture. This involves documenting all data types (e.g., text, images, integers, strings), their expected formats, and any constraints (like image resolution or text length limits). It also identifies relationships between different data features. This structural blueprint ensures that all incoming data, regardless of its type, adheres to the defined standards before further analysis.

- In-depth Statistical Characterization: Beyond just structure, the system calculates a rich statistical profile. For numerical data, this includes computing key descriptive statistics like the mean, median, and standard deviation. For categorical data (which could be derived from text or image classifications), it involves cataloguing the frequency distribution and overall cardinality (number of unique values) of each category. This statistical summary captures the central tendency, dispersion, and shape of the data's distribution across all data types.

- Deployment of Lightweight Anomaly Detection Models: Finally, the system trains specialized, computationally efficient anomaly detection models on the characteristics of the baseline data. These models learn the subtle patterns of what constitutes “normal" data, whether it's typical text structures, image features, or numerical ranges. Their lightweight design allows them to operate in real-time within a production pipeline without causing significant delays, quickly flagging any data that deviates from the learned "normal" patterns.

Together, these elements constitute a multi-faceted reference profile that provides a deep and robust understanding of “normal” data for the AI model.

During the operational phase, this system acts as a proactive security layer, analogous to a firewall. Every single instance of operational data intended for the main AI model is first intercepted and subjected to rigorous scrutiny by the protection system. It compares the incoming data point's structure and statistical properties against the established baseline profile. The lightweight anomaly detection models then provide a probabilistic assessment of whether the instance is anomalous or legitimate.

Suppose an input is detected as malicious or deviates significantly from the baseline, for instance, if a value falls far outside its expected statistical range, an unknown categorical value appears, or the data structure is malformed, it is immediately intercepted. Depending on the system's configuration, the anomalous input can be blocked outright, preventing it from ever reaching the core AI model, or it can be flagged and quarantined for manual review by a human operator.

To ensure accountability and facilitate rapid diagnostics, an automatic report is generated for every such exclusion. This report provides a transparently explanations of the rationale behind the decision, detailing which specific features were anomalous, the nature of the deviation, and the action that was taken.

In essence, this comprehensive approach ensures that the data processed by the model in its production environment remains statistically consistent with the data upon which it was originally trained. This not only

bolsters the model's resilience against poisoning and prompt-based attacks but also improves its general robustness and performance by preventing performance degradation caused by natural, non-malicious data drift. It represents an important practice for the responsible and secure deployment of AI systems.

6. System Integration and Stress Testing

This activity corresponds to task T5.4 that will start on month M18. This task is responsible for the overall integration of the MANOLO system (see also D1.3 MANOLO Architecture v.2 for integration details).

For optimization of the deployment process, T5.4 will plan and execute the preliminary stress testing procedures at in-lab environment. Moreover, it will test various aspects such as components communication, scalability, and overall performance. Once the integration of the MANOLO system is tested, it will be made available for the use cases (see section 6.2).

Multiple technical partners will test the integrated MANOLO toolset/suite/library in different environments, hardware and scenarios and will aggregate the benchmarks. For instance, in terms of networking scenarios, at the CRAAX lab (UPC), an infrastructure environment capable of accommodating a wide range of computing and networking requirements has been set up, including virtualization, container orchestration, and secure remote access. This infrastructure environment, see Figure 21, consists of a Proxmox Virtual Environment (Proxmox VE) running on a dedicated physical server. Proxmox VE is a powerful open-source platform for managing virtual machines (VMs). UPC will create a Kubernetes Cluster with a master (VM) and n workers (VMS). Also, it will be possible to create and give access to a VM independently from the Kubernetes Cluster. Once the initial version of each component is developed by the responsible partner, we will begin the integration process.

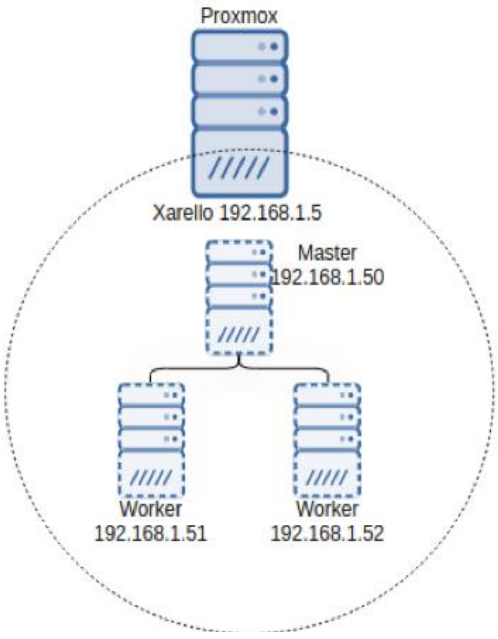


Figure 21: The infrastructure environment provided by UPC

.

6.1 Agile Integration Approach: Early functional prototypes and integration of MANOLO components

Given that most of the partners are still in the process of developing their respective components, the integration phase for the MANOLO system will adopt an agile and adaptive approach. This approach will allow

early functional prototypes of individual components to be integrated progressively as they become available. The integration process will:

- A skeleton and core functionality for MANOLO library and suite development has been implemented in python and uploaded to MANOLO GitHub repository where each component has its own floder to commit and common definitions and interfaces to facilitate integration, along with docker container scripts. This coordinates the whole integration process.
- Each component is being developed as a standalone module first, ensuring proper interfaces and communication protocols.
- A shared integration environment has been established using GitHub Actions to enable frequent and automated builds, integration tests, and deployment. This way commits or merge request are controlled following a central continuous integration/ continuous deployment pipeline to validate the updates and changes.
- For early detection of issues such as potential interface mismatches, data exchange problems, or functional incompatibilities will be identified and resolved promptly by integrating components early and often.
- Close collaboration with component developers will be established that will allow us to send back the integration results, accelerating optimization and stabilization.
- After the discussion with each responsible partner, the incremental versions of functional prototypes will be delivered with deadlines, enabling staged validation and system-level testing as the project progresses.

