Ecomap: Sustainability-Driven Optimization of Multi-Tenant DNN Execution on Edge Servers

Varatheepan Paramanayakam[1], Andreas Karatzas[1], Dimitrios Stamoulis[2], Iraklis Anagnostopoulos[1]
[1]School of Electrical, Computer and Biomedical Engineering, Southern Illinois University, Carbondale, IL, U.S.A.
[2]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, U.S.A.
Email: {varatheepan, andreas.karatzas, iraklis.anagno}@siu.edu, dstamoulis@utexas.edu

***Abstract*—Edge computing systems struggle to efficiently manage multiple concurrent deep neural network (DNN) workloads while meeting strict latency requirements, minimizing power consumption, and maintaining environmental sustainability. This paper introduces Ecomap, a sustainability-driven framework that dynamically adjusts the maximum power threshold of edge devices based on real-time carbon intensity. Ecomap incorporates the innovative use of mixed-quality models, allowing it to dynamically replace computationally heavy DNNs with lighter alternatives when latency constraints are violated, ensuring service responsiveness with minimal accuracy loss. Additionally, it employs a transformer-based estimator to guide efficient workload mappings. Experimental results using NVIDIA Jetson AGX Xavier demonstrate that Ecomap reduces carbon emissions by an average of 30% and achieves a 25% lower carbon delay product (CDP) compared to state-of-the-art methods, while maintaining comparable or better latency and power efficiency.**

***Index Terms*—Edge computing; Sustainability; Deep Neural Networks; Carbon intensity**

I. Introduction

In recent years, AI has emerged as a prominent field of computing, with significant projected growth in both scale and deployment. This trend has led to the widespread use of power-hungry hardware accelerators. For instance, Meta's infrastructure for AI inference has expanded by 2.5× in just 1.5 years to support trillions of daily inferences [1]. While AI enables transformative applications, this growth introduces critical sustainability challenges. These include not only operational carbon emissions (from energy consumed during inference) but also embodied carbon emissions (from device manufacturing, deployment, and disposal) [2], water consumption for cooling AI accelerators [3], and waste heat that further burdens thermal design and infrastructure. Although operational emissions have traditionally received more attention, recent studies show that embodied emissions are becoming increasingly dominant in the lifecycle footprint of AI systems [4].

This work focuses on reducing *operational carbon emissions* from AI inference workloads on edge computing systems. Edge deployments are becoming widely adopted across sectors such as healthcare, smart surveillance, and transportation, where on-device inference is needed to support fast, localized decisions. These systems operate under tight latency constraints and are typically powered by regional electricity grids. In this context, the carbon footprint (CF) of an edge system refers to the total CO_2 (i.e., equivalent emissions produced during inference), which is affected *not just by how much energy the device consumes, but also by the carbon intensity (CI)*, a measure of how clean or polluting the electricity source is [5]. CI varies significantly depending on location, time of day, and the local mix of energy sources (e.g., solar vs. coal), making operational carbon emissions highly dynamic and dependent on when and where inference occurs.

Edge computing has become essential for real-time applications by enabling computation closer to the user, thereby reducing latency and bandwidth costs [6]. Unlike centralized cloud data centers, edge devices typically handle real-time, *multi-tenant* AI workloads that cannot be deferred or batched. These devices are increasingly expected to execute multiple DNN-based services concurrently, such as object detection, tracking, and scene understanding, each with distinct latency and resource requirements [7]. This simultaneous execution introduces complex scheduling challenges, including resource contention, degraded QoS, and increased energy consumption. Traditional coarse-grain approaches that assign entire DNNs to a single processor are inadequate in such settings. Instead, modern systems must support fine-grained layer-level partitioning, allowing DNNs to be split across CPUs and GPUs to maximize utilization and reduce contention [8], [9].

The challenge becomes even more complex when incorporating sustainability into the scheduling objective. While reducing energy use is beneficial, it does not always lead to lower carbon emissions, especially when CI is high [4]. For instance, reducing the frequency of a GPU to save power may increase contention and unbalance resource utilization, leading to degraded performance and higher latency. As the GPU frequency drops, inference latency increases, leading to longer queues and contention when several models share the GPU. Meanwhile, the CPU may remain idle or underused, resulting in unbalanced resource utilization. These dynamics often invalidate previously optimal mappings, requiring re-optimization. To support real-time, sustainable multi-DNN execution under such non-stationary conditions, there is a clear need for intelligent runtime managers that can adapt workload placements, system configurations, and performance constraints on the fly.

A promising technique to support this adaptive runtime management is the use of *mixed-quality models*. These are pre-trained variants of the same model architecture that differ

in size, accuracy, and computational cost [10]. For example, a system may switch from ResNet-50 to ResNet-38 during periods of high load or carbon constraints, achieving faster inference and lower energy use with minimal accuracy loss [11]. The ability to dynamically substitute model variants provides a powerful knob to manage two common bottlenecks in edge AI systems: **(1)** contention from concurrent DNN execution, and **(2)** aggressive power capping under high-CI conditions. However, current runtime frameworks typically apply such model switches using static thresholds or simple heuristics. What is missing is a systematic framework that combines model quality adaptation with power-aware fine-grain mapping and real-time carbon optimization.

In this paper, we present **Ecomap**, a sustainability-driven framework for managing multi-DNN workloads on heterogeneous edge servers under strict latency constraints. Ecomap balances performance and environmental impact by combining fine-grained, power-aware layer mapping with dynamic workload adaptation. It integrates a transformer-based manager to distribute DNN layers across available components (CPU, GPU) and leverages mixed-quality models to maintain responsiveness with minimal accuracy loss. **The core contributions of Ecomap are threefold:** ❶ Fine-grained layer splitting to distribute concurrent DNNs across CPUs and GPUs, reducing contention and latency; ❷ Dynamic power control through real-time adjustment of frequencies and active CPU cores, guided by carbon intensity; ❸ Use of mixed-quality models to substitute heavier DNNs under contention or power constraints, preserving latency and accuracy. *This integration of mixed-quality models, power management, and fine-grained workload mapping makes Ecomap a comprehensive solution for sustainable multi-DNN management in edge systems.*

II. Related Work

Multi-DNN execution on resource constrained devices: Several studies have focused on improving inference throughput on heterogeneous edge platforms. For example, [12] explores inter-layer parallelism in DNNs, and [13] uses layer dimensions to guide mapping. However, both ignore power and sustainability. Recent methods like [14] use energy harvesting and renewable prediction for general edge workloads but do not address DNN-specific mapping. Adapi [15] supports model adaptivity for private inference but lacks support for concurrent multi-DNN execution or power control. Other frameworks target multi-DNN throughput. For instance, [16] builds a latency model for pipelined execution, and HaX-CoNN [7] considers contention-aware scheduling. Yet, they overlook energy efficiency. ODMDEF [17] uses regression and k-NN for workload pipelines but requires large datasets and lacks energy optimization. CARTAD [18] and ARM-CO-UP [19] address thermal or throughput goals but are not tailored for real-time, carbon-aware multi-DNN workloads. OmniBoost [8] introduces a neural cost model but does not consider power. MapFormer [20] improves power awareness with fine-grained layer splitting but lacks runtime sustainability controls. Although recent works begin addressing energy and sustainability, they typically target single-model inference or general workloads. To our knowledge, no existing method combines power-aware scheduling, real-time carbon adaptation, and efficient multi-DNN management as Ecomap does.

Mixed-quality ML models: Mixed-quality models have been explored extensively for efficient DNN execution. The method in [21] uses progressive bit-width allocation and joint training for compression-aware quantization. Similarly, [22] combines pruning with mixed-precision quantization to reduce latency and memory usage. AutoMPQ [23] automates this process via few-shot quantization adapters that dynamically tune per-layer precision. Edge-MPQ [24] introduces a hardware-aware, layer-wise quantization strategy to balance accuracy and efficiency on edge devices. In contrast, [25] uses diverse-precision hardware but relies on heuristics unsuitable for typical embedded systems. From a sustainability point of view, Clover [11] uses mixed-quality models and GPU partitioning to reduce emissions in cloud-scale inference, while PULSE [26] switches between model variants to cut latency and overhead in serverless settings. However, both target cloud environments rather than edge devices.

Sustainability-oriented edge computing: Carbon-aware strategies have been widely studied to improve sustainability in computing. For cloud systems, [27] proposes a scheduler balancing emissions, performance, and cost, while [28] explores scheduling during low-carbon periods using a regional simulation framework. However, these are cloud-focused and do not address edge-specific constraints. GreenScale [6] models carbon emissions in edge-cloud systems based on workload, renewables, and runtime variability, enabling efficient scheduling of edge tasks. In IoT contexts, CADTO [29] introduces a carbon-aware offloading scheme for NOMA-enabled edge systems, and LSCEA-AIoT [30] targets low-carbon data acquisition in AIoT environments. For DNNs, CarbonCP [31] applies conformal prediction for carbon-aware partitioning in edge-cloud scenarios. However, none of these methods tackle the combined challenges of multi-DNN execution, heterogeneous edge hardware, and real-time carbon adaptation.

III. Background

This section introduces key concepts for carbon-aware edge optimization. We define operational emissions and carbon intensity (CI), explain how CI varies over time and region, and formalize the carbon footprint (CF).

