TALON

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

Deliverable

D2.2 Experimental Verified & Optimized AI Theoretical Framework

Actual submission date: 31/03/2025

Funded by the European Union

Project Number:	101070181		
Project Acronym:	TALON		
Project Title:	Autonomous and Self-organized Artificial Intelligent Orchestrator for a Greener Industry 4.0		
Start date:	October 1st, 2022	**Duration:**	36 months
D2.2 Experimental Verified & Optimized AI Theoretical Framework			
Work Package:	WP2		
Lead partner:	UBITECH (UBI)		
Author(s):	George Theodorou (UBI); Ioannis Pastellas; Irem Doken (UBI); Sophia Karagiorgou (UBI)		
Reviewers:	Konstantinos Voulgaridis, Thomas Lagkas (DUTH; Magdalini Foti (NET)		
Due date:	31/03/2025		
Deliverable Type:	R — Document, report	**Dissemination Level:**	PU — Public
Version number:	1.0		

Revision History

Version	Date	Author	Description
0.1	02/12/2024	UBI	ToC release
0.2	20/12/2024	UBI	Contributions in all sections
0.3	20/02/2025	UBI	1st draft ready – sent for internal review
0.4	12/03/2025	DUTH, NET	Deliverable reviews with embedded comments
0.5	14/03/2025	UBI	Comments addressed; sent for quality review
0.6	24/03/2025	MINDS	Quality Review
0.7	25/03/2025	UBI	Addressed comments from quality review; sent to Coordinator
0.8	26/03/2025	UBI	Addressed comments from coordinator's review
1.0	28/03/2025	ENG	Final check before submission

Table of Contents

List of Figures

Definitions and Acronyms

AI	*Artificial Intelligence*
API	*Application Programming Interface*
CA	*Consortium Agreement*
CNN	*Convolutional Neural Networks*
CPU	*Central Processing Unit*
DAG	*Directed Acyclic Graph*
DoA	*Description of Action*
DT	*Digital Twin*
DX.X	*Deliverable X*
EC	*European Commission*
EC	*Evolutionary Computation*
EU	*European Union*
GA	*Grant Agreement*
GANs	*Generative Adversarial Networks*
GBMs	*Gradient Boosting Machines*
GDPR	*General Data Protection Regulation*
HW	*Hardware*
I/O	*Input – Output*
IoT	*Internet of Things*
KFP	*Kubeflow Pipelines*
kNN	*k-Nearest Neighbours*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MSE	*Mean Squared Error*
NN	*Neural Network*
PC	*Project Coordinator*
PRAUC	*Precision-Recall Area Under Curve*
RAM	*Random-access memory*
RFE	*Recursive Feature Elimination*
RMSE	*Root Mean Square Error*
RNNs	*Recurrent Neural Networks*
SVMs	*Support Vector Machines*
TFX	*TensorFlow*
TX.X	*Task X*
TC	*Technical Coordinator*
UCs	*Use Cases*
WP	*Work Package*
XAI	*Explainable AI*
YAML	*Yet Another Markup Language*

Disclaimer

Executive Summary

This document, henceforth referred to as D2.2, provides an extensive examination of Optimised Artificial Intelligence (AI) Theoretical Frameworks, spotlighting particularly on eXplainable AI (XAI) as well as practical hardware and energy benchmarking, together with their technological implementations. It imparts a detailed understanding of various post-hoc explainability techniques and theoretical AI frameworks, encompassing both system-centred and user-centred XAI methodologies.

Within D2.2, the conceptualization and advancement of AI theoretical frameworks are delineated, covering essential phases such as data loading, validation, transformation, model training, evaluation, and model deployment. Furthermore, it introduces the notion of federated learning at edge nodes, underscoring the pivotal role of distributed learning paradigms with privacy in contemporary AI systems. Moreover, D2.2 presents practical methods and performance assessments of the AI theoretical framework, furnishing insights into its efficacy, trustworthiness and confidentiality. Evolutionary computation algorithms have been used to assess the fitness over a collection of possible and competitive solutions to derive the most promising solution. The result is to solve the complex, non-linear, and multi-modal optimization problem of concurrently achieving high AI models accuracy with low energy consumption while training.

The deliverable concludes with key takeaways and prospective outlooks, accentuating the necessity for transparent and interpretable AI systems in moulding the prospective paradigm of artificial intelligence. In essence, D2.2 aspires to function as an authoritative guide in comprehending and implementing Optimised AI Theoretical Frameworks in practice measuring the trade-off metrics and deriving the optimal solutions under competitive conditions.

Introduction

This section introduces the deliverable and explains its overall scope within the TALON project and its objectives. It also documents the positioning of this deliverable within the TALON project, detailing its relationship with other deliverables (D), tasks (T), and work-packages (WPs). Specifically, it maps the technical specifications related to the theoretical framework and real-world performance modelling, while also explaining how the knowledge produced in other deliverables and work-packages served as input to the current deliverable.

1.1 Scope and Objectives

The aim of this deliverable is to report on TALON's theoretical framework, explaining the main components, functionalities, and the specific interfaces to be designed. This deliverable presents a structured approach to developing artificial intelligence (AI) systems that combine theoretical concepts with empirical validation and optimization techniques. The framework has been developed and improved throughout the progression of the TALON project.

We have thoroughly analysed the innovation strategy for administrative and management requirements for the procedures of the project TALON in the context of D1.1 and D5.4 (which is currently in progress). The primary objective was to design and develop TALON's novel theoretical framework, utilizing AI approaches to reduce energy consumption. This involves integrating theoretical concepts from these fields to explain and assess the operation of AI algorithms.

As TALON explores and implements a hybrid architecture that fuses data-driven and knowledge-centred techniques for optimal performance across various scenarios, the framework incorporates AI into Digital Twins (DTs) and modifies them to be non-intrusive reduced-order models capable of approximating future states. In addition, TALON utilizes data acquired from demonstrators to assess the performance of the developed theoretical framework in real-world scenarios. This includes the establishment of software evaluation methods, such as unit, integration, and system testing, to ensure robustness and reliability.

1.2 Relation to other work packages, tasks, and deliverables

Task 2.2 builds upon established user and system requirements, as well as AI metrics defined for TALON across different use cases and the overall system from T2.1.

D1.1, which defines the innovation strategy and procedures for improving the project's quality, contributes to this deliverable by providing a Performance Assessment Report. This report is crucial for benchmarking AI models deployed in the project and serves as a key input for D2.2, particularly in defining the main components, functionalities, and specific interfaces.

Additionally, this deliverable will contribute to WP3, WP4 and WP5, which focus on the transparency, explainability, reusability of models, and the instantiation of the components composing the TALON Conceptual Architecture. This process involves establishing seamless communication among system components and transforming the architecture into a fully functional and practical system.

Figure 1. Relation with other Deliverables and Work Packages.

1.3 Document Structure

The remainder of this document is organised as follows:

Section 1 – (Introduction): It presents the deliverable and explains its overall purpose.

Section 2 – (Literature Review): This section reviews the state of the art in AI-driven theoretical frameworks and highlights the novel contributions introduced in the TALON project.

Section 3 – (AI Theoretical Framework): It defines the AI Theoretical Framework, including its concepts and building blocks.

Section 4 – (Technical Implementation): It reports on the current development activities and the early prototype deployed in the TALON infrastructure to support the AI Theoretical Framework.

Section 5 – (Conclusion and Future Outlook): This section concludes the deliverable. It summarizes the main findings of the deliverable, outlining the next steps, potential enhancements, future research directions, and technological advancements for the consortium.

State-of-the-Art in Optimised AI Theoretical Framework

There has been a significant revolution in the industry due to advancements in Artificial Intelligence (AI) and Machine Learning (ML), leading to high accuracy levels, often surpassing human performance for different sort of problems. High accuracy rates are associated with models that have an immense number (millions or even billions) of weights (parameters), which are supposed to contain the learned information from training data. According to Angelov et al. [9], not only is the amount of data for these weights very large, but their high-dimensional and non-linear nested structure of input data makes AI non-explainable. Therefore, such models are considered non-transparent or opaque or “black box” models, as presented by Mittelstadt et al. [10].

However, alongside AI’s transformative potential, concerns regarding transparency, accountability, and trustworthiness have emerged, prompting the need for 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 is greater than ever.

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

- Transparency: A model is considered transparent if it is understandable, essentially the opposite of a “black box” model.
- Interpretability: The ability to provide interpretations that can be understood by people.
- Explainability: The capability of a system to provide accurate and comprehensible explanations for humans, as presented by Gilpin et al. [12].

2.1 The Taxonomy and Ontology of XAI

Even though those explanations are close to their semantic meanings, XAI taxonomy can be categorized as follows:

- Transparent model: Typical transparent models presented by Adadi et al. [13] include k-nearest neighbours (kNN), decision trees, rule-based learning, Bayesian network, and so on. The decisions from these models are often transparent, although transparency, as a property, does not guarantee that a model will be readily explainable, as detailed by Angelov et al. [9].
- Opaque model: Models such as random forest, neural networks, support vector machines (SVMs) typically offer high accuracy but lack transparency.
- Model agnostic: XAI approaches that are model-agnostic, as described by Dieber [14], are designed to be broadly applicable. These approaches prioritize flexibility, ensuring they can work independently of the specific model architecture. They focus on understanding the relationship between a model's input and output without being tied to the model's internal workings.
- Model-specific: Unlike model-agnostic XAI approaches, these techniques leverage specific knowledge about a particular model or models to provide transparency. The goal, as outlined by Bach [15], is to shed light on the inner workings of a specific type of model or a set of models.

2.1.1 Post-Hoc Explainability Techniques of Model Agnostic

Several methods have been developed to explain complex AI models post-hoc:

- Text explanations; This technique addresses the challenge of enhancing the explainability of a model by training it to produce text explanations that clarify the model's outcomes. This approach encompasses techniques that generate symbols representing how the model operates. These symbols can elucidate the algorithm's logic through a semantic mapping from the model to symbols.
- Explanation by simplification: This method involves simplifying a model through approximation, as proposed by Tritscher [16]. By creating simpler surrogate models like linear models or decision trees based on the original model's predictions, we can explain the complex model's predictions more understandably.
- Explanation by feature relevance: Similar to simplification, this approach focuses on evaluating the importance of features by considering their average expected marginal contribution to the model's decision. Researchers like Chen [17] and Pedreschi [18] explore this concept by assessing how different features impact the model's decisions.
- Visual explanation: XAI approaches based on visualization, as discussed by Chattopadhay [19], rely on data visualization 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: Local explanations, as introduced by Selvaraju [20], 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 like 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's (Arrieta [21]).

2.2 The Theoretical Frameworks for AI

Below some of the theoretical frameworks for Artificial Intelligence are presented that are most useful and relevant to our work for TALON:

- Explainable Artificial Intelligence (XAI): This framework focuses on providing explanations that are structurally sound. It combines 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 Behavior and Alignment: This framework explores the dynamics of emergent behavior and alignment in AI systems. It considers the interplay between system states, inputs, function rules, learning algorithms, environments, and historical data. It emphasizes the need for ongoing adaptation and control, robust alignment mechanisms, and empirical validation.
- AI as Originator and Facilitator of Innovation: This framework discusses how AI plays two roles 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, and identifies fourteen roles for AI in education, aligned with the model's features.

