Reliable AI and Data Optimization

D2.2 – RAIDO Green AI Framework and Optimised Pipeline

Submission date: 30/06/2025

Project Number:	101135800		
Project Acronym:	RAIDO		
Project Title:	Reliable AI and Data Optimization		
Start date:	January 1st, 2024	**Duration:**	36 months
Deliverable:	D2.2 – RAIDO Green AI Framework and Optimised Pipeline		
Work Package:	WP2		
Lead partner:	UBITECH-LIM (UBI)		
Author(s):	Christos Kotronis (UBI), Georgios Theodorou (UBI)		
Reviewers:	Ioanna Ntinou (QMUL)		
Due date:	30/06/2025		
Deliverable Type:	R—Document, report	**Dissemination Level:**	PU—Public
Version number:	0.6		

Document History

Version	Date	Author	Description
0.1	01/06/2025	Christos Kotronis (UBI)	First release of Table of Contents (ToC).
0.2	06/06/2025	Christos Kotronis (UBI), Georgios Theodorou (UBI)	First draft of Executive summary, Introduction.
0.3	12/06/2025	Christos Kotronis (UBI), Georgios Theodorou (UBI)	Contributions in all Sections.
0.4	20/06/2025	Christos Kotronis (UBI), Georgios Theodorou (UBI)	Preparation of final manuscript for internal review and quality control.
0.5	23/06/2025	Ioanna Ntinou (QMUL)	Internal review and recommendations for improvement.
0.6	29/06/2025	Christos Kotronis (UBI)	Preparation of final version and submission of the deliverable.

Table of Contents

List of Figures

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BD	Big Data
CO2	Carbon Dioxide
COAP	Core Optimized AI-Pipelines
CPU	Central Processing Unit
CT	CarbonTracker
D	Deliverable
DNN	Deep Neural Network
E2C	Edge-to-Cloud
EC	European Commission
EU	European Union
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
Grad-CAM	Gradient-weighted Class Activation Mapping
HITL	Human-in-the-Loop
HPC	High-Performance Computing
HW	Hardware
kNN	k-nearest neighbours
LIME	Local Interpretable Model-agnostic Explanations
M	Month
ML	Machine Learning
OOM	Out-Of-Memory
PID	Process Identification
RAI	Responsible AI
RAM	Random Access Memory
SHAP	SHapley Additive exPlanations
SVMs	Support Vector Machines
T	Task
TPU	Tensor Processing Unit
US	United States
WP	Work Package
XAI	Explainable Artificial Intelligence

Disclaimer

Executive summary

This document presents the “RAIDO Green AI Framework and Optimized Pipeline” (deliverable (D) 2.2), a key deliverable developed under Task (T) 2.2 of the RAIDO project. As Artificial Intelligence (AI) rapidly advances, it simultaneously intensifies demands on computational resources and raises significant environmental concerns. This framework directly addresses these challenges by outlining an innovative approach to optimizing AI workflows and datasets across both edge and cloud infrastructures, ensuring that AI systems are not only powerful but also sustainable.

The foundation of this framework is the Core Optimized AI-Pipelines (COAP) component. It is designed and built for seamless scalability and integrates cutting-edge technologies like Kubernetes for efficient orchestration, along with a robust monitoring stack comprising Prometheus, Grafana, InfluxDB, Kepler, and MLFlow. This comprehensive system continuously tracks energy consumption, hardware (HW) utilisation, and model performance. In addition to technical efficiency, the framework embeds principles of Explainable AI (XAI), Responsible AI (RAI), and Human-in-the-Loop (HITL) feedback. These elements ensure that AI deployments are trustworthy, transparent, and adaptable to real-world scenarios and contexts.

The framework has been rigorously benchmarked across diverse use cases, ranging from image-based applications in precision agriculture and robotics to tabular data scenarios in energy grid management and healthcare. These evaluations highlight crucial trade-offs between performance and energy efficiency, offering valuable insights for tailoring AI solutions to specific deployment environments, whether edge-constrained or cloud-based.

Ultimately, the “RAIDO Green AI Framework and Optimized Pipeline” establishes a foundational ecosystem for developing and deploying AI systems that are not only highly reliable and adaptable, but also environmentally responsible.

1. Introduction

The "RAIDO Green AI Framework and Optimised Pipeline" deliverable (D) 2.2 document represents the essence of Task (T) 2.2, "Core Optimised AI-Pipeline", a critical activity in Work Package (WP) two (2) within the RAIDO project. This task plays a pivotal role in addressing the growing challenges posed by the rapid evolution of Artificial Intelligence (AI), including increased computational resource demands, environmental impact, and the complexity of managing scalable AI systems.

As AI models continue to grow in scale and complexity, they require significant computational power, energy, and sophisticated deployment strategies. These demands span a wide range of diverse infrastructures, from energy-constrained edge devices to large-scale, expansive cloud environments. Addressing these multifaceted challenges is essential to ensure the sustainable, responsible, and effective advancement of AI technologies.

This document, D2.2: "RAIDO Green AI Framework and Optimized Pipeline", presents an innovative framework developed under T2.2 of WP2. Its goal is to deliver an optimized AI pipeline that not only achieves high performance but also prioritizes energy efficiency, ethical governance, and inherent trustworthiness. The framework introduces a holistic approach to managing and optimizing AI workflows and datasets across the entire Edge-to-Cloud (E2C) continuum. It incorporates advanced monitoring tools, flexible and dynamic deployment strategies, and human-centric feedback mechanisms to ensure adaptability and accountability.

Through the exposition of this framework's design, implementation, and benchmarking across real-world use cases, this deliverable demonstrates RAIDO's commitment to building scalable, transparent, and environmentally responsible AI solutions. It establishes a solid foundation for next-generation powerful and efficient AI development aligned with the principles of responsible innovation.

1.1 Objectives of the deliverable

The primary objective of D2.2 is to detail the architectural design, functional capabilities, and technical specifications of the RAIDO optimized AI pipeline. This document aims to demonstrate the pipeline's ability to support energy-efficient, trustworthy, and scalable AI solutions across different deployment scenarios, ranging from edge devices to cloud infrastructures.

1.2 Relation with other tasks and deliverables

The current deliverable, D2.2: "RAIDO Green AI Framework and Optimised Pipeline" (Month (M) 18) is a direct output of T2.2: "Core Optimised AI-Pipeline", which is active from M03 to M30 within the RAIDO workplan. This task specifically focuses on the

conceptualisation, design, and initial implementation of the core AI pipeline, integrating green AI principles and leveraging the E2C infrastructure. It builds upon the foundational guidelines provided by D1.1: "Project Management Handbook: Administration, Technical, Data, Business, and Dissemination" which encompasses administrative, technical, data management, business, and dissemination strategies.

1.3 Document structure

This document is structured to provide a comprehensive overview of the "RAIDO Green AI Framework and Optimized Pipeline" (D2.2), specifically focusing on its design, deployment, and performance benchmarking. It offers insights into its implementation strategies and its role within the broader RAIDO project. The structure is as follows:

- Section 1 ("Introduction"): This section introduces the deliverable's objectives, outlines its relationship with other tasks and deliverables, and presents the overall document structure.
- Section 2 ("Landscape of Big Data AI pipeline training & optimization"): It provides a detailed examination of the current state of AI pipeline development and optimization in Big Data (BD) environments. It outlines key challenges, emerging technologies, and strategic considerations for building scalable, efficient, and environmentally responsible AI systems.
- Section 3 ("Design of the AI theoretical framework"): This section describes the theoretical foundations and practical implementation of the Core Optimized AI-Pipelines (COAP) component. This includes the technology stack, integrated monitoring tools, AI training script integration, Kubernetes deployment configurations and strategies, and methods for visualizing energy and hardware (HW) consumption.
- Section 4 ("AI theoretical framework development & deployment"): This section presents the practical aspects of the framework's development and deployment, including the results of various analyses and implementations.
- Section 5 ("Benchmarking AI Theoretical Framework"): This dedicated section provides a detailed analysis of the benchmarking results for the AI theoretical framework. It includes benchmarking AI models for both image and tabular data modalities, detailing specific use cases, datasets, and observed performance vs. energy consumption trade-offs.
- Section 6 ("Conclusions"): This section summarizes the key findings of the document and outlines directions for future research within the RAIDO framework.
- Section 7 ("References"): This section lists all cited references that support the research and development insights presented throughout the document.

2. Landscape of Big Data AI pipeline training and optimization

There has been a significant shift in the industry due to advancements in Artificial Intelligence (AI) and Machine Learning (ML), leading to high accuracy levels that often surpass human performance across various problems. This high accuracy is typically associated with models possessing an immense number (millions or even billions) of weights (parameters), which are supposed to contain the learned information from training data. However, as Angelov et al. [1] point out, the sheer volume of this data combined with its high-dimensional and non-linear nested structure, makes AI models non-explainable. Therefore, these models are considered non-transparent, "black box" models, a concept highlighted by Mittelstadt et al. [2].

Despite AI's transformative potential, concerns regarding transparency, accountability, and trustworthiness have emerged, prompting the need for Explainable AI (XAI). With the enactment of regulations such as the General Data Protection Regulation (GDPR) in the European Union, the demand for transparent and interpretable AI systems has become more pressing than ever.

Understanding the different core terms related to explainability is crucial. This section clarifies key concepts:

- Transparency: A model is transparent if it is understandable, essentially the opposite of a "black box" model.
- Interpretability: How easily people can understand the reasoning behind a model's decisions.
- Explainability: The capability of a system to deliver accurate and comprehensible explanations for humans, as defined by Gilpin et al. [3].

2.1 The taxonomy and ontology of XAI

While these explanations are close to their semantic meanings, XAI taxonomy can be categorized as follows:

- Transparent models: As presented by Adadi et al. [4], typical examples include k-nearest neighbours (kNN), decision trees, rule-based learning, Bayesian networks, and so on. The decisions from these models are often transparent, though transparency itself, as a property, does not guarantee a model will be readily explainable, as detailed in Figure 1 by Angelov et al. [1].

- Opaque models: Models such as random forest, neural networks (NN), and support vector machines (SVMs) generally offer high accuracy but lack transparency.
- Model-agnostic approaches: Described by Dieber [5], these XAI approaches are designed for broad applicability. They prioritize flexibility, working independently of the specific model architectures. Their focus is on understanding the relationship between a model's input and output without being tied to its internal workings.
- Model-specific approaches: Unlike model-agnostic methods, these techniques leverage specific knowledge about a particular model or models to provide transparency. The goal, as outlined by Bach [6], is to shed light on the inner workings of a specific type or set of models.

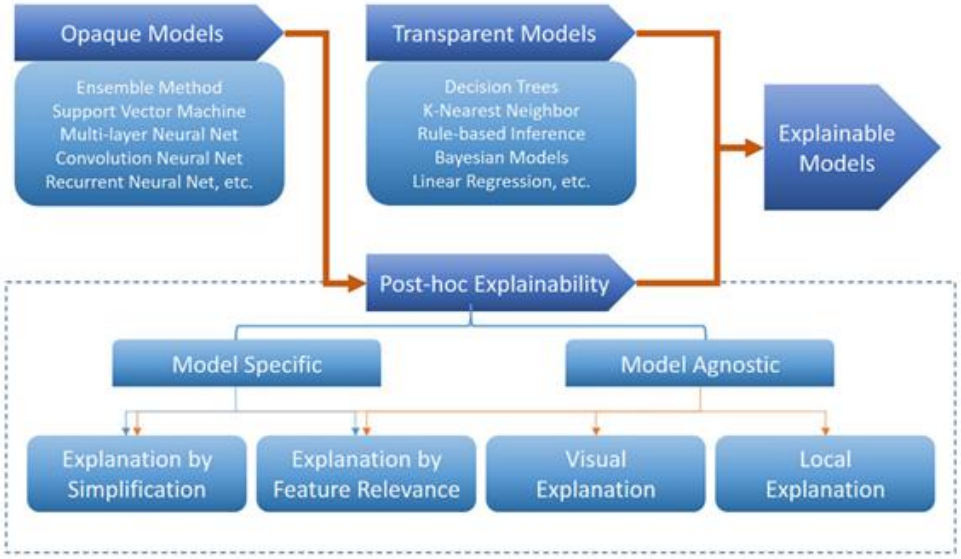


Figure 1: The high-level ontology of XAI approaches (Angelov et. al. [1]).