A. Operational Emissions and Carbon Intensity

The environmental impact of edge computing systems is largely determined by their operational emissions, which refer to the CO_2 released during active system operation. These emissions depend on two key factors: the total energy consumed and the carbon intensity (CI) of the electricity supplying that energy. While energy usage reflects how much electricity (in kWh) is consumed, it does not indicate how environmentally damaging that energy is.

Carbon intensity (CI) represents the average carbon dioxide emissions per unit of electricity generated and is a key metric

for evaluating the environmental impact of energy consumption. While energy consumption reflects how much electricity an edge device uses (in kWh), *it does not account* for how that electricity is produced. CI, measured in gCO_2/kWh, quantifies the emissions associated with generating one unit of electricity and therefore captures the environmental cost of energy usage.

In power systems that combine multiple generation sources, such as solar, wind, coal, or natural gas, the overall CI depends on the emission factor of each source and its contribution to the total supply. Mathematically, CI at time t is expressed as:

$$CI(t) = \frac{\sum_{i=1}^{G} E_i(t) \cdot CEF_i}{\sum_{i=1}^{G} E_i(t)} \quad (1)$$

where E_i is the electricity generated by source i, CEF_i represents the carbon emission factor of source i (in gCO_2/kWh), and G is the total number of electricity generation sources.

Different generation sources have widely different carbon intensities. For example, coal (820 gCO_2/kWh), oil (650 gCO_2/kWh), and natural gas (490 gCO_2/kWh) are among the most carbon-intensive sources, whereas wind (11 gCO_2/kWh), nuclear (12 gCO_2/kWh), and hydro (24 gCO_2/kWh) are much cleaner. Thus, minimizing emissions in edge systems requires reducing both energy use and the carbon intensity of the energy source.

B. Temporal and Spatial Variability of Carbon Intensity

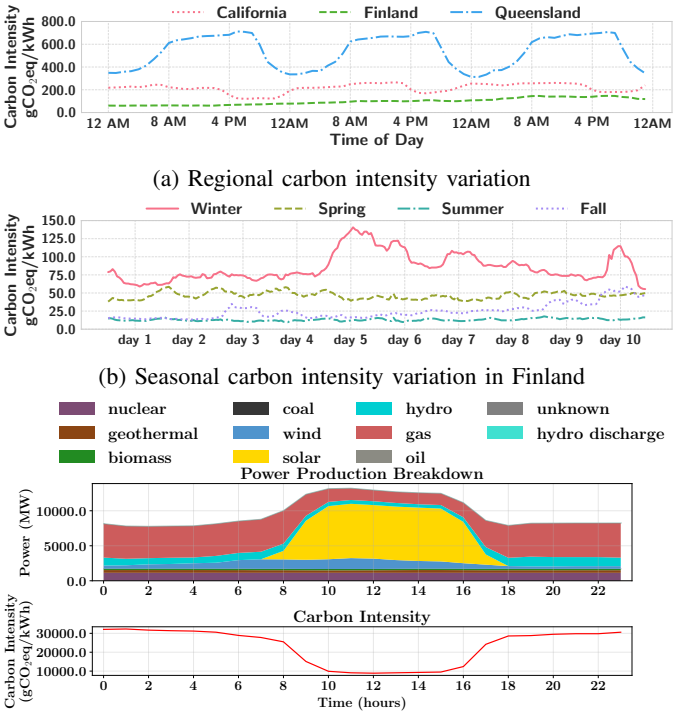
Equation 1 highlights that CI is both time- and region-dependent. Although Ecomap does not control the grid's generation mix or migrate workloads geographically, it operates under the realistic assumption that power grids dynamically adjust their energy mix based on time of day, demand, and weather. For instance, solar contributes more during the day, reducing CI, while fossil fuels often dominate in the evening or during peak demand, increasing CI. Similarly, regions with high renewable capacity (e.g., California) tend to have lower average CI than those dependent on fossil fuels. These shifts in energy composition cause CI to vary over time, even when a system's energy consumption remains constant. Figure 1 illustrates these variations by location, season, and grid composition.

C. Carbon Footprint

The carbon footprint (CF) of an edge computing system quantifies the total amount of CO_2 emitted during its operation due to energy use. When electricity is drawn from a power grid composed of multiple energy sources, each with its own emission characteristics, the resulting emissions reflect the combined environmental impact of that energy mix. As such, CF is determined by both the amount of energy consumed and the CI of the electricity used [4]:

$$CF = \int_t E(t) \times CI(t) \quad (2)$$

where CF represents the operational emissions of the system (gCO_2), $E(t)$ denotes the energy consumed by the system (kWh), and $CI(t)$ is the carbon intensity of the electricity



(a) Regional carbon intensity variation

(b) Seasonal carbon intensity variation in Finland

(c) CI variation with power source composition

Fig. 1: Examples of variation of Carbon Intensity over location, time, and energy mix of the power grid. (Data source [32])

source (gCO_2/kWh). As CI varies with region and time, identical energy consumption can result in varying amounts of CO_2 depending on the region and time. Minimizing CF in edge computing systems presents unique challenges compared to centralized cloud environments, primarily due to the real-time inference tasks that edge computing supports.

IV. Methodology

Ecomap is a sustainability-driven framework that optimizes multi-DNN execution on heterogeneous edge servers under strict latency constraints. *A key feature is its dynamic adjustment of the maximum power threshold ($\mathcal{P}^{threshold}$) based on real-time carbon intensity (CI).* This allows the system to lower emissions without sacrificing performance. Figure 2 shows a high-level overview of the framework.

Input and design space: Ecomap takes as input: (a) a set of DNNs to be executed concurrently; (b) the available computing components; (c) a list of hardware operational modes; and (d) CI forecast for the next 24 hours. Each mode defines a specific configuration, including active CPU cores, CPU/GPU/memory frequencies, and an associated maximum power cap(P_{max}) of that mode (Section IV-A). These inputs create a large design space of mappings and settings. To explore it efficiently, Ecomap uses the Latent Action Monte Carlo Tree Search (LA-MCTS) algorithm, which relies on a transformer-based estimator to predict throughput and power consumption and rank candidate solutions (Sections IV-B-IV-C).

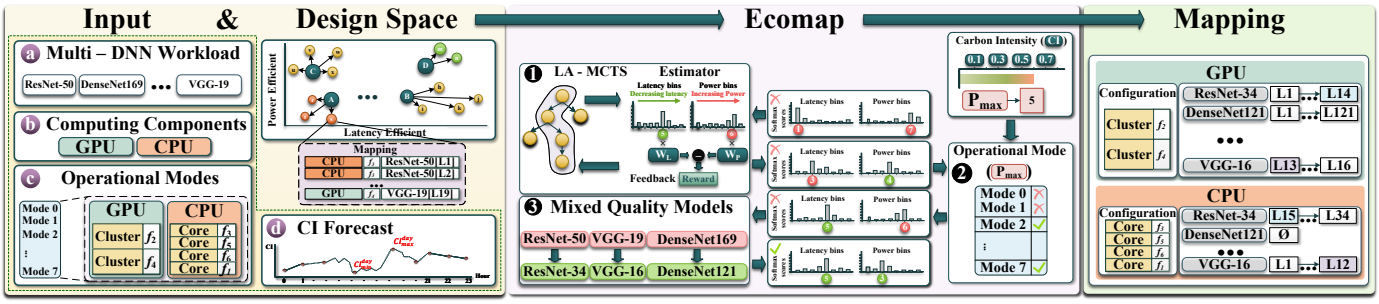


Fig. 2: Overview of Ecomap: The framework dynamically sets the edge server's power threshold based on current carbon intensity, reducing emissions while maintaining performance.

Runtime: At runtime, Ecomap adapts $\mathcal{P}^{threshold}$ to match real-time CI levels. This ensures emissions are reduced under changing grid conditions while maintaining required performance (Section IV-D). Once $\mathcal{P}^{threshold}$ is set, Ecomap uses its estimator and LA-MCTS to find the best DNN-to-hardware mapping and select an appropriate operational mode.

Enabling mixed-quality models: Ecomap monitors latency and power of running services. When latency thresholds are violated, it adapts by switching to lighter DNN variants from the same family, known as mixed-quality models. These replacements reduce delay with acceptable accuracy loss (Section IV-E). This mechanism ensures service-level agreements (SLAs) are met under changing workloads and CI levels.

A. Operating Modes

We define device-specific hardware operational modes to control power consumption. Each operational mode corresponds to a specific hardware configuration, defined by parameters such as the number of active CPU cores and the frequencies of the CPU, GPU, and memory. These modes are precomputed and stored in a lookup table (LUT), which is used at runtime (Section IV-D) to select appropriate modes.

