

# EcoMLS: A Self-Adaptation Approach for Architecting Green ML-Enabled Systems

Meghana Tedla

Software Engineering Research Center  
IIIT Hyderabad, India  
meghana.tedla@students.iiit.ac.in

Shubham Kulkarni

Software Engineering Research Center  
IIIT Hyderabad, India  
shubham.kulkarni@research.iiit.ac.in

Karthik Vaidhyanathan

Software Engineering Research Center  
IIIT Hyderabad, India  
karthik.vaidhyanathan@iiit.ac.in

**Abstract**—The sustainability of Machine Learning-Enabled Systems (MLS), particularly with regard to energy efficiency, is an important challenge in their development and deployment. Self-adaptation techniques, recognized for their potential in energy savings within software systems, have yet to be extensively explored in Machine Learning-Enabled Systems (MLS), where runtime uncertainties can significantly impact model performance and energy consumption. This variability, alongside the fluctuating energy demands of ML models during operation, necessitates a dynamic approach. Addressing these challenges, we introduce EcoMLS approach, which leverages the Machine Learning Model Balancer concept to enhance the sustainability of MLS through runtime ML model switching. By adapting to monitored runtime conditions, EcoMLS optimally balances energy consumption with model confidence, demonstrating a significant advancement towards sustainable, energy-efficient machine learning solutions. Through an object detection exemplar, we illustrate the application of EcoMLS, showcasing its ability to reduce energy consumption while maintaining high model accuracy throughout its use. This research underscores the feasibility of enhancing MLS sustainability through intelligent runtime adaptations, contributing a valuable perspective to the ongoing discourse on energy-efficient machine learning.

**Index Terms**—Self-Adaptation, Machine Learning-Enabled Systems, Sustainability, Green Software, Energy Efficiency, Machine Learning, Software Engineering

## I. INTRODUCTION

The growth of Artificial Intelligence (AI) has led to significant advancements as well as environmental concerns. Machine Learning-Enabled Systems (MLS), important for developing technologies such as autonomous vehicles and smart cities, consume considerable energy and generate high carbon emissions [1], [2]. Strubell et al. (2019) highlight the substantial energy required to train ML models<sup>1</sup>, equating the carbon footprint of training one model to that of five average cars [3]. The drive for more accurate models increases both energy use and emissions [4]. Therefore, adopting sustainable AI practices is vital to balance technological advancement with environmental preservation [5]–[7].

Despite awareness of the environmental impact of training and deploying ML models, the energy efficiency of model inference—particularly in terms of its significant energy consumption and its implications for Quality of Service (QoS)—has been less addressed. Green AI initiatives have

primarily focused on optimizing the training phase, with less attention given to the energy demands of inference in practical applications [8]–[11]. This gap highlights the need for strategies that reduce energy consumption without compromising performance and can adjust to varying operational demands. The potential of self-adaptation techniques, which balance energy efficiency with QoS, remains largely unexplored in this context [12]. As the ICT sector’s energy consumption is expected to increase, creating adaptive, energy-efficient MLS is paramount [13]. Our work seeks to bridge this gap, proposing a self-adaptive approach aiming to ensure MLS sustainability amidst environmental uncertainties.

Initial studies [14] explored how ML systems adapt to environmental shifts. Our further research [15] introduced the Machine Learning Model Balancer, which dynamically switches between ML models to ensure optimal QoS, yet overlooked energy efficiency—a critical aspect given ML deployments’ increasing energy demands and environmental impact. To bridge this gap, we extend the Machine Learning Model Balancer concept to include energy efficiency, aiming to create a more sustainable, adaptive MLS and introduce EcoMLS. This novel self-adaptive approach extends the Machine Learning Model Balancer concept, incorporating energy efficiency to tackle unexplored runtime uncertainties and the impact of dynamic environmental factors on model performance. EcoMLS distinguishes itself by dynamically adjusting energy consumption and model confidence through ML model switching, responding in real-time to operational conditions and request variability. Through the MAPE-K loop, it emphasizes energy efficiency alongside accuracy, involving: i) monitoring model and system energy parameters; ii) analyzing energy performance to pinpoint inefficiencies; iii) switching models based on energy-accuracy evaluations; iv) implementing strategies for sustainable MLS. EcoMLS emerges as a novel solution in self-adaptive MLS, optimizing energy consumption while aiming to maintain performance. Our case study on object detection with EcoMLS uses the underlying ML system of our SWITCH exemplar, designed for evaluating self-adapting ML-enabled systems. This practical application confirms EcoMLS’s ability to effectively manage the trade-off between energy efficiency and accuracy, showcasing its capability to be applied across a diverse set of ML models. The flexibility of EcoMLS allows it to navigate the trade-offs

<sup>1</sup>unless specified otherwise by model we imply ML model in this paper

between maximizing confidence scores with minimal energy increases and significantly reducing energy consumption with minimal impact on model confidence. The results show that EcoMLS outperforms both naive strategies and individual models in terms of energy consumption and confidence score trade-off, demonstrating the practicality of embedding energy efficiency within self-adaptive systems. This moves us towards a future where MLS can easily meet different needs, making them sustainable for various uses.

The remainder of the paper is structured as follows: Section II provides running example. Section III introduces the EcoMLS approach. Experimentation and results from its application are in Section IV. Threats to validity and Related work are discussed in Section V and VI respectively. Section VII concludes and discusses future work.

## II. RUNNING EXAMPLE

Our approach EcoMLS is exemplified through the adoption of the SWITCH [16] exemplar’s managed system and environment manager, an object detection web service. This service integrates a streamlined architecture for handling image processing requests: an *Image Ingestion Service* for emulating real-world asynchronous requests; an *Image Store* acting as a dynamic FIFO queue for incoming data; a *Data Preprocessor* preparing images for detection; a *Model Loader* for dynamic selection of machine learning models governed by underlying approach; a *Model Repository* housing a variety of preloaded models for quick deployment; an *ML Model* at the core of detection processing; a *Post Processor* refining detection results; and a *Result Storage* for temporary data retention before final transfer. This setup as shown in Figure 1 mirrors services akin to Google Cloud Vision or Amazon Rekognition, underscoring the practical applicability and flexibility in managing object detection tasks with efficiency and scalability.