2.1.1 Post-hoc explainability techniques for model-agnostic approaches

Several methods have been developed to explain complex AI models post-hoc (see Figure 2):

- Text explanations: This technique enhances model explainability by training it to produce text-based explanations of its outcomes. It includes techniques that generate symbols to represent how the model operates, clarifying the algorithm's logic through a semantic mapping from the model to these symbols.
- Explanation by simplification: This method, proposed by Tritscher [7], involves simplifying a model through approximation. By creating simpler surrogate models, such as linear models or decision trees, based on the original model's predictions, complex model's predictions become more understandable.

- Explanation by feature relevance: Similar to simplification, this approach evaluates the importance of features by considering their average expected marginal contribution to the model's decision. Researchers like Chen [8] and Pedreschi [9] explore how different features impact the model's decisions.
- Visual explanation: XAI approaches based on visualisation, as discussed by Chattopadhay [10], rely on data visualisation techniques to interpret model predictions or decisions. By visualizing the input data, these methods aim to provide insights into how the model operates.
- Local explanation: Introduced by Selvaraju [11], local explanations focus on approximating a model within a specific region around a particular instance of interest. This approach offers insights into how the model behaves when presented with inputs similar to the one being explained.

Model	Transparent ML Models			Post-hoc analysis
	Simulatability	Decomposability	Algorithmic Transparency	
Linear/Logistic Regression	Predictors are human readable and interactions among them are kept to a minimum	Variables are still readable, but the number of interactions and predictors involved in them have grown to force decomposition	Variables and interactions are too complex to be analyzed without mathematical tools	Not needed
Decision Trees	A human can simulate and obtain the prediction of a decision tree on his/her own, without requiring any mathematical background	The model comprises rules that do not alter data whatsoever, and preserves their readability	Human-readable rules that explain the knowledge learned from data and allows for a direct understanding of the prediction process	Not needed
K-Nearest Neighbors	The complexity of the model (number of variables, their understandability and the similarity measure under use) matches human naive capabilities for simulation	The amount of variables is too high and/or the similarity measure is too complex to be able to simulate the model completely, but the similarity measure and the set of variables can be decomposed and analyzed separately	The similarity measure cannot be decomposed and/or the number of variables is so high that the user has to rely on mathematical and statistical tools to analyze the model	Not needed
Rule Based Learners	Variables included in rules are readable, and the size of the rule set is manageable by a human user without external help	The size of the rule set becomes too large to be analyzed without decomposing it into small rule chunks	Rules have become so complicated (and the rule set size has grown so much) that mathematical tools are needed for inspecting the model behaviour	Not needed
General Additive Models	Variables and the interaction among them as per the smooth functions involved in the model must be constrained within human capabilities for understanding	Interactions become too complex to be simulated, so decomposition techniques are required for analyzing the model	Due to their complexity, variables and interactions cannot be analyzed without the application of mathematical and statistical tools	Not needed
Bayesian Models	Statistical relationships modeled among variables and the variables themselves should be directly understandable by the target audience	Statistical relationships involve so many variables that they must be decomposed in marginals so as to ease their analysis	Statistical relationships cannot be interpreted even if already decomposed, and predictors are so complex that model can be only analyzed with mathematical tools	Not needed
Tree Ensembles	✗	✗	✗	Needed: Usually *Model simplification* or *Feature relevance* techniques
Support Vector Machines	✗	✗	✗	Needed: Usually *Model simplification* or *Local explanations* techniques
Multi-layer Neural Network	✗	✗	✗	Needed: Usually *Model simplification*, *Feature relevance* or *Visualization* techniques
Convolutional Neural Network	✗	✗	✗	Needed: Usually *Feature relevance* or *Visualization* techniques
Recurrent Neural Network	✗	✗	✗	Needed: Usually *Feature relevance* techniques

Figure 2: Levels of ML model's explainability (Arrieta [12]).

2.2 The theoretical frameworks for AI

Below are some theoretical frameworks for AI most relevant to our work for RAIDO:

- Explainable AI (XAI): This framework focuses on providing structurally sound explanations by combining evidence from the model, its input-output mapping, and human interpretation. It emphasizes faithfulness (accurately describing the model's inner workings) and plausibility (convincing to the user).
- Emergent behaviour and alignment: This framework explores the dynamics of emergent behaviour and alignment in AI systems. It considers the interplay between system states, inputs, function rules, learning algorithms, environments, and historical data, emphasizing the need for ongoing adaptation and control, robust alignment mechanisms, and empirical validation.
- AI as originator and facilitator of innovation: This framework discusses AI's dual role in innovation: as a technology push and a market pull. It explores applications and implications for innovation theory and practice, including AI's contribution to new product development.
- Three-level model for AI in learning: This framework synthesizes existing learning theories and proposes a model that explains the roles of AI in promoting learning processes. It includes micro, meso, and macro levels, identifying fourteen roles for AI in education aligned with the model's features.

At the core of XAI lies a rich tapestry of theoretical frameworks and conceptual models that inform our understanding of transparency and interpretability in AI systems. Tielman et al.'s [32] conceptual framework outlines three (3) essential phases of explanation generation, communication, and reception, providing a roadmap for XAI research. Within this framework, system-centred and user-centred approaches to XAI emerge as distinct domains, each presenting unique challenges and opportunities for advancing the field.

2.2.1 System-centred XAI: Unravelling the 'black box'

System-centred XAI focuses on elucidating the inner workings of AI systems, particularly opaque "black-box" architectures that are not easily interpretable using traditional methods. The inscrutability of black-box systems poses a formidable challenge to transparency and accountability, prompting researchers to explore novel techniques for explanation generation. Recent advances in interpretable AI, such as model distillation and surrogate modelling, offer promising avenues for enhancing transparency without compromising predictive performance. Furthermore, the emergence of grey-box systems, which combine symbolic and sub-symbolic approaches, represents a synthesis of transparency and performance, offering a middle ground between complete opacity and full interpretability.

2.2.2 User-centred XAI: Bridging the explanation gap

User-centred XAI seeks to empower end-users with insights into the decision-making processes of AI systems, accommodating varying levels of understanding and information needs. Clear communication and user understanding are crucial, as users navigate complex concepts to make informed decisions based on AI-generated insights. Pragma-dialectical theory provides a roadmap for crafting persuasive, intelligible explanations tailored to end-users' cognitive capacities and goals. Meanwhile, Inference Anchoring Theory bridges inferential structures in argumentation with their corresponding illocutionary forces and dialogical processes, allowing explanations to adapt to specific user contexts.

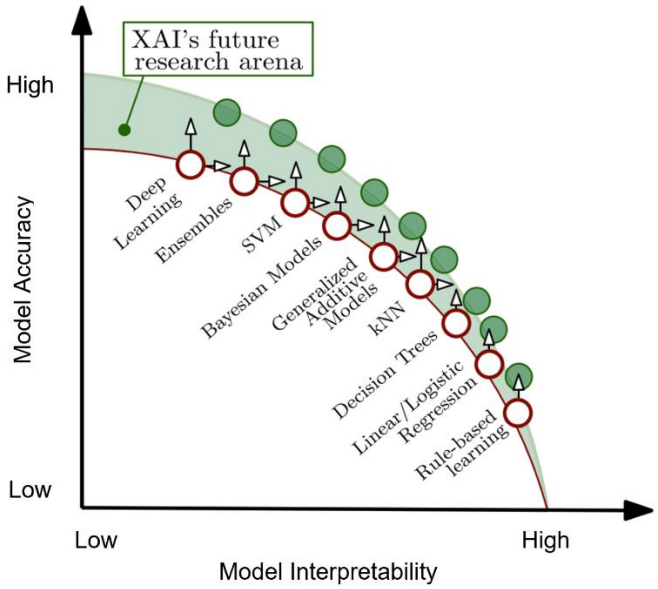


Figure 3: Trade-off graph for interpretability & performance (Arrieta et al. [12]).

Figure 3, influenced by prior research, conceptually illustrates how XAI can enhance the traditional trade-off between model interpretability and performance. An additional related point is the approximation dilemma: explanations crafted for an ML model must strike a balance between being sufficiently drastic and approximate to meet the intended audience's needs, ensuring the explanations accurately reflect the model under study without oversimplifying its key characteristics.

2.2.3 Energy tracking: Existing methodologies & frameworks

A comprehensive study highlights that training Ai with Big Data (BD) can vary widely in both performance and energy consumption, depending on the diverse objectives and type of Deep Neural Network (DNN) architectures [13] used.

When optimising the training phase, it is crucial to consider both the system architecture, including Central Processing Units (CPUs), Graphics Processing Units (GPUs), and Tensor Processing Units (TPUs), and the AI model complexity. While GPUs

and TPUs provide high throughput for tasks like image recognition and speech-to-text, their energy efficiency can greatly differ depending on specific hardware (HW) and applied optimization techniques. These findings underscore the necessity of considering both performance and energy consumption during HW selection for AI training, particularly in environments where cost, energy preservation, and efficiency are paramount.

The importance of evaluating energy consumption in Machine Learning (ML) is widely recognised for monitoring, understanding, and optimising its computational and environmental impact. However, there is no single universal approach that can address all use cases, and an ongoing debate exists regarding the best methods to evaluate energy consumption for specific applications. To address this, various methods with unique strengths and limitations have been developed. For RAIDO purposes, a systematic review of these approaches, designed to evaluate energy consumption during both training and inference, was conducted. This was followed by an experimental protocol to compare their effectiveness across diverse AI tasks, including vision and language models [14].

Several specialized libraries have emerged to track energy consumption in AI pipelines:

- eco2AI library [15] offers a powerful solution for monitoring energy usage and CO2 emissions during both training and inference phases. It tracks CPU, GPU, and RAM utilisation by gathering power consumption logs through process identification (PID) and system metrics retrieved via Linux commands (e.g., “top”).
- EfiMon tool [16] provides a granular, non-invasive method for process-level energy consumption tracking. It uses regression-based models to estimate energy usage with high precision, even in shared computing environments. This tool has demonstrated minimal deviations in measurements on Intel and AMD systems, making it a valuable resource for optimizing energy consumption in AI research and High-Performance Computing (HPC) [16].
- EIT tool [17] simplifies real-time monitoring of energy usage and carbon emissions during AI training. This tool facilitates the generation of standardized online reports and leaderboards, promoting responsible research practices, particularly for energy-efficient reinforcement learning algorithms.
- CarbonTracker (CT) [18] is a specialized tool for tracking carbon emissions produced during AI model training. This framework helps researchers measure the environmental impact of their models, offering valuable insights for developing more sustainable AI systems.

2.2.4 Edge-to-Cloud (E2C) multi-objective deployment and AI explainability

At the same time, the industry has witnessed a significant transformation driven by AI pipelines with a high level of accuracy performance, often surpassing human capabilities across various problem domains. This high accuracy rate is frequently attributed to models with vast numbers (millions or even billions) of weights (parameters), which are supposed to encapsulate learned information from training data. However, many modern AI methods have a “black-box" nature, which hinders their adoption by practitioners in many critical application fields. This issue has led to the emergence of a new research area in AI, setting the groundwork for: (i) extensive benchmarking over different algorithms and methods to understand their behaviour across different data modalities; (ii) utilising data and features of varying granularity and veracity to optimise the learning capability and AI model performance; (iii) monitoring the underlying compute resources to estimate the financial, computational, or energy costs of AI model training, and to derive trade-offs (e.g., through what-if analyses) regarding smart placement, energy consumption, and other business-defined objectives; and (iv) the need to explain the behaviour of an AI model, aiming to provide more understandable, interpretable, and justifiable AI-based decision-making processes and outcomes for humans.