This agile approach ensures that by the time all components reach maturity, the MANOLO system is already well integrated, reducing risks related to late-stage integration bottlenecks.

6.2 Stress-Testing Scenarios: Using benchmarking framework for monitoring integration

To ensure robust deployment, the task will implement comprehensive stress-testing procedures focusing on all components communication, scalability, and overall system performance. The plan includes:

- To test the realistic scenarios and monitor the behaviour of workloads, the necessary components, such as monitoring or benchmarking, will be deployed and integrated first.
- Initial test scenarios:
  - To test interconnection between the different subcomponents, various user requests with increasing workload will be tested on interconnected sub-components of each component.
  - The same process will be carried out for each component to check the components communication.
  - Scalability tests will be carried out to evaluate how the integrated MANOLO system handles the addition of new components, increased workload, multiple user requests, or expansion of infrastructure.
  - The user-performance requirements and policies, such as response time, resource usage, model performance, and many other metrics will be considered to evaluate the performance.
- An iterative stress testing approach will be adopted to test the integrated MANOLO system with more intensity and complexity.

- For preliminary in-lab testing, basic tests with initial version of the components from each partner will be conducted to fine-tune the required parameters before validating in real use case scenarios.

This systematic stress-testing approach will ensure that the MANOLO system can operate reliably and efficiently in realistic use-case scenarios.

7. Human Oversight Trustworthy Monitoring and Validation

7.1 Z-Inspection® Process: Trustworthy AI assessments with ethical and legal considerations

In MANOLO, especially in Task 5.5, we focus on developing methods and tools for human oversight in monitoring and validating AI systems within the cloud-edge continuum, emphasising trustworthiness and regulatory compliance. We use the EU High-Level Expert Group on AI's TAI principles and requirements[15] encompassing seven key aspects, including human agency, oversight, technical robustness, safety, privacy, data governance, transparency, diversity, non- discrimination, societal and environmental well-being, and accountability.

We employ the Z-Inspection® process for Trustworthy AI assessments, a comprehensive assessment framework considering ethical and legal factors.

The goal is to help evaluate alignment with ethical guidelines, legal regulations, and GDPR requirements, ensuring AI systems' accountability and responsibility throughout their lifecycle, reinforcing trustworthiness and compliance.

7.1.1 Methodology

For this task, we use Z-Inspection®[16], a methodology for assessing the trustworthiness of AI systems. Z-Inspection® uses a holistic, participatory approach, incorporating ethical principles from the EU framework and other sources, and employing socio-technical scenarios to identify potential risks and ethical tensions (part of this text has been taken from the official z-inspection.org web page).

The process involves assembling an interdisciplinary team, analysing claims and evidence, and mapping issues to ethical principles. Finally, Z-Inspection® offers recommendations for mitigating risks and promoting responsible AI development and deployment[17] (see Figure 22[18])

[15] European Commission AI HLEG. (2019). Ethics Guidelines for Trustworthy AI. https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=60419

[16] https://z-inspection.org/about/

[17] https://z-inspection.org/about/

[18] Zicari, Roberto V., et al. "Co-design of a trustworthy AI system in healthcare: deep learning based skin lesion classifier." *Frontiers in Human Dynamics* 3 (2021): 688152.

