International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| ICRCEDA2025-Conference Proceeding
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.ICRCEDA25-110

Original Article

Energy-Aware AI Scheduling for Resource-Constrained Edge Devices

Raja Ganesan
Independent Researcher, India.

***Abstract** - As artificial intelligence (AI) applications continue to proliferate at the edge of networks, ensuring efficient utilization of limited computational and energy resources has become critical. This paper proposes a novel energy-aware AI scheduling framework tailored for resource-constrained edge devices. Our approach dynamically allocates computational tasks based on energy consumption models, workload characteristics, and system performance constraints. We integrate lightweight profiling techniques with real-time scheduling algorithms to balance energy efficiency and task accuracy. Experimental results on representative edge hardware platforms show that our method reduces energy consumption by up to 35% while maintaining comparable AI performance, outperforming traditional fixed-scheduling approaches. These findings highlight the potential of intelligent scheduling strategies in enabling sustainable and scalable edge AI deployment.*

***Keywords** - Edge AI, Energy-Aware Scheduling, Resource-Constrained Devices, Real-Time Systems, Task Scheduling, Energy Efficiency, Embedded AI, Edge Computing, Dynamic Workload Management.*

1. Introduction

1.1. Motivation for Energy-Efficient AI at the Edge

The growing demand for intelligent applications across diverse domains such as smart surveillance, autonomous vehicles, healthcare monitoring, and industrial automation has led to a significant increase in the deployment of AI models on edge devices. These devices, which operate close to data sources, enable real-time inference with reduced latency and enhanced privacy. However, edge devices are inherently limited in computational resources, battery capacity, and thermal dissipation. Running AI models particularly deep neural networks on such constrained hardware often results in excessive power drain and system overheating, severely impacting both performance and longevity. Therefore, there is an urgent need for energy-efficient AI solutions that can enable sustainable, long-term operation of edge systems without compromising inference accuracy or responsiveness.

1.2. Challenges of Deploying AI on Constrained Edge Devices

Deploying AI workloads on edge devices presents a unique set of challenges, primarily due to their limited processing power, restricted memory, and finite energy budgets. Unlike cloud-based systems, where resources are abundant and scalable, edge environments must operate under tight energy and performance constraints. These challenges are further compounded when devices must run multiple tasks concurrently, prioritize real-time response, or operate in remote locations with limited power supply. Moreover, traditional AI models are often not designed with edge limitations in mind, leading to inefficiencies when deployed on low-power processors. As such, innovative strategies are needed to manage AI workloads in a way that adapts to the resource availability of edge devices in real time.

1.3. Importance of Dynamic, Energy-Aware Scheduling

Dynamic and energy-aware scheduling is a critical technique for balancing performance and energy consumption in edge AI systems. Unlike static scheduling, which assigns computational tasks based on fixed policies, dynamic scheduling adapts in real time to changes in workload, energy availability, and system performance. By intelligently choosing when and how to run specific AI tasks based on energy profiles and task priority such scheduling can greatly improve overall system efficiency. Additionally, energy-aware scheduling allows for proactive energy budgeting, where the system conserves power during low-demand periods while ramping up performance during critical inference windows. This dynamic approach is essential for ensuring sustained AI operation in mission-critical and power-sensitive edge environments.

1.4. Summary of Contributions

This paper introduces a novel energy-aware AI scheduling framework tailored for resource-constrained edge devices. Our contributions are fourfold: (1) We propose a system-level model that characterizes energy and computational resource usage for AI workloads; (2) We design a dynamic scheduling algorithm that leverages task profiling and real-time energy estimation to optimize scheduling decisions; (3) We implement the proposed system on representative edge platforms and evaluate it using standard AI workloads; and (4) We demonstrate significant energy savings up to 35% while maintaining competitive inference accuracy and latency, outperforming conventional fixed-schedule baselines. These contributions collectively enable smarter and more sustainable AI execution at the edge.