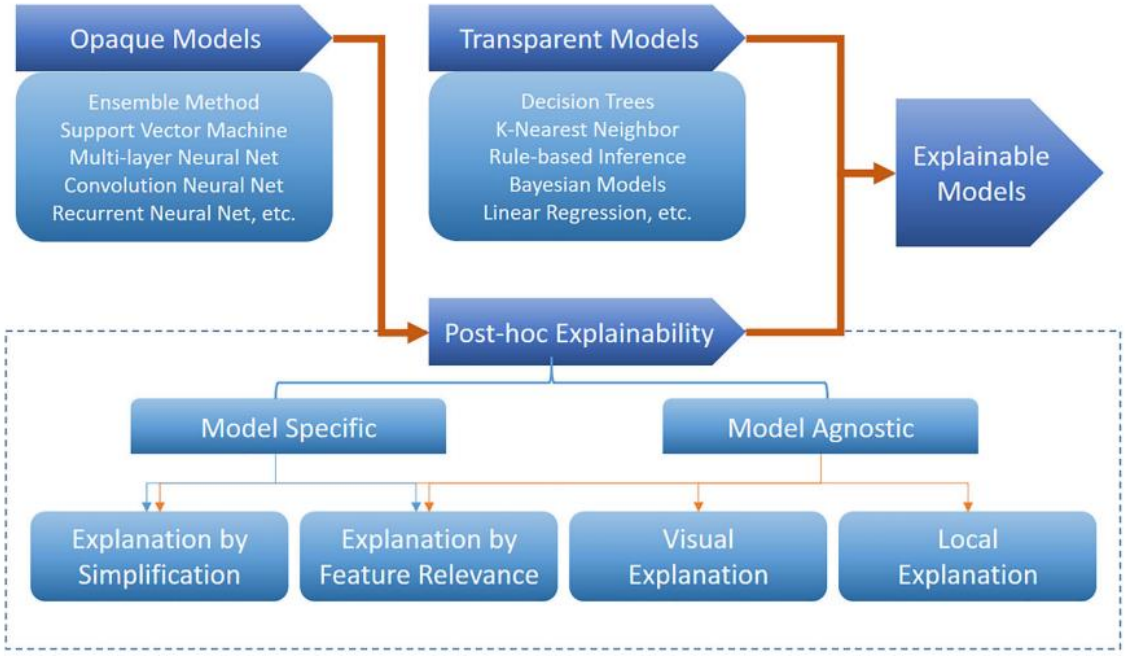


Figure 3. The High-level Ontology of Explainable AI Approaches (Angelov et. al., [9]).

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. Neerincx et al.'s [1] conceptual framework delineates three essential phases of explanation generation, communication, and reception, providing a roadmap for research in XAI. Within this framework, system-centred and user-centred approaches to XAI emerge as distinct domains, each posing 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 defy traditional methods of interpretation. Black-box systems, characterized by their inscrutability, pose 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 modeling, 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 opacity and 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, catering to diverse informational needs and cognitive capacities. In this domain, the communication and reception phases of explanation assume paramount importance, as users

navigate complex concepts and seek to make informed decisions based on AI-generated insights. Pragma-dialectical theory offers a roadmap for crafting persuasive, intelligible explanations tailored to end-users' cognitive capacities and informational needs, while Inference Anchoring Theory bridges inferential structures in argumentation with their corresponding illocutionary forces and dialogical processes, enabling the adaptation of explanation to specific user contexts.

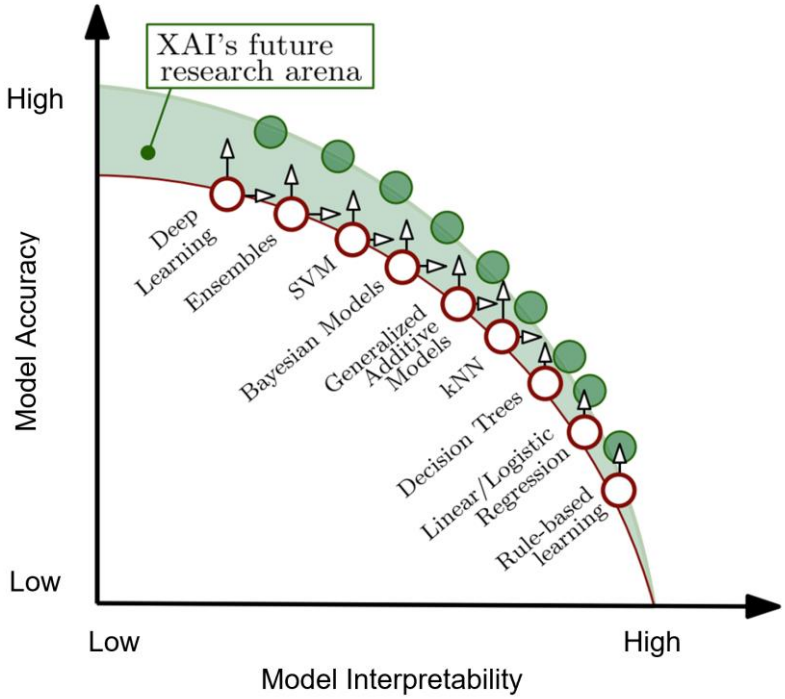


Figure 4. Trade-off Graph for Interpretability and Performance (Arrieta et al., [21]).

The illustration presented in Figure 4, offers a conceptual depiction influenced by prior research, illustrating how XAI can enhance the traditional trade-off between model interpretability and performance. An additional point to highlight, closely related to model interpretability and performance, is the approximation dilemma: explanations crafted for a machine learning model must strike a balance between being sufficiently drastic and approximate to meet the needs of the intended audience, ensuring that the explanations accurately reflect the model under study without oversimplifying its key characteristics.

2.2.3 Energy Tracking Existing Methodologies/Frameworks

A comprehensive study on the aspects that affect the Big Data AI pipeline training, considering different objectives, highlights significant variations in both performance and energy consumption during Deep Neural Networks (DNN) training [22].

Both the system architecture, including CPUs, GPUs, and Tensor Processing Units (TPUs), and the AI model complexity should be considered during benchmarking when optimising the training phase. While GPUs and TPUs provide high throughput for tasks such as image recognition and speech-to-text, energy efficiency can greatly differ depending on the hardware and the optimization techniques applied. These findings underscore the necessity of considering both performance and energy consumption when selecting hardware for AI training, particularly in environments where cost, energy preservation, and efficiency are critical factors.

The importance of evaluating energy consumption in machine learning (ML) is widely recognized for monitoring, understanding, and optimizing its computational and environmental impact. However, there is no single approach that can address all use cases, and there is an ongoing debate about the best methods to evaluate energy consumption for specific applications. In the meantime, various methods, each with unique strengths and limitations, have been developed. A systematic review of these approaches, designed to evaluate energy consumption during both training and inference, was conducted for TALON purposes, followed by an experimental protocol to compare the effectiveness of these methods across diverse AI tasks, including vision and language models [23].

Several libraries have emerged to address the challenge of tracking energy consumption in AI pipelines. One prominent example is the eco2AI library [24], which offers a powerful solution for monitoring energy usage and CO2 emissions during both training and inference phases. This library tracks CPU, GPU, and RAM utilization by gathering power consumption logs through process identification (PID) and system metrics retrieved using Linux commands, such as top.

Similarly, the EfiMon tool [25] provides a granular, non-invasive method for tracking energy consumption at the process level. EfiMon uses regression-based models to estimate energy usage with high precision, even in shared computing environments. This tool has shown small deviations in its measurements on Intel and AMD systems, making it a valuable resource for optimizing energy consumption in AI research and high-performance computing (HPC) [25].

EIT [26] is another tool that 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, especially in the context of energy-efficient reinforcement learning algorithms.

CarbonTracker (CT) [27] is a specialized tool for tracking the 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 Multi-Objective Deployment based on Hardware and User Requirements

At the same time, there has been an evident revolution in the industry thanks to the improvement of AI pipelines with a high level of accuracy performance even surpassing human capabilities for different sorts of problems. Achieving a high accuracy rate is closely related to models that have a vast amount (millions or even billions) of weights (parameters) that are supposed to contain the information learnt from training data. However, many modern AI methods have a black-box nature, which hinders their adoption by practitioners in many application fields. This issue raises a recent emergence of a new research area in AI, setting the ground for (i) extensive benchmarking over different algorithms and methods to understand their behaviour across different data modalities; (ii) use of data and features of different granularity and veracity to optimise the learning capability and thus the performance of an AI model; and (iii) monitoring the underlying compute resources to dimension the financial, computing or energy costs of AI model training and to derive trade-offs (i.e. through what-if analyses) regarding smart placement, energy consumption, and other business-defined objectives. In addition, there is the need to explain the behaviour of an AI model, aiming at providing more understandable, interpretable, and justifiable for humans AI-based decision-making processes and outcomes. Several theoretical frameworks for AI have been introduced to tackle different directions, focusing on (i) Explainable Artificial Intelligence (XAI); (ii) Emergent Behaviour and Alignment of AI; (iii) AI as Originator and Facilitator of Innovation; and (iv) Three-Level Model for AI while Learning.

Tchuente et al. [28] proposed a new methodological and theoretical framework for XAI decomposed into six steps that can be followed by all practitioners or stakeholders to improve the implementation

and adoption of XAI in their business applications. They highlighted the need to rely on domain field and analytical theories to explain the entire analytical process, from the relevance of the business question to the robustness checking and validation of explanations provided by XAI methods.

Rizzo et al.[29] fit explanations of an AI model into the properties of 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 operationalization of these properties, and provides new insights into common explanation methods that they analyse as case studies. They also discuss the impact of their framework in biomedicine, a very sensitive application domain where XAI can have a central role in generating trust.

Freund et al. [30] explore the complex dynamics of emergent behaviour and alignment within AI systems and present a comprehensive framework for conceptualizing and modelling these phenomena. Their framework incorporates the multilevel and time-dependent nature of emergent behaviour and alignment, considering the interplay between system states, inputs, function rules, learning algorithms, environments, and historical data. The proposed framework sheds light on the challenges and opportunities associated with achieving and maintaining alignment in AI systems.

Brem et al.[31] introduced a two-part conceptual AI framework: The first part views AI as a technology that can fulfil different roles within a company, and the second looks at AI and its use along the company's innovation processes. They also discussed these two views using examples from existing field applications and described potential areas for future research and limitations of the proposed framework.

Gibson et al [32] introduced a three-level model that synthesizes and unifies existing learning theories to model the roles of AI in promoting learning processes. The model, drawn from developmental psychology, computational biology, instructional design, cognitive science, complexity, and sociocultural theory, includes a causal learning mechanism that explains how learning occurs and works across micro, meso, and macro levels. The model also explains how information gained through learning is aggregated, or brought together, as well as dissipated, or released and used within and across the levels.

Last, Haidar [33] proposes a novel integrative theoretical framework for Responsible AI (RAI), which addresses four 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. The first embeds elements like anticipation and reflexivity into corporate culture, and the latter examines AI-specific laws from the European Union and the United States, providing a comparative perspective on legal frameworks governing AI. The study's findings are helpful for businesses seeking to responsibly integrate AI, developers who focus on creating responsibly compliant AI, and policymakers looking to foster awareness and develop guidelines for RAI.

Within this expansive domain of Big Data and edge computing, AI stands as a beacon, transforming raw data into actionable insights and automating a myriad of complex tasks. However, the intricate relationship between AI and Big Data gives rise to various technical challenges, like the number of training epochs and time, over-/under-fitting, and data leakage, which can influence the efficacy of AI models. Last, the inherent characteristics of AI models and the energy-constrained edge devices further contribute to technical challenges while optimizing AI models for smart placement, cost, energy reduction, and more.

Design of the AI Theoretical Framework

The role of the AI Theoretical Framework is to employ ML and XAI to explain and assess the operation of AI algorithms. To achieve this, we first introduced a pipeline covering all the phases of learning (both at the training and inference phase) and extended to federated learning to model the learning performed at the edge devices.

The novelties of the proposed AI Theoretical Framework are highlighted as follows:

- Transparency of the internal learning of diverse AI models covering different learning tasks and types of data;
- Explainability; and
- Abstraction and Extensibility.

Figure 5 depicts the high-level illustration of the AI Theoretical Framework which models both centralized and decentralized learning tasks. The centralised phases of the learning pipeline denote that computing and resource allocation is made in a centralised cloud infrastructure, while the decentralised phases are being performed by federated edge computing nodes.