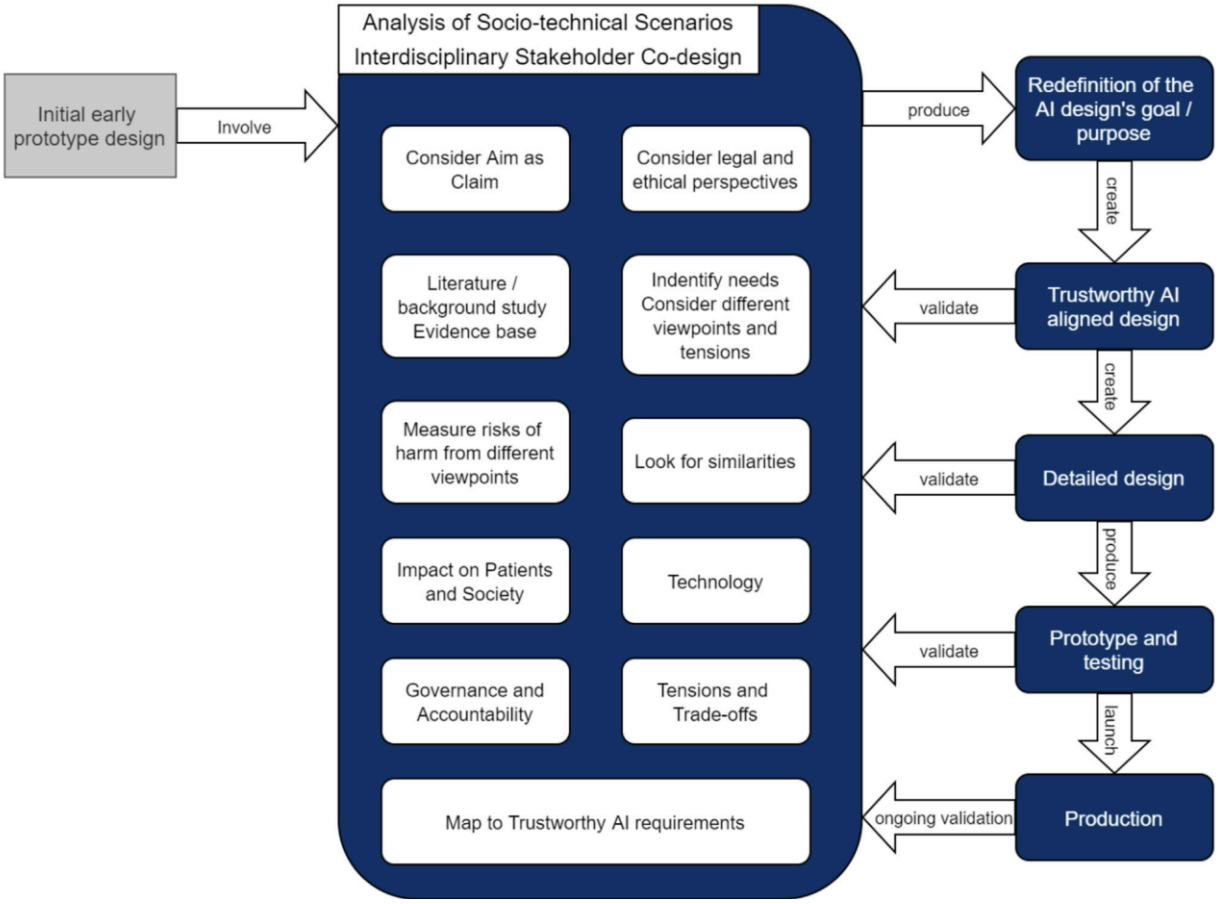


Figure 22: Trustworthy AI co-creation process

7.2 Human Oversight Trustworthy Monitoring and Validation

7.2.1 The Challenge

In MANOLO we are facing the challenge of addressing two sets of developments.

On one hand the design and development of standalone AI components which form the core of the MANOLO architecture.

On the other hand, the design and implementation of a number of Use Cases (UCs) with the goal to test and verify some of the underlying MANOLO architecture.

Monitoring and validating in this context means to continuously iterate the Trustworthy AI co-creation process that we started already in WP 1 and WP6.

7.2.2 The Approach

To iterate the Trustworthy AI co-creation process, we pursue two related approaches.

We created a new dedicated section in the socio-technical scenarios defined for the Use Cases (WP6) asking the UC Stakeholders which components of the MANOLO architecture they plan to test and the benefits they expect from them.

We also consider various requirements specified by the UC stakeholders grouped in a so-called Matrix.
The combination of the two (socio-technical scenarios and matrix) offers a view of how the UCs potentially use and test some selected AI components (and AI pipelines) under development in the MANOLO architecture.

Such a combined view is then mapped to the EU trustworthy AI 4 ethical principles and 7 requirements (including the sub-requirements), providing a so-called closed vocabulary understood by all partners.

At the same time, we also created a new section in each socio-technical scenario for the AI components in WP3, asking the designers which components under development will have a benefit in their view in the context of the various UCs and how the designers of such components consider them relevant for some of the UCs. The Trustworthy AI co-creation is then applied by helping compare the two views, (one top down from UC to components) and one bottom up (from components to UCs) and looking for possible gaps and semantic interpretation. This process helps engaging stakeholders to clarify commonalities and differences of expectation.
A consensus should be reached by using what we call a Triangulation. A triangulation consists of considering the Trustworthy AI design of AI standalone components (as defined in WP3 for example) adding the socio-technical context defined by the respective Use Case(s) where these components are planned to be used and tested. The triangulation process, which is a new element we added to the Z-Inspection® process, is a dynamic one and it is constantly updated as the project development proceeds and more clarity which components of the architecture will be used and tested.

7.2.3 Use Cases

In this deliverable, we will focus on the BitBRAIN use case developed as part of WP6.

BITBRAIN use case

Overview of the use case

To learn more about the use case, the use case owners were asked to complete a Sociotechnical scenario template developed by the ARCADA team. The information provided and completed through an iterative process (see Figure 22) provided the basis for a more thorough understanding of the use case and the identification of technical, ethical, as well as legal issues and tensions.

The BitBrain use case, IoT – Wearables in Healthcare, involves neurotechnology that monitors EEG brain activity and auditory stimulation during sleep that seeks to improve memory consolidation, i.e. the process by which newly acquired information is stabilized and integrated into long-term memory. The approach is based on research that indicates that slow oscillations in brains play an important role for memory consolidation during sleep, and that auditory stimulation synchronized with the phase of slow waves can strengthen and improve memory. The hope is that the approach proves beneficial for older people to reduce age-associated

cognitive deterioration and memory loss, for people with conditions like mild cognitive impairment (MCI) or dementia, but also for healthy individuals who seek to optimize learning.

The owners of the use case seek to integrate the MANOLO technology within the cloud-edge continuum to leverage the strengths of both cloud computing and edge devices to optimize and personalize the intervention that the use case implements. They plan to use the edge device for real-time monitoring and inference, while employing the cloud infrastructure for advanced processing and continuous model training. They hope that integrating MANOLO technology allows them to have a model with low computational cost that is able to reduce battery consumption, running efficiently in the wearable device and keeping it functional for the whole night. In addition, they expect better performance as a result of reduced and stable latency, something which is critical for the required closed-loop operation, i.e., the dynamic, real-time interaction between the brain's responses and the system's actions. In this context, the system continuously monitors brain activity, processes this information, and gives feedback, adjusting auditory stimulation accordingly to achieve the desired outcomes. They also hope to achieve benefits in usability caused by an enhanced user experience with faster and smoother interactions, as well as by the ability to function offline. The operation mode is independent of central servers or cloud, ensuring functionality even in case of network failures or connectivity issues, resulting in improved reliability and availability. Another advantage could consist in user experience tailored to individual user preferences and behaviours in real-time, providing more personalized services, as well as higher flexibility, allowing for scalable deployments and quick adaptation to changing conditions in the local environment.