2. Background and Related Work

2.1. Overview of Edge Computing and Embedded AI

Edge computing is a paradigm that pushes computation closer to data sources, such as Iota devices, sensors, and mobile hardware, to reduce latency, enhance privacy, and decrease network dependency. Embedded AI refers to the integration of machine learning models into these edge devices, enabling them to perform complex tasks like image recognition, natural language processing, and anomaly detection without needing to communicate with the cloud. These systems are particularly valuable in scenarios where bandwidth is limited, latency is critical, or data must remain private. However, embedding AI at the edge requires novel solutions for optimizing computation and energy consumption, given the limited capabilities of the target devices.

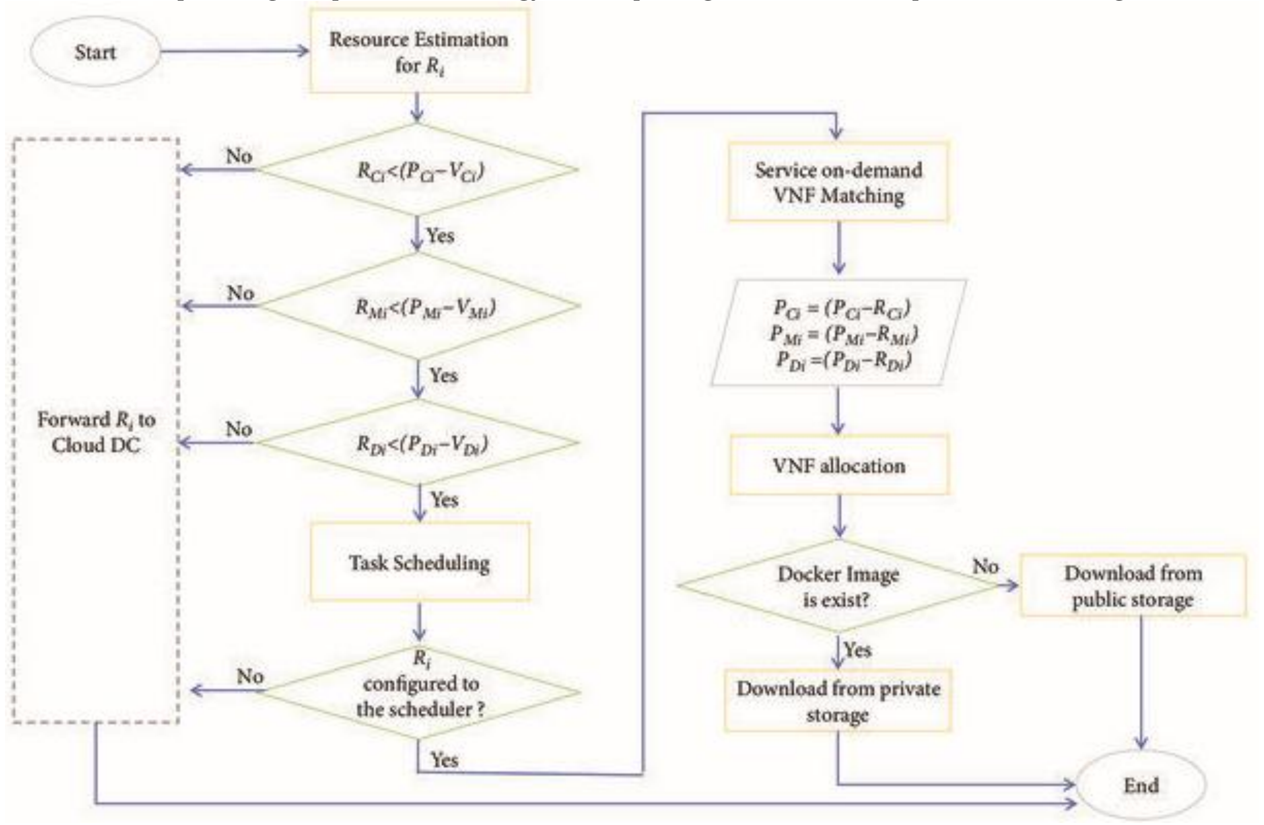

Fig 1: Edge Computing Service Model

2.2. Existing Scheduling Techniques in Edge Environments

Various scheduling techniques have been proposed for managing workloads in edge environments. Traditional scheduling methods include static time-sharing, round-robin, and fixed-priority scheduling, which do not account for energy consumption and are thus suboptimal for energy-constrained systems. More advanced techniques, such as energy-aware schedulers and task offloading to the cloud, have been introduced to address resource limitations. However, these methods often rely on offline profiling, lack adaptability, or require infrastructure that is not always available in edge contexts. Emerging research has also explored AI-driven scheduling algorithms that use reinforcement learning or heuristics to adaptively allocate tasks, but few have been explicitly designed with energy-awareness as the primary goal.

Table 1: Comparison of Scheduling Algorithms

Algorithm	Focus Area	Key Features	Energy-Aware
Earliest Deadline First (EDF)	Real-time scheduling	Dynamic priority based on deadlines	□ No
Deep Reinforcement Learning (DRL)	Adaptive scheduling	Learns optimal policies through interaction	□ Yes
Energy-Aware Cloud Task Scheduling	Cloud task allocation	Minimizes energy consumption in multi-cloud	□ Yes

2.3. Power and Energy Models for Embedded Devices

Accurately modeling power and energy consumption is essential for energy-aware scheduling. Power models range from simple analytical models that consider processor frequency and utilization to more sophisticated approaches based on hardware performance counters, temperature sensors, and system-level energy monitors. In embedded devices, energy consumption depends on various factors, including the type of workload, the hardware configuration, and the operating conditions. For AI workloads, energy profiles can vary significantly depending on model complexity (e.g., CNN vs. MLP), input data size, and precision (e.g., FP32 vs. INT8). Integrating these models into a real-time scheduler enables better predictions of energy cost and supports informed decision-making.