In this system, we define a set of machine learning models  $M$ , where each model  $m_j$  in  $M$  represents a different configuration of the YOLO algorithm [17], varying in size and computational efficiency. These models are evaluated on two critical metrics: energy consumption, reflecting the electricity used during inference, and the confidence score, which measures the probability of accurate object identification and classification, serving as an indicator for model accuracy. Here,  $M$  includes the YOLOv5 models: YOLOv5n (nano), YOLOv5s (small), YOLOv5m (Medium), and YOLOv5l (Large), provided by Ultralytics [18], each pretrained on the COCO 2017 training dataset [19]. The differentiation among these models primarily lies in their number of parameters, influencing both their energy consumption ( $E_j$ ) and detection confidence score ( $c_j$ ), where  $j$  indexes the model within  $M$ .

The variants within  $M$ —ranging from YOLOv5n to YOLOv5l—exhibit a trade-off between  $c_j$  and  $E_j$ . This trade-off highlights the balance required in practical applications between model accuracy, indicated by the confidence score, and operational efficiency, as represented by energy consumption. Specifically, YOLOv5n, optimized for efficiency, consumes 2 mJ of energy ( $E_1$ ) with a mAP of 45.7 ( $c_1$ ), while YOLOv5l,

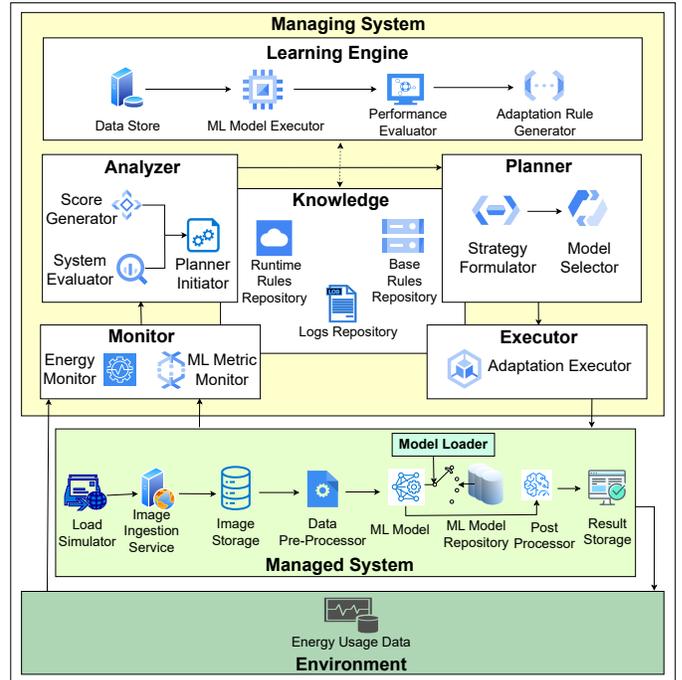


Fig. 1. Architecture of EcoMLS

aimed at higher accuracy, consumes 16 mJ ( $E_n$ ) achieving a mAP of 68.9 ( $c_n$ ). This delineation showcases the balance between computational demand and detection accuracy.

In operational terms, the system processes a stream of image requests  $i$ , selecting a model  $m_j$  for each  $i$ , based on optimizing the balance between energy efficiency ( $E_i$ ) and detection confidence ( $c_i$ ). Our approach, EcoMLS aims to dynamically switch between the models to maintain an effective balance between  $E_i$  and  $c_i$ , enhancing both the sustainability and the performance of the object detection service.

## III. APPROACH

Our EcoMLS approach focuses on improving machine learning-enabled systems with an eye on sustainability and performance. The first step is to explore the Learning Engine, as shown in Figure 1.

### A. Learning Engine

The Learning Engine (LE) within EcoMLS as shown in Figure 1 is structured to evaluate and adapt a diverse set of machine learning models  $M$ , to enhance both sustainability and performance within the MLS framework. This engine is built to work with a variety of models and datasets, offering a robust foundation for dynamic adaptation across different scenarios. Within the LE, the *Data Store* ( $DS$ ) serves as the repository for both the evaluation dataset and the models  $M = \{m_j\}$ . These models are assessed using the evaluation dataset to determine their performance. The *ML Model Executor* ( $MLE$ ) deploys the models against the evaluation dataset. For each model  $m_j \in M$ , it records key performance metrics across all evaluated requests in the dataset. These metrics include energy consumption ( $E_j$ ), confidence score ( $c_j$ ), and

processing time ( $t_{mp,j}$ ). Logging these metrics is essential for evaluating each model’s operational efficiency and prediction/detection accuracy.

For performance analysis, the *Performance Evaluator (PE)* constructs a performance matrix  $\mathbf{P}_j$  for each model  $m_j$ . Suppose the evaluation dataset contains  $r$  requests. In that case,  $\mathbf{P}_j$  will have  $r$  rows, with each row corresponding to a request in the dataset and columns representing the key metrics: energy consumption ( $E_j$ ), confidence score ( $c_j$ ), and processing time ( $t_{mp,j}$ ). This component enables a thorough comparison of model performance.