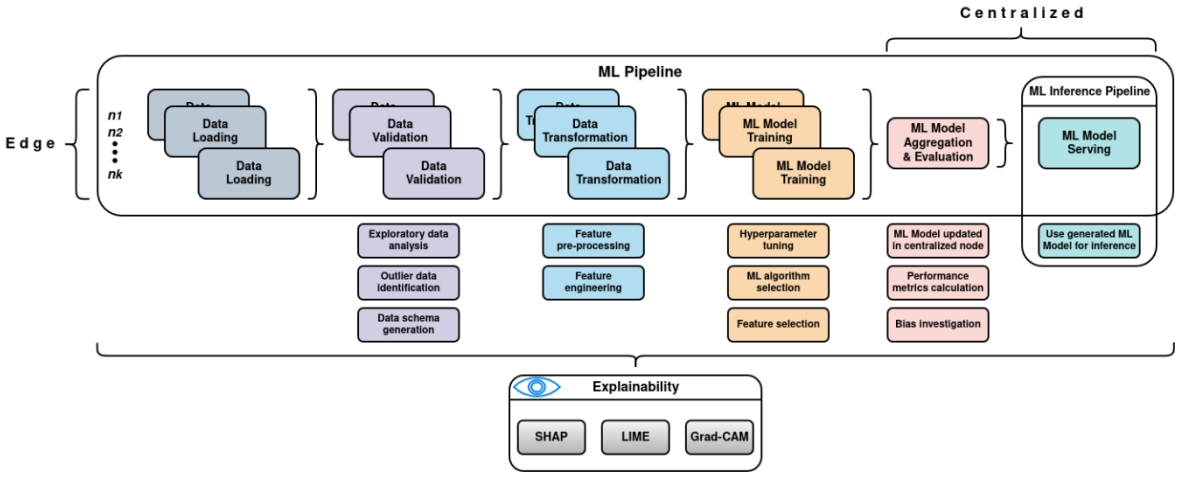


Figure 5. Abstracted Pipeline for Studying and Explaining AI Algorithms (Theodorou et al, [42]).

The AI Theoretical Framework is pivotal for our research as it accomplishes two significant objectives. Firstly, it provides a comprehensive framework that caters to a wide range of scenarios necessitating AI deployment, including tasks involving image processing, natural language processing with text data, and handling structured tabular data, among others. This versatility is crucial in today's dynamic AI landscape, where diverse data types and applications are prevalent. Secondly, the pipeline's inherent flexibility allows us to customize each component according to the specific requirements of individual use cases. This adaptability ensures that the pipeline remains efficient and effective across various domains, enabling us to achieve optimal results and address distinct challenges with precision and scalability. The design of the pipeline depicted in Figure 5 represents a significant advancement in AI development and implementation. By encompassing multiple data modalities such as images, text, and tabular data, the pipeline offers a holistic approach to AI-driven solutions. This is particularly advantageous in real-world scenarios where data sources are diverse and complex, allowing us to build robust models that can handle heterogeneous data inputs seamlessly. Furthermore, the modular nature of the pipeline enables us to fine-tune each component to suit the specific needs of different use cases. Whether it's adjusting model architectures for better performance or integrating

specialized preprocessing techniques, the flexibility of the pipeline ensures that we can optimize our AI solutions for maximum accuracy and efficiency. Overall, this pipeline not only enhances our ability to tackle a wide range of AI tasks but also empowers us to adapt and innovate in response to evolving challenges and requirements in the field of artificial intelligence.

The segmentation of the pipeline into two distinct parts, one operating in the cloud and the other at the edge, represents a strategic approach to AI implementation. This division enables a centralized training process capable of aggregating results from numerous edge devices and consolidating them on a centralized server. Such a setup not only enhances scalability but also promotes efficient data management and analysis across distributed environments. Moreover, this architecture underscores the versatility of our AI algorithms, accommodating various development methodologies whether they are designed for edge computing or direct deployment on server infrastructure. By incorporating both cloud and edge computing capabilities, our pipeline demonstrates a comprehensive understanding of modern AI paradigms, ensuring adaptability and effectiveness across different deployment scenarios. The division of the pipeline into cloud-based and edge-based components reflects a sophisticated strategy tailored to the demands of contemporary AI systems. This bifurcation enables a centralized training mechanism that can harness the computational power of multiple edge devices, consolidating their outputs on a centralized server for streamlined analysis and decision-making. Moreover, this approach acknowledges the diverse pathways through which AI algorithms can be constructed and deployed, whether optimized for edge computing environments or designed for direct execution on powerful server infrastructures. This dual-capability architecture not only ensures the scalability and resilience of our AI solutions but also underscores our commitment to leveraging cutting-edge technologies in tandem, thereby maximizing performance and adaptability across a spectrum of use cases and operational scenarios.

The genesis of generic AI pipeline framework draws inspiration from the TFX user guide, which illuminates TensorFlow's adept handling of diverse AI pipelines spanning text analysis, image recognition, and beyond. TensorFlow stands as a cornerstone in the realm of AI development, boasting a robust framework capable of accommodating a wide array of use cases and data modalities. Leveraging the rich functionalities and best practices delineated by TensorFlow, we have crafted our own adaptable framework, making nuanced adjustments to align with the specific requirements and operational criteria outlined in our work package. By embracing the paradigm established by TensorFlow while tailoring our approach to suit our unique context, we ensure that our AI pipeline not only harnesses industry-leading techniques but also remains finely attuned to the intricacies of our project's objectives and constraints. The design philosophy behind our generic AI pipeline framework is deeply rooted in the principles and methodologies outlined in the TFX user guide. TensorFlow's comprehensive approach to handling various AI pipelines, ranging from text and image processing to other domains, serves as a beacon of best practices and efficient implementation strategies. As TensorFlow stands as the industry standard for developing diverse forms of AI pipelines, we have meticulously studied and integrated its proven techniques into our framework. However, recognizing the unique nuances and requirements of our work package, we have tailored and fine-tuned certain aspects to ensure seamless integration and optimal performance within our specific context. By amalgamating the established practices of TensorFlow with our customized adaptations, we strive to create a robust and versatile AI pipeline framework that excels in handling a multitude of tasks while remaining agile and responsive to evolving demands and challenges in the AI landscape.

3.1 Data Loading

The initial phase of any AI pipeline is dedicated to loading the pertinent data that will serve as the foundation for training and analysis. Data holds unparalleled significance in AI implementations,

serving as the lifeblood that fuels model development and decision-making processes. By prioritizing robust data loading mechanisms, we ensure that our pipeline is equipped with the necessary information to derive meaningful insights and drive informed decisions. Moreover, efficient data loading offers several key benefits, including improved data integrity, enhanced model accuracy through comprehensive dataset representation, and streamlined data preprocessing workflows. By establishing a solid data loading foundation at the forefront of our AI pipeline, we lay the groundwork for successful AI model development and deployment across diverse applications and use cases. The initial step in any AI pipeline, data loading, is crucial for several reasons. Firstly, it ensures that the AI pipeline has access to the necessary information required for training and inference tasks. Without properly loaded data, the pipeline cannot function effectively or produce meaningful results. Secondly, robust data loading procedures contribute to data integrity and quality, minimizing errors and inaccuracies that could impact the performance of AI models. Additionally, efficient data loading streamlines the preprocessing and transformation of raw data into formats suitable for model training and evaluation. This not only accelerates the development process but also enhances the scalability and adaptability of the AI pipeline to handle large and diverse datasets. Overall, prioritizing data loading as the initial step in the AI pipeline sets a strong foundation for subsequent stages, ensuring reliable and accurate outcomes in AI-driven applications.

3.2 Data Validation

The validation of loaded data constitutes the second critical step in the AI pipeline. This phase involves conducting a series of checks and assessments to ensure that the data contains accurate and relevant information necessary for subsequent stages. The validation process is multifaceted, tailored to the specific modality of the loaded data—whether it's images, text, tabular data, or other formats. Moreover, considerations such as the AI model to be trained, computational resources available, and the desired level of data accuracy play a pivotal role in determining the validation criteria. By meticulously validating the loaded data, we uphold data quality standards, which directly correlate with the accuracy and reliability of the AI models trained downstream.

Effective data validation serves as a cornerstone for robust AI systems, contributing significantly to the overall performance and effectiveness of the pipeline. Through comprehensive validation checks, potential errors, inconsistencies, and outliers within the data can be identified and addressed proactively. This proactive approach not only enhances data quality but also minimizes the risk of biases or inaccuracies influencing the training and evaluation of AI models. Additionally, data validation contributes to ensuring compliance with regulatory standards and ethical considerations, fostering trust and transparency in AI-driven decision-making processes.

Furthermore, integrating automated validation mechanisms into the AI pipeline streamlines the validation process, reducing manual efforts and accelerating the overall development cycle. Leveraging advanced techniques such as data profiling, anomaly detection, and statistical analysis, we can establish robust validation protocols tailored to the unique characteristics of each data modality. This proactive validation strategy not only enhances data quality assurance but also empowers us to adapt and respond effectively to evolving data challenges and complexities in the AI landscape.

Integrating automated validation mechanisms into the AI pipeline streamlines the validation process, reducing manual efforts and accelerating the overall development cycle. Various checks can be employed to ensure the quality and integrity of the loaded data across different modalities. For image data, checks may include verifying image resolution, aspect ratio, and colour channels consistency, as well as detecting corrupted or incomplete images. Text data validation may involve identifying and correcting spelling errors, handling missing values, and ensuring uniform text encoding across the dataset. Tabular data validation encompasses checks for data completeness, consistency, and

correctness, such as detecting outliers, duplicates, or discrepancies in numerical values and categorical labels.

In addition to these modality-specific checks, overarching data quality assessments can be performed to evaluate the overall coherence and reliability of the dataset. This may involve assessing data distribution and statistical properties, detecting data imbalances or biases, and ensuring adherence to predefined schema or formatting standards. Anomaly detection techniques can also be employed to identify outliers or anomalies that deviate significantly from the expected data patterns, potentially indicating data errors or anomalies.

Furthermore, data validation extends beyond structural and statistical checks to encompass semantic validation, ensuring that the data aligns with the intended semantics or domain-specific constraints. This may involve validating relationships between different data attributes, verifying data consistency across related datasets, and enforcing domain-specific rules or constraints. For instance, in healthcare data, semantic validation may involve verifying the consistency of patient demographics across different medical records or ensuring compliance with privacy regulations such as HIPAA.

By leveraging a combination of these validation techniques, the AI pipeline can establish comprehensive quality assurance protocols tailored to the unique characteristics and requirements of the loaded data. This proactive validation strategy not only enhances data quality and integrity but also fosters confidence in the subsequent stages of AI model training and evaluation, ultimately leading to more accurate and reliable AI-driven insights and decisions.

3.3 Data Transformation

The third pivotal step in the AI pipeline involves transforming the loaded data into a format optimized for leveraging its inherent features and characteristics to train the most effective machine learning models. This transformation process is multifaceted, relying on a range of transformation functions tailored to the specific attributes of the data. Several factors influence the choice of transformation functions, including the data modality (e.g., images, text, tabular data), the shape and format of the loaded data, the desired AI model architecture, and the computational resources available. This transformation process can be divided into two key components: feature engineering and feature selection.

Feature engineering constitutes the first part of the data transformation process, focusing on creating new features or representations from the existing data. This step is crucial for enhancing the predictive power of AI models by extracting meaningful patterns and relationships from the data. Depending on the data modality and the nature of the AI model being developed, various feature engineering techniques can be applied. For example, in image data, feature engineering may involve extracting visual descriptors or features using techniques like convolutional neural networks (CNNs) or image preprocessing methods. Similarly, in text data, feature engineering may encompass techniques such as word embeddings, n-grams, or text vectorization to represent textual information in a format suitable for model training.

The second part of the data transformation process is feature selection, which involves determining the most relevant and informative features to include in the model while discarding irrelevant or redundant ones. This step is essential for optimizing model performance, reducing overfitting, and improving interpretability. Feature selection techniques vary based on the data type and the ML model's requirements. For instance, in tabular data, feature selection methods like feature importance ranking, correlation analysis with the target label, or recursive feature elimination (RFE) can be utilized to identify and retain the most influential features for predictive modelling. By leveraging these transformation techniques effectively, the AI pipeline can preprocess and prepare the data in a

manner that maximizes the model's predictive capabilities and generalization performance, ultimately leading to more accurate and robust AI-driven solutions.