7.2.4 Trustworthy AI Monitoring and Validation

As mentioned, the combination of the socio-technical scenarios and matrix offers a view of how the UCs potentially use and test some selected AI components (and AI pipelines) under development in the MANOLO architecture. Such a combined view is then mapped to the EU trustworthy AI 4 ethical principles and 7 requirements (including the sub-requirements), providing a so-called closed vocabulary understood by all partners.

7.2.5 Analysis of Socio Technical Scenarios: Identified Technical, Ethical, and Regulatory issue and Tensions

Using socio technical scenarios, we were able to identify a number of technical, ethical, and regulatory issues and tensions, as part of WP6. The following is the list of Technical, Ethical and Regulatory Issues and Tensions for the BITBRAIN use case. This table is taken from the Manolo D6.1 deliverable" Use Cases Design and Deployment Plans" and reproduced here to provide clarity in terms of methodological progress in our co-design.

Claim ID	Claim Description	Relevant Technical Issues	Evidence from Case Log	Issues
CLAIM-1	The headband is easy to self-place, capable of working reliably during the night in terms of signal quality and comfort	Hardware Design (Sensor Placement Signal Quality User Experience)	Requires validation study	Needs empirical validation of user experience across different demographics and sleep conditions. Reliability metrics should be quantified.

Claim ID	Claim Description	Relevant Technical Issues	Evidence from Case Log	Issues
CLAIM-2	DL reference model achieved an 87.08% match between human-consensus labels and network-provided labels for PSG data, and 86.64% match for wearable EEG data	ML Model Performance (Classification Accuracy Validation)	Published dataset (BOAS) on OpenNeuro; Esparza-Iaizzo et al. (2024) bioRxiv paper	Performance metrics can be broken down by sleep stage and patient demographics.
CLAIM-3	Auditory closed-loop stimulation enhances memory consolidation	Therapeutic Efficacy (Real-time Processing Intervention Timing)	Requires validation study	Need larger-scale validation specifically for target population (elderly/MCI). Long-term effects not yet established.
CLAIM-4	The system maintains <100ms latency in real-time processing	System Performance (Edge Computing Latency)	No direct evidence cited	Requires comprehensive benchmarking across different deployment scenarios and hardware configurations.
CLAIM-5	The battery lasts >10 hours	Hardware Performance (Power Management)	No direct evidence cited	Need validation across different usage patterns and environmental conditions. Impact of processing load on battery life should be quantified.
CLAIM-6	The cloud-edge approach improves system performance while maintaining data privacy	System Architecture (Edge Computing Data Privacy)	No direct evidence cited	Formal security analysis needed (e.g. trust model). Trade-offs between performance and privacy should be quantified.
CLAIM-7	Model compression techniques maintain accuracy while reducing computational requirements	Model Optimization (Resource Efficiency)	No direct evidence cited	Quantitative benchmarks needed comparing original vs. compressed models across different metrics.
CLAIM-8	The system provides explainable insights to doctors	Explainability (Model Interpretability Clinical Relevance)	Requires validation study	Need validation of explanation quality with healthcare professionals. Usability studies required.
CLAIM-9	Real-time volume adjustment algorithm effectively personalizes stimulation without disrupting sleep	Adaptive Systems (Personalization Real-time Adaptation)	Requires validation study	Need empirical evidence of algorithm effectiveness across different sleep patterns and user profiles.
CLAIM-10	The system maintains robust performance across diverse home environments	Environmental Robustness (Signal Quality Noise Resistance)	Requires validation study	Need systematic testing across different environmental conditions

Claim ID	Claim Description	Relevant Technical Issues	Evidence from Case Log	Issues
				and interference scenarios.
CLAIM-11	The system provides interpretable insights to doctors through domain expert translation of AI decisions	Explainability (Model Interpretability Clinical Communication)	No direct evidence cited	Need validation studies on effectiveness of expert-mediated explanations. Impact on clinical decision-making should be quantified.
CLAIM-12	Domain experts can effectively translate complex AI decisions into clinically actionable insights	Knowledge Translation (Technical Communication Clinical Relevance)	No direct evidence cited	Requires formal evaluation of translation accuracy and clinical utility. Need metrics for assessing explanation quality.
CLAIM-13	The system maintains data privacy and security in cloud-edge deployment	System Security (Data Protection Privacy Controls)	No direct evidence cited	Need comprehensive security audit and privacy impact assessment. Specific security measures should be detailed.
CLAIM-14	The system adapts to individual user characteristics through personalization algorithms	Adaptive Systems (User Modeling Personalization)	Requires validation study	Need empirical validation across diverse user profiles. Adaptation mechanisms should be quantified and validated.
CLAIM-15	The system operates reliably across different home environments and usage conditions	Environmental Robustness (Signal Quality Interference Resistance)	No direct evidence cited	Systematic testing across different environmental conditions needed. Performance metrics in varied settings required.