TABLE I: Operating modes

o_j	c	f_{CPU}	f_{GPU}	f_{mem}	$P_{\max}$
1	8	2.2GH	1.3GH	2.1GH	30W
2	6	2.2GH	1.3GH	2.1GH	26W
3	4	2.2GH	1.3GH	2.1GH	22W
4	8	1.8GH	828MH	2.1GH	16W
5	6	1.8GH	828MH	2.1GH	13W
6	4	1.8GH	828MH	2.1GH	11W
7	8	1.2GH	675MH	1.2GH	8W
8	6	1.2GH	675MH	1.2GH	6W

We define the LUT of operational modes as $\mathcal{O} = o_1, \ldots, o_k$, where each o_j is a distinct mode and k is the total number of modes. Each mode is described by the tuple:

$$o_j = (c, f_{\text{CPU}}, f_{\text{GPU}}, f_{\text{mem}}, P_{\max}) \tag{3}$$

Here, j is the ID of the mode, c is the number of active CPU cores, f_{CPU}, f_{GPU}, and f_{mem} are the operating frequencies of the CPU, GPU, and memory, respectively, and $P_{\max}$ is the corresponding power cap. Table I shows eight precomputed operational modes we developed for the NVIDIA Jetson AGX Xavier board, spanning 8 W to 30 W in small increments.

Considering the multitude of possible combinations that the parameters in Equation 3 can have, iteratively tuning CPU and GPU settings to meet required power caps is not efficient. This is because adjusting one component may require coordinated changes to the others. These adjustments must also consider how each DNN uses system resources. By using predefined operational modes with fixed hardware configurations and power caps, Ecomap avoids runtime tuning via trial and error. Each mode in the LUT is derived from offline profiling across diverse workloads. Although Table I shows one representative $P_{\max}$ per mode, actual consumption may vary. For example, mode o_1 may run below 26W or 22W depending on workload and partitioning. These values are statistical abstractions that guide exploration. At runtime (Section IV-D), the power estimator predicts a discrete power class for each mapping. MCTS then prioritizes modes that previously supported mappings in that class, avoiding exhaustive exploration while *still adapting to workload changes*. Adding finer-grained DVFS steps would expand the design space with little benefit, as many new configurations yield minimal power gains due to architectural limits like shared memory and voltage regulators.

B. Latency and Power Estimator

As mentioned before, Ecomap takes as input: (**i**) a set of DNNs to be executed simultaneously; (**ii**) the set of available computing components (e.g., CPU, GPU); and (**iii**) the supporting operational modes. To process this data, we transform it into numerical vector representations using a learnable composite embedding module [33] that incorporates the latent representations of: (**i**) each DNN layer within the workload, (**ii**) each available computing component, and (**iii**) each operational mode o_j. This generated vector representations are used as the input sequences ($\mathcal{S}$) to our estimators. Ecomap utilizes layer partitioning to break down any DNN model into smaller sub-DNNs, requiring a layer-block level input representation, where each block consists of one or more fundamental layers of similar type. Thus, for each layer-block in the workload, we apply our tailored embedding module to create a sequence of tuples,

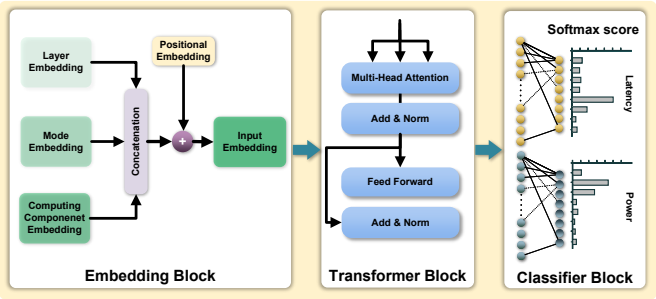


Fig. 3: Architecture of our transformer based estimator.

each consisting of a layer-block, a computing component, and its corresponding operational mode o_j.

The input sequence $\mathcal{S}$ is then processed by our transformer-based estimator, which consists of three core components. First, a **composite embedding module** creates the input representation for each token using: (i) a global layer index embedding (to uniquely identify each DNN layer); (ii) a computing component embedding (e.g., CPU as 0, GPU as 1); (iii) a mode ID embedding representing the active operational mode; and These components are summed to form the final input vector per token. Second, a **multi-head self-attention layer** captures dependencies across layers and resources in the mapping sequence. Lastly, a **feed-forward network** produces latency and power outputs, using separate classification heads with quantile-based bins. Figure 3 illustrates this architecture.

Unlike previous methods [8], [17], our distributed embedding vectors are learnable, enhancing the transformer's ability to estimate latency and power consumption more accurately. When the DNNs are split into sub-DNNs, additional communication delays will be introduced between computing components, making appropriate selection of split points and computing components crucial. Especially for latency prediction, these communication delays can be significant. However, explicit profiling of these delays often involves oversimplified assumptions of linear correlation [34]. To mitigate this, our method models this implicitly, opting for end-to-end prediction. Our learnable embedding vectors provide more context to the estimator to effectively model communication delays implicitly.

Building on input sequence $\mathcal{S}$, we use the transformer-based estimator [35] presented above to assess any mapping $\mathcal{M}$ and predict its latency and power consumption under each different mode o_j. The choice of a transformer-based estimator is due to its ability to identify long-sequence patterns, which is crucial for managing higher-order multi-DNN workloads. Estimators from previous studies [8], [17], although effective for smaller workloads, tend to underperform with larger multi-DNN workloads, often resulting in sub-optimal mappings.

A major differentiator of Ecomap from previous state-of-the-art approaches is that it is designed for classification rather than regression. While estimating exact values for latency and power consumption could potentially yield better multi-DNN mappings, it also requires significantly larger datasets to manage the imbalances in target values, especially due to the relatively rare nature of optimal mappings in the mapping space. To address this, we define the estimator's target as a quantile distribution of N discrete classes(bins), transforming the problem into a classification task. The N quantiles are equal in sample size, which helps overcome data imbalance. Furthermore, to manage the multi-objective nature [36] (latency and power), we employ two separate fully connected layers, each with N output neurons corresponding to the target classes.

To ensure that the quantile-based classes(bins) adequately represent the latency and power spaces, we used a systematic mapping generation strategy to build the estimator dataset. This strategy explores a wide range of DNN combinations and resource allocations. The estimator does not predict the latency of individual DNNs, which is difficult due to cache and scheduling effects on CPUs. Instead, it models the average system-level latency of the entire workload, defined as:

$$\text{Average latency} = \frac{U}{\sum_{i=1}^{U} \text{Throughput}_i} \tag{4}$$

where U is the number of concurrently running DNNs. This latency proxy correlates well with system responsiveness and supports consistent class labeling. In addition, we formed latency classes(bins) in latency decreasing order and power classes(bins) in power increasing order. This allows the MCTS to search for mappings with higher latency class predictions (corresponding to lower actual latency) and lower power class predictions (corresponding to lower actual power), creating a clear distinction between optimization objectives. The use of quantile-based classification avoids regression instability and handles class imbalance effectively. Furthermore, during runtime, the statistical nature of the MCTS algorithm allows the estimator to still identify and classify mappings with values outside the training range into the nearest valid quantile class, maintaining robustness under unseen workloads.

C. LA-MCTS Module

Our estimator module is the mechanism for evaluating any candidate mapping. To address the exploration of the mappings, we integrate the Latent Action-MCTS (LA-MCTS) [37] algorithm, a highly efficient space exploration module. MCTS is a heuristic approach that efficiently navigates extensive design spaces by iteratively interacting with its decision tree within a set computational budget [38]. This tree holds all possible mappings for a given design space. Although traditional MCTS effectively minimizes a cost function through stochastic processes, it tends to converge slowly.

To enhance the convergence rate of MCTS, we adopted LA-MCTS, which iteratively learns to partition the design space hierarchically. For the most part, LA-MCTS follows the same steps as MCTS. Initially, it generates random mappings, and these mappings are then evaluated using the estimators to obtain the reward values. The reward values are then used for the traditional MCTS exploration and exploitation. On top of traditional MCTS, in each iteration, LA-MCTS examines specific regions of the decision tree and applies k-means algorithm with $k = 2$ to categorize them into two clusters,

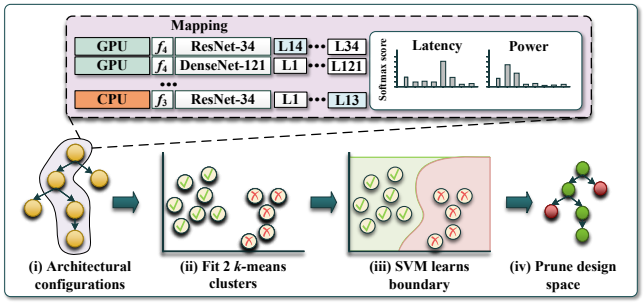


Fig. 4: Design space pruning via LA-MCTS.

distinguishing between promising (good) and less promising (bad) solutions. Subsequently, LA-MCTS uses Support Vector Machines (SVM) to learn a decision boundary that extrapolates the patterns identified by the k-means to the broader design space. We use $k = 2$ because the goal of clustering in LA-MCTS is to support binary classification for adaptive pruning. The clustering step simply separates explored configurations into two broad categories, those that appear more promising and those that do not, based on their reward values. This binary distinction is sufficient for the SVM to learn a clear decision boundary, allowing the search to prune unpromising regions efficiently. Using more clusters (e.g., $k = 4$ or $k = 8$) would increase the complexity of the classification model and may not improve search effectiveness, especially given the limited samples available at each iteration. Prior work on LA-MCTS [37] has shown that a two-cluster approach balances pruning performance and convergence stability in high-dimensional design spaces. As each simulation step involves an estimator inference, by effectively pruning less promising nodes, LA-MCTS eliminates wasteful estimator inferences and generates solutions quickly. Figure 4 provides a high-level overview of the iterative process of LA-MCTS and how it prunes the design space to focus on more viable solutions. Overall, LA-MCTS consistently converges faster than standard MCTS by reducing the number of estimator evaluations, making it well-suited for complex, time-constrained multi-DNN mapping tasks.