Building on the insights gained from each  $\mathbf{P}_j$ , the *Adaptation Rule Generator* synthesizes a set of base adaptation rules. It does so by averaging the metrics in  $\mathbf{P}_j$  for each model  $m_j$ , resulting in aggregated values of minimum energy ( $E_{min,j}$ ), maximum energy ( $E_{max,j}$ ), and average confidence score ( $C_{avg,j}$ ) across all requests. These aggregated values are then represented in a matrix  $\mathbf{A}$ :

$$\mathbf{A} = \begin{bmatrix} E_{min,1} & E_{max,1} & C_{avg,1} \\ \vdots & \vdots & \vdots \\ E_{min,n} & E_{max,n} & C_{avg,n} \end{bmatrix}$$

In the matrix  $\mathbf{A}$ , each row corresponds to a distinct model  $m_j$  within the set  $M$ , where  $j$  ranges from 1 to  $n$ , indicating the sequence of models considered in the evaluation. The columns  $E_{min,j}$ ,  $E_{max,j}$ , and  $C_{avg,j}$  represent, respectively, the minimum energy consumption, maximum energy consumption, and average confidence score calculated for each model  $m_j$  across all requests in the evaluation dataset. The matrix  $\mathbf{A}$  gives a clear performance comparison of each model, guiding the selection based on efficiency and accuracy. In the context of our running example II,  $m_j$  refers to YOLOv5 variants [18] -YOLOv5n, YOLOv5s, YOLOv5m, and YOLOv5l, evaluated against the COCO Test 2017 [19] to benchmark performance. The requests here are images, illustrating how the LE applies to object detection use case.

## B. MAPE-K Loop

1) *Knowledge*: In the EcoMLS’s MAPE-K (Monitor, Analyze, Plan, Execute - Knowledge) framework as depicted in Figure 1, the Knowledge component ( $K$ ) integrates three key repositories essential for the system’s adaptive behavior: the *Log Repository*, the *Base Rules Repository*, and the *Runtime Rules Repository*. These repositories collectively maintain the data necessary for informed decision-making and dynamic adaptation of the system. The *Log Repository* stores real-time logs of each processed request, including energy consumption ( $E_j$ ), confidence score ( $c_j$ ), request processing time ( $t_{sys}$ ), and detection outcomes such as the number of detected bounding boxes ( $b$ ) for running example explained in Section II. This repository is continuously updated, ensuring a comprehensive record of system performance metrics. The *Base Rules Repository* is directly derived from the Learning Engine’s output, specifically referencing the matrix  $\mathbf{A}$  previously discussed. This matrix, generated for the initial evaluation of models against the evaluation dataset, establishes a baseline for energy

efficiency ( $E_{min,j}$  and  $E_{max,j}$ ) and confidence score ( $C_{avg,j}$ ) for each model  $m_j$ . It serves as the foundation for model selection and system adaptation during the initial operational phase. Post the initial model evaluations, the *Runtime Rules Repository (B)* updates real-time performance metrics for each model  $m_j$  as new requests are processed. The structure of this repository is formalized as a matrix that dynamically reflects changes in model performance and energy usage:

$$\mathbf{B} = \begin{bmatrix} E_{min,1} & E_{max,1} & E_{latest,1} & C_{avg,1} \\ E_{min,2} & E_{max,2} & E_{latest,2} & C_{avg,2} \\ \vdots & \vdots & \vdots & \vdots \\ E_{min,n} & E_{max,n} & E_{latest,n} & C_{avg,n} \end{bmatrix}$$

In the matrix  $\mathbf{B}$ , each row corresponds to a model  $m_j$  in the set  $M$ , with columns for minimum and maximum energy consumption ( $E_{min,j}$  and  $E_{max,j}$ ), observed from initial evaluations.  $E_{latest,j}$  represents the most recent energy consumption for model  $m_j$ .  $C_{avg,j}$  is the running average of confidence score for model  $m_j$ , calculated over the latest  $k$  processed requests. This repository’s dynamic nature allows for continuous refinement of adaptation rules based on the most current data, facilitating an adaptive and responsive system capable of optimizing model selection in real-time for enhanced performance and energy efficiency. It allows the Knowledge component to provide real-time insights into model efficacy, facilitating adaptive system responses to changing operational conditions.

2) *Monitor*: The Monitor in the MAPE-K framework is divided into two specialized components: the *Energy Monitor* and the *ML Metric Monitor*, each focusing on distinct aspects of system performance monitoring. The *Energy Monitor* specifically tracks the energy consumption metrics ( $E_m$ ) of the system, including the average energy consumption for the last  $k$  processed requests ( $\bar{E}_k$ ), providing a near-real-time insight into the system’s power efficiency. Also, the *ML Metric Monitor* focuses on the detection performance metrics of the currently active model ( $m_{current}$ ), including the average confidence score ( $\bar{C}_k$ ) across the last  $k$  images processed by the model, where  $k$  represents a variable number of recent requests considered for generating a moving average. This component not only monitors confidence score but also logs additional detection outcomes like the count of bounding boxes into the *Log Repository* in Knowledge, facilitating a thorough performance review. Together, these components compile a continuous log of energy and ML metrics. The averaged data from the latest  $k$  observations is continuously sent to the Analyzer for further examination and to assess the need for adaptation. This steady stream of up-to-date information is essential for the adaptive functions of the EcoMLS approach.

3) *Analyzer*: The Analyzer component evaluates the real-time data from the Monitor and decides system adaptations. Its functionality is segmented into three components: *System Evaluator*, *Score Generator*, and *Planner Initiator*, each contributing to the adaptive decision-making process. *System Evaluator* analyses the performance metrics provided

by the Monitor, specifically focusing on the averaged energy consumption ( $\bar{E}_k$ ) and confidence score ( $\bar{C}_k$ ) for the last  $k$  processed requests. It ensures the *Runtime Rule Repository* (**B**) is up-to-date with the latest performance data before any adaptation decisions are made. Following the evaluation, the *Score Generator* calculates a performance score ( $\text{Score}_m$ ) for the currently active model ( $m_{\text{current}}$ ), using the formula  $E_j \times (1 - c_j)$ . The aim is to minimize this score, where a lower score indicates a more efficient balance between energy consumption and confidence score. Before any updates are made to the *Runtime Rules Repository* (**B**), the Analyzer provides the Planner with the current state of **B**, allowing it to devise a strategic adaptation plan based on the most recent data. This step ensures that the Planner’s decisions are informed by the latest model performance insights. The *Planner Initiator* then triggers the Planner to evaluate whether a model switch or any other adaptation action is necessary. If the performance score ( $\text{Score}_m$ ) for the currently active model ( $m_{\text{current}}$ ) exceeds the threshold defined by its corresponding metrics in the *Runtime Rule Repository* (**B**), using  $E_{\text{avg}}$ , where  $E_{\text{avg}}$  is the average of  $E_{\text{min}}$  and  $E_{\text{max}}$ , and  $C_{\text{avg}}$  from **B**, it indicates inefficiency or ineffectiveness of the current model in use. This discrepancy triggers the system to initiate an adaptation, leveraging the latest data from **B** to optimize the system’s performance. This approach ensures the Analyzer updates the Knowledge with the latest metrics and works with the Planner for prompt adaptations. This collaboration enables dynamic, informed decision-making, improving the EcoMLS system’s adaptability and optimization.