Table 7: Identification of Technical Issues and Tensions

Issue	User privacy
Description	It is unclear how user privacy can be guaranteed and how anonymization or pseudonymization of brain data can be achieved.
Tension	Tension between the potentially positive memory-enhancing effects and sleep-disruption
	The tool seeks to improve memory consolidation by stimulating the brain with sound, but by doing so risks disrupting a person's sleep
Tension	Tension between privacy / data protection and increased performance of the tool

Issue	User privacy
	The cloud-edge approach promises to improve performance but risks negative effects with regard to data protection and privacy.
Tension	Tension between energy efficiency and accuracy
Issue	Unclear therapeutic benefits and side effects
	The therapeutic or health-related benefits for people with Mild Cognitive Impairments (MCI), dementia or other memory-related problems are currently not proven. The potential side effects of this approach are unclear. The approach lacks detailed information on how potential therapeutic effects or side effects will be assessed.
Issue	Unclear enhancement effects and side effects
	The enhancement-related benefits for healthy individuals are currently not proven. The potential side effects of this approach are unclear. The approach lacks detailed information on how potential enhancement effects or side effects will be assessed.
Tension	Data needs, balanced data set versus patient privacy
	The approach needs a large data set including particularly sensitive brain data that will have to stem from the patients. The patients have to agree to give away their personal data and brain data. This involves privacy issues.
Issue	Patient informed consent for data usage
	The team has to achieve free and informed consent from the patients / users. The team has to make sure that the users are fully aware of data sharing and that there is no pressure involved. It will not be acceptable to make the willingness to share data a precondition for using the tool, or to automatically use patient data to refine the approach.
Issue	Patient informed consent for study participation
	The primary user group for the study is patients with memory deficiencies, i.e. a vulnerable group. Only patients who are competent to give and withdraw their informed consent can participate in the study.
Issue	Security risks

Issue	User privacy
	Compared to an approach that uses a lab-based laptop, the cloud-edge approach involves additional security risks.
Issue	Legal compliance
	If used in the European Union, the approach has to be compliant with the EU AI Act, the GDPR, the Medical Device Law, and other legal regulations.
Issue	Transparency issues
	Users / Patients need information on how the tool functions, whether and how the tool brings about health-related benefits, what its side effects are, and how all of these will be assessed. They need information on whether and how their data will be used and how anonymization or pseudonymization will be achieved, if possible.

Table 8: Identification of Ethical Issues and Tensions

Issue	Description
Issue	Compliance with GDPR
	If developed in the EU or used by EU citizens, the processing made by the AI model has to be compliant with the GDPR. Notably, this entails: ● Clearly define who is data processor and/or data controller ● Clearly define the legal basis on which the data processing is performed, for every type of processing activity ● Establish the necessity (or not) of a Data Protection Impact Assessment ● Need to consider the informed consent by the patient (examine the case of a patient being incapable for providing consent - relatives' involvement)
Issue	Compliance with the AI Act
	● How will the reliance on the MANOLO framework impact the risk management requirements of the AI Act, as the use-case is considered high-risk
Issue	Compliance with Medical device and Cybersecurity regulation
	● What framework will be used to ensure security of the system? Is this in line with current EU regulation such as NIS II?
Issue	Compliance with domestic/sectorial regulation
	● The MDR is identified as an applicable sectorial regulation ● Are there any specific national legislation applying in the country where the solution will be deployed that may apply?
Issue	Compliance with the Cyber Resilience Act (EU 2024/2847)

Issue	Description
	The Regulation refers to harmonised rules and cybersecurity requirements for bringing into the market products or software with a digital component (sensors, like those used in our use case are included) - full applicability from 11-12-2027

Table 9: Identification of Regulatory Issues and Tensions

7.2.6 BitBrain integration of MANOLO technology

A section to the socio-technical scenarios has been added to help asking stakeholders of the BITBRAIN use case how they envision their Use Case to use and test some of the MANOLO AI components composing the underlying MANOLO architecture.

The output of this new section is a list of AI Components identified by stakeholders of the BITBRAIN Use case likely to be used and tested within their Use Case. This is a starting point for the work which aims to enable monitoring of the components in the UCs. Bitbrain has provided this initial list to be evaluated during the next phase.

- Data Quality Assessment: Reduction of dataset bias for increased trustworthiness, Dataset reduction through cleaning (WP2: T2.2)
- Data Inspection and Distillation: Data pseudonymization (WP2: T2.3)
- Frugal-Oriented Neural Architecture Growth (WP3: T3.4)
- HW-Aware Neural Architecture Search (WP3: T3.3)
- Model Compression: pruning, quantization, knowledge distillation (WP3: T3.1)
- Neuromorphic Computing Algorithms (WP3: T3.5)
- Model selection based on desired efficiency and/or interpretability/explainability characteristics (WP4: T4.2)
- Resource allocation for inference: minimum response time through deployment in proper resources, confidentiality provisions (privacy, secrecy) by the allocated cloud resources (WP4: T4.2, T4.3, T4.4)
- Model learning allocation: resources for model training, model training process start, monitoring and control (WP4: T4.2)
- Explainability Assessment: input-output interpretability methods, surrogate explainable models (WP5)
- Infrastructure Oriented Model and Learning Allocation: allocation of resources based on requirements (WP5)
- Resource and Infrastructure Mapping: mapping memory resources for model parameters and quantization schemes (WP5: T5.1, T5.2, T5.3)

7.2.7 Mapping of Feature and Specification Matrix

Compare Matrix with the Trustworthy AI requirements

In task T6.2, a matrix was developed by various partners, the MANOLO use case feature matrix. It contains a broad spectrum of features and metrics specifications to be used in the MANOLO use cases.

In order to have a common vocabulary, we have mapped the MANOLO use case feature matrix with the seven Trustworthy AI requirements (including the sub requirements)[19]:

1. **Human agency and oversight** (Including fundamental rights, human agency and human oversight);
2. **Technical robustness and safety** (Including resilience to attack and security, fall back plan and general safety, accuracy, reliability and reproducibility);
3. **Privacy and data** governance (Including respect for privacy, quality and integrity of data, and access to data);
4. **Transparency** (Including traceability, explainability and communication);
5. **Diversity, non-discrimination and fairness** (Including the avoidance of unfair bias, accessibility and universal design, and stakeholder participation);
6. **Societal and environmental wellbeing** (Including sustainability and environmental friendliness, social impact, society and democracy); and
7. **Accountability** (Including auditability, minimisation and reporting of negative impact, trade-offs and redress).