To address the multi-objective nature of our problem, we formulate a reward function $\mathcal{V}$. This function evaluates any mapping $\mathcal{M}$ by calculating the weighted difference between the predicted latency class and the predicted power consumption class. Additionally, to ensure the satisfaction of the power constraint ($\mathcal{P}^{threshold}$), the reward for any mapping that estimated to exceed the maximum allowable power consumption is set to negative infinity, effectively removing it from consideration as a viable solution. The classification model provides an initial estimate, while the MCTS search explores candidate mappings within and around the predicted class boundaries. This enables Ecomap to adapt to new or edge-case workloads by probabilistically sampling viable solutions, even if exact matches were not present during training. The reward function is formulated as:

$$\mathbf{V}(\mathcal{M}) = \begin{cases} \mathbf{W}_L \cdot \mathcal{L}(\mathcal{M}) - \mathbf{W}_P \cdot \mathcal{P}(\mathcal{M}), \\ \qquad \text{if } \mathcal{P}(\mathcal{M}) \leq \mathcal{P}^{threshold} \\ -\infty, \text{otherwise} \end{cases} \tag{5}$$

where $\mathbf{W}_L$ represents the weight assigned to latency, $\mathcal{L}(\mathcal{M})$ denotes the estimated latency class for the mapping $\mathcal{M}$, $\mathbf{W}_P$ is the weight assigned to power consumption, and $\mathcal{P}(\mathcal{M})$ indicates the predicted power consumption class. The weights $\mathbf{W}_L$ and the $\mathbf{W}_P$ are determined at runtime based on the latency requirement and power threshold. The maximum allowable power consumption, determined by CI (Section IV-A) is denoted by $\mathcal{P}^{threshold}$. It is important to note that the LA-MCTS does not directly use the CI, but instead bases its solutions on the $\mathcal{P}^{threshold}$, which is computed from the CI.

The reward function provides a scalar feedback to the LA-MCTS algorithm, enabling it to compare and rank different mappings efficiently during search. For feasible mappings (i.e., those satisfying the power constraint), this scalar reflects a weighted trade-off between latency and power classes. Infeasible mappings are immediately excluded by assigning them a reward of $-\infty$. These scalar reward values are used in LA-MCTS as follows: (i) they are used to rank candidate mappings within each iteration, (ii) they drive the k-means clustering to distinguish promising vs. non-promising mappings, and (iii) they serve as labels for training the SVM classifier, which then predicts which regions of the search space are worth exploring further. The mapping with the highest reward among all feasible configurations is ultimately selected as the final output.

D. Runtime

At runtime, Ecomap incorporates a 24-hour CI prediction for the electricity grid to dynamically manage the operational power thresholds. For this Ecomap uses a highly accurate carbon intensity forecasting method presented in [39]. This method uses a two-tier approach for forecasting. The first tier employs multilayer perceptron (MLP) networks and predicts electricity generation for each source in the grid using historical data for each source and weather forecasts for the next 96 hours. The second tier employs a hybrid CNN-LSTM based network and predicts carbon intensity based on the first-tier predictions and carbon intensity data from the past 24 hours. Using this method, Ecomap forecasts the CI value for the next 24 hours in an hourly granularity, and based on this forecast, Ecomap determines the minimum ($CI^{\text{day}}_{\text{min}}$) and maximum ($CI^{\text{day}}_{\text{max}}$) values of CI over the next 24 hours (for the day).

During the runtime, Ecomap uses real-time CI data from [32], which provides region wise CI values in an hourly granularity. The CI from this source provides a more accurate value as this source obtains the data from the relevant electricity production entities. Additionally, as described in Section III-B, while CI generally fluctuates throughout the day, its pattern varies significantly by region and energy mix. Ecomap adapts to these dynamics by leveraging the CI forecasts to schedule high-power modes when emissions are low and transitioning to lower-power configurations during high-CI periods. When

CI is at its minimum, the edge device operates at the highest power threshold, delivering services with minimal delays and maximum performance while taking advantage of the low environmental impact. As CI increases throughout the day, Ecomap dynamically adjusts the maximum allowable power threshold ($\mathcal{P}^{threshold}$) and transition between operational modes through MCTS exploration. The decision to use granular power thresholds, as shown in Table I, is crucial for maintaining a balance between sustainability and performance.

Typically, the default operational mode configurations showcase significant maximum power deviations. With these high variations, the computational capability of the system reduces drastically. Therefore, while employing the default modes can result in significant emission reductions, they may degrade system responsiveness to unbearable levels. Through carefully preselected core combinations and frequencies, EcoMap ensures that the frequencies and core utilization of the computing components produce viable combinations that remain functional across a wide variety of workloads. Conversely, a smaller difference in P_{max} limits the reduction in computational capability, which in turn allows the workloads to identify mappings that do not exhibit abrupt performance degradation. By defining operational modes with small, incremental differences in P_{max}, Ecomap ensures smooth transitions between configurations, enabling more precise control of power consumption while adapting to varying carbon intensity levels.

The variability of CI is generally low over short time intervals, and CI data re typically reported or forecasted at an hourly resolution [40]. Exploiting this property, Ecomap updates the maximum power threshold at most once per hour. The CI value at the beginning of each hour is used throughout that hour to guide decisions. This periodic scheduling reduces the overhead of frequent remapping while maintaining alignment with CI trends, ensuring stable and efficient runtime behavior. To avoid erratic behavior (e.g., frequent back-and-forth changes in P_{max}), Ecomap updates the power threshold only when CI changes by at least 10% of the predicted range. These fine-grained adjustments allow the system to remain stable and responsive to considerable changes in CI without significant disruptions to service performance.

Ecomap also handles dynamic service arrivals. For each new service, it uses the latency and power estimator with LA-MCTS to find a mapping that satisfies the current power threshold ($\mathcal{P}^{threshold}$) set by CI. When multiple mappings meet this constraint, Ecomap applies the reward function from Subsection IV-C to prioritize lower latency while respecting power and carbon constraints.

As discussed in III-C, CI varies throughout the day due to the evolving energy mix in the grid, even when energy consumption remains constant. While Ecomap does not control the grid's sources or migrate workloads geographically, it does exploit temporal CI variation. At runtime, Ecomap adapts to these CI changes by adjusting the system's power budget and re-evaluating workload mappings accordingly. *This is not a simple DVFS problem.* Each new power budget introduces a different set of available hardware configurations (e.g., CPU/GPU modes), and the system must determine whether DNN models should be split, remapped, or substituted with lighter alternatives. This triggers a multi-objective scheduling process that jointly considers latency, power, and model quality.

E. Enabling Mixed-Quality Models

Ecomap ensures that all running services meet predefined latency and power thresholds by actively monitoring their performance in real time. Changes in the maximum allowable power threshold ($\mathcal{P}^{threshold}$), driven by variations in carbon intensity (CI), or the arrival of new service requests, can lead to resource contention and latency violations. To address these issues, Ecomap dynamically adapts by leveraging mixed-quality models, which replace computationally intensive DNNs with lighter alternatives from the same model family. These alternatives, referred to as mixed-quality models, offer reduced computational requirements with slightly lower accuracy for lower quality service levels. Generally, model size and accuracy within a model family showcase a trade-off relationship, where accuracy exhibits diminishing returns as model size increases. Consequently, strategically selected model variants serve as suitable backbones without significant accuracy drops. These model switches occur only when necessary to preserve system responsiveness under tighter power budgets or high contention. This allows Ecomap to meet real-time latency constraints even under unfavorable CI conditions, while ensuring that service quality remains high unless adaptation is essential.

Ecomap monitors services continuously and detects latency violations triggered by changes in $\mathcal{P}^{threshold}$ or the addition of new service requests. When a latency violation is detected, Ecomap handles it by using a two-level prioritization policy. First, Ecomap begins by identifying the service experiencing the highest latency relative to its threshold. Let R denote this service, with its associated DNN represented as D^j, where j is the service level. Ecomap replaces D^j with the next available lightweight alternative from the set of mixed-quality models, $Q(D) = D^1, D^2, \ldots, D^q$. The selection is guided by the following optimization:

$$\text{Find } D^k \in Q(D) \text{ such that } L(D^k) \leq L_{\text{max}} \text{ and } \Delta A(D^k_j) \leq \epsilon, \tag{6}$$

where j is the current service level, k is the target service level, q is the number of service levels, $L(D^k)$ is the latency of D^k, L_{max} is the maximum allowable latency, $\Delta A(D^k_j)$ is the accuracy drop of D^k compared to D^j, and ϵ is the maximum acceptable accuracy drop. If latency constraints are still not met after the first replacement, Ecomap identifies the most computationally intensive service in the workload that is in a higher QoS level, and replaces that with a lightweight alternative to reduce contention and free up resources for other services. Ecomap alternates between these two DNN switching approaches until the constraints are met, all services have reached their lowest allowable model variant, or a predefined retry limit is reached. This two-level prioritization, first by latency then by resource usage, ensures that adjustments are effective while avoiding unnecessary model downgrades.