4) *Planner*: The Planner component strategizes adaptations based on insights from the Analyzer, specifically addressing energy efficiency and confidence score through model selection under two scenarios as explained in Algorithm 1. The algorithm navigates decision-making under two primary scenarios—high energy and low confidence—leveraging a unified score metric for model selection (lines 11-16). The score ( $\text{score}_m[i]$ ), calculated for each model, combines energy efficiency and confidence inversely (line 10), guiding the selection process. In high energy scenarios ( $E_{m_{\text{current}}} > E_{\text{avg}, m_{\text{current}}}$ ), models less energy-intensive than the average are considered, selecting the one with the lowest score to ensure energy efficiency (lines 11-13). Conversely, for low confidence ( $C_{m_{\text{current}}} < C_{\text{avg}, m_{\text{current}}}$ ), models that promise confidence improvement without excessive energy use are preferred, again choosing the lowest score to balance confidence and efficiency (lines 14-16). The algorithm initiates with an  $\epsilon$ -greedy approach (line 6), randomly selecting an action if below a certain threshold  $\epsilon$ , otherwise it evaluates each model’s score based on their energy and confidence metrics (lines 7-16). The executor performs model switching action (line 17), aiming to optimize the system for either energy efficiency or confidence enhancement as per the current operational requirement.

5) *Executor*: The Executor is responsible for actualizing the adaptation strategy decided by the Planner. Upon receiving a directive for a model switch from the Planner, the *Adaptation Executor* activates the selected model ( $m_{\text{best}}$ ).

---

**Algorithm 1** Planner: Algorithm for Model Selection

---

```

1: procedure FORMULATOR( $m_{\text{current}}, E_{m_{\text{current}}}, C_{m_{\text{current}}}$ )
  ▷ Input: Current model  $m_{\text{current}}$ , its energy consumption
   $E_{m_{\text{current}}}$ , and confidence score  $C_{m_{\text{current}}}$ 
2:    $A \leftarrow \{NA : -1, m_1 : 1, m_2 : 2, m_3 : 3, m_4 : 4\}$ 
3:    $p \leftarrow \text{random}(0, 1)$ 
4:   if  $p < \epsilon$  then
5:     ▷ Perform exploration
6:      $\text{action} \leftarrow \text{randint}(1, 4)$ 
7:   else
8:     ▷ Perform exploitation based on the runtime context
9:     for  $i$  in 1 to 4 do
10:       $\text{score}_m[i] = \min(E_{\text{avg}, m_i}, E_{\text{latest}, m_i}) \times (1 - C_{\text{avg}, m_i})$ 
11:     if  $E_{m_{\text{current}}} > E_{\text{avg}, m_{\text{current}}}$  then
12:       ▷ Aim for energy efficiency improvement
13:        $\text{model} = \text{argmin}_{i \in [1, 4], \text{s.t.}, E_{\text{avg}, m_i} < E_m} \text{score}_m[i]$ 
14:     else
15:       ▷ Aim to improve confidence
16:        $\text{model} = \text{argmin}_{i \in [1, 4], \text{s.t.}, C_{\text{avg}, m_i} > C_m} \text{score}_m[i]$ 
17:      $\text{action} \leftarrow A[\text{model}]$ 
18:   return  $\text{action}$ 

```

---

In the absence of such an instruction, it continues with the currently active model ( $m_{\text{current}}$ ). This adaptability is crucial for sustaining the system’s efficiency and accuracy in real-time environments. In this discussion, ‘requests’ and ‘metrics’ are generalized, but for our running example as explained in section II, they refer to images and detection boxes in object detection. This illustrates the approach’s versatility, applicable to wide range of ML tasks, with further details and results explored in the following section.

#### IV. EXPERIMENTATION AND RESULTS

The objective of our evaluation is to assess the effectiveness and efficiency of the approach by answering:

**RQ1.** How does the EcoMLS perform in comparison to other naive baselines and non-adaptive solutions in MLS?

**RQ2.** What is the effectiveness of EcoMLS in balancing objectives trade-offs within ML-Enabled Systems (MLS)?

**RQ3.** How effective is EcoMLS in managing energy and time efficiency in MLS environments, including the specific energy and time impacts of the EcoMLS adaptation process?

In the remainder of this section, we first describe our experimental setup used for the evaluation of the approach, followed by a discussion of the evaluation questions informed by our results.