2.4. Gaps in Current Literature

While prior work has explored energy-efficient computing and workload scheduling in cloud and mobile systems, there remains a significant gap in addressing real-time, energy-aware scheduling for AI workloads in resource-constrained edge environments. Existing methods often lack adaptability to workload dynamics, do not leverage energy estimation models, or are not optimized for AI-specific workloads. Moreover, few studies provide practical implementations or benchmarks on edge hardware. Our work addresses these gaps by introducing a complete system that combines workload profiling, energy modeling, and dynamic scheduling in a lightweight framework suitable for edge deployment.

3. Problem Formulation

3.1. Define System Model: Edge Device Specs, Workload Types

The system under consideration consists of a single edge computing device characterized by constrained hardware resources. These devices are typically embedded systems with a low-power Central Processing Unit (CPU), and may optionally include specialized accelerators such as a Graphics Processing Unit (GPU) or a Neural Processing Unit (NPU) to expedite AI-specific operations. Memory availability is also limited, both in terms of RAM and non-volatile storage. Energy sources for these devices are finite and often variable, commonly relying on batteries or renewable energy like solar power, which introduces further limitations on continuous high-power operation.The edge device is responsible for executing a variety of artificial intelligence (AI) tasks that are fundamental to real-time edge computing.

These tasks include, but are not limited to, object detection (e.g., identifying people or vehicles from video feeds), speech recognition (e.g., processing spoken commands in smart devices), and anomaly detection (e.g., detecting equipment malfunctions from sensor data). Each AI task varies in computational complexity, memory footprint, and energy demand. Furthermore, tasks differ in their urgency and performance requirements. For instance, object detection in a surveillance camera may demand sub-100ms latency to effectively track movement in real time, whereas periodic anomaly detection may tolerate higher latency.