Several theoretical frameworks for AI have been introduced to tackle these multifaceted challenges, focusing on key directions:

- XAI: Tchuente et al. [19] proposed a new methodological and theoretical framework for XAI, decomposed into six (6) steps for practitioners or stakeholders to improve XAI implementation and adoption in their business applications. They highlighted the need to rely on domain-specific and analytical theories to explain the entire analytical process, from business question relevance to the robustness of XAI methods explanations. Rizzo et al. [20] defined AI model properties as faithfulness (i.e. the explanation is an accurate description of the model’s inner workings and decision-making process) and plausibility (i.e. how much the explanation seems convincing to the user). Their theoretical framework simplifies the operationalisation of these properties, offering new insights into common explanation methods that they analyse as case studies. They also discussed its impact in biomedicine, where XAI is crucial for building trust.
- Emergent behaviour and alignment of AI: Freund et al. [21] explored the complex dynamics of emergent behaviour and alignment within AI systems. They presented a comprehensive framework for conceptualising and modelling these phenomena, incorporating the multilevel and time-dependent nature of emergent behaviour and alignment. Their work considers the interplay between system states, inputs, function rules, learning algorithms, environments, and historical

data, shedding light on challenges and opportunities in achieving and maintaining alignment in AI systems.

- AI as originator and facilitator of innovation: Brem et al. [22] introduced a two-part conceptual AI framework. The first part views AI as fulfilling different roles within a company, while the second part examines AI's use throughout the company's innovation processes. They also discussed these two views using field examples and described potential future areas and limitations of the proposed framework.
- Three-level model for AI while learning: Gibson et al. [23] introduced a three-level model that synthesises and unifies existing learning theories to model AI's roles in promoting learning processes. Drawn from diverse fields, including developmental psychology, computational biology, instructional design, cognitive science, complexity, and sociocultural theory, the model includes a causal learning mechanism that explains how learning occurs and operates across micro, meso, and macro levels. It also explains how learned information is aggregated, or brought together, and disseminated within and across these levels.
- Responsible AI (RAI): Most recently, Haidar [24] proposed a novel integrative theoretical framework for RAI, addressing four (4) key dimensions: technical, sustainable development, responsible innovation management, and legislation. The responsible innovation management and the legal dimensions form the foundational layers of the framework, embedding elements like anticipation and reflexivity into corporate culture, and examining AI-specific laws (from the European Union (EU) and the United States (US)) for a comparative legal AI perspective. This study's findings are helpful for businesses integrating AI responsibly, developers creating compliant AI, and policymakers fostering awareness and developing guidelines for RAI.

Within this expansive domain of BD and edge computing, AI transforms raw data into actionable insights and automates complex tasks. However, the intricate relationship between AI and BD gives rise to various technical challenges. These include managing the number of training epochs and time, addressing over-/under-fitting, and preventing data leakage, all of which can influence the efficacy of AI models. Furthermore, the inherent characteristics of AI models and the energy-constrained nature of edge devices present additional technical challenges when optimizing AI models for smart placement, cost reduction, energy efficiency, and other objectives.

3. Design of the AI theoretical framework

This document further presents the Core Optimized AI-Pipelines (COAP) component, a foundational element of the RAIDO project's advanced architectural framework; outlined in previous deliverable D2.3. Designed to deliver a cutting-edge ML ecosystem, COAP emphasises a balanced integration of efficiency, ethical governance, and inherent trustworthiness. It is built for seamless scalability across both localized edge deployments and centralised cloud environments. Its strategic design influences key RAIDO WPs (WP3: “Data Enrichment & Trustworthy AI Architectures”; WP4: “Human-Centred AI”; WP5: “Energy-Efficient E2C Deployment & Green AI Orchestration”), establishing a cohesive continuum from data generation to optimized AI deployment.

At its core, COAP is designed to balance the strengths of edge and cloud computing: it leverages the agile, low-latency capabilities of edge computing for localized data processing, validation, and preliminary model training, while simultaneously utilising centralized intelligence for comprehensive model aggregation, evaluation, and global deployment. This dual approach not only maximises resource utilisation and minimises latency but also enhances adaptability across diverse operational environments. A continuous monitoring framework is embedded to meticulously track performance and proactively detect, identify, and mitigate algorithmic biases, reinforcing fairness throughout the pipeline. Moreover, the component embeds ethical AI principles, incorporating robust privacy-preserving mechanisms and secure data storage, into every developmental phase, to protect user data and ensure compliance with regulatory standards. Additionally, COAP integrates explainability tools such as SHAP, LIME, and/or Grad-CAM to clarify complex predictions, build stakeholder trust, and provide transparent insights into AI decision-making. Through ongoing monitoring and iterative model updates, COAP ensures that deployed models remain resilient and effective in dynamic real-world scenarios. This directly aligns with the project's overarching objective: to deliver energy-efficient, ethical, and reliable AI/ML systems that effortlessly operate across the edge-to-cloud continuum.

3.1 COAP component architecture

The COAP component, designated as C01 and interchangeably referred to as "AI-Pipelines" in the RAIDO reference architecture, functions as the algorithmic engine of the RAIDO project. Its essence lies in a modular, meticulously engineered ML pipeline that orchestrates the entire lifecycle of AI models and datasets: from initial data ingestion, preparation, and preprocessing, through model training and evaluation, to final deployment across distributed edge nodes and centralized processing hubs. Beyond standard ML workflows, COAP is designed to support advanced learning paradigms including:

- Data distillation for model efficiency,

- Lifelong learning for continuous adaptation,
- Transfer learning for leveraging pre-existing knowledge, and
- Few-shot learning for rapid model adaptation with limited and minimal data.

Its architectural blueprint is designed to incorporate ethical AI principles and XAI capabilities, ensuring transparency, fairness, and energy-efficient AI development. Implemented primarily in Python, the component's design promotes flexibility and extensibility, allowing seamless adaptation to evolving RAIDO research and deployment needs.

From an operational perspective, COAP relies on:

- Ingestion prerequisites: Access to raw data streams originating from edge nodes, along with curated labeled and/or unlabeled datasets essential for model training. Configurable parameters for AI model tuning and optimization are also crucial inputs.
- Actionable outputs: The component yields optimized, globally aggregated, evaluated, and deployable ML models primed for inference at either edge or centralised locations. Critically, it also generates comprehensive insights, including model bias assessments and diagnostics, detailed performance metrics, and interpretable explainability reports. All processed data and trained models are securely stored as a post-condition.
- Interoperability: Its Application Programming Interface (API) is currently under active development, designed to enable seamless interaction with other RAIDO components.

The high-level architecture of the COAP component is divided into an Edge and a Centralized Layer, a design meticulously crafted to balance distributed processing efficiency with centralized oversight and intelligence. This layered structure, illustrated in Figure 4, is key to achieving and delivering optimal scalability, performance, fairness, and transparency at scale.

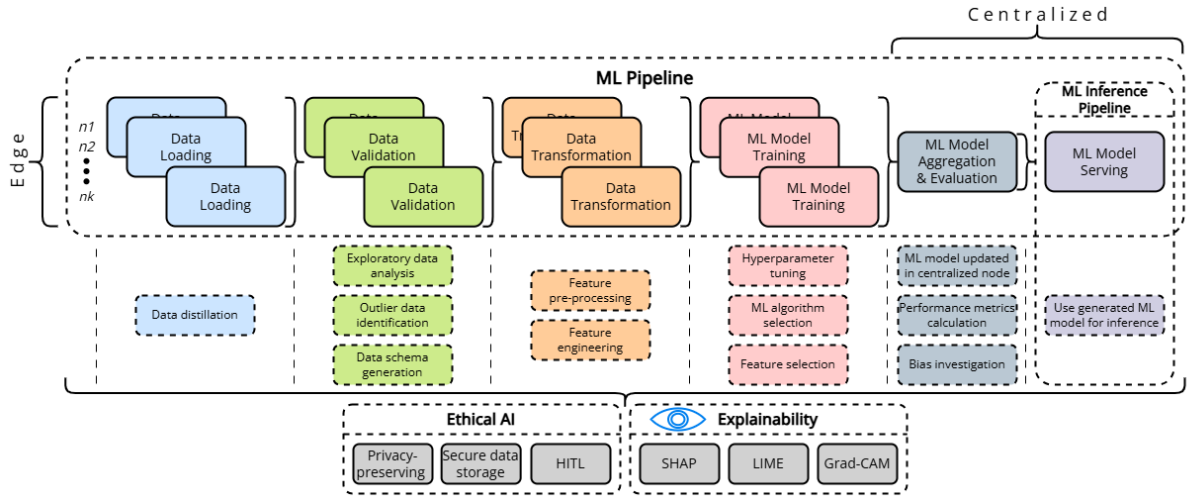


Figure 4: COAP architecture.

The flow of data and operations within COAP follows a clearly defined trajectory:

- Distributed data capture: Raw data is initially collected at the edge (see nodes n_1 to n_k) where it undergoes preliminary processing close to its source.
- Pipeline initiation: The processed data then enters the core ML pipeline, beginning with Data Loading, followed by critical Data Validation and necessary Data Transformation stages.
- Local model development: Each edge node independently trains ML models using the localized, processed data and selected algorithms.
- Global model synthesis: These locally trained models are then sent to the centralized layer, where they are aggregated and comprehensively evaluated to form a unified, global model.
- Explainability of the AI predictions: An end-to-end explainability process is performed spanning a broad set of actions depending on the data modality that is processed for each use-case this pipeline is used. Aspects such as model performance, model variance or bias, fairness and reliability of the AI model. The libraries that are being exploited for that end are SHAP/Lime for tabular and time-series data as well as Grad-CAM for image
- Performance and ethical audit: Within the centralized environment (processing), detailed performance metrics are computed, and crucial targeted assessments are conducted to detect, identify and address any inherent model biases.
- Deployment and refinement: The finalized, updated models are then managed centrally and can be redeployed back to the edge nodes for localized inference or served for global inference scenarios and tasks.
- Continuous operation: Once deployed, the ML models actively process new incoming data, supporting robust and scalable AI operations across the entire edge-to-cloud ecosystem.

In more detail, the internal structure of the COAP component, as detailed in Figure 4, reveals a finely segmented pipeline, orchestrating data and model progression through distinct, interconnected stages.