This is a short overview of the mapping:

In the MANOLO use case feature matrix, from the perspective of Trustworthy AI requirements, of particular relevance are the following specifications:

Data Quality Assessment,
Data Inspection and Data Distillation,
Resource Allocation for Inference,
Model Learning Allocation,
Explainability Assessment,
Resource and Infrastructure Mapping,

and the following UC-KPIs:

Intelligent Task Performance,
Intelligent Task Efficiency,
AI Model Explainability, System and Infrastructure Operation, and
Usability and User Experience.

Overall, the MANOLO project clearly stresses Energy Efficiency and Model Performance Analysis (T.5.2), Explainability (T.5.3.).

Energy Efficiency can be seen as correlated with the Trustworthy AI Requirement No. 6 *environmental wellbeing, including sustainability and environmental friendliness*.

Model Performance is reflected in the Trustworthy AI requirement No. 2 *Technical Robustness and Safety" and the sub-requirements accuracy, reliability, and reproducibility.*

[19] European Commission AI HLEG. (2020). The Assessment List for Trustworthy Artificial Intelligence (ALTAI) for self assessment. https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=68342

Task T.5.3. is dedicated to *Explainability,* one of the four ethical principles of the TAI related to requirement No. 4. *Transparency*, and considers a broad spectrum of explainability-related specifications.

Privacy-related aspects are of crucial relevance in MANOLO's cloud-edge approach that allows for federated learning which facilitates privacy protection.

Data Governance, in particular the quality and integrity of data, is represented in a number of MANOLO features and methods.

Both are related to the TAI requirement No. 2 Privacy and data governance (*Including respect for privacy, quality and integrity of data, and access to data);*

The MANOLO team takes measures to avoid discrimination and unfair bias, in line with the TAI requirements NO. 5 *Diversity, Non-Discrimination and Fairness*.

BitBrains intentions

For the BitBrain use case, the result of the detailed mapping is presented in the table below.

Requirement	Sub-requirement	BitBrain Matrix
Human Agency and Oversight	Human agency and autonomy	Helper models for trustworthiness co-design aspects (ethical, legal, social robustness)
	Human oversight	Model selection based on desired efficiency and/or interpretability/explainability characteristics; Cloud model training process start, monitoring and control; Input-output interpretability methods (e.g., feature attribution heatmaps)
Technical Robustness and Safety	Resilience to attack and security	Encrypted storage
	Fallback plan and general safety	Infrastructure recovery time in secs
	Accuracy	Classification Top-1 Accuracy; Classification Top-5 Accuracy; Classification Precision; Classification Recall; Classification Average F1 Score; Classification Confusion Matrix; Classification TPs, TNs, FPs, FNs
	Reliability	Performance consistency across different datasets and subject profiles

Requirement	Sub-requirement	BitBrain Matrix
	Reproducibility	
Privacy and Data Governance	Respect for privacy	Data pseudonymization (e.g., use image samples from distilled datasets instead of original R,G,B); Confidentiality provisions (privacy, secrecy) by the allocated cloud resources
	Quality and integrity of data	Dataset reduction through cleaning (e.g., remove noisy samples and outliers based on a threshold)
	Access to data	
	Data governance	Data tier deployed in a container
Transparency	Traceability	Global feature importance metrics Feature Importance Spread Metric
	Explainability	Input-output interpretability methods (e.g., feature attribution heatmaps); Surrogate explainable models; Explainer fidelity Fidelity to Input (FTI) Metric; Surrogate explainer output fidelity Fidelity to Output (FTO) Metric; Explanation determinism - Fidelity of Cause-to-Effect (FCE) Metric
	Communication	Visualizations (E.g. highlight input regions that experience greatest activation)
	Awareness of AI interaction	AI interaction latency; Time of UX update minus start time of user action
Diversity, Non-discrimination and Fairness	Avoidance of unfair bias	Reduction of dataset bias for increased trustworthiness
	Accessibility and universal design	
	Stakeholder participation	

Requirement	Sub-requirement	BitBrain Matrix
Societal and Environmental Well-being	Sustainable and environmentally friendly AI	Reduced power consumption through deployment in proper resources; Cloud energy consumption for training; Cloud energy consumption for inference; Edge energy consumption for inference; Battery consumption for inference per function in mW; Model access from different software implementations; System cost reduction
	Social impact	
	Society and democracy	Alternative models for increasing efficiency and trustworthiness
	Impact on environment and living beings	
Accountability	Auditability	Stability of global explanations Average of Jacard Similarities Metric
	Minimization and reporting of negative impacts	Performance degradation by increasing efficiency
	Trade-offs	Ratio of accuracy to model parameters Relative Model Efficiency (RME) Metric; Efficiency of quantized model; Ratio of accuracy to training sample count; Ratio of accuracy to training time
	Redress	

Table 10: BitBrains intentions

The combination of the socio-technical scenarios and matrix offers a view of how the BITBRAIN Use Case plans to use and test some selected AI components (and AI pipelines) under development in the MANOLO architecture. According to this initial monitoring, the BitBrain use case builds on a broad spectrum of MANOLO features and assessment tools that can be seen in line with the seven Trustworthy AI requirements.