3.4 ML Model Training

The subsequent step in the AI pipeline is the actual creation of the AI model itself. This stage involves utilizing the transformed data from the previous step to train the model, enabling it to understand and learn from the underlying patterns and features within the data. Each preceding component of the pipeline contributes to the smooth and optimal execution of this step. Model training is a pivotal stage that consumes significant time and computational resources, representing the core of the AI pipeline where the true magic of machine learning unfolds. The choice of algorithm for training the model is influenced by several key factors.

One of the primary factors dictating the selection of the AI model algorithm is the modality of the data being used. For instance, in the case of image data, Convolutional Neural Networks are widely regarded as one of the most effective and popular algorithms due to their ability to capture spatial dependencies and hierarchical features within images. Similarly, for text data, algorithms like Recurrent Neural Networks (RNNs) or Transformer models are often employed to process sequential data and capture contextual dependencies effectively.

Another crucial factor shaping the type of AI algorithm chosen is the specific task that the model is intended to perform. For supervised learning tasks, where the model learns from labelled data and aims to predict or classify based on predefined target labels, algorithms like Support Vector Machines (SVMs), Decision Trees, or Gradient Boosting Machines (GBMs) are commonly utilized. On the other hand, for unsupervised learning tasks, where the model learns patterns and structures from unlabelled data, algorithms such as K-means clustering, Hierarchical clustering, or Generative Adversarial Networks (GANs) may be more suitable.

Furthermore, the available computing power and resources play a significant role in determining the complexity and scalability of the AI algorithm chosen for model training. Deep learning algorithms, for instance, often require substantial computational resources, especially when dealing with large-scale datasets or complex neural network architectures. Therefore, considerations such as parallel processing capabilities, GPU utilization, and distributed computing frameworks may influence the decision-making process regarding the selection of AI algorithms for model training within the AI pipeline.

During the ML model training phase within the AI pipeline, the transformed data is used to iteratively adjust the model's parameters and optimize its performance. This process typically involves splitting the data into training, validation, and testing sets to evaluate the model's generalization ability accurately. The training phase consists of feeding the training data into the model, which then learns to recognize patterns, correlations, and features relevant to the task at hand. The model's performance is continuously evaluated using the validation set to prevent overfitting (when the model performs well on the training data but poorly on new, unseen data) and to fine-tune hyperparameters for optimal performance.

The ML model training process is iterative and often involves hyperparameter tuning, regularization techniques (to prevent overfitting), and optimization algorithms (such as Gradient Descent) to minimize the model's loss function and improve its predictive accuracy. Additionally, ensemble methods like Bagging and Boosting can be employed to combine multiple base models for improved generalization and robustness.

It's essential to monitor and evaluate the model's performance throughout the training process using metrics such as accuracy, precision, recall, F1-score (for classification tasks), Mean Squared Error

(MSE), Root Mean Squared Error (RMSE), or R-squared (for regression tasks). This allows fine-tuning the model's parameters, adjusting the learning rate, or exploring different optimization strategies to achieve the desired level of performance and generalization.

3.5 ML Model Evaluation

The final step in the AI pipeline entails evaluating the quality and performance of the generated AI model. This evaluation process is crucial as it determines the model's effectiveness in solving the specified problem and meeting the predefined task criteria. Various machine learning model evaluation metrics are available, each focusing on different aspects of model performance. Some widely recognized evaluation metrics include Precision, Recall, F1 score, PRAUC (Precision-Recall Area Under Curve), Accuracy, and many others. These metrics provide insights into different attributes of the model's behavior, such as its ability to make correct predictions, identify relevant instances, and balance between false positives and false negatives.

The choice of evaluation metric depends on the specific requirements and objectives of the task at hand. For example, in scenarios where precision is of utmost importance, such as medical diagnostics or financial fraud detection, a high Precision metric is desired to minimize false positives and ensure reliable decision-making. Conversely, in situations where recall is prioritized, such as anomaly detection or search engines, a high Recall metric is crucial to capture as many relevant instances as possible, even at the cost of some false positives.

Another factor influencing the selection of evaluation metrics is the type of ML algorithm used for model training. For supervised learning algorithms, evaluation metrics like Accuracy, Precision, Recall, and F1 score are commonly applied to assess classification and regression model performance. In contrast, for unsupervised learning algorithms, different evaluation metrics such as Silhouette score (for clustering) or Reconstruction Error (for autoencoders) may be used to evaluate clustering quality or reconstruction accuracy, respectively.

Ultimately, the evaluation step serves as a critical checkpoint to determine whether the produced AI model is ready for deployment in real-world scenarios. Based on the evaluation results and the task requirements, decisions can be made regarding model refinement, hyperparameter tuning, or deployment readiness. This final evaluation step ensures that the AI model meets the desired performance standards and can deliver accurate and reliable predictions when deployed in production environments with new, unseen data.

3.6 ML Model Serving

Once all the preceding steps of the AI pipeline have been meticulously executed and the AI model has been fine-tuned and validated, it transitions into the inference phase. The inference process involves utilizing the trained AI model to generate predictions, classifications, or recommendations based on new, unseen data inputs. This phase is where the AI model's true utility and effectiveness are put to the test in real-world applications.

During inference, the AI model takes input data and applies the learned patterns, features, and relationships acquired during the training phase to make predictions or decisions. This could involve tasks such as image classification, sentiment analysis, anomaly detection, predictive maintenance, or personalized recommendations, depending on the nature of the AI model and its intended application. The model's ability to generalize and provide accurate predictions on unseen data is a key determinant of its success in the inference phase.

The inference process typically involves deploying the AI model in a production environment, where it interacts with live data streams or user inputs to generate real-time insights or actions. This may require integrating the AI model into existing software systems, applications, or platforms to enable

seamless data processing and decision-making. Additionally, considerations such as model latency, scalability, and reliability become paramount during inference, ensuring that the AI model can handle varying workloads and deliver timely responses without compromising performance.

Continuous monitoring and evaluation of the AI model's performance during the inference phase are essential to detect any drift or degradation in predictive accuracy over time. This may involve monitoring key performance metrics, analysing prediction errors, and retraining the model periodically to adapt to changing data patterns or evolving user requirements. By leveraging the insights and feedback gathered during the inference phase, organizations can iteratively improve their AI models, enhance decision-making processes, and drive tangible business outcomes.

3.7 Federated Learning at Edge Nodes

A key part of our proposed AI Theoretical Framework is the use of federated learning for allowing de-centralized training on the edge. Federated learning has emerged as a transformative paradigm in machine learning, offering a decentralized approach to model training while preserving data privacy and security. This methodology is particularly relevant in scenarios where sensitive data is involved, such as healthcare, finance, and IoT networks. By distributing the training process to edge devices or local servers, federated learning enables organizations to collaboratively improve machine learning models without centrally storing or sharing raw data. This not only addresses privacy concerns but also enhances data locality, reducing latency and bandwidth requirements for model updates.

One of the key advantages of federated learning is its applicability in mobile and IoT environments, where data privacy is of utmost importance. In these contexts, devices can autonomously train AI models using locally generated data, such as user interactions or sensor readings, without compromising individual privacy. The trained models' updates are then securely aggregated and merged at a central server or cloud, ensuring that sensitive information remains decentralized and protected. This approach not only enhances data privacy but also empowers edge devices to contribute to collective intelligence, leading to more personalized and efficient AI applications.

Moreover, federated learning has implications beyond privacy preservation. It also addresses challenges related to data heterogeneity, scalability, and regulatory compliance. By leveraging federated learning techniques, organizations can harness the full potential of distributed data sources while mitigating risks associated with centralized data storage and processing. However, federated learning also poses technical challenges, such as communication overhead, model synchronization, and federated optimization strategies, which require ongoing research and development efforts to optimize the performance and scalability of federated learning systems.

In constructing our AI pipeline, a fundamental goal is to prioritize explainability and trustworthiness, ensuring transparency in the inner workings of the pipeline and providing clarity on the processes involved when executing an AI pipeline. To achieve this, our initial step is to meticulously measure the hardware consumption resources utilized throughout the pipeline's operations. By quantifying and analysing these resource allocations, we can offer comprehensive explanations regarding the computational resources utilized, the efficiency of resource allocation strategies, and the overall impact on performance metrics. This focus on measuring and explaining hardware consumption not only enhances transparency but also enables stakeholders to gain a deeper understanding of the computational intricacies and optimizations within the AI pipeline.

In the upcoming chapter, we will delve into the actual implementation work undertaken to create and subsequently measure hardware (HW) consumption within our AI pipeline. This phase encompasses a series of methodical steps aimed at developing robust mechanisms for monitoring and quantifying the computational resources utilized during pipeline execution. We outline the methodologies, tools, and frameworks employed to construct the HW consumption measurement framework, ensuring

accuracy, reliability, and scalability. Moreover, we detail the implementation process, including data collection procedures, instrumentation of hardware monitoring components, and integration with the AI pipeline architecture. Through this chapter, readers will gain insights into the technical intricacies involved in creating and measuring HW consumption, paving the way for enhanced transparency and understanding of the AI pipeline's operational dynamics.

AI Theoretical Framework Development and Deployment

The experimental framework, as illustrated in Figure 6 , effectively captures the metrics of energy consumption, performance efficiency, and hardware utilization.

4.1 Technological Stack

This framework is hardware and platform vendor-agnostic, designed to cover a broad range of scenarios and use cases. We place particular emphasis on ensuring our framework is both highly scalable and extensive, making it practical for real-world Edge-to-Cloud deployments. By tracking system operations on a per-second basis, our framework effectively captures dynamic workloads and fluctuating resource availability, even in cases of abrupt changes. However, we record the technical specifications of the target hardware and platform based on their objective capacity. The first step includes the abstraction and containerization of the Big Data AI pipeline within a Kubernetes cluster [35]. Proper configuration and setup of the cluster are necessary for monitoring the execution environment of the Big Data application under evaluation.

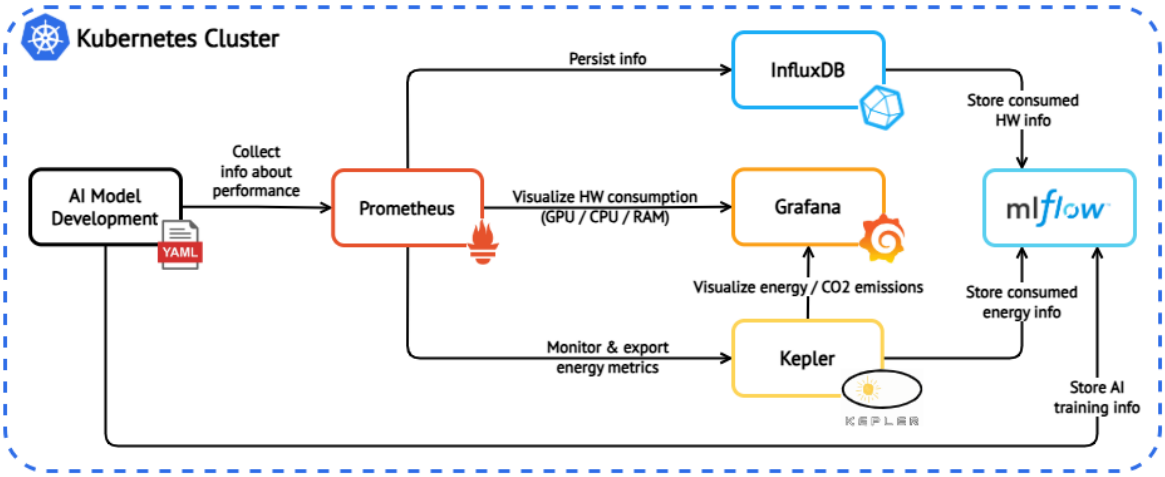


Figure 6. Technology stack for Benchmarking the AI Theoretical Framework (Theodorou et al, [42]).