- Edge Processing Layer (Localised Intelligence)
    - Located on the left side of the architecture, this layer serves as the first line of data handling. It enables distributed, low-latency processing directly at the source, minimising bandwidth usage and maximising responsiveness. Key modules include:
        - Data Loading (Blue Modules): This initial phase orchestrates distributed data collection across the network of edge nodes (n_1 to n_k). It encompasses the fundamental preparation of raw data for subsequent ML processes, and it may include Data Distillation to reduce redundancy and compress or refine datasets for efficient downstream use.
        - Data Validation (Green Modules): A critical quality assurance stage that involves exploratory data analysis to characterize data properties, outlier data detection to flag and address anomalies, and data schema generation to standardize data formats for cross-node consistency.
        - Data Transformation (Orange Modules): This stage refines and prepares the data, incorporating feature pre-processing (e.g., cleaning, normalization) and feature engineering to craft new features that enhance ML model performance.
        - ML Model Training (Pink Modules): Here, local ML models are built. This involves hyperparameter tuning to optimize model configurations and performance, intelligent ML algorithm selection tailored to task-specific requirements, and feature selection to reduce dimensionality and improve generalization for more efficient learning.
- Centralized Processing Layer (Global Intelligence & Oversight):
    - Positioned on the right side of the architecture, this layer serves as the central hub for aggregation, evaluation, and deployment. In essence, it aggregates and synthesises knowledge from edge models, providing robust evaluation, bias correction, and continuous performance monitoring, before deploying the refined ML models into production inference pipelines. This layer comprises:
        - Model Aggregation & Evaluation (Grey Modules): This crucial stage unifies locally trained models (from edge) into a single, high-performing global model. It computes precise performance metrics (e.g., accuracy, efficiency, robustness) and conducts diligent bias

investigation and audits to ensure fairness and address any detected model biases. It also updates models centrally in preparation for redeployment.
        - ML Inference Pipeline (Purple Module): It operationalises the trained models for real-world usage. It handles model serving to facilitate inference tasks at both edge and centralized locations.
- Cross-Cutting Enablers (Ethicality & Transparency):
    - Underpinning the entire COAP pipeline, as illustrated at the bottom of the architecture, the following foundational capabilities ensure COAP complies with the highest standards of ethical AI and model interpretability:
        - Ethical AI: it embeds safeguards and accountability mechanisms, including privacy-preserving techniques, secure data storage solutions, and Human-in-the-Loop (HITL) processes for critical oversight and intervention at various stages.
        - Explainability: It leverages advanced tools like SHAP, LIME, and Grad-CAM to deliver clear, interpretable insights and explanations into model predictions, fostering user/stakeholder trust and understanding, as well as regulatory compliance.

This architecture design exemplifies COAP's core value proposition: a scalable, ethical, transparent, and explainable AI pipeline that seamlessly bridges edge and centralized capabilities, delivering optimal performance and trustworthiness across diverse deployment contexts.

Here, we need to mention that the COAP component operates not as a standalone module, but as a deeply integrated part of the broader RAIDO ecosystem. Its full functionality depends on seamless interaction with several key components, previously identified and detailed in D2.3, which collectively support RAIDO's overarching vision of scalable, ethical, and energy-efficient AI. In this deliverable, we focus specifically on COAP's core integration points with these components, highlighting how AI Pipelines function in synergy with the larger system.

- Network & Resource Managers: COAP interfaces dynamically with RAIDO's Network and Resource Management subsystems, enabling real-time orchestration of computational resources such as CPU, GPU, memory, and network bandwidth across both edge and centralised nodes. This collaboration is crucial for ensuring consistent quality of service for model training and inference, enabling elastic scaling based on workload demands, and supporting energy-aware scheduling to align with RAIDO's sustainability goals. This interface empowers COAP to balance performance with efficiency, especially during high-load AI training and model distribution tasks.

- Green-AI integration: Through its integration with the Green-AI component, COAP incorporates energy-efficient strategies directly into its ML workflow. These include data distillation to reduce training volume, model distillation for compact and efficient inference models, and hyperparameter optimization techniques that minimise computational overhead. This direct alignment with Green-AI ensures that every stage of the pipeline, from data preprocessing to model deployment, is consciously optimised to lower environmental impact, without compromising model performance.
- Data lake collaboration: COAP maintains a continuous, bidirectional interface with the RAIDO data lake, which functions as the central, high-availability repository for all data and model assets. This integration enables efficient ingestion and retrieval of both raw and processed datasets, version-controlled storage of trained models, and a unified source of truth for monitoring data lineage and ensuring auditability. The data lake not only facilitates a streamlined data flow within COAP but also ensures alignment with data governance and traceability requirements across the project.

4. AI theoretical framework development & deployment

As illustrated in Figure 5, our experimental framework effectively captures key metrics including energy consumption, performance efficiency, and HW utilisation.

4.1 Technology stack

This framework is hardware and platform vendor-agnostic, designed for broad applicability across various scenarios and use cases. We prioritize its high scalability and extensibility to ensure practicality for real-world E2C deployments. By tracking system operations on a per-second basis, our framework accurately captures dynamic workloads and fluctuating resource availability, even during abrupt changes. Technical specifications of the target hardware and platform are recorded based on their objective capacity.

The first step includes the abstraction and containerisation of the BD AI pipeline within a Kubernetes cluster (see Figure 6). Proper configuration and setup of this cluster are necessary for monitoring the execution environment of the BD application under evaluation.

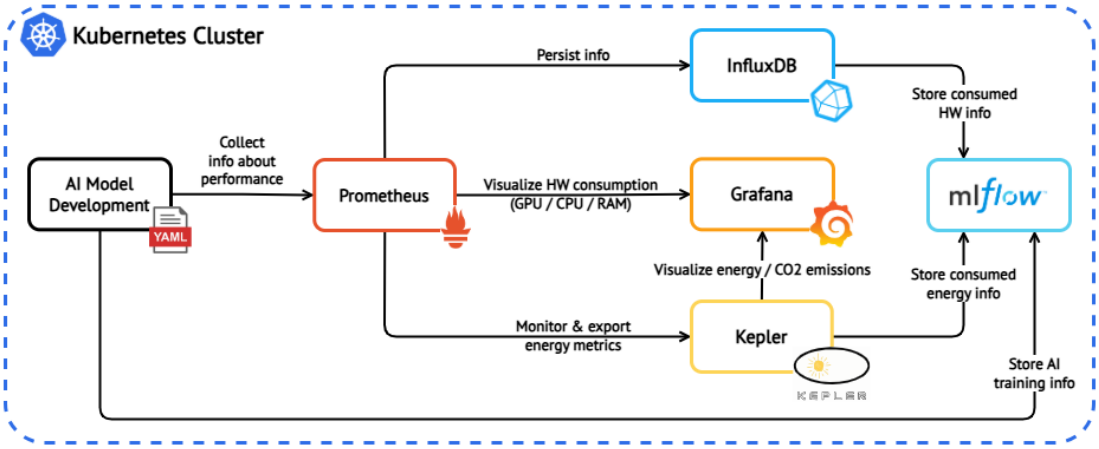


Figure 5: Technology stack for benchmarking the AI theoretical framework (Theodorou et al. [25]).

The Kubernetes cluster supports E2C applications, providing scalability, easy integration of edge devices, and dynamic resource allocation. We specifically study the case of BD AI pipelines, which often require dynamic resource adaptation due to potential random fluctuations during the training and inference phases. In addition, containerisation within Kubernetes ensures process isolation and minimises interference from other processes like the operating system or neighbouring applications. This leads to more accurate and reproducible measurements of energy and computing resource consumption. Moreover, Kubernetes' native monitoring tools, such as Prometheus and Grafana, enable real-time

tracking and visualisation of HW metrics (CPU, GPU, RAM) and energy usage for a given AI pipeline. This distributed approach facilitates monitoring of federated workloads across multiple nodes, offering more comprehensive and granular insights into system performance compared to traditional PID-based monitoring. Lastly, by leveraging Kubernetes' cloud-native infrastructure, the framework can be easily extended to larger, multiple, and more complex environments, supporting diverse AI applications and varying computational demands.

We have deployed and integrated a comprehensive suite of libraries and tools to form our monitoring infrastructure, ensuring precise tracking of energy consumption and HW utilisation throughout the BD AI pipeline. Each component has been carefully selected to address specific resource monitoring challenges. In the following, we briefly overview the tools employed, providing necessary background and explaining their role within the overall architecture.

To develop such a comprehensive AI framework focused on measuring hardware consumption, we leverage a suite of cutting-edge technology tools, as shown in Figure 5. These include Prometheus, Grafana, InfluxDB, and Kepler, each playing a pivotal role in enabling end-to-end energy and HW consumption tracking.

- Prometheus [26], integrated into the Kubernetes cluster, serves as the primary monitoring tool for collecting real-time HW utilisation data. It tracks vital system metrics like CPU, GPU, and RAM usage, offering a highly reliable and scalable method for capturing these metrics. Its integration with Kubernetes ensures continuous monitoring and accurate recording of resource fluctuations over time. Prometheus also gathers diverse statistics beyond application-specific metrics, providing high flexibility for custom, use case-specific monitoring requirements, such disk throughput, filesystem I/O, and out-of-memory (OOM) errors.
- Once captured by Prometheus, hardware utilization data is stored in InfluxDB [27], a high-performance time-series database. InfluxDB is chosen for its ability to persistently handle large volumes of time-stamped data over long periods, facilitating efficient querying and analysis of historical HW consumption trends. This long-term storage is crucial for monitoring and understanding the evolving energy consumption patterns of AI applications over extended periods, especially in BD scenarios.
- For visualisation, Grafana [28] is employed to create intuitive and interactive dashboards. We utilise two (2) purposefully developed Grafana dashboard templates [29]: for HW consumption visualisation, and [30] for energy and CO2 emissions visualisation, with Grafana connecting directly to Prometheus for data retrieval. This visualisation capability provides developers and researchers with real-time insights into the AI application's resource usage, enabling them to

monitor trends and make informed decisions regarding optimizations, resource allocation, and scheduling.

- Kepler [30] extends Prometheus's capabilities by leveraging system telemetry data to compute energy consumption and carbon emissions. Kepler seamlessly integrates with Prometheus, offering real-time insights into the environmental impact of the AI application, including power usage and CO2 emissions. This integration promotes sustainable computing practices by providing both technical and environmental metrics critical for optimizing energy consumption in containerised AI workloads.
- Finally, the AI models, their performance efficiency, and associated metadata are stored and managed using MLFlow [31]. MLFlow ensures reproducibility by tracking experiment runs, saving model versions, and maintaining all relevant information for future reference. This allows for consistent monitoring and comparison of model performance, energy consumption, and HW usage across different experiments, providing a complete lifecycle management system for AI development within the proposed framework.

By harnessing the collective power of Prometheus, Grafana, InfluxDB, Kepler, and MLFlow, we not only create an end-to-end AI pipeline but also establish robust mechanisms for measuring and optimizing HW consumption. This integrated toolset empowers us to monitor, analyse, and enhance the performance and efficiency of our AI infrastructure, ultimately leading to more transparent, scalable, and resource-efficient AI solutions.