7.2.8 AI Standalone Components

As mentioned, we created a new section in each socio-technical scenario for the AI components in WP3, asking the designers which components under development will have a benefit in their view in the context of the various UCs and how the designers of such components consider them relevant for some of the UCs.
In what follows, as an illustrative example, we consider the following MANOLO component:

T.3.3: HW-aware Neural Architecture Search

The following questions have been asked to the possible user of this T3.3 component. We report as an example, the answers given below by the Bitbrain designers how they foresee using the T.3.3 component.

Q1. How will you likely use /test the HW-aware Neural Architecture Search component in MANOLO?

We will use the HW-aware Neural Architecture Search component to optimize deep learning models for EEG-based real-time sleep stage decoding, slow-wave detection, and closed-loop auditory stimulation. The component will be tested by benchmarking model performance, energy consumption, and inference latency on neuromorphic hardware at the edge. The evaluation will compare standard architectures vs. architectures optimized via HW-aware NAS in terms of accuracy, efficiency, and adaptability to resource constraints.
In particular, the focus could be on:

- Selecting HW-specific models that balance accuracy, energy efficiency, and computational cost for edge deployment.
- Comparing Once-for-All (OFA) and PrototypeNAS approaches to assess energy savings and inference speed in a neuromorphic computing setting.
- Validating the impact of HW-optimized models on real-time performance, specifically ensuring that auditory stimulation is delivered with low latency.

Q2. Please indicate in your words, how you believe your BITBRAIN use case can use and test the HW-aware Neural Architecture Search component?

The HW-aware NAS will be applied to refine the AI models detecting slow-wave sleep patterns, ensuring that they:

- Minimize power consumption to enable extended operation of the EEG wearable headband.
- Fit within neuromorphic HW constraints, ensuring real-time inference on low-power edge devices.
- Maintain high accuracy despite compression, ensuring that slow waves are detected reliably for effective stimulation.
- Guarantee model robustness across different home environments and sleep conditions.

We can test the proposed OFA approach to evaluate its flexibility in selecting optimized models dynamically and PrototypeNAS to determine if predefined HW-specific models better match our neuromorphic deployment.

Q3. How is the HW-aware Neural Architecture Search connected to some of the other components of the BITBRAIN use case? Please specify.

The HW-aware NAS connects with multiple key MANOLO components, including:

- Frugal-Oriented Neural Architecture Growth: HW-aware NAS defines the base model optimizing its structure before frugal worth refines it.
- Model Compression, Quantization & Knowledge Distillation: Once the architecture is selected, additional techniques are applied to reduce memory and computational overhead/needs.
- Neuromorphic Computing Deployment: The optimized models are tailored for execution on a low-power edge device with a neuromorphic chip.

- Data Curation & Training Pipelines: Pre-processed EEG data is used to train and evaluate HW-optimized models.

Q4. With what components of the BITBRAIN use case is the HW-aware Neural Architecture Search supposed to interact?

- EEG Signal Processing & Feature Extraction: to ensure high-quality input data for model training.
- Energy & Performance Benchmarking: to compare models trained with NAS vs. standard models.
- Real-Time Edge Inference on Neuromorphic Chips (to validate deployment efficiency of HW-aware models in real-time settings, ensuring low latency).
- Explainability Tools: to ensure optimized models remain interpretable for domain experts.
- In case of selecting MANOLO Usage Scenario 2, Federated Learning & Continuous Model Adaptation: to ensure the optimized model remains effective under real-world conditions. NAS models evolve over time based on new data for continuous adaptation based on subject-specific EEG responses.

Q5. What benefits does the use of the HW-aware Neural Architecture Search bring to the BITBRAIN use case?

- Optimized model selection and deployment on neuromorphic HW: Seamless execution on resource-constrained devices by disposing of DNN architectures matched to the target HW. Improved energy efficiency: Power consumption reduction, extending overnight battery life for the EEG headband.
- Lower latency: Real-time closed-loop auditory stimulation at the edge without delays. Adaptive architecture selection: Adapts model complexity to different edge devices, allowing scalability without compromising accuracy.

Q6. In your view, how does the integration of the HW-aware Neural Architecture Search modify the use case?

- Enhances real-time performance by selecting AI models best suited for neuromorphic computing, making them more efficient.
- Enables longer autonomous operation in edge devices.
- Reduces computational costs by designing and selecting architectures that are tailored to edge processing and require fewer resources.
- Provides a framework for continuous model adaptation, ensuring optimal performance despite hardware constraints.
- Enhances flexibility, allowing deployment across different edge hardware configurations.

Q7. Do you see any problems in using the HW-aware Neural Architecture Search for the three MANOLO use cases?

- UC1 (PAL): Robots need to balance multiple AI workloads simultaneously (perception, navigation, object manipulation, human interaction), with limited power and computational capacity. NAS must balance real-time decision-making and energy savings. The heterogeneity of HW across different robots (e.g., embedded GPUs, TPUs, CPUs) makes it difficult to generalize NAS results across all deployments.
- UC2 (ARX.NET): AI models for privacy-preserving image processing must be executed locally on mobile devices due to encryption constraints while still maintaining high performance. HW-aware NAS

optimizations may focus on energy efficiency at the cost of accuracy, which could impact the quality of face recognition or object detection tasks. NAS must adapt models to multiple mobile HW platforms.

- UC3 (BITBRAIN): The real-time nature of the system (delivering auditory stimuli precisely timed to brain activity) requires ultra-low latency and efficient execution on neuromorphic HW. Standard HW-aware NAS techniques may not yet be optimized for neuromorphic architectures, requiring additional adaptations to ensure compatibility and lightweight AI models suitable for on-device execution.

Q8. What other MANOLO components do you plan on using or testing in the BITBRAIN use case?