The Kubernetes cluster supports E2C applications and provides scalability, easy integration of edge devices, and dynamic resource allocation. We study the case of Big Data AI pipelines that require dynamic resource adaptation, as the training and inference phases may experience random fluctuations. In addition, containerization within Kubernetes ensures process isolation and minimizes interference from other processes (e.g., from the operating system or neighbouring applications), leading to more accurate and reproducible measurements of energy and computing resource consumption. Moreover, Kubernetes' native tools for monitoring, such as Prometheus and Grafana, allow for real-time tracking and visualization of hardware metrics (CPU, GPU, RAM) and energy usage of a given AI pipeline. This distributed approach enables the monitoring of federated workloads across multiple nodes, offering a more comprehensive and granular insight into the system's performance compared to traditional PID-based monitoring. Lastly, 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 the monitoring infrastructure, ensuring precise tracking of energy consumption and hardware utilization during the various steps of a Big Data AI pipeline. Each of these components has been carefully selected for its ability to address specific challenges in resource monitoring. In the following, we provide a brief

overview of the tools employed, offering necessary background for those unfamiliar with them, and explain their role within the overall architecture.

In our quest to develop a comprehensive AI framework with a focus on measuring hardware consumption, we have leveraged a suite of cutting-edge technology tools as shown in Figure 6. Among these tools are Prometheus, Grafana, InfluxDB, and Kepler, each playing a pivotal role in enabling end-to-end energy and HW consumption tracking.

Prometheus [34], integrated into the Kubernetes cluster, serves as the primary monitoring tool to collect real-time hardware utilization data. It tracks vital system metrics, including CPU, GPU, and RAM usage, and provides a highly reliable and scalable method for capturing these metrics. By integrating with the Kubernetes environment, Prometheus ensures continuous monitoring, accurately recording resource fluctuations over time. Moreover, Prometheus gathers a wide variety of statistics on top of the application it monitors, providing high flexibility for any use case-specific monitoring requirements, such as information on disk throughput, filesystem I/O, and out-of-memory (OOM) errors.

Once the hardware utilization data is captured by Prometheus, it is stored in InfluxDB [36], a high-performance time-series database chosen for its ability to persist and handle large volumes of time-stamped data for long periods. InfluxDB ensures long-term data storage and facilitates efficient querying and analysis of historical hardware consumption trends. This long-term storage is crucial for monitoring the energy consumption patterns of AI applications over extended periods, particularly in Big Data scenarios where these patterns evolve.

For visualization, Grafana [37] is employed to create intuitive and interactive dashboards. We make use of two purposefully developed Grafana dashboard templates [38] for visualization of HW consumption and [39] for visualization of energy and CO2 emissions, that enables Grafana to connect and pull data from Prometheus. This visualization capability allows developers and researchers to gain real-time insights into the AI application's resource usage, enabling them to monitor trends and make informed decisions regarding optimizations, as well as optimal resource allocation and scheduling.

Kepler [40] extends the capabilities of Prometheus by leveraging system telemetry data to compute energy consumption and carbon emissions. Kepler integrates seamlessly with Prometheus, offering real-time insights into the environmental impact of the AI application, including power usage and CO2 emissions. This promotes sustainable computing practices, offering both technical and environmental metrics that are critical for optimizing energy consumption in containerized AI workloads.

Finally, the AI models, their performance efficiency, and any associated metadata are stored and managed using MLFlow [41]. MLFlow enables reproducibility by tracking experiment runs, saving model versions, and maintaining all relevant information for future reference. This ensures that model performance, energy consumption, and hardware usage can be monitored and compared across different experiments, providing a complete lifecycle management system for AI development through the proposed framework.

By harnessing the collective power of Prometheus, Grafana, InfluxDB, and Kepler, 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.

```
gtheodorou@ubitech:~$ kubectl get all -n kubeflow
NAME                                                        READY   STATUS    RESTARTS   AGE
pod/alertmanager-prometheus-kube-prometheus-alertmanager-0  2/2     Running   0          80m
pod/influxdb-0                                              1/1     Running   0          79m
pod/influxdb-deployment-kepler-c57ff7db8-z27cs              1/1     Running   0          78m
pod/kepler-5vmt4                                            1/1     Running   0          78m
pod/mlflow-server-674d99d496-xssr9                          1/1     Running   0          73m
pod/prometheus-grafana-78dd74577f-xtx9q                     3/3     Running   0          81m
pod/prometheus-kube-prometheus-operator-9456cddb-r5xmj      1/1     Running   0          81m
pod/prometheus-kube-state-metrics-7446b747c6-txtbf          1/1     Running   0          81m
pod/prometheus-prometheus-kube-prometheus-prometheus-0      2/2     Running   0          80m
pod/prometheus-prometheus-node-exporter-wd5ps               1/1     Running   0          81m

NAME                                                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
service/alertmanager-operated                       ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP   80m
service/influxdb                                    ClusterIP   10.104.131.190   <none>        8086/TCP,8088/TCP            79m
service/influxdb-kepler                             NodePort    10.104.0.184     <none>        8086:30543/TCP               78m
service/kepler                                      ClusterIP   10.107.44.138    <none>        9102/TCP                     78m
service/mlflow-server                               ClusterIP   10.107.120.152   <none>        5000/TCP                     73m
service/prometheus-grafana                          ClusterIP   10.109.65.218    <none>        80/TCP                       81m
service/prometheus-kube-prometheus-alertmanager     ClusterIP   10.110.105.228   <none>        9093/TCP,8080/TCP            81m
service/prometheus-kube-prometheus-operator         ClusterIP   10.98.57.178     <none>        443/TCP                      81m
service/prometheus-kube-prometheus-prometheus       ClusterIP   10.104.96.43     <none>        9090/TCP,8080/TCP            81m
service/prometheus-kube-state-metrics               ClusterIP   10.100.114.152   <none>        8080/TCP                     81m
service/prometheus-operated                         ClusterIP   None             <none>        9090/TCP                     80m
service/prometheus-prometheus-node-exporter         ClusterIP   10.98.75.219     <none>        9100/TCP                     81m

NAME                                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/kepler                               1         1         1       1            1           kubernetes.io/os=linux   78m
daemonset.apps/prometheus-prometheus-node-exporter  1         1         1       1            1           kubernetes.io/os=linux   81m

NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/influxdb-deployment-kepler          1/1     1            1           78m
deployment.apps/mlflow-server                       1/1     1            1           73m
deployment.apps/prometheus-grafana                  1/1     1            1           81m
deployment.apps/prometheus-kube-prometheus-operator 1/1     1            1           81m
deployment.apps/prometheus-kube-state-metrics       1/1     1            1           81m

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/influxdb-deployment-kepler-c57ff7db8  1       1         1       78m
replicaset.apps/mlflow-server-674d99d496            1         1         1       73m
replicaset.apps/prometheus-grafana-78dd74577f       1         1         1       81m
replicaset.apps/prometheus-kube-prometheus-operator-9456cddb  1  1      1       81m
replicaset.apps/prometheus-kube-state-metrics-7446b747c6  1       1         1       81m

NAME                                                READY   AGE
statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager  1/1  80m
statefulset.apps/influxdb                           1/1     79m
statefulset.apps/prometheus-prometheus-kube-prometheus-prometheus  1/1  80m
```

Figure 7. Kubernetes Namespace Orchestrating AI Pipeline Energy and Hardware Consumption Monitoring.

4.2 AI Training Script

The solution framework for monitoring the consumption of energy and hardware is model agnostic. This means that it can be applied irrespective of the AI model that the user would like to measure. This very important attribute comes as an added benefit of encapsulating our AI model inside a Kubernetes pod and measuring the pod's footprint. Additionally, the developed framework can be utilized either on the edge or in the cloud depending on the user requirements and the available hardware for training.