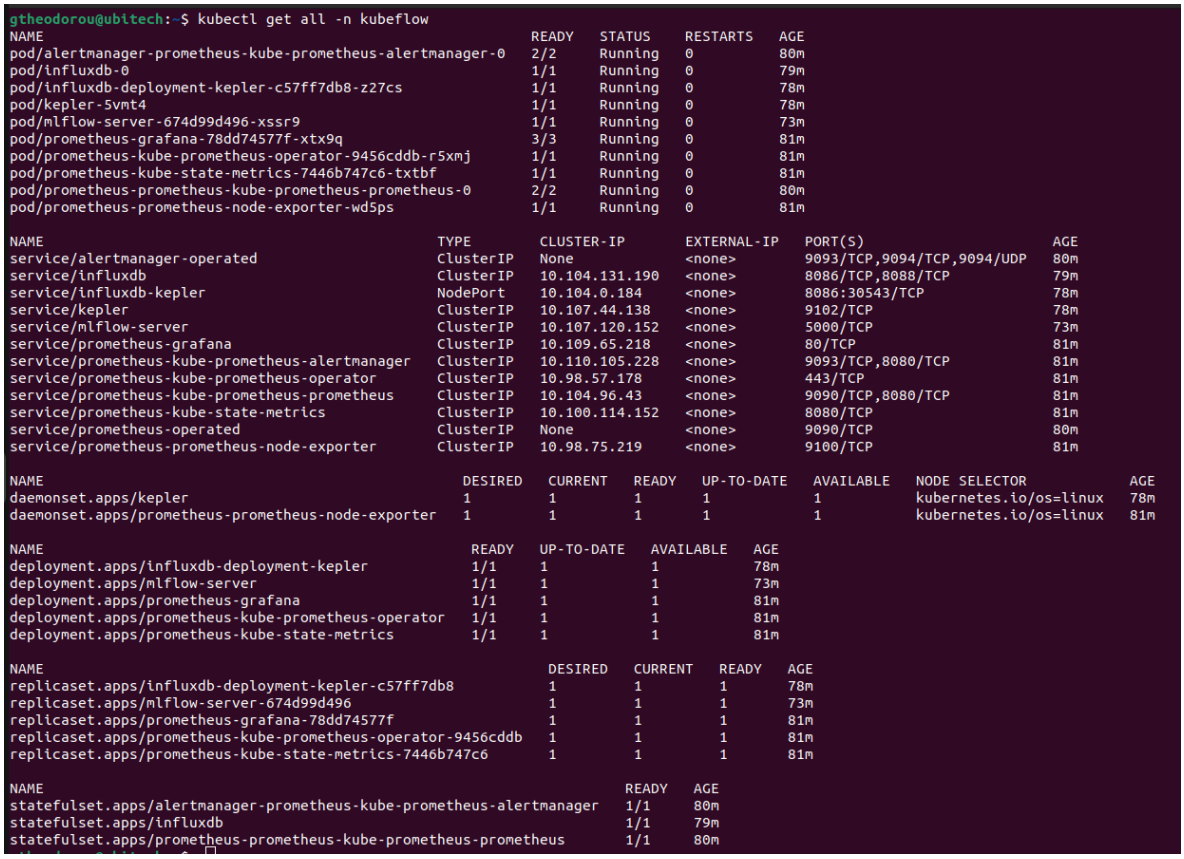
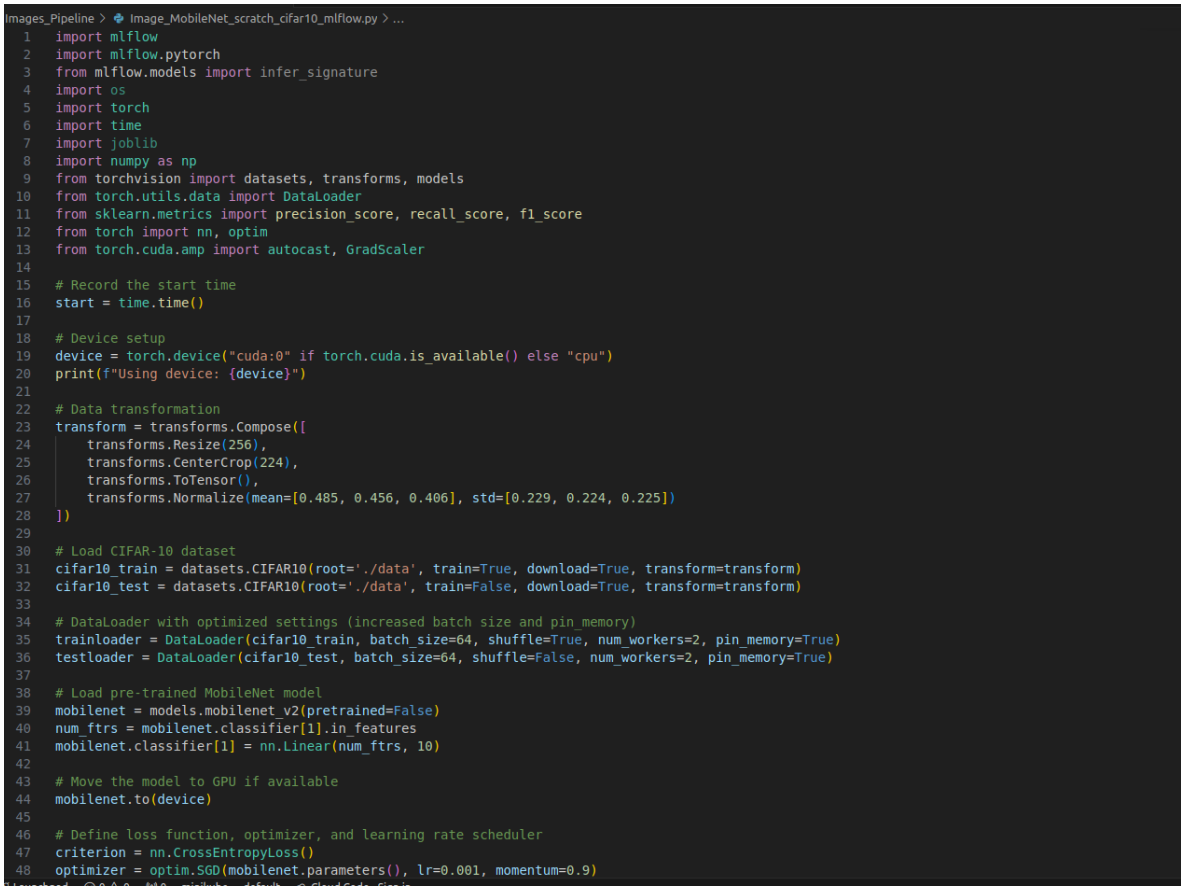


Figure 6: Kubernetes namespace orchestrating AI pipeline energy & HW consumption monitoring.

4.2 AI training script

The solution framework for monitoring the consumption of energy and HW is model agnostic (meaning that it can be applied irrespective of the AI model that the user would like to measure). This crucial attribute stems from encapsulating our AI model inside a Kubernetes pod and measuring the pod's footprint. In addition, the developed framework can be utilised on either the edge or in the cloud, depending on user requirements and available training HW.

The framework does not make any hard requirements on the AI training pipeline script itself. This allows users to leverage the rich array of AI/ML frameworks available for developing AI models, such as PyTorch, TensorFlow, Scikit-Learn, etc. This flexibility for the AI engineer is one of the most critical components of the developed framework. Figure 7 shows an example of an AI training script that is being monitored with the framework.



```
Images_Pipeline > Image_MobileNet_scratch_cifar10_mlflow.py > ...
import mlflow
import mlflow.pytorch
from mlflow.models import infer_signature
import os
import torch
import time
import joblib
import numpy as np
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
from sklearn.metrics import precision_score, recall_score, f1_score
from torch import nn, optim
from torch.cuda.amp import autocast, GradScaler

# Record the start time
start = time.time()

# Device setup
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")

# Data transformation
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# Load CIFAR-10 dataset
cifar10_train = datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
cifar10_test = datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)

# DataLoader with optimized settings (increased batch size and pin_memory)
trainloader = DataLoader(cifar10_train, batch_size=64, shuffle=True, num_workers=2, pin_memory=True)
testloader = DataLoader(cifar10_test, batch_size=64, shuffle=False, num_workers=2, pin_memory=True)

# Load pre-trained MobileNet model
mobilenet = models.mobilenet_v2(pretrained=False)
num_ftrs = mobilenet.classifier[1].in_features
mobilenet.classifier[1] = nn.Linear(num_ftrs, 10)

# Move the model to GPU if available
mobilenet.to(device)

# Define loss function, optimizer, and learning rate scheduler
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(mobilenet.parameters(), lr=0.001, momentum=0.9)
```

Figure 7: Example AI training script to be monitored with use of framework.

4.3 Kubernetes deployment configuration

After setting up the entire Kubernetes namespace with the frameworks as explained above, the next step is to monitor an AI model's training and track its energy and HW consumption. To streamline this, we generate and upload a YAML file (see Figure 8), which encapsulates the instructions required for creating the pipeline. This YAML file serves as a blueprint, outlining the sequence of tasks, dependencies, and configurations necessary for seamlessly orchestrating and then monitoring the entire pipeline's progress.

The YAML file instructs the namespace to create a batch job that sets up a pod containing the Python script where the AI training has been defined. By executing this script, we abstract the complexity of pipeline creation, ensuring consistency, reproducibility, and ease of maintenance across different environments and deployments.

The YAML file contains detailed specifications for each component of the pipeline, including GPU resource definition, MLFlow port connection, memory resource limitations, and restarting policy definition. In addition, the setup is configured to load the Docker image containing the AI training specification from Docker Hub if it's not found locally. This streamlined approach to pipeline development and automation

enhances efficiency, reduces manual intervention, and accelerates the energy and HW tracking of AI pipelines within Kubernetes-based environments.