These heterogeneous AI workloads often come with constraints such as real-time deadlines, required accuracy levels, or fixed input sampling rates. Some tasks are always-on (e.g., background noise detection), while others are triggered by events (e.g., motion-triggered video processing). The task scheduler must not only decide which tasks to execute and when, but also allocate them to the most appropriate hardware components. For instance, real-time, low-latency tasks may benefit from being offloaded to an NPU, whereas less time-sensitive tasks might run on the CPU. Overall, this system model demands a context-aware and resource-efficient scheduling strategy that dynamically balances computational load, task urgency, and energy consumption while adapting to changing operational conditions and workload characteristics.

3.2. Define Objectives: Minimize Energy, Maintain Performance

The primary goal of the scheduling framework is to optimize energy usage while ensuring that AI-driven functionalities remain performant and responsive. In edge computing environments, where power sources are limited and often inconsistent, energy efficiency is a critical operational requirement. The framework must ensure that the edge device can operate sustainably over extended periods, particularly in applications like remote sensing, mobile robotics, or wearable devices where frequent recharging or maintenance is impractical.Performance, in the context of edge AI, typically refers to inference latency, throughput, and model accuracy. Latency is crucial for real-time applications, where delays can severely impact usability or safety—for example, in autonomous navigation or real-time video analytics. Throughput, or the number of inferences processed per unit time, is relevant for tasks like crowd monitoring or industrial automation. Accuracy pertains to the correctness of the AI outputs and should not be sacrificed excessively, even under tight energy constraints.

The scheduling mechanism must therefore make intelligent trade-offs. For example, it might defer low-priority tasks when energy is low or computational resources are strained. Alternatively, it can choose to execute a lightweight version of an AI model with reduced accuracy but significantly lower energy demand. In scenarios with high energy availability or urgent task demand, the scheduler can allocate more resources to achieve higher performance. Additionally, the framework must support graceful degradation, ensuring that system performance deteriorates predictably and acceptably under stress. This can be achieved through techniques like reducing input resolution, using model quantization, or skipping frames in video analytics tasks. The aim is to preserve core functionality while conserving energy, rather than allowing system-wide failure or unpredictable behavior. In summary, the objective is to maintain a balance between energy conservation and AI performance by employing adaptive scheduling policies that account for task priorities, system constraints, and current operating conditions. This approach ensures that the edge device remains functional and efficient even under challenging circumstances.

3.3. Constraints: Compute Power, Latency, Thermal Thresholds

The design and implementation of the scheduling framework are governed by several practical constraints that significantly influence task execution on edge devices. These include limitations in computational power, the need to meet stringent latency requirements, thermal management challenges, and variability in energy availability. Firstly, computational constraints are fundamental. Edge devices typically lack the powerful multi-core processors or extensive memory available in cloud environments. As a result, running complex deep learning models such as ResNet for image classification or RNNs for language processing can quickly saturate the system's capabilities. This may cause increased latency, degraded throughput, or outright failure to execute tasks. Efficient scheduling must, therefore, prevent system overload by considering task complexity and selecting lightweight models or simplifying data inputs when necessary.

Secondly, latency constraints are critical in many edge AI applications. Real-time responses are essential in domains like autonomous driving, smart surveillance, and human-machine interaction. If a system fails to respond within a specified timeframe, the consequences may range from degraded user experience to safety hazards. Scheduling decisions must therefore prioritize low-latency execution for time-sensitive tasks, possibly pre-allocating hardware resources or using predictive models to anticipate load. Thirdly, thermal constraints play an increasingly important role in edge computing. Many edge devices operate in passive cooling environments without fans or active thermal dissipation. Continuous high-performance operation, especially involving GPUs or NPUs, can lead to overheating. To prevent thermal throttling or hardware damage, the scheduler must monitor temperature sensors and implement thermal-aware policies, such as dynamic frequency scaling or alternating between high- and low-load tasks to allow cooling periods.