The framework does not make any hard requirements on the AI training pipeline script per se. Therefore, the user is enabled to make use of the very rich AI/ML frameworks available for developing AI models such as PyTorch, TensorFlow, Scikit-Learn etc. This is one of the most crucial components of the framework developed as it allows for very high flexibility on the AI Engineer's side. 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 > ...

```python
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 8. Example AI Training script to be monitored with the use of the framework.

4.3 Kubernetes Deployment Configuration

After setting up the whole Kubernetes namespace, with the frameworks as explained above, the next step is to monitor an AI model training and track its energy and hardware consumption. To accomplish this, we have streamlined the process by generating and uploading a YAML file, Figure 8, that 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 orchestrating the entire pipeline seamlessly and then monitoring its 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 the script, we abstract the complexity of pipeline creation, ensuring consistency, reproducibility, and ease of maintenance across different environments and deployments.

The YAML file, as depicted in Figure 9, contains detailed specifications for each component of the pipeline, including GPU resource definition, mlflow port connection memory resource limitation, and restarting policy definition. Additionally, the way that the solution is set-up, it loads the docker image containing the AI training specification from the Docker Hub if it cannot be found locally. This streamlined approach to pipeline development and automation enhances efficiency, reduces manual intervention, and accelerates the hardware and energy 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 9. YAML File with Instructions for Deploying the AI Pipeline to monitor.

4.4 Energy Consumption Visualization

As previously mentioned, our framework utilizes Kepler to track the energy consumed during the AI model development. Once the energy has been logged from Kepler to InfluxDB, it can be visualized for better understanding of the AI energy efficiency.

There are two ways that this can be achieved. On the one hand, the energy consumed can be readily viewed from within the Influx DB UI as per Figure 11. The end user can choose the specific AI training that he is interested in viewing as well as the timeframe (start-end) to see the corresponding results.

On the other hand, the energy consumed can be visualized from Grafana, as per Figure 10. Grafana can display the energy consumed broken down per device. PKG refers to the energy from CPU, DRAM refers to the energy from the RAM, GPU refers to the energy from the GPU, OTHER refers to any other device consuming energy. Another important visualization offered by Grafana is the break-down of the energy consumed to CO2 emissions Coal, Petroleum and Natural Gas. This allows the user to better understand the energy footprint of the AI training.

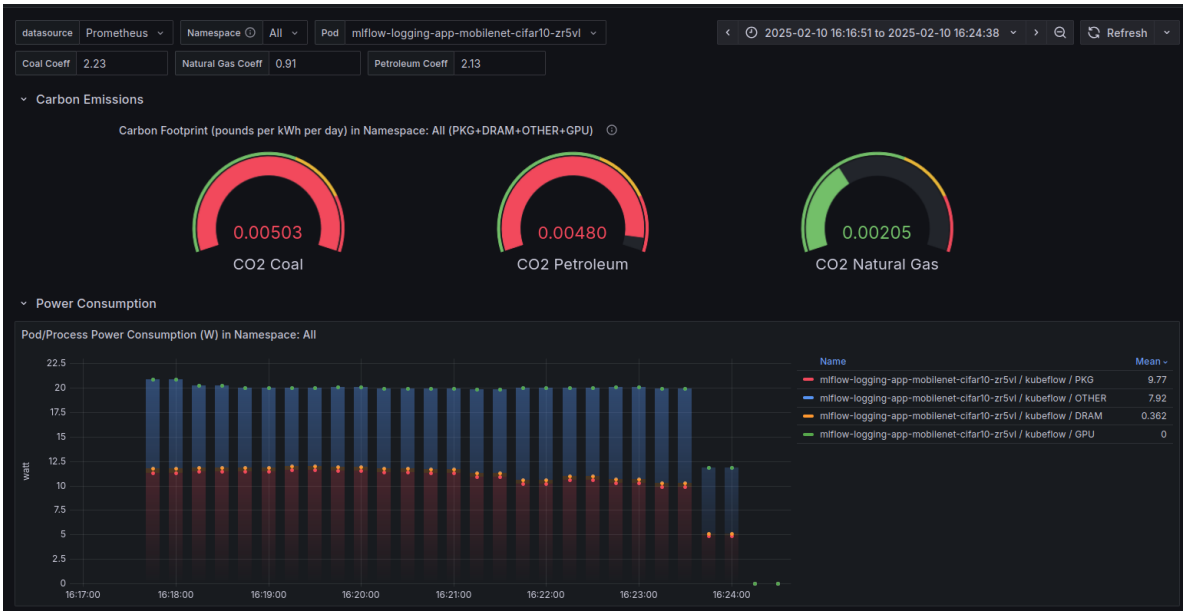


Figure 10. Grafana showing the energy consumed by the AI training.

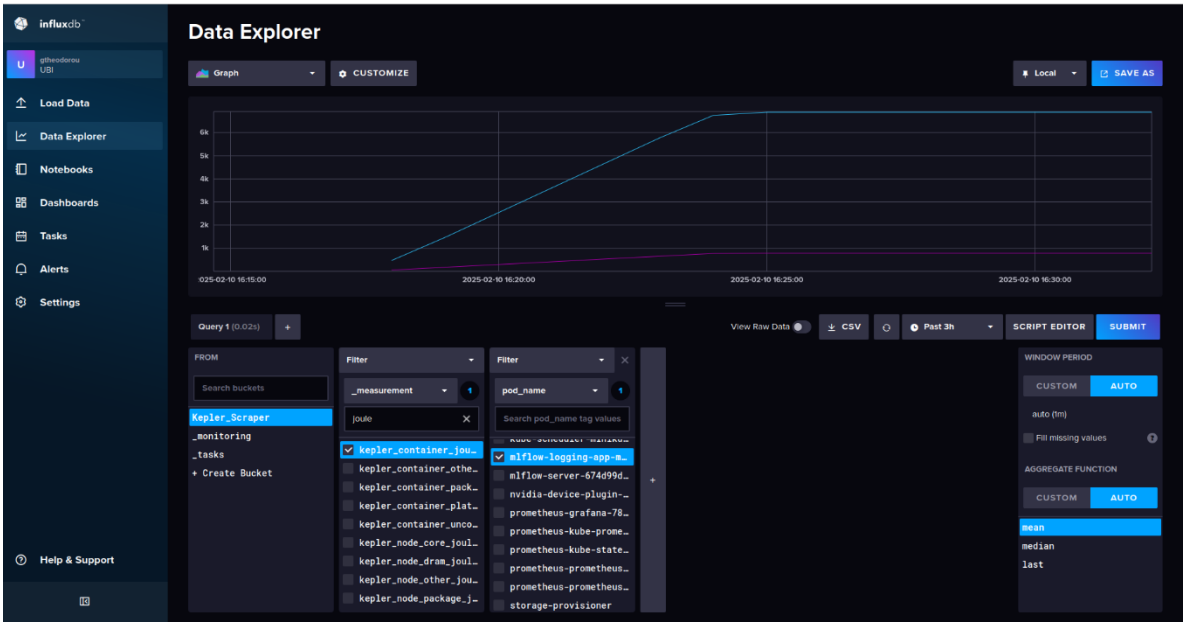


Figure 11. The InfluxDB UI showing the amount a linear graph of the total energy consumption.

4.5 Hardware Consumption Visualization

During the execution of the AI pipeline, in addition to monitoring hardware consumption using tools like Prometheus, we leverage Grafana to visualize real-time metrics. This dynamic visualization provides us with a visually appealing dashboard that offers insights into the ongoing activities and resource utilization within our system. An illustrative example of this real-time monitoring can be observed in Figure 12, where we gain visibility into the CPU and RAM usage, particularly during the execution of the most resource-intensive step of the pipeline: ML Model Training.

In Figure 12, the Grafana dashboard showcases a graphical representation of 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 visualization empowers 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 utilization in real-time, identify potential performance bottlenecks, and optimize resource allocation strategies to maximize pipeline efficiency and reliability.

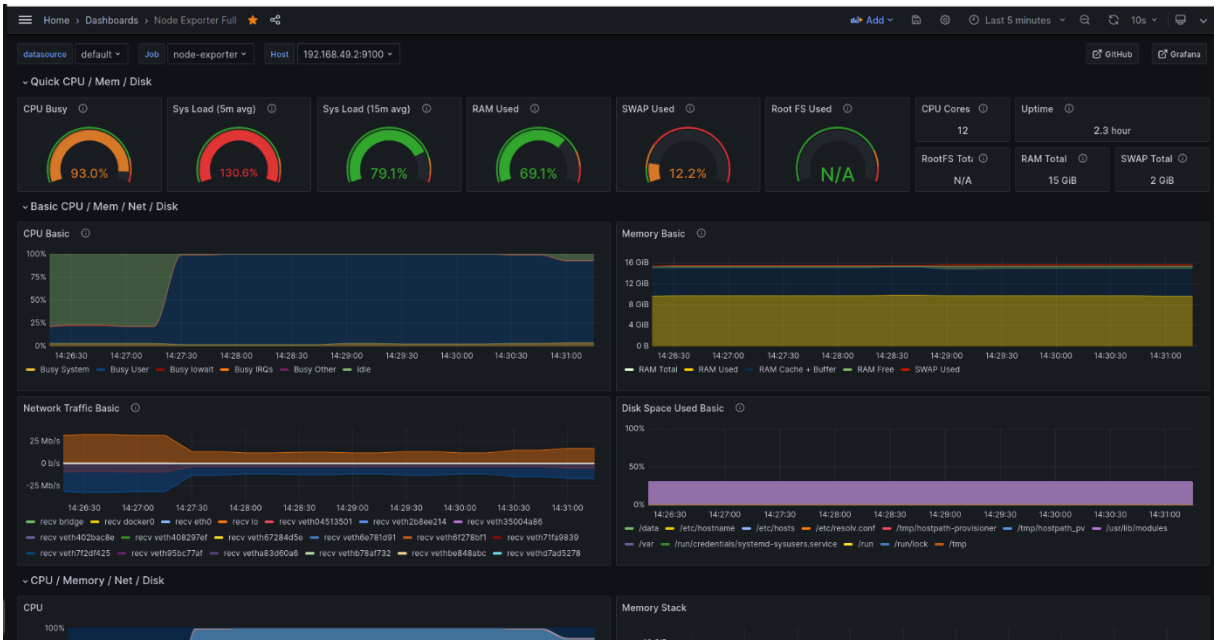


Figure 12. Grafana Dashboard Displaying HW Consumption of the AI Pipeline.

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 us with a comprehensive dataset that can be leveraged for further analysis, performance optimization, and future reference. Figure 13 exemplifies how these logs are structured, showcasing a specific pod's total CPU usage in seconds as an illustrative example.

In Figure 13, the logs retrieved from InfluxDB offer insights into the CPU utilization 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.

Moreover, the logged HW consumption metrics serve as a valuable resource for conducting retrospective analyses, diagnosing performance issues, and benchmarking system performance over time. The availability of historical HW consumption 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 that empowers us to capture, analyse, and leverage HW consumption metrics effectively. This logging infrastructure not only facilitates real-time monitoring but also enables us to derive actionable insights, optimize resource utilization, and ensure optimal performance of our AI pipeline and Kubernetes cluster.

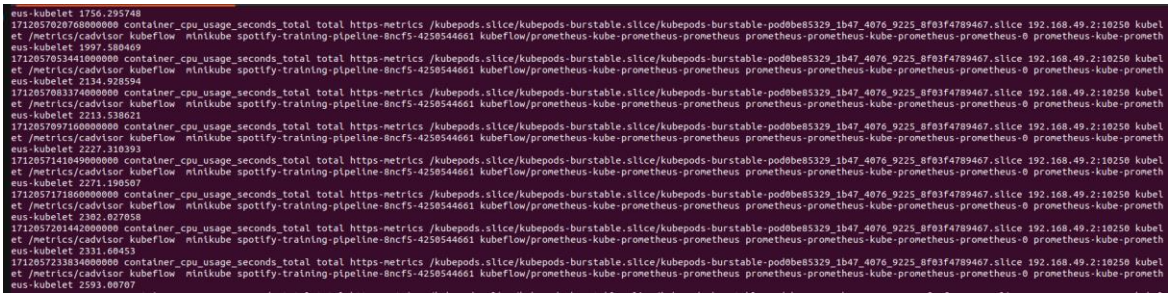


Figure 13. InfluxDB Logged Information from Prometheus Node Exporter.

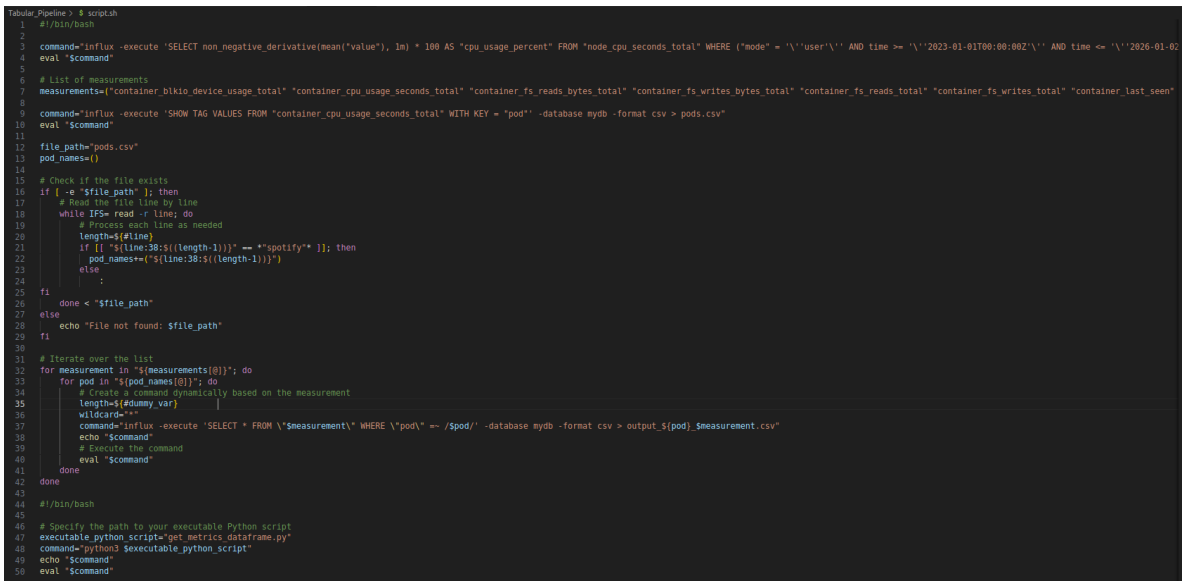


Figure 14. Bash Script For Retrieving Logs from InfluxDB.

Upon the completion of the AI pipeline's execution, the next step involves retrieving the stored logs from InfluxDB to facilitate further analyses and insights. To streamline this process, we have developed a custom bash script, Figure 14, 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.

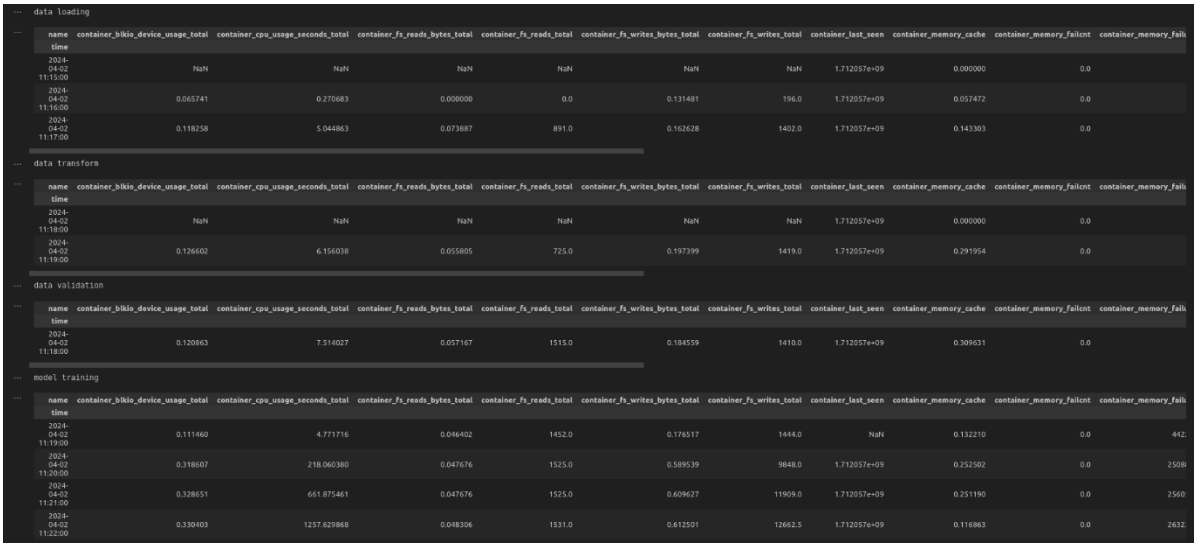
The bash script executes a series of steps, starting with the retrieval of logs from InfluxDB based on specified criteria and time ranges. Next, it processes the retrieved logs, applying data transformation

and filtering techniques to extract key metrics related to HW consumption, such as CPU usage, memory utilization, disk I/O, and network traffic. This processing step is crucial as it allows us to focus on relevant information that is essential for performance analysis, resource optimization, and system monitoring.

Once the logs are processed, the bash script compiles the extracted data into a structured CSV file format, ensuring readability, organization, 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.

Additionally, during the processing stage, we carefully select and prioritize the information to be included in the CSV file, considering 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 the 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.



data loading

name time	container_blkio_device_usage_total	container_cpu_usage_seconds_total	container_fs_reads_bytes_total	container_fs_reads_total	container_fs_writes_bytes_total	container_fs_writes_total	container_last_seen	container_memory_cache	container_memory_failcnt	container_memory_fail
2024-04-02 11:15:00	NaN	NaN	NaN	NaN	NaN	NaN	1.712057e+09	0.000000	0.0	
2024-04-02 11:16:00	0.065741	0.270683	0.000000	0.0	0.131481	196.0	1.712057e+09	0.057472	0.0	
2024-04-02 11:17:00	0.118258	5.044863	0.073887	891.0	0.162628	1402.0	1.712057e+09	0.143303	0.0	

data transform

name time	container_blkio_device_usage_total	container_cpu_usage_seconds_total	container_fs_reads_bytes_total	container_fs_reads_total	container_fs_writes_bytes_total	container_fs_writes_total	container_last_seen	container_memory_cache	container_memory_failcnt	container_memory_fail
2024-04-02 11:18:00	NaN	NaN	NaN	NaN	NaN	NaN	1.712057e+09	0.000000	0.0	
2024-04-02 11:19:00	0.126602	6.156038	0.055805	725.0	0.197399	1419.0	1.712057e+09	0.291954	0.0	

data validation

name time	container_blkio_device_usage_total	container_cpu_usage_seconds_total	container_fs_reads_bytes_total	container_fs_reads_total	container_fs_writes_bytes_total	container_fs_writes_total	container_last_seen	container_memory_cache	container_memory_failcnt	container_memory_fail
2024-04-02 11:18:00	0.120863	7.514027	0.057167	1515.0	0.184559	1410.0	1.712057e+09	0.309631	0.0	

model training

name time	container_blkio_device_usage_total	container_cpu_usage_seconds_total	container_fs_reads_bytes_total	container_fs_reads_total	container_fs_writes_bytes_total	container_fs_writes_total	container_last_seen	container_memory_cache	container_memory_failcnt	container_memory_fail
2024-04-02 11:19:00	0.111460	4.771716	0.046402	1452.0	0.176517	1444.0	NaN	0.132210	0.0	442
2024-04-02 11:20:00	0.318607	218.060380	0.047676	1525.0	0.589539	9848.0	1.712057e+09	0.252502	0.0	2508
2024-04-02 11:21:00	0.328651	661.875461	0.047676	1525.0	0.609627	11909.0	1.712057e+09	0.251190	0.0	2560
2024-04-02 11:22:00	0.330403	1257.629868	0.048306	1531.0	0.612591	12662.5	1.712057e+09	0.116863	0.0	2632

Figure 15. Various HW Consumption Metrics as Logged During the AI Pipeline Execution.

Figure 15 provides a detailed and granular view of the HW metrics captured during each step of the AI pipeline execution. This visualization offers a comprehensive overview of key HW consumption parameters that we consider crucial and monitor closely throughout the pipeline's lifecycle. By capturing 1-minute granularity for each step, we gain insights into real-time performance trends, resource utilization patterns, and system behaviour during the pipeline's execution.

For each step of the AI pipeline depicted in Figure 15, the graph showcases essential HW metrics such as CPU usage, memory utilization, disk I/O, and network traffic. These metrics are tracked at a fine-grained level, enabling us to monitor resource consumption trends, detect anomalies, and optimize resource allocation strategies based on actual execution data.