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: mlflow-logging-app-efficientnet-cifar10
  labels:
    app: mlflow-logging-app-efficientnet-cifar10
spec:
  template:
    metadata:
      labels:
        app: mlflow-logging-app-efficientnet-cifar10
    spec:
      containers:
      - name: mlflow-logging-app-efficientnet-cifar10
        image: gtheodorou/efficientnet_mlflow:latest
        resources:
          limits:
            nvidia.com/gpu: 1 # Request 1 GPU
        ports:
        - containerPort: 8080 # Adjust if your app uses a different port
        env:
        - name: MLFLOW_TRACKING_URI
          value: "http://mlflow-server:5000" # Adjust based on your MLflow server's service name and port
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
          sizeLimit: 1Gi
      restartPolicy: Never
```

Figure 8: YAML file with instructions for deploying AI Pipeline to monitor.

4.4 Energy consumption visualisation

As aforementioned, our framework utilises Kepler to track the energy consumed during the AI model development. Once the energy has been logged from Kepler to InfluxDB, it can be visualised for better understanding of the AI's energy efficiency.

There are two (2) primary ways to achieve this visualisation: (i) the energy consumed can be readily viewed directly from the Influx DB UI, as shown in Figure 9. The end-user can select the specific AI training run of interest and define the timeframe (start-end) to view the corresponding results; (ii) alternatively, the energy consumption can be visualized in Grafana, as depicted in Figure 10. Grafana can display the energy consumed broken down per device. "PKG" (from CPU), "DRAM" (energy from RAM), "GPU" (from GPU), "OTHER" (any other device consuming energy). Another important visualisation offered by Grafana is the breakdown of the energy consumed into CO_2 emissions based on fuel sources (i.e. coal, petroleum, and natural gas). This allows users to better understand the environmental footprint of their AI training.

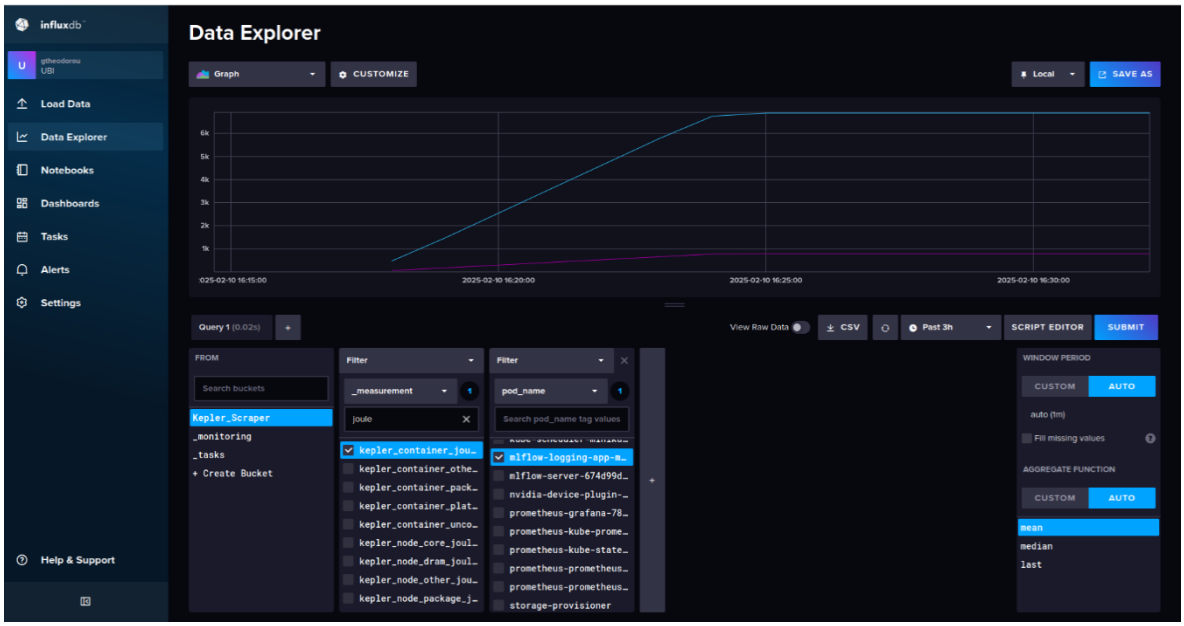


Figure 9: InfluxDB UI, showing a linear graph with total energy consumption.

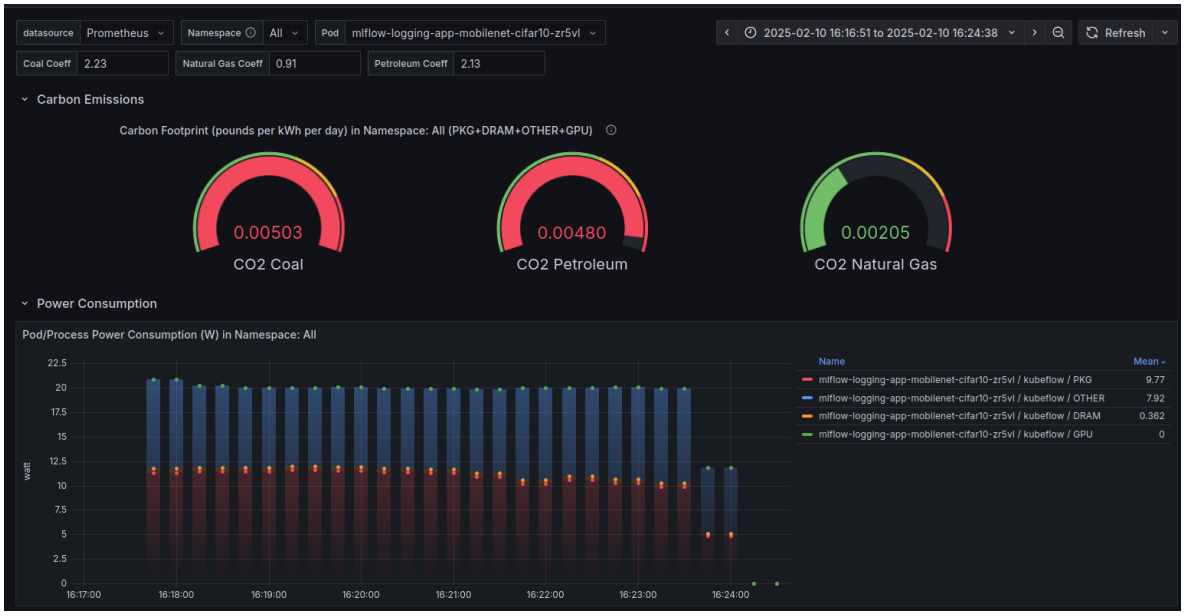


Figure 10: Grafana showing energy consumed by AI training.

4.5 Hardware consumption visualisation

During the execution of the AI pipeline, in addition to monitoring HW consumption using tools like Prometheus, we leverage Grafana to visualize real-time metrics. This dynamic visualisation provides an intuitive dashboard that offers insights into ongoing activities and resource utilisation within our system. An illustrative example of this real-time monitoring can be observed in Figure 11, where we gain visibility into CPU and RAM usage, particularly during the execution of the most resource-intensive step of the pipeline: *ML Model Training*.

In Figure 11, the Grafana dashboard graphically represents CPU and RAM usage, highlighting that these resources are predominantly maxed out during the ML Model Training phase. This insight is critical as it allows us to identify resource bottlenecks, optimize resource allocation, and ensure the efficient execution of the AI pipeline. The real-time visualisation allows us to make informed decisions, proactively address performance issues, and maintain system stability throughout the pipeline execution. By integrating Grafana into our monitoring stack, we gain a comprehensive understanding of HW consumption trends, performance metrics, and system behaviour during the AI pipeline's execution. This enhanced visibility enables us to monitor resource utilisation in real-time, identify potential performance bottlenecks, and optimize resource allocation strategies to maximise pipeline efficiency and reliability.

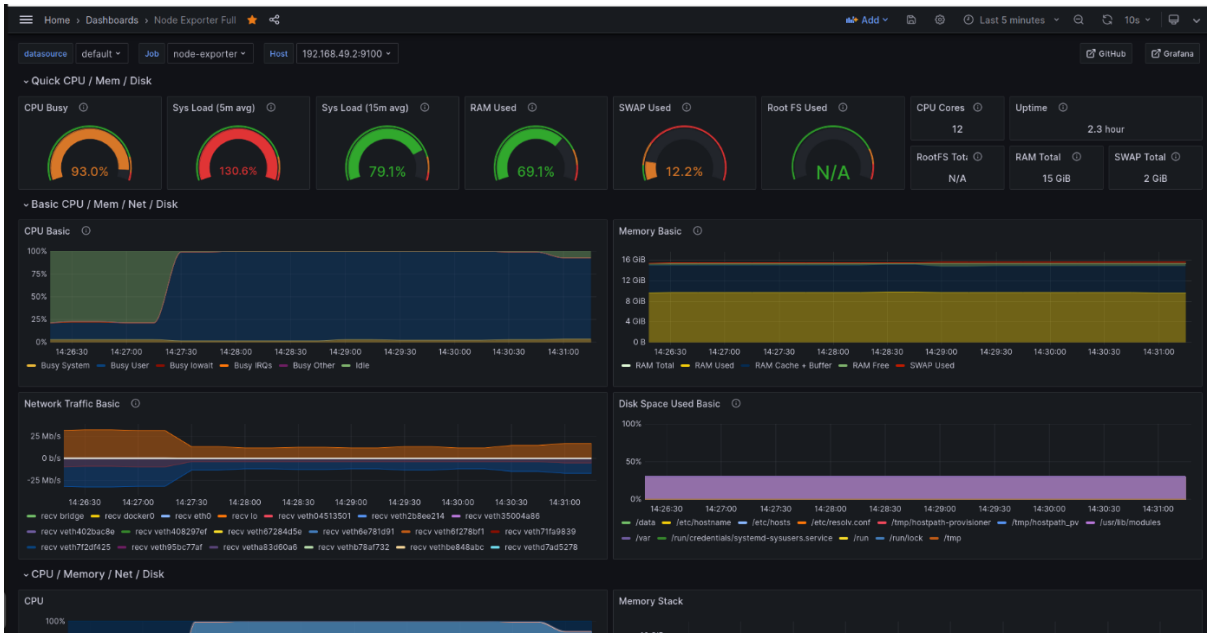


Figure 9: Grafana dashboard displaying HW consumption of AI Pipeline.

As outlined earlier, our monitoring strategy involves logging HW consumption metrics collected by the Node Exporter component of Prometheus to InfluxDB. This logging mechanism captures detailed information about system resource usage, providing a comprehensive dataset for in-depth analysis, performance optimization, and future reference. Figure 12 illustrates the log structure, exemplifying a specific pod's total CPU usage in seconds.

The logs retrieved from InfluxDB (as shown in Figure 12) offer insights into the CPU utilisation metrics for a particular pod within our Kubernetes cluster. By tracking CPU usage in seconds, we gain a granular understanding of resource allocation patterns, workload demands, and performance trends at a micro level. This data becomes invaluable for identifying resource-intensive tasks, detecting anomalies, and optimizing resource allocation strategies to enhance system efficiency and reliability.

Furthermore, these logged HW consumption metrics are a crucial resource for conducting retrospective analyses, diagnosing performance issues, and benchmarking system performance over time. The availability of historical data enables us to establish baseline performance metrics, track deviations, and make data-driven decisions to improve the overall stability and scalability of our infrastructure.

By seamlessly integrating Prometheus, Node Exporter, and InfluxDB into our monitoring stack, we establish a robust logging and monitoring framework. This framework empowers us to effectively capture, analyse, and leverage HW consumption metrics, facilitating not only real-time monitoring but also the derivation of actionable insights, optimised resource utilisation, and optimal performance for our AI pipeline and Kubernetes cluster.

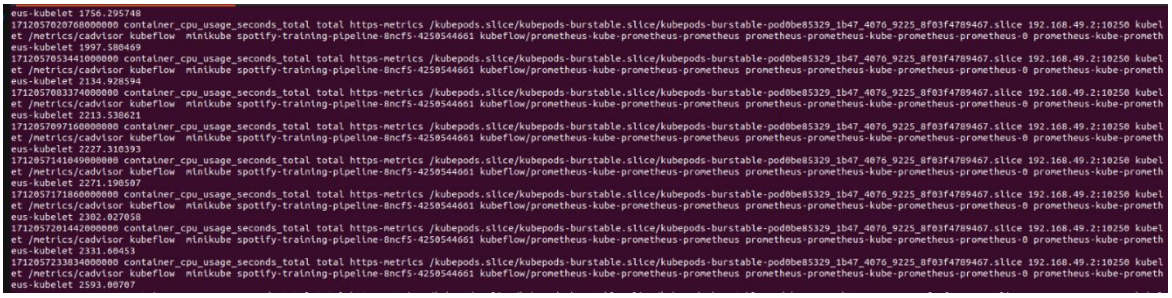


Figure 10: InfluxDB logged information from Prometheus Node Exporter.

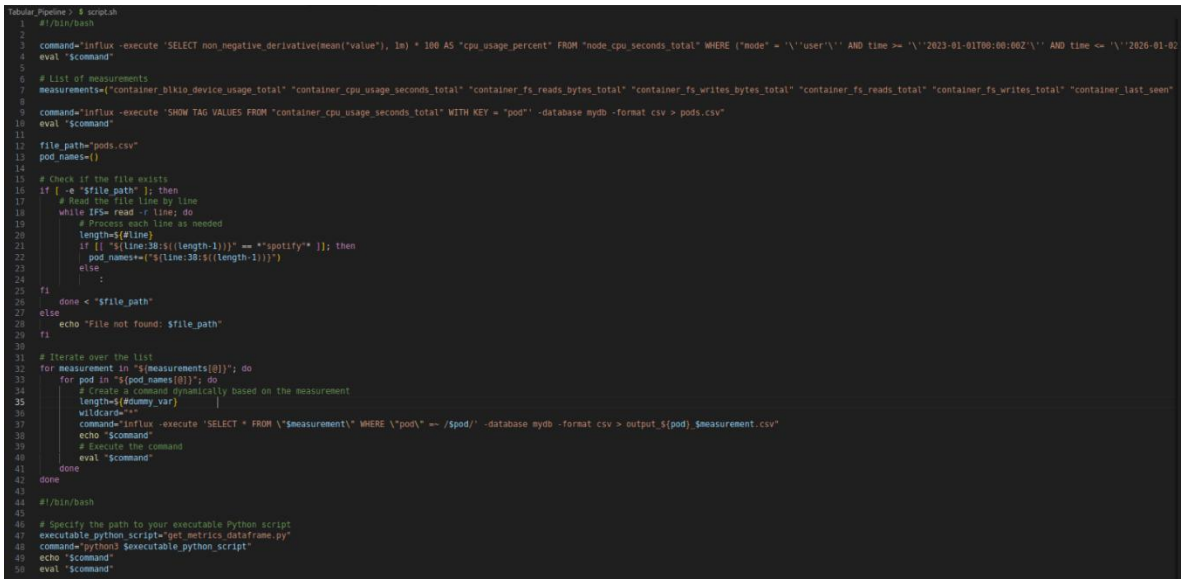


Figure 11: Bash script for retrieving logs from InfluxDB.

Upon completion of the AI pipeline's execution, the stored logs are retrieved from InfluxDB for further analysis. To streamline this process, we have developed a custom bash script (see Figure 13) that serves as a robust automation tool for loading, processing, and organizing the logs into a structured CSV format for future reference and analysis. This script plays a pivotal role in extracting pertinent information from the

logged HW consumption metrics, filtering out irrelevant data, and transforming it into a usable format.

The bash script executes a series of steps:

- Retrieval: Logs are retrieved from InfluxDB based on specified criteria and time ranges.
- Processing: The retrieved logs undergo data transformation and filtering to extract key HW consumption metrics, such as CPU usage, memory utilisation, disk I/O, and network traffic. This step is crucial for focusing on relevant information essential for performance analysis, resource optimization, and system monitoring.
- Compilation: The bash script then compiles the extracted data into a structured CSV file format, ensuring readability, organisation, and compatibility for future analyses and reference. This CSV file serves as a comprehensive repository of HW consumption metrics, providing a historical record of system performance, trends, and anomalies over time.

During processing, we carefully select and prioritise the information to be included in the CSV file, aligning it with the specific requirements and objectives of the analysis. This selective approach ensures that only relevant and meaningful data points are captured, enhancing the efficiency and effectiveness of subsequent analyses and decision-making processes.

Overall, the development of this bash script for log retrieval, processing, and CSV file generation streamlines the post-execution workflow of the AI pipeline, enabling us to extract actionable insights, perform in-depth analyses, and maintain a structured record of HW consumption metrics for ongoing monitoring and optimization efforts.

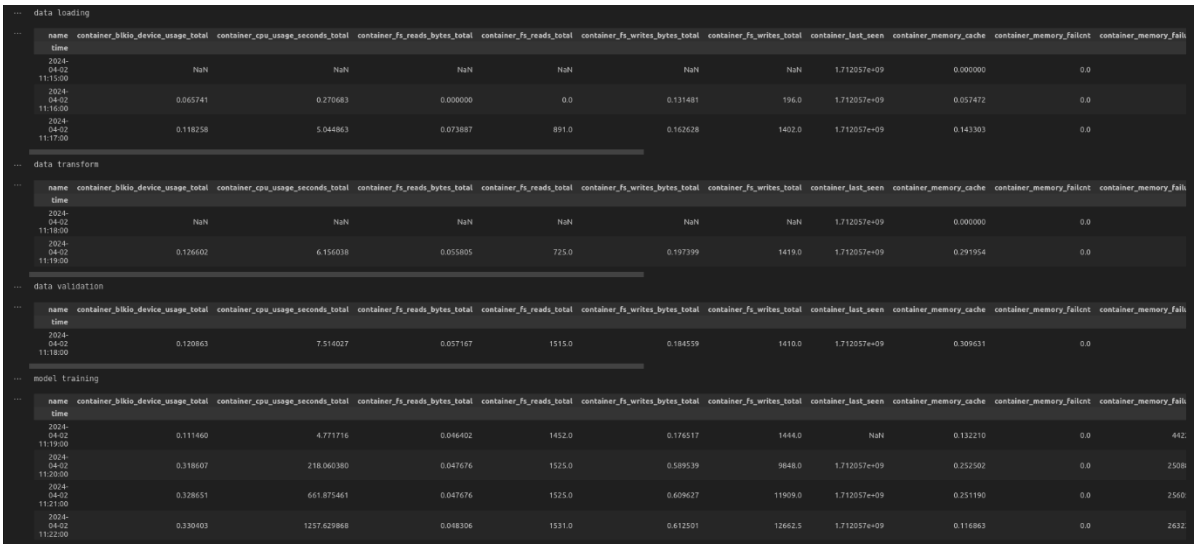


Figure 12: Various HW consumption metrics as logged during AI pipeline execution.

Figure 14 provides a detailed and granular view of the HW metrics captured during each step of the AI pipeline execution. This visualisation offers a comprehensive overview of key HW consumption parameters that are closely monitored throughout the pipeline's lifecycle. By capturing data with 1-minute granularity for each step, we gain immediate insights into real-time performance trends, resource utilisation patterns, and system behaviour during the pipeline's execution.

For each step of the AI pipeline depicted in Figure 14, the graph showcases essential HW metrics such as CPU usage, memory utilisation, disk I/O, and network traffic. Tracking these metrics at a fine-grained level allows us to monitor resource consumption trends, detect anomalies, and prioritise resource allocation strategies based on actual execution data.

The detailed information presented in Figure 14 facilitates several critical insights and observations from the AI pipeline's execution. For instance, we can identify peak resource usage periods, assess the impact of specific pipeline tasks on HW consumption, and optimize resource allocation based on workload demands. The data's granularity also enables us to conduct root cause analysis, diagnose performance bottlenecks, and make data-driven decisions to enhance system performance and efficiency.

In summary, the figure serves as an invaluable tool for visualising and analysing HW metrics at a granular level, providing actionable insights and facilitating informed decision-making throughout the AI pipeline's execution and optimization process.

5. Benchmarking AI theoretical framework

5.1 Benchmarking AI models for image data modality

Our AI theoretical framework for energy and hardware monitoring can assess an AI model's footprint regardless of its data modality. To demonstrate this, we have applied the framework to benchmark RAIDO's precision agriculture and robotics use cases, both of which involve image data. We trained, monitored, and benchmarked state-of-the-art models recognised for their superior performance in their respective tasks.

In the first use case, the objective is to enable autonomous and AI-powered monitoring and operations. This involves leveraging Computer Vision for tasks like multispectral imaging to detect plant diseases early and predict crop growth. Given that many such systems might be deployed on edge devices (e.g., drones), it is crucial to distinguish between the AI model's training and deployment environments. Deploying computationally heavy models that achieve extremely high accuracy at the expense of energy efficiency is often not feasible due to hardware limitations. Therefore, developing an AI model optimised for edge deployment, prioritizing energy efficiency, is a primary criterion when selecting a suitable model for this use case.