TABLE II: Supported services and mixed-quality models

Service	Default DNN (Level-1)	Level-2	Level-3
Object Detection	MNASNet1_3	MNASNet1_0	MNASNet0_75
Object Classification	EfficientNet_v2_s	EfficientNet_b1	EfficientNet_b3
Object Tracking	ResNet152	ResNet101	ResNet50
Depth Estimation	ResNet152	ResNet101	ResNet50
Abnormal Behavior Detection	VGG19	VGG16	VGG13
Facial Expression Recognition	DenseNet169	DenseNet161	DenseNet121

Once a new mapping is generated due to workload changes or CI changes, the mapping is evaluated to measure the latency of each DNN. If service level changes need to be performed, instead of conducting LA-MCTS exploration in the entire configuration space, Ecomap narrows the search to configurations directly affected by the updated DNN to enhance the speed of the search process. In addition, Ecomap utilizes a refined search space for each DNN which is constructed by evaluating the behavioral patterns under various mapping configurations. This space deprioritizes the regions where layer splitting leads to latency increases exceeding 30%. At runtime, this tailored search significantly reduces computational overhead, as it focuses on high-probability configurations while avoiding suboptimal areas. This strategy is particularly effective because in this scenario Ecomap adjusts only one DNN at a time, ensuring that the tailored search remains fast and precise.

Ecomap identifies two main cases where remapping may be necessary. In **Scenario 1**, the system is under pressure, due to either a rise in service requests or a reduction in the power budget (from a higher CI). In this case, Ecomap performs a full remapping to ensure all services can meet their constraints. In **Scenario 2**, the system operates under relaxed conditions, such as fewer requests or lower CI. Here, Ecomap performs targeted remapping: it only replaces DNNs that currently violate their latency limits or can be upgraded to higher-quality versions. This selective strategy avoids unnecessary changes and reduces the number of model switches. On average, fewer than five model transitions occur per day, and each service switch completes in about 30 seconds (see Section V-C). By dynamically adapting services through mixed-quality models and leveraging tailored search, Ecomap maintains latency compliance even under dynamic conditions and balances power efficiency and sustainability, providing an efficient solution for managing multi-DNN workloads in edge computing environments.

V. Experimental Evaluation

In this section, we evaluate Ecomap's performance in latency, power, emissions, and sustainability using the NVIDIA Jetson AGX Xavier (JAX) edge server. JAX features (**i**) a Volta GPU with 512 CUDA and 64 Tensor cores (10 TFLOPS peak), (**ii**) a Carmel CPU with $\times 4$ ARMv8.2 dual-core clusters at 2.26 GHz, and (**iii**) 32 GB LPDDR4x memory.

Ecomap is developed using PyTorch, which supports the integration of diverse DNN architectures and enables fine-grained partitioning of multi-DNN workloads. To manage these workloads, we created a custom PyTorch-powered compute library that enables dynamic mapping of DNNs onto the edge server's computing components. The training dataset for Ecomap's transformer-based estimator consists of 8,000 mappings, all evaluated through real execution on the NVIDIA Jetson AGX Xavier platform. Each workload includes 5 to 10 DNNs and is executed across the 8 predefined operational modes, with 1,000 mappings collected per mode to ensure broad coverage. To ensure the dataset captures both efficient and inefficient mappings, we avoided fully random generation, which tends to overrepresent poor configurations. Instead, we used a rule-based strategy. First, we profiled each individual DNN by testing various ways of splitting it between CPU and GPU, measuring end-to-end throughput for each split configuration. This analysis revealed that only certain split positions maintain acceptable performance, while others introduce significant communication overhead and performance degradation. We prioritized the viable splits identified in this profiling phase when generating workload mappings. For multi-DNN workloads, we created mapping scenarios by varying the number of DNNs assigned to each component and the number of models split across CPU and GPU. The rule-based strategy ensured that mappings captured a range of behaviors, from efficient to inefficient, enabling the estimator to learn distinctions across the performance spectrum. Each mapping was executed for 30 seconds on the device, following a 5-second warm-up period to allow transient effects to settle. For each configuration, we recorded per-DNN throughput, total system power consumption, and average system latency. We consider only execution latency, assuming images are available at intended frequencies, without including image capturing.

To ensure a comprehensive evaluation, we leveraged a large set of models available in the `torchvision.models` library, resulting in a total space of 50 widely used DNNs. These models are categorized into the following families: (**i**) AlexNet, (**ii**) DenseNet, (**iii**) EfficientNet, (**iv**) GoogLeNet, (**v**) InceptionV3, (**vi**) MNASNet, (**vii**) MobileNetV2, (**viii**) MobileNetV3, (**ix**) RegNet, (**x**) ResNet, (**xi**) ShuffleNetV2, (**xii**) SqueezeNet, (**xiii**) VGG, and Ecomap's design ensures compatibility with most of the models defined in PyTorch, making it adaptable to diverse application requirements. We trained our estimator for 100 epochs with 80% of our dataset using `AdamW` optimizer with 0.0001 learning rate and `CosineAnnealingLR` scheduler for smooth approximation of the most optimal model parameter set. For validation, we evaluated our estimator on the remaining and unseen test subset. We evaluated the trained model using top-1 and top-3 accuracy metrics. Our top-3 definition considers predictions accurate if they match the true class or either adjacent class, which is meaningful since classes represent sorted value ranges. Top-1 accuracies were 78.59% (latency) and 81.69% (power), while top-3 accuracies reached 98.79% and 99.17%, respectively, demonstrating estimator robustness.

For our experiments, we utilized three distinct 5-day periods to evaluate the server's performance under varying carbon intensity (CI) conditions and workloads. For this we extracted real world carbon intensity data from *Electricity Maps* website [32], which provides data from real energy supply agencies from many countries around the world. Week-1 and Week-3 exhibit significant variability in CI, reflecting fluctuating

energy grid dynamics, whereas Week-2 demonstrates relatively stable CI with minimal fluctuation. These scenarios allow us to test Ecomap's adaptability to different environmental and operational conditions.

Regarding user-based service requests, we selected six types of services: (**i**) object detection, (**ii**) object classification, (**iii**) object tracking, (**iv**) depth estimation, (**v**) abnormal behavior detection, and (**vi**) facial expression recognition. Each service supports mixed-quality models to ensure adaptability under latency violations. Table II shows the default DNN (level-1) and the mixed-quality models (level-2 and level-3) used for each service. In our evaluation, we focus on the backbone DNNs commonly used in each service task, without including task-specific layers such as detection heads or post-processing modules. This is justified for two reasons: (i) task-specific layers contribute minimally to overall compute time, while the backbone dominates inference latency on edge devices, and (ii) adding these layers would increase system complexity without affecting the mapping and power optimization challenges that Ecomap addresses. The selected models (e.g., EfficientNet, ResNet, VGG) are widely used as backbones for tasks such as classification, detection, and tracking [41], [42], and reflect real-world edge deployment practices. Since Ecomap is task-agnostic and focuses on scheduling and adaptation across heterogeneous workloads, evaluating the backbone models alone effectively captures the relevant system behavior.

Each week also varies in the number of service requests received by the server. In Weeks 1 and 2, the maximum number of requests the server could handle without significant delays or becoming unresponsive was capped at 15 concurrent service instances. For Week-3, the maximum number of requests was reduced to 10 to evaluate system performance under medium-to-heavy workloads. The workload patterns for all the weeks were generated by mimicking the workload variation patterns found in the *Microsoft Azure Traces* [43], which provides comprehensive statistics and performance metrics from its cloud server resources. The weekly characteristics, including CI variability and workload intensity, are summarized in Table III.

TABLE III: Weekly experiment characteristics

Week Name	CI Variability	Workload Intensity
Week-1	High	High
Week-2	Low	High
Week-3	High	Medium

In our experiments, we evaluated Ecomap under two configurations that reflect different latency thresholds: $Ecomap_R$ which operates under a relaxed threshold of 2 seconds and $Ecomap_S$ which operates under a strict deadline of 500 milliseconds. These thresholds reflect different quality-of-service requirements, allowing us to assess Ecomap's ability to balance latency and sustainability under varying constraints.

For a comprehensive evaluation, we compare Ecomap against four state-of-the-art multi-DNN management frameworks: (**i**) **OmniBoost** [8], a greedy throughput optimization framework for multi-DNN workloads, which serves as the baseline for comparison; (**ii**) **ODMDEF** [17], a manager utilizing a combination of linear regression and k-NN classifiers for DNN scheduling; (**iii**) **Hax-Conn** [7], a contention-aware scheduling framework designed for concurrent DNN execution; and (**iv**) **MapFormer** [20], a power-efficient framework aimed at optimizing resource usage for multi-DNN workloads.