The detailed information presented in Figure 15 allows us to draw several insights and observations from the AI pipeline's execution. For example, 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. Additionally, the granularity of the data enables us to conduct root cause analysis, diagnose performance bottlenecks, and make data-driven decisions to enhance system performance and efficiency.

Overall, 4 serves as a valuable tool for visualizing 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.

Benchmarking AI Theoretical Framework with TALON Pilots

5.1 Benchmarking with UC1: Automatic UATVs Coordination and UC4: Human-Robot Collaboration (images)

As explained, the developed AI Theoretical Framework for energy and hardware monitoring can retrieve the AI model's footprint irrespective of the modality. Hence, when using the framework to benchmark Use Cases 1 & 4 where the AI modality is images, we have trained, monitored and benchmarked 8 state-of-the-art models that have been deemed as superior in the respective task.

In the first use case, where the objective is to detect fires using a drone and promptly notify responders, it is crucial to distinguish between the training and deployment environments of the AI model. Given the hardware limitations of drones, deploying computationally heavy models that achieve extremely high accuracy at the expense of energy efficiency is not feasible. Therefore, it is essential to develop an AI model optimized for edge deployment, prioritizing energy efficiency as the primary criterion when selecting a suitable model for this use case.

In the fourth use case, where the objective is to detect whether factory workers comply with safety regulations by wearing helmets and vests, it is equally important—just as in the first use case—to ensure that the trained AI model is deployable on the edge. 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.

The dataset utilized consists of 60,000 images with a total size of 170 MB. These images are representative of the use cases, specifically focusing on fire detection and compliance with safety regulations (helmet detection). The models benchmarked for these tasks, along with their corresponding benchmarking results, are presented in Figure 16 and Figure 17.

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 16. Comparison of imagery data for AI models performance and 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 17. Comparison of imagery data for AI models Hardware and Energy consumption.

As shown in Figure 16, EfficientNet is the most efficient network, achieving the highest precision, accuracy, and F1 score across the four training epochs. It outperforms the second-best model, VGGNet, 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 better architecture in balancing performance and resource utilization.

Interestingly, while MobileNet completes the 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, and ResNet152, outperform those with a medium number of weights, such as ConvNext and YOLOv8. 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 20, 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, 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 utilization. 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 utilization, 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 with UC2: I5.0 Automation and Planning and UC3: AR/VR for Training and Maintenance (time-series and categorical)

For Use Cases 2 and 3, 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 Use Case 2, the objective is to minimize waste generated from a production line while optimizing the trade-off between energy efficiency and maintenance costs. This requires the analysis of time-series data to enable proactive maintenance strategies and optimize the production line's lifecycle.

In Use Case 3, the objective is to develop an AR/VR system that facilitates on-the-job training by guiding users through a questionnaire and enabling real-time coordination with a remote engineer. This system provides step-by-step instructions for learning specific tasks related to operating the Nakamura machine.

Unlike the first set of use cases, where energy efficiency was a primary constraint, Use Cases 2 and 3 prioritize achieving the highest possible accuracy to ensure reliable predictions. Since edge deployment constraints do not apply in these cases, the focus shifts from optimizing energy consumption to maximizing model performance and prediction accuracy.

The dataset utilized consists of 4.4 million records, amounting to a total size of 767 MB. 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 Figure 18 and Figure 19.

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 18. Comparison of tabular data for AI models performance and 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 19. Comparison of tabular data for AI models Hardware and Energy consumption.

As shown in Figure 18, XGBoost 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, 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 19 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 utilization. This suggests that XGBoost is optimized for resource management, maintaining high performance while minimizing 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 utilization 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 utilize 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 Big Data nature of AI pipelines, and most importantly the phase that mostly contributes to greater execution times, energy consumption, and hyperparameter tuning.

5.3 Evolutionary Computation Methods for Multi-objective Optimisations of the AI Theoretical Framework

Evolutionary Computation (EC) methods are powerful tools that help solve problems with multiple goals in AI systems. They work by exploring many possible solutions at once, which is very useful when you need to balance different, sometimes conflicting, objectives.

The main idea behind these methods is Pareto Optimality. Pareto Optimality aims to find the perfect balance between competitive goals. Instead of finding just one best solution, EC methods try to create a set of good solutions that offer different balances between the goals. For example, an AI system might need to balance accuracy and energy consumption. These methods help find a range of solutions that trade off these objectives in different ways.

Evolutionary algorithms work by mimicking natural evolution to improve solutions over time. They start with a diverse group of candidate solutions and evaluate each one using several criteria. In multi-objective scenarios, these criteria often conflict—for example, one solution might be very efficient while another might be highly accurate, as in the accuracy, energy consumption.

Next, the algorithm creates new solutions by combining features of the selected ones and making small random changes, a process similar to biological reproduction and mutation. Over many iterations (or generations), the population evolves, gradually finding solutions that offer better trade-offs among the competing objectives. In the end the evolved population will consist of best solutions that form a Pareto Front. Pareto Front shows all the best possible choices, where it has computed the most of all goals in an evolutionary way. It helps derive the trade-offs more clearly.

In essence, evolutionary algorithms offer a systematic way to explore a wide range of possible solutions, allowing decision-makers to choose from a variety of options that best meet their diverse requirements. In the TALON context, we use the data from Figure 16 and Figure 18.

To select the most optimal algorithms that have great accuracy and lower energy consumption, using EC algorithms as described above. Below are the steps of the EC algorithm:

1. Pre-Computed Metrics:

• Two dictionaries hold metrics for different models: one for tabular models and one for image models.

• Each model has values for accuracy, training time, and energy consumption, values derived from previous sections.

• There are also “low” and “high” reference points for normalization purposes. “low” refers to the min value of Accuracy (least optimal) and Energy (most optimal). “high” refers to max value of Accuracy (most optimal) and Energy (least optimal)

2. Choosing a Modality:

• The code allows you to choose whether to optimize for tabular or image models.

• Based on this choice, it selects the corresponding metrics and creates a list of available model names.

3. Setting Up the Evolutionary Environment:

• **Individuals:** Each individual in the population represents one model (its name).

• **Fitness:** The fitness is defined as a pair: one value is the negative accuracy (so that higher accuracy is better by minimizing a negative value), and the other is the energy (which we want to minimize).

• **Population:** A population is created by randomly selecting models from the chosen list.

4. Evaluation Function:

• This function calculates the fitness for each model by getting its accuracy and energy values.

• Energy is normalized so that it fits into a common scale.

• The fitness is returned as (-accuracy, energy) because we want to maximize accuracy (by minimizing its negative) and minimize energy.

5. Genetic Operators:

• **Crossover (Mating):** Two individuals may swap their chosen models with a 50% chance.

• **Mutation:** An individual can change its model to a different random model.

6. Evolutionary Loop:

• The algorithm starts with an initial population and evaluates each individual.

• Over several generations, it selects, mates, and mutates individuals to create new offspring.

• Each new generation is evaluated, and the best individuals (based on fitness) are selected for the next generation.

7. Extracting the Pareto Front:

• At the end, the code finds the Pareto-optimal set of models. This is a set where you cannot improve one criterion (like accuracy) without worsening the other (like energy).

Figure 20 depicts the definition of parameters (number of generations (NGEN), population size (MU), crossover and mutation probability (CXPB, MUTPB)), training loop and application of each step (crossover, mutation, evaluation of population, and the generation/evolution of new population.