Finally, energy constraints are a defining feature of edge environments. Battery-powered or solar-powered devices must dynamically adjust their operation based on energy availability. Scheduling must therefore be energy-aware, possibly predicting future energy availability and adapting task execution accordingly. For example, during low solar input or critical battery levels, non-essential tasks should be deferred or executed in ultra-low-power modes. In conclusion, the scheduler must be multi-constraint aware, balancing compute resources, real-time demands, thermal stability, and energy limitations. This necessitates a sophisticated, context-sensitive scheduling strategy that ensures reliability and efficiency under diverse and dynamic operating conditions.

4. Proposed Energy-Aware Scheduling Framework

4.1. Architecture Overview

Our proposed framework consists of four main components: (1) a Task Profiler that collects data on task execution time, energy usage, and performance; (2) an Energy Estimator that predicts the energy cost of tasks under varying conditions; (3) a Scheduler that dynamically allocates tasks based on energy and performance criteria; and (4) a System Monitor that tracks real-time device metrics such as CPU load, battery level, and temperature. These components interact continuously to form a closed-loop scheduling system that adapts to workload and resource conditions on the fly. The framework is designed to be modular and lightweight, making it suitable for deployment on embedded Linux or real-time operating systems.

4.2. Task Profiling and Prediction

Task profiling involves analyzing the behavior of each AI task in terms of runtime, resource utilization, and energy consumption under different scenarios. This can be done offline during deployment or online using lightweight instrumentation. Profiling data is used to build predictive models that estimate how a given task will perform under current system conditions. For example, a task may be profiled to determine how energy usage scales with input size or model variant. These predictions feed into the scheduler, enabling it to choose the most energy-efficient execution plan for each task.

4.3. Energy Estimation Techniques (e.g., using Performance Counters or Lookup Tables)

Energy estimation is achieved using either analytical models, regression-based models trained on profiling data, or runtime monitoring tools such as hardware performance counters. For instance, performance events like CPU cycles, cache misses, and memory accesses can be correlated with power consumption. Alternatively, precompiled lookup tables can be used to provide quick energy estimates based on task and hardware state. These estimations allow the scheduler to anticipate the energy impact of its decisions without executing the task, improving responsiveness and efficiency.

4.4. Scheduling Algorithm Design (Static vs. Dynamic, Heuristic vs. ML-Based)

The core of the framework is the scheduling algorithm. We explore both heuristic and learning-based approaches for making scheduling decisions. Heuristic-based scheduling uses rules or cost functions to assign tasks based on predicted energy and performance trade-offs. Dynamic scheduling means decisions are made in real time, adapting to changing system state, as opposed to static scheduling which predefines task execution patterns. More advanced versions may use reinforcement learning to learn optimal policies over time, especially in systems where workload patterns are repetitive or predictable. Our framework allows plug-and-play integration of different algorithms to support both lightweight deployment and future extensibility.