##### A. Experimental Setup

To evaluate the EcoMLS approach, our experimental setup is adopted from the SWITCH [16] exemplar. It employs an object detection system, as detailed in Section II, utilizing YOLOv5 models and FastAPI simulating requests using the FIFA98 World Cup log dataset, processing 25,000 image requests. For evaluation, as detailed in our approach, the

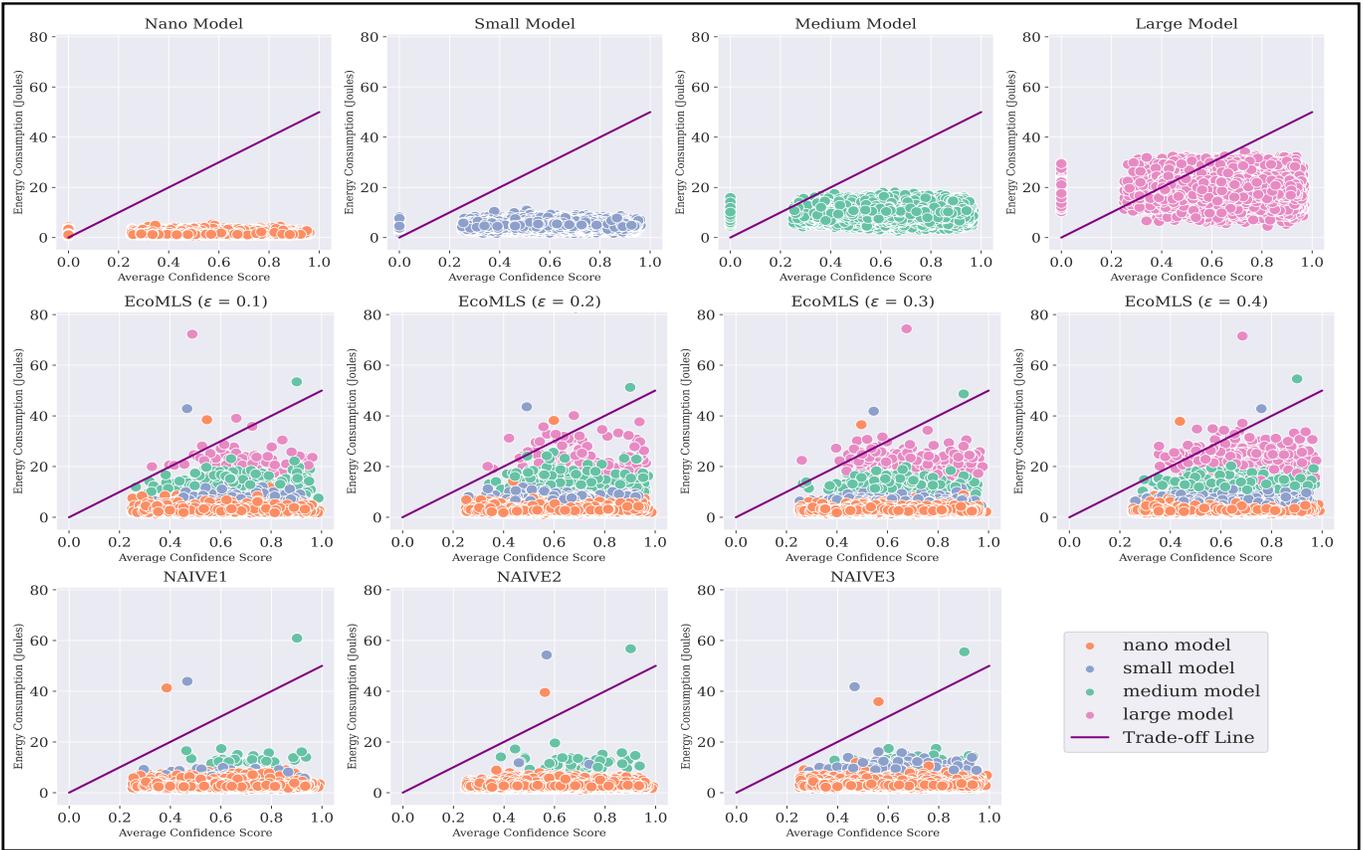


Fig. 2. Trade-off between energy consumption and the average confidence score of individual models (first row), EcoMLS with varying  $\epsilon$  (second row), and naive baselines (third row).

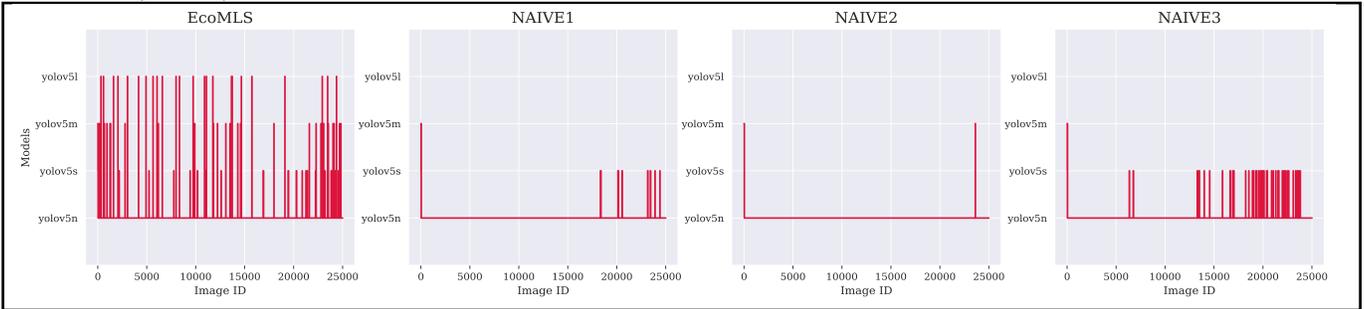


Fig. 3. Model switching: Naive baselines Vs. EcoMLS

COCO 2017 test dataset is utilized. For the testing phase, which includes performance results, the COCO 2017 unlabelled dataset is employed. The YOLOv5 models (YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l) used are pre-trained by Ultralytics on the COCO 2017 training dataset. The experiments were conducted on a system equipped with an Intel Core i7-11370H processor, NVIDIA GeForce RTX 3050 Ti 4GB Graphics, 16GB DDR4 3200MHz SDRAM, and developed using Python 3.11. To measure energy consumption, we utilized pyRAPL<sup>2</sup>, a Python package specifically designed for assessing the energy consumption and power usage of software applications running on Intel processors. The EcoMLS

framework’s evaluation included varying the  $\epsilon$  value to analyze its impact on balancing exploration and exploitation. We compared EcoMLS’s adaptive model selection with individual YOLOv5 model performances and three naive strategies: (1) using fixed knowledge for model switching (naive 1), (2) updating knowledge based on average confidence (naive 2), and (3) incorporating dynamic updates of both confidence and energy metrics in knowledge (naive 3). Our approach builds on naive 3 by adding an  $\epsilon$ -greedy mechanism, enhancing exploration through probability. The complete specifics of our implementation, parameters and results can be found here<sup>3</sup>.