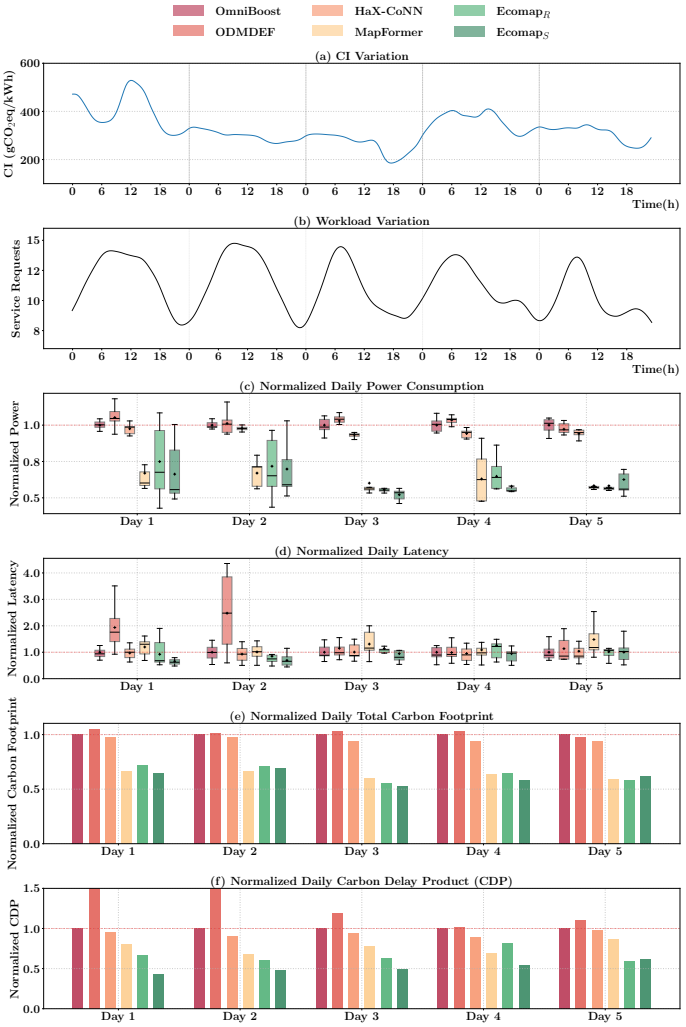


Fig. 5: Normalized comparison of Ecomap ($Ecomap_R$ and $Ecomap_S$) during Week-1 across (a) CI, (b) workload, (c) power, (d) latency, (e) emissions, and (f) CDP over 5 days. Lower is better in all charts.

To evaluate the performance of Ecomap, we compared it against all the aforementioned methods using a comprehensive set of metrics. Specifically, we measured latency, power consumption, daily carbon footprint, and the Carbon-Delay Product (CDP). The daily carbon footprint quantifies the total operational emissions over a 24-hour period, providing a measure of the environmental impact. Finally, the CDP, a product of latency and carbon footprint, offers an integrated metric to evaluate the trade-off between performance and sustainability.

A. Sustainability-Oriented Comparison

Figures 5-7 depict the comparison between all methods. For Week-1, depicted in Figure 5, the CI exhibits significant

variability, ranging from approximately 200 gCO_2/kWh to 500 gCO_2/kWh. For each box plot used, it shows the median (center line), interquartile range (box edges), and minimum/maximum values, summarizing variability across the 24-hour period.

Power consumption: Ecomap shows strong power efficiency in both $Ecomap_R$ and $Ecomap_S$. On average, $Ecomap_R$ reduces power by 35% vs. OmniBoost and 32% vs. Hax-Conn, while $Ecomap_S$ achieves 39% and 32% reductions, respectively. MapFormer, as expected, remains competitive in terms of power consumption due to its focus on power optimization.

Latency: Ecomap effectively balances latency in both $Ecomap_R$ and $Ecomap_S$ configurations. $Ecomap_R$ ensures low power while maintaining acceptable service responsiveness. $Ecomap_S$, under the strict 500 ms latency constraint, achieves lower latency values across all days but incurs slightly higher power usage. Compared to OmniBoost and Hax-Conn, $Ecomap_R$ maintains the latency with a slight increase of about 2%, while $Ecomap_S$ achieves 17% lower latency due to Ecomap's dynamic adaptation and mixed-quality models. In contrast, due to its power-centric design, MapFormer performs poorly. This underscores Ecomap's ability to handle multi-DNN workloads effectively under varying latency constraints.

Daily total emissions: Ecomap achieves significant reductions in normalized daily total emissions for both configurations. $Ecomap_R$, benefiting from its relaxed constraints, reduces emissions by 35% compared to OmniBoost and 33% compared to Hax-Conn on average across all days. $Ecomap_S$, despite stricter latency requirements, achieves 39% and 36% lower emissions than OmniBoost and Hax-Conn, respectively. These results demonstrate Ecomap's effectiveness in minimizing emissions even under challenging operational constraints.

CDP: Ecomap excels in terms of normalized Carbon Delay Product (CDP), which integrates latency and emissions to measure sustainability efficiency. Both $Ecomap_R$ and $Ecomap_S$ outperform MapFormer significantly. $Ecomap_R$ achieves 13% lower CDP than MapFormer, while $Ecomap_S$ achieves 36% lower CDP. Despite MapFormer's exceptional power efficiency, its inability to adapt to latency constraints results in higher latency, which increases its CDP. In contrast, despite the stricter latency thresholds, the use of mixed-quality models in $Ecomap_R$ and $Ecomap_S$ improves both carbon efficiency and latency, making both configurations more sustainable compared to other methods. In summary, Ecomap, in both $Ecomap_R$ and $Ecomap_S$ configurations, outperforms state-of-the-art frameworks in terms of power consumption, latency, emissions, and sustainability efficiency. These results highlight Ecomap's adaptability and ability to balance performance and sustainability in dynamic edge computing environments.

In Week-2 (Figure 6), with low CI variability (450–550 gCO_2/kWh), Ecomap performs well across all metrics. In terms of **power**, $Ecomap_R$ slightly exceeds MapFormer on some days but averages a 1% reduction, while $Ecomap_S$ reduces power by 9%. Both significantly outperform OmniBoost and Hax-Conn by 34% and 32%, respectively. For **latency**, $Ecomap_S$ achieves 18% and 16% lower latency than OmniBoost and Hax-Conn, while MapFormer shows higher

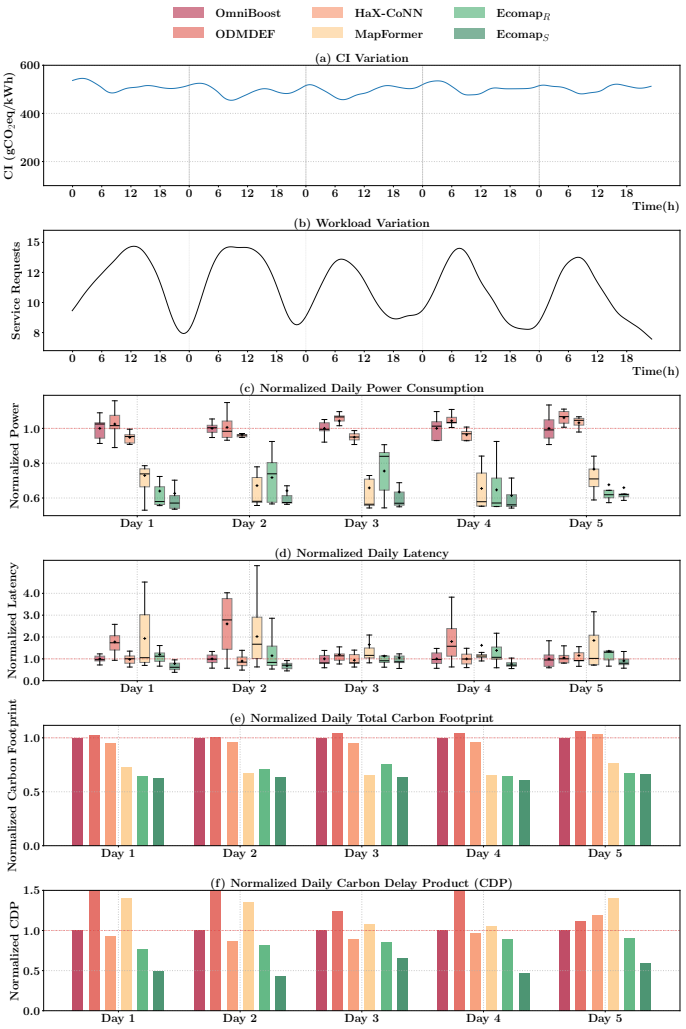


Fig. 6: Normalized comparative analysis of Ecomap during Week-2. For all comparison charts, lower is better.

delays due to limited latency handling. Regarding **carbon footprint**, $Ecomap_R$ cuts emissions by 32% vs. OmniBoost and 30% vs. Hax-Conn; $Ecomap_S$ achieves 37% and 35% reductions. In **CDP**, Ecomap clearly outperforms MapFormer, with $Ecomap_R$ and $Ecomap_S$ achieving 34% and 60% lower values, respectively.