WIP - within task 6.2, the 3 UCs are going to build a mapping table of the MANOLO functionalities and modules that we plan to implement.

7.2.9 Triangulation

A triangulation consists of considering the Trustworthy AI design of AI standalone components (as defined in WP3) adding the socio-technical context defined by the respective Use Case(s) where these components are planned to be used and tested.

These two views need to be reconciled, by identifying possible gaps, semantic differences, and different expectations.

The triangulation process is a dynamic one and it is constantly updated as the project development proceeds and more clarity which components of the architecture will be used and tested.

The triangulation is performed as follows:

- *Step 1. Compare the two approaches above and arrive at a consensus, resulting in a set of AI Components that are likely to be used and tested by the Use Cases.*
- *Step 2. For such selected AI components Iterate the Trustworthy AI co-design using the socio-technical context of the Use Cases where they are planned to be used and tested*
- *Step 3. Iterate this process over time taking into account changes.*

7.3 Training Materials: Development of materials for educating stakeholders

A comprehensive set of educational material on the Trustworthy AI co-design process has been developed and shared to all stakeholders in the MANOLO project, via a number of dedicated Zoom meetings and workshops.

7.4 Comments from the Reviewer

1) *How can generalization of the models be ensured in this sense? So, may --would it be more valuable to conduct Z-Inspection® across multiple distinct use cases to gain insights into how the AI system maintains its trustworthiness across various scenarios?*

   a) We created socio-technical scenarios that go beyond the specific test cases and ensure sustainability of the solutions/technology beyond project`s end. We start with BitBrain and will continue with the other UCs

2) *Does evaluating an AI system through Z-Inspection® within a particular use case lead to optimizations that enhance performance under specific conditions but potentially reduce its adaptability in different contexts?*

   a) If we discover some "issues"/"risks"/ "tensions" we give recommendations. So, the answer is possibly yes, and this should be made transparent to the stakeholders designing the UCs.

3) *How can Z-Inspection® be complemented with other validation methods, such as robustness testing and fairness evaluation across diverse datasets, to ensure the system's reliability beyond a single application domain?*

   a) Several tools should and could be used while testing some of the "issues" (e.g. robustness testing and fairness) during the process. The process allows to use any tools that is supposed to help.

   b) Metrics and benchmarking are the key technologies we have discussed.

4) *The Technical Report should clarify how generalization is ensured to adopt trustworthy principles by design, addressing how optimizations for specific scenarios do not undermine the broader adaptability of the AI system.*

   a) In this deliverable 5.5. we are extending Z-Inspection® with a top down and bottom up approach and a so-called triangulation. In particular, with the bottom up approach, we focus on the MANOLO components, i.e., what we come up with here is not so much at risk of being relevant only in the context of a specific use case.
   b) We also would like to stress that the Z-Inspection® process NEEDS sociotechnical scenarios and uses contexts.

8. Conclusion and Future Work

MANOLO Deliverable D5.1 “Trustworthy Efficiency-Performance Assessment v.1” is the first report where the project looks at how to evaluate the MANOLO system in terms of performance, eco-efficiency, and trustworthiness in a way that MANOLO AI developers can rely on.

In the context of WP5, the consortium started by building a set of tools to measure how fast and efficient the system is, how much energy it consumes, and how we can understand what the AI models are doing. This is indeed an early step, but it already provides a useful starting point to continue improving the overall process.

In Task 5.1, we defined and implemented a first version of the MANOLO benchmarking orchestrator/interpreter using an adaptation of KOBE. This allows us to test different AI workloads across a range of hardware platforms, in a controlled and repeatable way, and to compare results effectively.

We also built a monitoring subsystem that integrates several eco-efficiency tracking tools including Kepler, CodeCarbon, and Alumet. These tools help us to understand where energy is being used, and to estimate the environmental impact of running AI tasks. This aspect is especially relevant in edge computing scenarios, where devices are smaller and energy efficiency becomes a key constraint.

In parallel, under Task 5.3, we started working on making AI more transparent. In other words, we want to understand not just whether models give the right answers, but also *why* they make certain decisions. We tested different explainability techniques, such as LIME and Grad-CAM, and explored how to detect data drift situations where the incoming data starts to differ from the data the model was trained on. These capabilities are particularly important in real-world operations and changing environments.

In coordination with WP4, we also worked on tools to enable continuous monitoring and alerting — for example, when model performance drops or when data characteristics change significantly. This will support more resilient and adaptive AI systems.

Now that most of these tools are in place, the next step is to bring everything together. We’ll start testing the system as a whole, simulating a controlled infrastructure put at disposal by Task 5.4 to see how well it performs under stress.

This includes system-level integration, stress testing, checking component interoperability, and running early validations using selected use cases. We’ll also continue refining the explainability and monitoring tools, making them easier to integrate and more helpful for developers and operators.

Finally, as part of Task 5.5, and using the Z-Inspection methodology, the continuous design of MANOLO component is monitored under the context of the seven TAI principles and against the requirements of the Use Cases.

To sum up: the foundations are ready, the tools are working, and the main ideas of the project are starting to take shape up.

MANOLO

CLOUD-EDGE EFFICIENT & TRUSTWORTHY AI

GA 101135782

CeADAR Ireland's Centre for AI

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Atos

EVIDEN

Technische Universität Braunschweig

DEMOKRITOS

FOUR DOT INFINITY

LAUREA

Inria

université PARIS-SACLAY

ARCADA

KU LEUVEN

Fraunhofer IIS

PAL

Bitbrain

arx.net MOBILE INNOVATIONS

Q-PLAN INTERNATIONAL

eit Digital

Funded by the European Union