<sup>2</sup><https://pypi.org/project/pyRAPL/>. Latest version released on Dec 19, 2019

<sup>3</sup><https://github.com/sa4s-serc/EcoMLS>

Approach name	$C_{avg}$	$E_{avg}$	$E_{monitor}$	$E_{analyzer}$	$E_{planner}$	$E_{executor}$	$E_{mape-k}$	$E_{avg} + E_{mape-k}$	No. of Switches
nano	0.536	1.61	-	-	-	-	-	1.61	0
small	0.611	4.327	-	-	-	-	-	4.327	0
medium	0.652	8.918	-	-	-	-	-	8.918	0
large	0.675	17.705	-	-	-	-	-	17.705	0
EcoMLS ( $\epsilon = 0.1$ )	0.61	2.762	1.284	0.001	0.001	0.0	1.285	4.047	160
EcoMLS ( $\epsilon = 0.2$ )	0.612	3.044	1.166	0.001	0.001	0.0	1.168	4.212	324
EcoMLS ( $\epsilon = 0.3$ )	0.613	2.912	1.035	0.001	0.001	0.0	1.037	3.949	313
EcoMLS ( $\epsilon = 0.4$ )	0.616	3.564	0.959	0.001	0.001	0.001	0.961	4.525	676
NAIVE1	0.609	2.47	1.226	0.001	0.001	0.0	1.228	3.697	20
NAIVE2	0.609	2.399	1.21	0.001	0.001	0.0	1.213	3.612	5
NAIVE3	0.609	3.319	1.658	0.001	0.002	0.0	1.661	4.98	106

TABLE I  
COMPARISON OF ENERGY METRICS AND CONFIDENCE SCORES ACROSS DIFFERENT APPROACHES

## B. Results

*RQ1: How does the EcoMLS perform in comparison to other naive baselines and non-adaptive solutions in MLS?*

We examine EcoMLS’s effectiveness by comparing its performance to state-of-the-art baselines and non-adaptive models, focusing on energy consumption and confidence scores balance. Table II showcases that within the score range of 0-1, EcoMLS ( $\epsilon = 0.1$ ) processes 13,040 images, showcasing its capability to frequently achieve lower scores, indicative of a balanced trade-off between energy and accuracy. In contrast, the nano model processes 20,265 images within the same score range, suggesting a tendency towards lower energy consumption but at a compromise of average confidence ( $C_{avg} = 0.536$ ), as highlighted in Table I.

The large model, which is the most accurate but also the most energy-consuming, achieves an average confidence ( $C_{avg}$ ) = 0.675 with a high energy consumption of 17.705. EcoMLS, with  $\epsilon = 0.1$ , strikes a balance with average confidence ( $C_{avg}$ ) = 0.61 and significantly lower average energy consumption of 2.762, showcasing an efficient trade-off with a 14% improvement in confidence over the nano model and an 84% reduction in energy consumption compared to the large model as observed from Figure 4. EcoMLS’s adaptive strategy aims to select the most energy-efficient models without compromising confidence. Unlike no switching, EcoMLS consistently ensures detections fall below a reference trade-off line, avoiding low-confidence results and effectively using the  $\epsilon$ -greedy mechanism for adaptation. EcoMLS’s adaptability is further highlighted by its frequency of model switching, contrasting with the minimal switching in baseline models due to their static nature. EcoMLS engages in more frequent model changes as shown in Figure 3, evidenced by 160 switches with  $\epsilon = 0.1$ , indicating an understanding of runtime conditions. EcoMLS outperforms traditional models by smartly balancing energy use and accuracy, using adaptive strategies like model switching and  $\epsilon$ -greedy exploration. It slightly exceeds the nano model in energy consumption, depicted in Figure 4, but compensates with higher confidence levels. Compared to all other models and approaches, EcoMLS achieves better performance scores, effectively managing the trade-offs between energy and accuracy. This underscores EcoMLS’s superior efficiency and its ability to

deliver high-quality outcomes while using resources optimally.

*RQ2. What is the effectiveness of EcoMLS in balancing objective trade-offs within ML-Enabled Systems (MLS)?*

We assess the balance EcoMLS achieves between energy consumption and model confidence in MLS, using the performance score  $Score_m = E_j \times (1 - C_j)$ , with a lower score indicating a more favorable balance. According to Table I, EcoMLS<sup>4</sup>, particularly with  $\epsilon = 0.1$ , optimally balances energy and confidence, exhibiting an average energy consumption ( $E_{avg}$ ) of 2.762 and an average confidence score ( $C_{avg}$ ) of 0.61 (Table I). Incremental analysis from  $\epsilon = 0.1$  to 0.4 reveals a 29% rise in  $E_{avg}$  for a mere 1% improvement in  $C_{avg}$ , justifying  $\epsilon = 0.1$  as the optimal setting due to the diminishing returns of higher exploration.

The effectiveness of EcoMLS in managing trade-offs is further evidenced by plotting energy versus confidence scores as shown in Figure 2, with a reference line connecting the lowest and highest energy and confidence values across all models. EcoMLS’s results predominantly fall below this line, especially at  $\epsilon = 0.1$ , affirming its efficiency in maintaining a balance between energy use and accuracy, despite the tendency for more detections to cross the line as  $\epsilon$  increases, indicating more exploratory actions. EcoMLS demonstrates effective trade-off management between energy efficiency and model confidence, with  $\epsilon = 0.1$  proving to be the most effective. This conclusion is underpinned by detailed quantitative analysis, illustrating EcoMLS’s capability to optimize the exploration-exploitation trade-off while ensuring sustainable MLS operation.