In Week-3 (Figure 7), with medium workload and high CI variability (250–600 gCO_2/kWh), Ecomap maintains strong performance. In terms of **power**, $Ecomap_R$ averages 10% higher than MapFormer and $Ecomap_S$ only 2% higher, while both outperform OmniBoost and Hax-Conn by 34% and 32%, respectively. **Latency** remains low: $Ecomap_S$ is 6% lower than OmniBoost and about 2% higher than Hax-Conn, with $Ecomap_R$ showing similar performance under relaxed constraints. For **carbon footprint**, $Ecomap_R$ reduces emissions by 33% vs. OmniBoost and 31% vs. Hax-Conn; $Ecomap_S$ achieves 38% and 36% reductions. In **CDP**, Ecomap outperforms MapFormer with 7% lower CDP for $Ecomap_R$ and 28% lower for $Ecomap_S$.

It is important to note here that, while the daily total

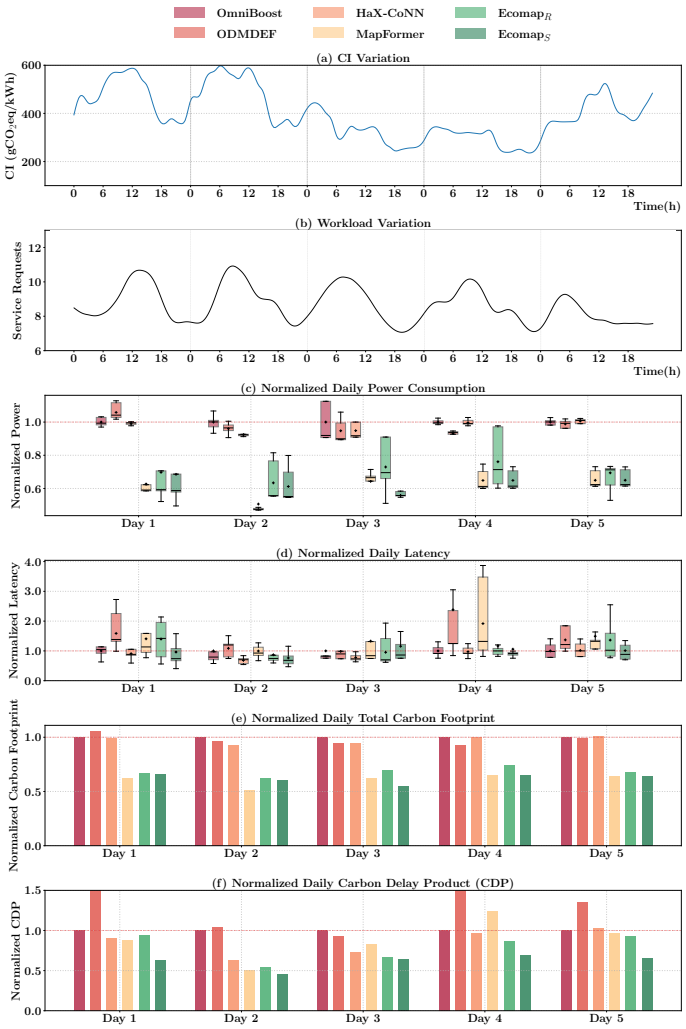


Fig. 7: Normalized comparative analysis of Ecomap during Week-3. For all comparison charts, lower is better.

emissions are correlated with energy consumption (power $\times$ latency), they are not directly proportional due to carbon intensity (CI) fluctuations which is EcoMap's fundamental control variable. For example, on day 1 of week 1, $Ecomap_R$ used 12.9% less energy between 10 AM -11 AM than between 11 AM-12 PM, yet produced 6.4% more carbon emissions during the earlier period due to a 22% higher carbon intensity. In our experiments, since CI is updated at hourly granularity, the daily emissions reported in Figures 5-7 aggregate the hourly variations and smooth out these differences.

B. Mixed-Quality Models Analysis

Mixed-quality models allow Ecomap to adapt to varying latency requirements by replacing high-quality DNNs with lighter alternatives whenever latency constraints are violated. The analysis for Week-1 (Figure 8) reveals distinct trends in how $Ecomap_R$ and $Ecomap_S$ utilize mixed-quality models. In general, $Ecomap_R$ relies more heavily on default models compared to $Ecomap_S$, which frequently switches to lighter models

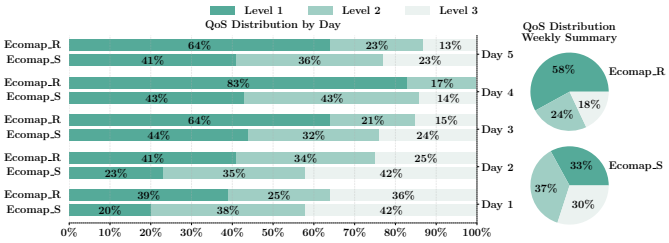


Fig. 8: Mixed-quality model usage by Ecomap in Week-1

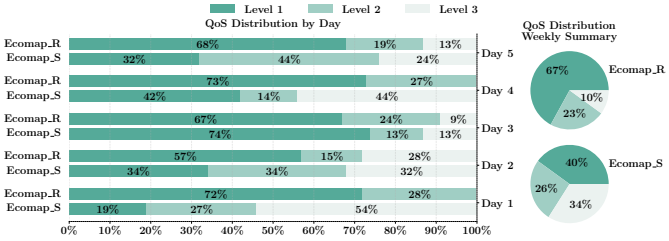


Fig. 9: Mixed-quality model usage by Ecomap in Week-2

to meet its stricter constraints. Over the week, $Ecomap_R$ processes an average of 58% of tasks with default models, while $Ecomap_S$ only achieves 33%, reflecting the additional adaptations required under tighter latency thresholds. $Ecomap_S$ demonstrates a higher reliance on lightweight models, with default models used the least on Day 1, accounting for only 20% of tasks. The analysis for Week-2 (Figure 9) also shows consistent model utilization behavior of Ecomap. $Ecomap_R$ utilized default (level-1) models for a significant portion of tasks (averaging 67%), whereas $Ecomap_S$ relies more heavily on lightweight alternatives, with 40% of tasks processed using level-1 models, demonstrating the strict constraints.

In Week-3 (Figure 10), with a medium workload intensity, there is a noticeable increase in the use of level-1 models for $Ecomap_S$, averaging 43% across the week compared to 33% in Week-1. This shift indicates that the reduced workload intensity allows $Ecomap_S$ to accommodate more tasks with default models while still adhering to its strict latency constraints. The increased use of level-1 models in Week-3 demonstrates how Ecomap efficiently balances workload demands and latency requirements while minimizing the need for lightweight alternatives under less intensive conditions.

In addition to showing that Ecomap$_R$ utilizes higher-quality models more frequently than Ecomap$_S$, Figures 8-10 reveal a

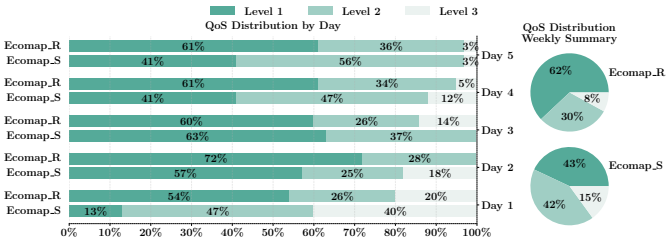


Fig. 10: Mixed-quality model usage by Ecomap in Week-3.

deeper insight into how model selection is influenced by the interplay between CI dynamics and workload complexity. For example, in Week 1, CI fluctuates heavily on day 1, while it remains relatively stable on day 2. This is reflected in the QoS distribution: higher-level models are used more on day 2 compared to day 1. *This suggests that CI variation, and not absolute CI, is a stronger driver of model selection decisions*.

C. Estimator Overhead

In our experiments, both $Ecomap_R$ and $Ecomap_S$ spent an average of 13 minutes per day on MCTS-based exploration and mapping evaluations, with each MCTS session taking about 30 seconds. This accounts for *less than 1% of the total daily runtime*. Due to the lightweight design of our estimator, the power consumption during exploration is minimal compared to the inference workload, resulting in negligible impact on runtime performance or energy usage. On average, the power budget was updated 3 times per day, with observed variations ranging from 2 to 5 changes. Similarly, operational mode changes occurred 5 times per day on average, ranging from 1 to 10. These values indicate that Ecomap adapts effectively to variations in workload and carbon intensity, while keeping reconfiguration frequency moderate and manageable.

VI. Conclusion

This paper introduces Ecomap, a sustainability-focused framework for managing multi-DNN workloads on edge devices. Unlike conventional methods that prioritize either throughput or power, Ecomap dynamically adjusts power thresholds based on carbon intensity (CI), balancing low latency and reduced emissions. Experiments show Ecomap outperforms existing methods in minimizing emissions and optimizing the carbon delay product under varying workloads and CI levels. While matching others in power efficiency, Ecomap excels in sustainability by adapting to real-time CI, maintaining latency, and using mixed-quality models.

Acknowledgments

This work is supported by grant NSF CCF 2324854.

References

[1] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai *et al.*, "Sustainable ai: Environmental implications, challenges and opportunities," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 795–813, 2022.

[2] A. M. Panteleaki and I. Anagnostopoulos, "Carbon-aware design of dnn accelerators: Bridging performance and sustainability," in *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2024, pp. 515–520.

[3] H. Amrouch, G. Zervakis, S. Salamin, H. Kattan, I. Anagnostopoulos, and J. Henkel, "Npu thermal management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3842–3855, 2020.