For the Robotics Domain, the objective is to develop AI models for plant fiber characterization within the context of Industry 5.0 and bio-based composites. This typically involves Computer Vision for tasks such as identifying and analysing fiber structures. Similar to the precision agriculture use case, it's equally important to ensure that the trained AI model is deployable on the edge, potentially on robotic systems with constrained computational resources. In both scenarios, deploying a computationally heavy model, despite its higher accuracy and lower error rate, is not feasible due to hardware constraints. Conversely, a model that prioritizes energy efficiency but fails to achieve sufficient accuracy for the task would be ineffective. Therefore, AI model development for these use cases presents a multi-objective challenge: balancing energy efficiency with accurate detection to ensure reliable and practical deployment.

To benchmark the models and validate our AI energy and hardware monitoring framework, we used a proxy dataset consisting of 60,000 RGB images (~170 MB total). Although this dataset is not directly from the agriculture or robotics domains, it includes binary classification tasks, specifically, fire detection and helmet compliance, that closely resemble the structure and computational demands of our target applications. Since domain-specific datasets for smart farming and robotics are still being developed, this toy dataset allows us to test our system effectively. The classification task with two (2) classes parallels the plant disease detection problem in smart farming. Similarly, once relevant data becomes available for the robotics use case, we plan to extend our framework to classify different types of fiber structures. This approach enables us to

demonstrate and validate the framework across representative tasks before applying it to real-world data. The benchmarking results, covering both training and edge deployment metrics, are presented in Figures 15 and 16.

AI Model	Modality	Records	Size (MB)	Num. Weights	Precision	Accuracy	F1 Score	Training Time (sec)
ConvNext (Tiny)	Images	60K	163MB	28.6M	32.5%	30%	26%	1197
DenseNet121	Images	60K	163MB	8M	64%	63%	63%	1128
EfficientNetb0	Images	60K	163MB	5.3M	72%	71.5%	71.5%	596
MobileNet(v2)	Images	60K	163MB	3.5M	63%	63%	62.6%	371
ResNet152	Images	60K	163MB	60.2M	54%	52%	51%	2467
VGGNet16	Images	60K	163MB	138M	66%	65%	64%	2199
Vision Transformer (Base)	Images	60K	163MB	86M	59%	56%	55.8%	2577
YOLOv8	Images	60K	163MB	27.3M	38%	37%	33%	2641

Figure 13: Comparison of imagery data for AI models performance & training time.

AI Model	Energy Consumed (Joules)	Avg CPU Usage %	Avg Memory Usage (GB)	Avg Threads
ConvNext (Tiny)	79248	80.25%	0.76	12
DenseNet121	74445	91.78%	1.45	16
EfficientNetb0	39331	106.3%	0.78	11
MobileNet(v2)	32268	149%	1.17	13
ResNet152	159031	79%	0.91	14
VGGNet16	135813	101.28%	0.89	15
Vision Transformer (Base)	169813	103%	1.02	16
YOLOv8	167321	144%	6.89	90

Figure 14: Comparison of imagery data for AI models HW & energy consumption.

As shown in Figure 15, EfficientNet [33] is the most efficient network, achieving the highest precision, accuracy, and F1 score across the four (4) training epochs. It outperforms the second-best model, VGGNet [34], by nearly 10% in accuracy, while completing the task in a significantly shorter time frame and using 73% less energy. This remarkable efficiency in both time and energy consumption highlights EfficientNet's superior architecture in balancing performance and resource utilisation.

Interestingly, while MobileNet [35] completes training the fastest, EfficientNet offers a better combination of accuracy and energy efficiency. The comparison reveals an interesting trend where AI models with either a relatively small number of parameters, like EfficientNet and MobileNet, or many parameters, such as VGGNet, Vision Transformer [36], and ResNet152 [37], outperform those with a medium number of weights, such as ConvNext [38] and YOLOv8 [39]. This observation suggests that small networks are well-suited for scenarios where quick execution and energy-aware placement are crucial. In contrast, medium-sized networks may struggle to balance speed and performance effectively, justifying the preference for smaller architectures in time-sensitive or energy-constrained application cases.

As depicted in Figure 16, MobileNet demonstrated the lowest energy consumption, requiring approximately 32K Joules, followed closely by EfficientNet. In contrast, the most energy-intensive networks were Vision Transformer and YOLOv8, with YOLOv8 being particularly noteworthy for its subpar performance relative to its energy usage, making it the least optimal model among those evaluated.

Interestingly, the number of parameters does not appear to be the primary driver of energy consumption; instead, the time required for execution plays a more significant

role. For instance, despite ConvNext having more than three times the number of parameters as DenseNet [40], the two models completed their training at nearly the same time and exhibited very similar energy consumption levels. This suggests that model architecture and execution efficiency have a more pronounced impact on energy usage than the sheer number of parameters.

Moreover, an intriguing observation is that YOLOv8, unlike the other models, exhibited significantly higher RAM and CPU utilisation. This points to a potential inefficiency in resource allocation, particularly when considering its lower performance in comparison to the other networks. These findings underscore the importance of not only evaluating accuracy but also considering resource efficiency when selecting models for deployment in energy-constrained environments.

It was observed that, on average, 90% of the total energy consumed by the networks could be attributed to GPU utilisation, with the smaller networks having a ratio of ~82% and bigger ones ~90%, underscoring the significant energy demands of GPUs compared to other hardware components such as CPU and RAM. This finding highlights the energy-intensive nature of GPU operations in AI training and suggests a pressing need for further research into optimizing GPU energy efficiency. Addressing this imbalance is critical for reducing the overall energy footprint of AI models, especially as their deployment becomes more widespread across diverse environments, from cloud data centres to edge devices.

5.2 Benchmarking AI models for tabular data modality

For RAIDO's energy grid and healthcare use cases, the data modality is tabular, and the developed framework is designed to monitor both the energy consumption and hardware performance of the AI system.

In the energy grid domain, the objective is power and energy grid management for AI-enabled optimal planning. This requires the analysis of complex time-series and categorical data from various sensors and systems to enable proactive management strategies, optimize grid operations, and integrate renewable energy sources.

In healthcare, the objective is to develop digital health solutions for personalized preventive pharmacogenetics. This involves analysing large datasets of genetic information and patient records to predict individual responses to medications and optimize treatment plans.

Unlike the image-based use cases where energy efficiency was a primary constraint due to edge deployment, these use cases here prioritise achieving the highest possible accuracy to ensure reliable predictions and critical decision-making. Since strict edge deployment constraints generally do not apply in these cases (often relying on cloud or

powerful on-premise infrastructure), the focus shifts from optimizing energy consumption to maximising model performance and prediction accuracy.

The models for these tabular tasks were benchmarked using a specific dataset consisting of 4.4 million records, amounting to a total size of ~767 MB. This dataset is representative of the scale and complexity of data encountered in these critical applications. Each model was trained using hyperparameter tuning via a random Grid Search approach, ensuring optimal model configurations and enabling a fair and rigorous comparison of the algorithms. The models benchmarked for these tasks, along with their corresponding benchmarking results, are presented in Figures 17 and 18.

AI Model	Modality	Records	Size (MB)	Accuracy	Training Time (mins)
CatBoost	Tabular	4.4M	767MB	76.9%	45
LGBM	Tabular	4.4M	767MB	73.5%	155
Gradient Boosting	Tabular	4.4M	767MB	77.5%	70
XGBoost	Tabular	4.4M	767MB	77.1%	7.35

Figure 15: Comparison of tabular data for AI models performance & training time.

AI Model	Energy Consumed (Joules)	Avg CPU Usage %	Avg Memory Usage (GB)	Avg Threads	CPU Energy Impact
CatBoost	124,544	406%	17	150	50%
LGBM	488,931	484%	14	97	44.2%
Gradient Boosting	162,169	450%	13	56	54.2%
XGBoost	17,451	368%	11	42	31%

Figure 16: Comparison of tabular data for AI models HW & energy consumption.

As shown in Figure 17, XGBoost [41] achieves an optimal balance between execution time and precision, completing the task significantly faster than the other models while maintaining impressive accuracy. Its execution time makes it highly suitable for scenarios where speed and efficiency are critical. Surprisingly, LightGBM [42], which is often praised for its speed, was considerably slower than expected, taking more than twice as long as the other models to complete the task.

Figure 18 provides further insight into the energy consumption patterns of the AI models under evaluation. XGBoost not only completes the task in the shortest amount of time but also demonstrates remarkable energy efficiency, consuming significantly less energy than the other models. In stark contrast, LightGBM (LGBM) emerges as the least efficient model, consuming 27 times more energy than XGBoost. This substantial disparity in energy consumption highlights the complexity differentiation in the underlying architectures and their efficiency in handling computational tasks.