*RQ3: How effective is EcoMLS in managing energy and time efficiency in MLS environments, including the specific energy and time impacts of the EcoMLS adaptation process?*

In RQ3, we evaluate the energy efficiency of our adaptive approach, focusing on the MAPE-K technique. The analysis shows that the MAPE-K loop’s additional energy consumption is minimal compared to traditional models. Specifically, integrating the MAPE-K loop, EcoMLS consumes 77.14% less energy than the ‘large’ model, 54.62% less than ‘medium’,

<sup>4</sup>unless specified otherwise by EcoMLS we imply EcoMLS with  $\epsilon = 0.1$

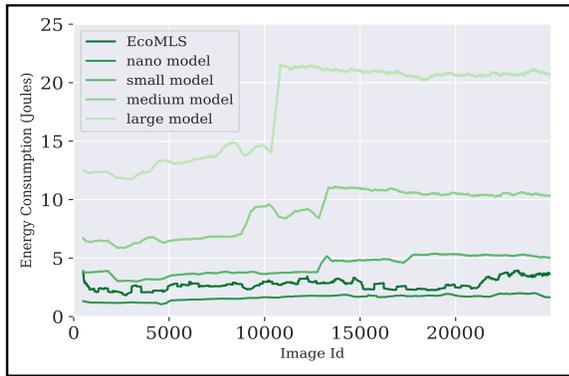


Fig. 4. Trend of energy consumption with processed image requests

and 6.46% less than 'small' as shown in Table I. The MAPE-K loop itself only adds a negligible time overhead of 0.007 seconds for each adaptation. EcoMLS continuously monitors system performance, with the analyzer evaluating the need for adaptation every second. This process triggers the planner and executor as needed, ensuring optimal system operation with minimal delay. For instance, with  $\epsilon = 0.1$ , EcoMLS's energy consumption for model operations is 2.762 Joules, with the MAPE-K loop contributing an additional 1.285 Joules, for a total of 4.047 Joules. This efficiency persists across various  $\epsilon$  settings, demonstrating that while the adaptive mechanism introduces a slight energy cost for adaptations, it significantly lowers overall energy usage compared to static models, without compromising performance. With 160 switches at  $\epsilon = 0.1$ , EcoMLS showcases effective adaptability and operational efficiency, adeptly balancing energy use and accuracy. These values, measured in Joules by PyRAPL, indicate the precise energy impact of the MAPE-K loop, continuous monitoring and adaptation process. This evaluation shows EcoMLS's capability to maintain high performance with energy efficiency.

## V. THREATS TO VALIDITY

Threats to *external validity* concerns the focus on a single type of task and a select group of machine learning models, and by limiting our examination to the inference phase. To address the first challenge, we chose a range of YOLOv5 models, varying in complexity (from YOLOv5n to YOLOv5l), and used diverse datasets, including COCO 2017. This strategy aimed to cover different visual data types and model sizes. However, our decision to focus only on the inference phase,

without considering the full lifecycle of machine learning models like training and tuning, was intentional. This choice was made to study energy consumption during inference specifically, acknowledging its narrow scope in reflecting the entire machine learning process.

A threat to the *internal validity* could be the impact of varying hardware conditions like temperature changes on the results of the evaluations. To tackle this, we implemented a 24-hour sleep period before each test to stabilize hardware conditions and performed a warm-up run to maintain consistency throughout our experiments. The threats *construct validity* could be constituted by the accuracy of our energy consumption measurements. To mitigate this, we utilized the pyRAPL library in Python (pyRAPL makes use of the RAPL library provided by Intel) which is a well-known library for measuring energy consumption. Concerning *conclusion validity*, the main threat is the potential low statistical power of our tests, which we addressed by conducting multiple experiments across different settings and conditions. Additionally, we took precautions to minimize the impact of extraneous variables, such as background tasks, on our energy consumption measurements by ensuring a clean experimental environment.

## VI. RELATED WORK

The field of Green AI has attracted substantial research interest, focusing on energy efficiency in Machine Learning (ML) systems. Verdecchia et al. [13] reviewed 98 studies, highlighting a dominant emphasis on energy efficiency mechanisms. Despite this, the practical application of these findings, particularly in making ML-enabled systems sustainable, has been limited. Studies like those by Jarvenpaa et al. [10] propose tactics for sustainability, mainly at the design stage, covering data-centric methods [20] and optimization of algorithms and models. However, the aspect of runtime energy efficiency has been relatively underexplored. The concept of self-adaptation in software, originating from IBM's autonomic computing [21], has evolved to include Machine Learning Systems (MLS). Research in this area [14], [22], [23] categorizes self-adaptation into software design approaches (SDA) and system engineering approaches (SEA), focusing on design-time solutions. Recent studies [15], [24] explore enhancing adaptability at runtime, including unsupervised learning and model switching. Yet, these approaches often limit systems to pre-set configurations, as noted by Tundo et al. [25].

Our approach diverges by offering a dynamic, self-adaptive framework that addresses runtime uncertainties in ML-enabled systems. It employs a learning algorithm to monitor and adapt based on historical data, allowing for real-time model switching. This adaptive planning considers incoming request contexts and model performance indicators, optimizing for accuracy, service quality and minimal energy consumption. Our work fills the gap in runtime energy efficiency and contributes to the development of self-adaptive, sustainable ML-enabled systems.