[4] U. Gupta, M. Elgamal, G. Hills, G.-Y. Wei, H.-H. S. Lee, D. Brooks, and C.-J. Wu, "Act: Designing sustainable computer systems with an architectural carbon modeling tool," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 784–799.

[5] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Chasing carbon: The elusive environmental footprint of computing," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 854–867.

[6] Y. G. Kim, U. Gupta, A. McCrabb, Y. Son, V. Bertacco, D. Brooks, and C.-J. Wu, "Greenscale: Carbon-aware systems for edge computing," *arXiv preprint arXiv:2304.00404*, 2023.

[7] I. Dagli and M. E. Belviranli, "Shared memory-contention-aware concurrent dnn execution for diversely heterogeneous system-on-chips," in *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, 2024, pp. 243–256.

[8] A. Karatzas and I. Anagnostopoulos, "Omniboost: Boosting throughput of heterogeneous embedded devices under multi-dnn workload," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023.

[9] A. Karatzas, D. Stamoulis, and I. Anagnostopoulos, "Rankmap: Priority-aware multi-dnn manager for heterogeneous embedded devices," in *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, 2025, pp. 1–7.

[10] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, and D. Marculescu, "Single-path nas: Designing hardware-efficient convnets in less than 4 hours," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 481–497.

[11] B. Li, S. Samsi, V. Gadepally, and D. Tiwari, "Clover: Toward sustainable ai with carbon-aware machine learning inference service," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–15.

[12] C.-Y. Hsieh, A. A. Sani, and N. Dutt, "The case for exploiting underutilized resources in heterogeneous mobile architectures," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1265–1268.

[13] S. Wang, G. Ananthanarayanan, Y. Zeng, N. Goel, A. Pathania, and T. Mitra, "High-throughput cnn inference on embedded arm big. little multicore processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2254–2267, 2019.

[14] M. Alhartomi, A. Salh, L. Audah, S. Alzahrani, and A. Alzahmi, "Enhancing sustainable edge computing offloading via renewable prediction for energy harvesting," *IEEE Access*, 2024.

[15] T. Zhou, J. Zhao, Y. Luo, X. Xie, W. Wen, C. Ding, and X. Xu, "Adapi: Facilitating dnn model adaptivity for efficient private inference in edge computing," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.

[16] E. Baek, D. Kwon, and J. Kim, "A multi-neural network acceleration architecture," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 940–953.

[17] C. Lim and M. Kim, "Odmdef: on-device multi-dnn execution framework utilizing adaptive layer-allocation on general purpose cores and accelerators," *IEEE Access*, vol. 9, pp. 85 403–85 417, 2021.

[18] D. Liu, S.-G. Yang, Z. He, M. Zhao, and W. Liu, "Cartad: Compiler-assisted reinforcement learning for thermal-aware task scheduling and dvfs on multicores," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 6, pp. 1813–1826, 2021.

[19] E. Aghapour, D. Sapra, A. Pimentel, and A. Pathania, "Arm-co-up: Arm co operative u tilization of p rocessors," *ACM Transactions on Design Automation of Electronic Systems*, 2024.

[20] A. Karatzas and I. Anagnostopoulos, "Mapformer: Attention-based multi-dnn manager for throughout & power co-optimization on embedded devices," in *2024 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2024.

[21] X. Wang, W. Fei, W. Dai, C. Li, J. Zou, and H. Xiong, "Mixed-precision deep neural network quantization with multiple compression rates," in *2023 Data Compression Conference (DCC)*. IEEE, 2023, pp. 371–371.

[22] B. A. Motetti, M. Risso, A. Burrello, E. Macii, M. Poncino, and D. J. Pagliari, "Joint pruning and channel-wise mixed-precision quantization for efficient deep neural networks," *IEEE Transactions on Computers*, 2024.

[23] K. Xu, X. Shao, Y. Tian, S. Yang, and X. Zhang, "Autompq: Automatic mixed-precision neural network search via few-shot quantization adapter," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024.

[24] X. Zhao, R. Xu, Y. Gao, V. Verma, M. R. Stan, and X. Guo, "Edge-mpq: Layer-wise mixed-precision quantization with tightly integrated versatile inference units for edge computing," *IEEE Transactions on Computers*, 2024.

[25] O. Spantidi, G. Zervakis, S. Alsalamin, I. Roman-Ballesteros, J. Henkel, H. Amrouch, and I. Anagnostopoulos, "Targeting dnn inference via efficient utilization of heterogeneous precision dnn accelerators," *IEEE Transactions on Emerging Topics in Computing*, vol. 11, no. 1, pp. 112–125, 2022.

[26] K. Sankaranarayanan, R. B. Roy, and D. Tiwari, "Pulse: Using mixed-quality models for reducing serverless keep-alive cost," in *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024, pp. 99–109.

[27] W. A. Hanafy, Q. Liang, N. Bashir, A. Souza, D. Irwin, and P. Shenoy, "Going green for less green: Optimizing the cost of reducing cloud carbon emissions," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2024, pp. 479–496.

[28] P. Wiesner, I. Behnke, D. Scheinert, K. Gontarska, and L. Thamsen, "Let's wait awhile: How temporal workload shifting can reduce carbon emissions in the cloud," in *Proceedings of the 22nd International Middleware Conference*, 2021, pp. 260–272.

[29] Y. Yang, Y. Chen, K. Li, and J. Huang, "Carbon-aware dynamic task offloading in noma-enabled mobile edge computing for iot," *IEEE Internet of Things Journal*, 2024.

[30] Z. Song, M. Xie, J. Luo, T. Gong, and W. Chen, "A carbon-aware framework for energy-efficient data acquisition and task offloading in sustainable aiot ecosystems," *IEEE Internet of Things Journal*, 2024.

[31] H. Ke, W. Jin, and H. Wang, "Carboncp: Carbon-aware dnn partitioning with conformal prediction for sustainable edge intelligence," *arXiv preprint arXiv:2404.16970*, 2024.

[32] "Electricity maps: Live and forecasted electricity emissions data." [Online]. Available: https://app.electricitymaps.com/

[33] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[34] M. Han, J. Hyun, S. Park, J. Park, and W. Baek, "Mosaic: Heterogeneity-, communication-, and constraint-aware model slicing and execution for accurate and efficient inference," in *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, 2019, pp. 165–177.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[36] R. Hu and A. Singh, "Unit: Multimodal multitask learning with a unified transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1439–1449.

[37] L. Wang, R. Fonseca, and Y. Tian, "Learning search space partition for black-box optimization using monte carlo tree search," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 511–19 522, 2020.

[38] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, "Monte carlo tree search: A review of recent modifications and applications," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2497–2562, 2023.

[39] D. Maji, P. Shenoy, and R. K. Sitaraman, "Carboncast: multi-day forecasting of grid carbon intensity," in *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2022, pp. 198–207.

[40] G. J. Miller, K. Novan, and A. Jenn, "Hourly accounting of carbon emissions from electricity consumption," *Environmental Research Letters*, vol. 17, no. 4, p. 044073, 2022.

[41] A. Al-Dhamari, R. Sudirman, and N. H. Mahmood, "Transfer deep learning along with binary support vector machine for abnormal behavior detection," *Ieee Access*, vol. 8, pp. 61 085–61 095, 2020.

[42] W. Hayale, P. S. Negi, and M. H. Mahoor, "Deep siamese neural networks for facial expression recognition in the wild," *IEEE Transactions on Affective Computing*, vol. 14, no. 2, pp. 1148–1158, 2021.

[43] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 153–167.

Varatheepan Paramanayakam received the Bachelor of Science of Engineering degree from the Department of Electronic and Telecommunication Engineering, University of Moratuwa, Sri Lanka, in 2021. He is currently pursuing the Doctor of Philosophy degree at the School of Electrical, Computer and Biomedical Engineering at Southern Illinois University, Carbondale, Illinois, as a member of the Embedded Systems Software Laboratory. His research interests include embedded systems, sustainable AI, and deep learning.

Andreas Karatzas received the Integrated Master degree (Diploma) from the department of Computer Engineering and Informatics (CEID), University of Patras, Patras, Greece, in 2021. He is currently pursuing the Ph.D. degree at the School of Electrical, Computer and Biomedical Engineering at Southern Illinois University, Carbondale, Illinois, as a member of the Embedded Systems Software Lab. His research interests include embedded systems, approximate computing, and deep learning.

Dimitrios Stamoulis is a Special Faculty member in the Dept. of Electrical and Computer Engineering (ECE) at The University of Texas at Austin, Austin, TX. Previously, he founded and led the *CoStrategist* R&D Group at Microsoft Mixed Reality. He received his PhD in ECE from Carnegie Mellon University, where he specialized on hardware-aware AutoML. He also holds a MEng in ECE from McGill University and a Diploma in ECE from the National Technical University of Athens.

Iraklis Anagnostopoulos is an Associate Professor at the School of Electrical, Computer and Biomedical Engineering at Southern Illinois University, Carbondale. He is the director of the Embedded Systems Software Lab, which works on run-time resource management of modern and heterogeneous embedded many-core architectures. He received his Ph.D. in the Microprocessors and Digital Systems Laboratory of National Technical University of Athens. His research interests lie in the area of machine learning and heterogeneous hardware accelerators.