An interesting observation is that XGBoost, being both the fastest and the most energy-efficient model, also exhibits the lowest CPU and RAM utilisation. This suggests that XGBoost is optimised for resource management, maintaining high performance while minimising its hardware footprint. Conversely, LGBM, the least energy-efficient model, placed the heaviest demand on CPU resources, particularly due to its extensive use of multithreading.

The intensive CPU utilisation in LGBM contributes to excessive energy consumption and longer execution times. Therefore, an intriguing trend can be observed: energy consumption appears to closely correlate with CPU usage. AI models that utilise less CPU power tend to consume less energy and, correspondingly, are less memory intensive. This relationship underscores the importance of CPU efficiency in determining overall energy consumption, making it a crucial factor when optimizing AI models for large-scale tasks in energy-constrained environments.

Throughout the benchmarking of Use Cases, Data Loading, Validation, Transformation, and Model Evaluation were also monitored for hardware and energy consumption. However, these phases are allocating computing time, and thus consuming energy, which is identical to the data size. The primary focus of this work is to benchmark the BD nature of AI pipelines, and most importantly the phase that mostly contributes to greater execution times, energy consumption, and hyperparameter tuning.

6. Conclusions

This document presents the “RAIDO Green AI Framework and Optimized Pipeline”, a robust and innovative solution designed to optimize AI workflows and datasets across the full E2C continuum. The framework effectively captures and analyses critical metrics such as energy consumption, performance efficiency, and hardware utilisation. It demonstrates its capability to support energy-efficient AI training and deployment through optimised data management strategies and HITL feedback mechanisms.

Built on a model-agnostic and HW/platform vendor-agnostic approach, the framework leverages a Kubernetes-orchestrated technology stack, including Prometheus, Grafana, InfluxDB, Kepler, and MLFlow, to enable fine-grained, real-time monitoring and analysis of AI pipeline footprints (performance and environmental impact). This comprehensive monitoring facilitates actionable insights into resource allocation, system performance bottlenecks, and environmental indicators such as CO2 emissions. A custom bash script further streamlines the post-execution analysis by transforming raw logs into structured, analyzable data.

Extensive benchmarking across diverse RAIDO use cases, encompassing both image and tabular data modalities, highlights the complex trade-offs in AI model selection. For edge-constrained applications (e.g., precision agriculture, robotics), energy efficiency is a primary optimization driver and target, where models like EfficientNet demonstrate a superior balance between accuracy and resource consumption. In contrast, in cloud-based applications (e.g., energy grid, healthcare), the focus shifts primarily towards maximising predictive accuracy. However, energy-efficient models such as XGBoost still exhibit favorable energy-to-performance ratios. Notably, the observed correlations between CPU utilisation and energy consumption across tabular data models further emphasizes the importance of holistic, system-level resource optimisation.

The document also details the Core Optimized AI-Pipelines (COAP) component, highlighting its modular architecture that supports advanced learning paradigms like data distillation, lifelong learning, and few-shot learning. COAP's integrated approach to ethical AI governance, XAI, and continuous monitoring for bias detection further reinforces the framework’s commitment to building trustworthy, transparent, and responsible AI systems.

In conclusion, the “RAIDO Green AI Framework and Optimized Pipeline” lays a strong foundation for the development and deployment of sustainable, scalable, and ethical AI solutions. Future work will focus on refining current methodologies, exploring and integrating more advanced optimisation techniques, and expanding the framework's capabilities to meet the emerging challenges of energy efficient and responsible AI at scale.

7. References

[1] (2021), Plamen P. Angelov, Eduardo A. Soares, Richard Jiang, Nicholas I. Arnold, Peter M. Atkinson; Explainable artificial intelligence: an analytical review, doi: 10.1002/widm.1424

[2] Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in AI. In Proceedings of the conference on fairness, accountability, and transparency (pp. 279–288). Atlanta, GA: ACM.

[3] Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA). IEEE (pp. 80–89).

[4] Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). IEEE Access, 6, 52138–52160.

[5] Dieber, J., & Kirrane, S. (2020). Why model why? Assessing the strengths and limitations of lime. arXiv preprint arXiv:2012.00093.

[6] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS One, 10, e0130140.

[7] Tritscher, J., Ring, M., Schlr, D., Hettinger, L., & Hotho, A. (2020). Evaluation of post-hoc XAI approaches through synthetic tabular data. In International symposium on methodologies for intelligent systems (pp. 422–430). Springer.

[8] Chen, H., Lundberg, S., & Lee, S.-I. (2019). Explaining models by propagating Shapley values of local components. arXiv preprint arXiv: 1911.11888.

[9] Pedreschi, D., Giannotti, F., Guidotti, R., Monreale, A., Ruggieri, S., & Turini, F. (2019). Meaningful explanations of black box AI decision systems. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, pp. 9780–9784).

[10] Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018). Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In 2018 IEEE Winter conference on applications of computer vision (WACV) (pp. 839–847).

[11] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618–626).

[12] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, Francisco Herrera, Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion, Volume 58, 2020, Pages 82-115, ISSN 1566-2535, https://doi.org/10.1016/j.inffus.2019.12.012. (https://www.sciencedirect.com/science/article/pii/S1566253519308103)

[13] Wang, Y., Wang, Q., Shi, S., He, X., Tang, Z., Zhao, K., & Chu, X. (2020, May). Benchmarking the performance and energy efficiency of AI accelerators for AI training. In 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID) (pp. 744-751). IEEE.

[14] Rodriguez, C., Degioanni, L., Kameni, L., Vidal, R., & Neglia, G. (2024). Evaluating the energy consumption of machine learning: Systematic literature review and experiments. arXiv preprint arXiv:2408.15128.

[15] Budennyy, S. A., Lazarev, V. D., Zakharenko, N. N., Korovin, A. N., Plosskaya, O. A., Dimitrov, D. V. E., ... & Zhukov, L. E. E. (2022, December). Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai. In Doklady mathematics (Vol. 106, No. Suppl 1, pp. S118-S128). Moscow: Pleiades Publishing.

[16] León-Vega, L. G., Tosato, N., & Cozzini, S. (2024, September). Efimon: A process analyser for granular power consumption prediction. In Latin American High Performance Computing Conference (pp. 112-126). Cham: Springer Nature Switzerland.

[17] Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., & Pineau, J. (2020). Towards the systematic reporting of the energy and carbon footprints of machine learning. Journal of Machine Learning Research, 21(248), 1-43.
[18] Anthony, L. F. W., Kanding, B., & Selvan, R. (2020). Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. arXiv preprint arXiv:2007.03051.
[19] Tchuente, D., Lonlac, J., & Kamsu-Foguem, B. (2024). A methodological and theoretical framework for implementing explainable artificial intelligence (XAI) in business applications. Computers in Industry, 155, 104044.
[20] Rizzo, M., Veneri, A., Albarelli, A., Lucchese, C., Nobile, M., & Conati, C. (2023, August). A theoretical framework for ai models explainability with application in biomedicine. In 2023 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB) (pp. 1-9). IEEE.
[21] Freund, L. (2023). Towards a comprehensive theory of aligned emergence in ai systems: Navigating complexity towards coherence.
[22] Brem, A., Giones, F., & Werle, M. (2021). The AI digital revolution in innovation: A conceptual framework of artificial intelligence technologies for the management of innovation. IEEE Transactions on Engineering Management, 70(2), 770-776.
[23] Gibson, D., Kovanovic, V., Ifenthaler, D., Dexter, S., & Feng, S. (2023). Learning theories for artificial intelligence promoting learning processes. British Journal of Educational Technology, 54(5), 1125-1146.
[24] Haidar, A. (2024). An integrative theoretical framework for responsible artificial intelligence. International Journal of Digital Strategy, Governance, and Business Transformation (IJDSGBT), 13(1), 1-23.
[25] Theodorou, G., Karagiorgou, S., & Kotronis, C. (2024, December). On Energy-aware and Verifiable Benchmarking of Big Data Processing targeting AI Pipelines. In 2024 IEEE International Conference on Big Data (BigData) (pp. 3788-3798). IEEE.
[26] Prometheus. Available at: https://prometheus.io/docs/introduction/overview/
[27] InfluxDB. Available at: https://www.influxdata.com/index/
[28] Grafana. Available at: https://grafana.com/
[29] Node Exporter. Available at: https://grafana.com/grafana/dashboards/1860-node-exporter-full/
[30] Kepler. Available at: https://github.com/sustainable-computing-io/kepler/blob/main/grafana-dashboards/Kepler-Exporter.json
[31] Mlflow. Available at: https://mlflow.org/
[32] Tielman, M. L., Suárez-Figueroa, M. C., Jönsson, A., Neerincx, M. A., & Siebert, L. C. (2024). Explainable AI for all-A roadmap for inclusive XAI for people with cognitive disabilities. Technology in Society, 79, 102685.
[33] Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.
[34] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
[35] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
[36] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.
[37] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[38] Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I. S., & Xie, S. (2023). Convnext v2: Co-designing and scaling convnets with masked autoencoders. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 16133-16142).

[39] Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). Real-time flying object detection with YOLOv8. arXiv preprint arXiv:2305.09972.
[40] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
[41] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
[42] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.

Funded by
the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101135800