Name	score						
	0-1	1-2	2-3	3-4	4-5	5-6	6-7
nano	20265	4480	237	16	1	0	0
small	5301	11870	6139	1258	290	102	31
medium	2394	4447	6018	5319	3670	1870	706
large	645	2398	2304	2691	3204	3148	2693
EcoMLS	13040	10740	841	180	61	37	31
NAIVE1	14301	9996	562	96	27	9	4
NAIVE2	14860	9560	491	62	12	4	3
NAIVE3	9188	12408	2698	492	156	43	9

TABLE II  
MODEL SCORE FREQUENCY TABLE

## VII. CONCLUSIONS AND FUTURE WORK

This paper introduces EcoMLS, an innovative strategy designed to improve the sustainability of Machine Learning-Enabled Systems (MLS) by employing dynamic runtime model switching to enhance energy efficiency while ensuring model confidence is not compromised. Our evaluations reveal that EcoMLS significantly boosts energy efficiency in MLS by adapting model selection in response to real-time conditions, thereby maintaining high model accuracy and substantially lowering energy consumption. Through the EcoMLS approach, which utilizes the Machine Learning Model Balancer concept within a MAPE-K loop, we address environmental uncertainties linked to MLS deployment, focusing on reducing energy consumption in runtime operations. Currently demonstrated within the computer vision domain through object detection, future works will extend EcoMLS's applicability to Natural Language Processing (NLP), autonomous systems, and beyond, exploring its potential across a broader spectrum of ML applications. Looking ahead, we intend to investigate EcoMLS's applicability in sustainable computing, particularly its integration with edge computing and lightweight AI models. This exploration will encompass the application of software engineering methodologies attuned to the sustainability dimensions of software systems, covering economic, social, environmental, and technical aspects. Our objective is to evolve EcoMLS into a tool that facilitates sustainability-aware decision-making, enabling architects, developers and businesses to create greener, more sustainable ML-Enabled systems. In summary, EcoMLS marks an advancement towards achieving energy-efficient and sustainable ML-Enabled systems. By effectively balancing energy consumption with model confidence, EcoMLS seeks to set a new standard towards greener AI. Future efforts will focus on extending its impact, with EcoMLS continuing to lead innovation in an environmentally conscious way.

## REFERENCES

- [1] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey and benchmarking of machine learning accelerators," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sep. 2019. [Online]. Available: <http://dx.doi.org/10.1109/HPEC.2019.8916327>
- [2] Y. Mehta, R. Xu, B. Lim, J. Wu, and J. Gao, "A review for green energy machine learning and ai services," *Energies*, vol. 16, no. 15, 2023. [Online]. Available: <https://www.mdpi.com/1996-1073/16/15/5718>
- [3] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *arXiv preprint arXiv:1906.02243*, 2019.
- [4] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grah, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0743731518308773>
- [5] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, and S. Wagner, "Software engineering for ai-based systems: A survey," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 2, 2022.
- [6] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," *arXiv preprint arXiv:1910.09700*, 2019.

- [7] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 610–623. [Online]. Available: <https://doi.org/10.1145/3442188.3445922>
- [8] T. Yigitcanlar, R. Mehmood, and J. M. Corchado, "Green artificial intelligence: Towards an efficient, sustainable and equitable technology for smart cities and futures," *Sustainability*, vol. 13, no. 16, 2021.
- [9] E. Barbierato and A. Gatti, "Towards green ai. a methodological survey of the scientific literature," *IEEE Access*, pp. 1–1, 2024.
- [10] H. Järvenpää, P. Lago, J. Bogner, G. Lewis, H. Muccini, and I. Ozkaya, "A synthesis of green architectural tactics for ml-enabled systems," *arXiv preprint arXiv:2312.09610*, 2023.
- [11] R. Verdecchia, L. Cruz, J. Sallou, M. Lin, J. Wickenden, and E. Hotellier, "Data-centric green ai an exploratory empirical study," in *2022 International Conference on ICT for Sustainability (ICT4S)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2022, pp. 35–45. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/ICT4S55073.2022.00015>
- [12] F. A. Moghaddam, P. Lago, and I. C. Ban, "Self-adaptation approaches for energy efficiency: a systematic literature review," in *Proceedings of the 6th International Workshop on Green and Sustainable Software*, ser. GREENS '18. Association for Computing Machinery, 2018. [Online]. Available: <https://doi.org/10.1145/3194078.3194084>
- [13] R. Verdecchia, J. Sallou, and L. Cruz, "A systematic review of green ai," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, p. e1507, 2023.
- [14] M. Casimiro, P. Romano, D. Garlan, G. A. Moreno, E. Kang, and M. Klein, "Self-adaptation for machine learning based systems." in *ECSA (Companion)*, 2021.
- [15] S. Kulkarni, A. Marda, and K. Vaidhyanathan, "Towards self-adaptive machine learning-enabled systems through qos-aware model switching," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023.
- [16] A. Marda, S. Kulkarni, and K. Vaidhyanathan, "Switch: An exemplar for evaluating self-adaptive ml-enabled systems," 2024.
- [17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [18] G. Jocher, "Yolov5 by ultralytics," May 2020. [Online]. Available: <https://github.com/ultralytics/yolov5>
- [19] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.
- [20] R. Verdecchia, L. Cruz, J. Sallou, M. Lin, J. Wickenden, and E. Hotellier, "Data-centric green ai an exploratory empirical study," in *2022 International Conference on ICT for Sustainability (ICT4S)*. Los Alamitos, CA, USA: IEEE Computer Society, 2022, pp. 35–45. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/ICT4S55073.2022.00015>
- [21] T. Wong, M. Wagner, and C. Treude, "Self-adaptive systems: A systematic literature review across categories and domains," *Information and Software Technology*, vol. 148, 2022.
- [22] H. Muccini and K. Vaidhyanathan, "Software architecture for ml-based systems: What exists and what lies ahead," in *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, 2021, pp. 121–128. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/WAIN52551.2021.00026>
- [23] F. A. Moghaddam, P. Lago, and I. C. Ban, "Self-adaptation approaches for energy efficiency: a systematic literature review," in *Proceedings of the 6th International Workshop on Green and Sustainable Software*. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: <https://doi.org/10.1145/3194078.3194084>
- [24] I. Gerostathopoulos, C. Raibulet, and P. Lago, "Expressing the adaptation intent as a sustainability goal," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '22, 2022. [Online]. Available: <https://doi.org/10.1145/3510455.3512776>
- [25] A. Tundo, M. Mobilio, S. Ilager, I. Brandic, E. Bartocci, and L. Mariani, "An energy-aware approach to design self-adaptive ai-based applications on the edge," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE Computer Society, 2023.