```
random.seed(42)
pop_size = 20  # Since we have only a few models.
population = toolbox.population(n=pop_size)

NGEN = 8     # Number of generations.
MU = pop_size
LAMBDA = 20    # Number of offspring per generation.
CXPB = 0.7     # Crossover probability.
MUTPB = 0.3    # Mutation probability.

# Evaluate the initial population.
for ind in population:
    ind.fitness.values = toolbox.evaluate(ind)

print("Initial Population:")
for ind in population:
    print(ind, ind.fitness.values)

# Begin the evolutionary process.
for gen in range(1, NGEN + 1):
    offspring = toolbox.select(population, LAMBDA)
    offspring = list(map(toolbox.clone, offspring))

    # Apply crossover.
    for child1, child2 in zip(offspring[::2], offspring[1::2]):
        if random.random() < CXPB:
            toolbox.mate(child1, child2)
            del child1.fitness.values
            del child2.fitness.values

    # Apply mutation.
    for mutant in offspring:
        if random.random() < MUTPB:
            toolbox.mutate(mutant)
            del mutant.fitness.values

    # Evaluate individuals with invalid fitness.
    invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
    for ind in invalid_ind:
        ind.fitness.values = toolbox.evaluate(ind)

    # Select the next generation population.
    population = toolbox.select(population + offspring, MU)

    print(f"\nGeneration {gen} Fitnesses:")
    for ind in population:
        print(ind, ind.fitness.values)

return population
```

Figure 20. Code snippet of the overall procedure of the evolutionary computation algorithm.

Figure 21 illustrates the implementation of crossover, selection and mutation for the TALON context. More specifically, since the data are in the form of dictionary (with key being the name of AI algorithm in the benchmark) the crossover is simply to change names of individuals. The mutation is to select an algorithm that does not exist in the current population randomly. The selection process is the NSGA-II that is supported by DEAP.

```
# ============================================================
#  Genetic Operators (Discrete Search Space)
#  crossover -> cx_model
#  mutation -> mut_model
# ============================================================
def cx_model(ind1, ind2):
    if random.random() < 0.5:
        # Swap model names between individuals.
        ind1[0], ind2[0] = ind2[0], ind1[0]
    return ind1, ind2

toolbox.register("mate", cx_model)

def mut_model(individual):
    current_model = individual[0]
    # Mutate by randomly selecting a different model.
    ne Loading... random.choice([m for m in models_list if m != current_model])
    individual[0] = new_model
    return (individual,)

toolbox.register("mutate", mut_model)

# Use NSGA-II for selection.
toolbox.register("select", tools.selNSGA2)
```

Figure 21. Code snippet of the implementation of crossover, selection and mutation for TALON context.

Figure 22 presents the Pareto Front of the current data, for image modality for the objectives of accuracy, and normalized energy consumption (0-100 range). The closer we are in the top left corner the best solution for high-accuracy and low energy consumption. In the plot, dominant solution for the multi-objective of high-accuracy and low energy consumption is the EfficientNet, as well as MobileNet algorithms, that the evolutionary algorithm also selects.

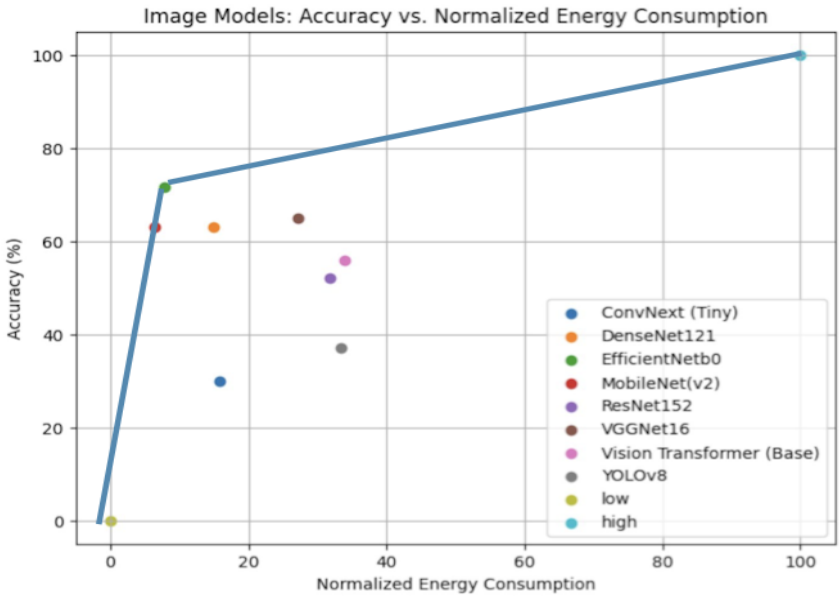


Figure 22. Pareto front of Image Modality for accuracy and normalized energy consumption.

Figure 2323 illustrates the Pareto Front of the current data, for tabular modality for the objectives of accuracy, and normalized energy consumption (0-100 range). In the plot, dominant solution for the multi-objective of high-accuracy and low energy consumption is the XGBoost algorithm, that the evolutionary algorithm also selects.

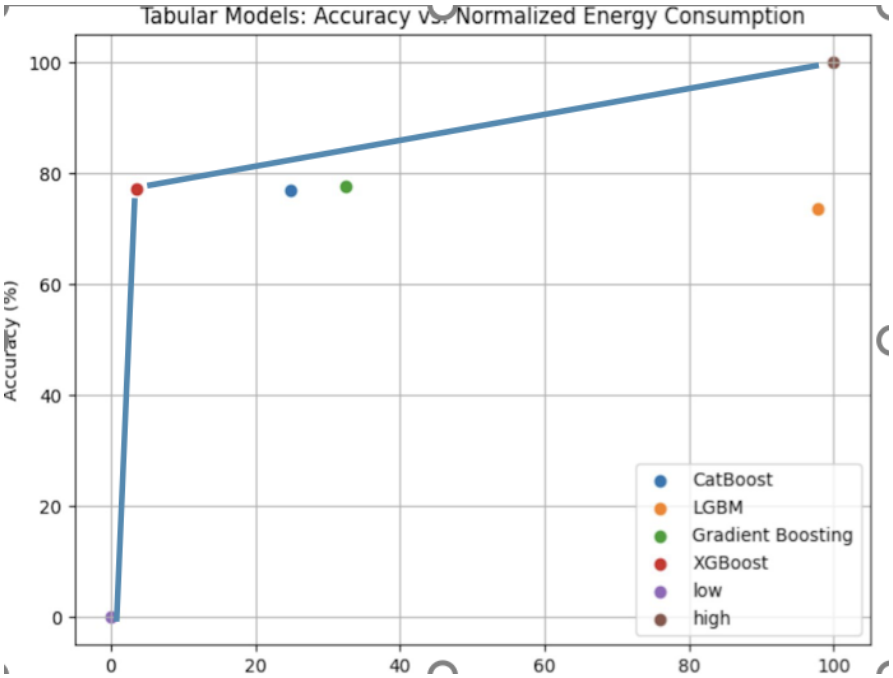


Figure 23. Pareto front of Tabular Modality for accuracy and normalized energy consumption.

Conclusion and Future Outlook

The proliferation of AI applications in E2C computing environments has led to a growing demand for efficient and scalable execution. However, understanding the behavior and performance of AI algorithms in these dynamic contexts remains a significant research challenge, requiring empirical modelling, tuning, and optimizing their performance. The proposed theoretical and software frameworks efficiently address both performance and energy preservation of Big Data AI pipelines, providing a detailed analysis of optimal model selection and placement within E2C environments.

In summary, this overview has delved into the realm of eXplainable Artificial Intelligence, emphasizing its critical role in facilitating the adoption of machine learning methods in practical, real-world applications. By clarifying various concepts related to model explainability and exploring the motivations behind the pursuit of more interpretable ML techniques, we have laid a foundation for a systematic review of recent literature on XAI.

Our examination has categorized XAI approaches into two main perspectives: transparent ML models and post-hoc techniques designed to enhance interpretability. This classification has resulted in the development of comprehensive taxonomies that group different methodologies under unified criteria. Particularly, our exploration of Deep Learning models has led to an alternative taxonomy tailored to the specific domains where explainability is crucial for such models.

Moreover, we have extended our discussions beyond traditional XAI boundaries to address the concept of Responsible AI, which emphasizes principles such as fairness, transparency, and privacy in AI implementation. The implications of XAI in data fusion and fairness have been thoroughly examined, highlighting both its potential and challenges.

The proposed theoretical framework, alongside its associated benchmarking methods, effectively addresses the need for enhanced performance optimization and energy preservation in AI pipelines. By furnishing comprehensive analyses of optimal model selection and strategic placement within E2C environments, it affords organizations the opportunity to achieve maximum efficiency while concurrently minimizing compute resource allocation. Moreover, this research elucidates the notions of model explainability and examines the incentives underpinning interpretable ML methodologies, thereby fostering higher transparency and user confidence. Through the systematic categorization of XAI approaches and the development of exhaustive taxonomies, we establish a structured framework for the comprehension and implementation of explainable models, including those integrated within deep learning architectures.

Looking ahead, promising directions for future research include focusing on prototype-based models, which provide a highly explainable form of representation, especially when combined with deep architectures. Key open research questions include determining optimal network architectures, feature extraction methods, distance metrics, optimization techniques, and identifying the most suitable set of prototypes for data representation in prototype-based methods. This research direction will drive innovation in creating more understandable and trustworthy AI systems.

Ultimately, our vision underscores the importance of addressing model interpretability alongside considerations of data privacy, model confidentiality, fairness, and accountability for the responsible implementation and utilization of AI methods across various organizations and institutions. We extend beyond traditional XAI to address Responsible AI, emphasizing fairness, privacy via federation, and transparency. By focusing on learning with privacy constraints, model confidentiality, fairness, and accountability, we pave the way for responsible AI systems deployment for a wide array of applications and execution contexts.

References

[1] Mark A. Neerincx, Jasper van der Waa, Frank Kaptein, and Jurriaan van Diggelen. 2018. Using perceptual and cognitive explanations for enhanced human-agent team performance. In Engineering Psychology and Cognitive Ergonomics, pages 204–214, Cham. Springer International Publishing.

[2] Matteo, Rizzo., Alberto, Veneri., Andrea, Albarelli., Claudio, Lucchese., Marco, S., Nobile., Cristina, Conati. (2022). A Theoretical Framework for AI Models Explainability with Application in Biomedicine.

[3] (2023). The AI Digital Revolution in Innovation: A Conceptual Framework of Artificial Intelligence Technologies for the Management of Innovation. IEEE Transactions on Engineering Management.

[4] Tomasz, Wójtowicz. (2023). Towards a Comprehensive Theory of Aligned Emergence in AI Systems: Navigating Complexity towards Coherence.

[5] (2022). A Theoretical Framework for AI Models Explainability with Application in Biomedicine.

[6] David, R., Gibson., Vitomir, Kovanović., D., Ifenthaler., Sara, Dexter., Shihui, Feng. (2023). Learning theories for artificial intelligence promoting learning processes. British Journal of Educational Technology.

[7] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521, 436–444

[8] Pasquale, F. (2015). The black box society. Harvard University Press.

[9] (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

[10] 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.

[11] Emmert-Streib, F., Yli-Harja, O., & Dehmer, M. (2020). Explainable artificial intelligence and machine learning: A reality rooted perspective. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(6), e1368.

[12] 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).

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

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

[15] 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.

[16] 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.

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

[18] 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).

[19] 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).

[20] 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).

[21] 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)

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Freund, L. (2023). Towards a comprehensive theory of aligned emergence in ai systems: Navigating complexity towards coherence.

[31] 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.

[32] 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.

[33] 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.

[34] Prometheus. Available at: https://prometheus.io/docs/introduction/overview/

[35] Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, omega, and kubernetes. Communications of the ACM, 59(5), 50-57.

[36] InfluxDB. Available at: https://www.influxdata.com/index/

[37] Grafana. Available at: https://grafana.com/

[38] Node Exporter. Available at: https://grafana.com/grafana/dashboards/1860-node-exporter-full/

[39] Kepler. Available at: https://github.com/sustainable-computing-io/kepler/blob/main/grafana-dashboards/Kepler-Exporter.json

[40] Amaral, M., Chen, H., Chiba, T., Nakazawa, R., Choochotkaew, S., Lee, E. K., & Eilam, T. (2023, July). Kepler: A framework to calculate the energy consumption of containerized applications. In 2023 IEEE 16th International Conference on Cloud Computing (CLOUD) (pp. 69-71). IEEE.

[41] Mlflow. Available at: https://mlflow.org/

[42] 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.

[43]

Funded by
the European Union

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