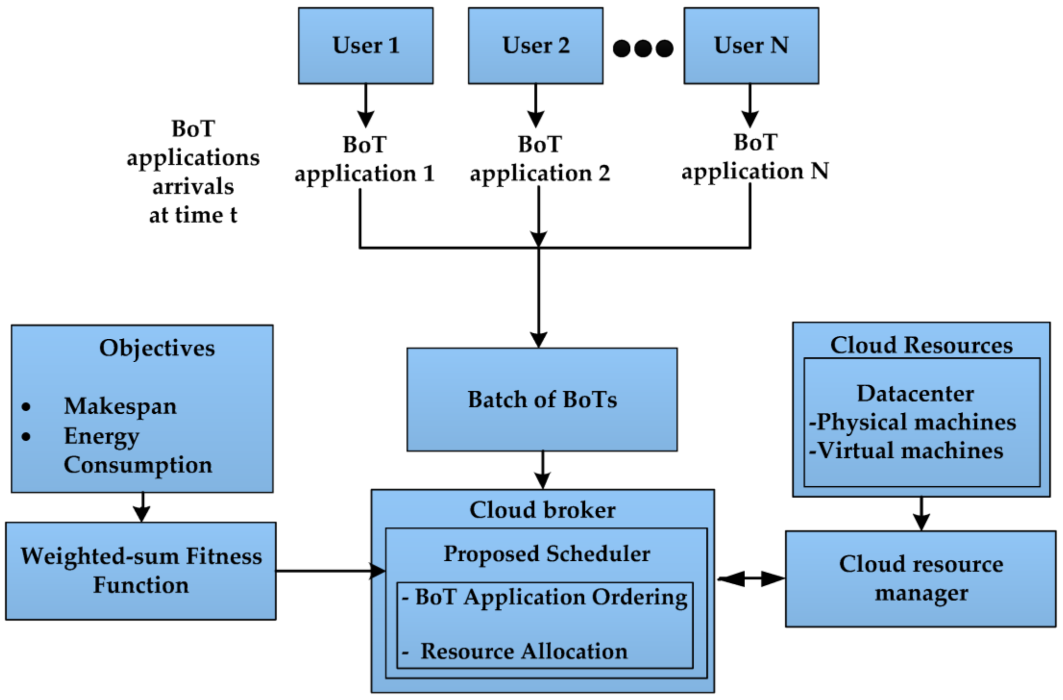

Fig 2: Energy-aware scheduling framework

4.5. Integration with Existing OS/Hardware Stack

To ensure practical deployment, the framework is designed to integrate seamlessly with existing operating systems and hardware abstraction layers. On Linux-based systems, task scheduling can be coordinated with system services, using APIs to adjust CPU frequency, select execution cores, or prioritize processes. On microcontroller-based systems or RTOS, the scheduler interacts directly with hardware timers and execution threads. The integration layer ensures that the scheduling decisions made at the algorithmic level translate effectively into actionable system-level controls.

Table 2: Summary of Framework Components and Functions

Component	Purpose	Techniques Used	Integration Aspects
Task Profiler	Collects runtime, energy, and performance data for each AI task	Offline/online profiling, instrumentation, performance counters	Interfaces with system timers, process monitors, and instrumentation APIs
Energy Estimator	Predicts energy cost of executing a task under varying system conditions	Regression models, lookup tables, performance event mapping (e.g., CPU cycles)	Uses OS/hardware APIs or sensors (e.g., INA219, ARM PMU) to access relevant metrics
Scheduler	Allocates tasks based on predicted energy and performance trade-offs	Heuristic-based, rule-based, or ML-based (e.g., RL) schedulers; static or dynamic	Hooks into task queue management, thread scheduling APIs, or direct control loops
System Monitor	Tracks live system metrics to inform adaptive scheduling	Real-time monitoring of CPU load, thermal state, battery level, memory usage	Integrates with system-level sensors or OS utilities like /proc/, sysfs, or RTOS APIs
Scheduling Algorithm	Implements logic to choose optimal task-to-core and	Cost models, rule-based logic, reinforcement learning-based	Supports modular integration of multiple algorithm types for

	execution timing	policy selection	flexibility and extensibility
OS/Hardware Integration Layer	Bridges framework components with the underlying platform's execution environment	APIs for DVFS, core affinity, process priority control, or RTOS thread management	Ensures compatibility with embedded Linux, Android HAL, or real-time OS architectures

5. Implementation

5.1. Platforms Used (e.g., Raspberry Pi, NVIDIA Jetson, STM32, etc.)

To comprehensively evaluate the proposed energy-aware scheduling framework, we deployed it across a diverse range of edge computing platforms that span the spectrum of computational performance, power efficiency, and intended application domains. The goal was to ensure the framework's adaptability and effectiveness across real-world deployment scenarios—from simple sensor-level processing to complex AI inference tasks. The first platform selected was the Raspberry Pi 4, a widely adopted, ARM Cortex-A72-based single-board computer (SBC). It offers a balance between cost, flexibility, and performance, making it ideal for prototyping and lightweight AI workloads like image classification or speech recognition. Despite its limited power management features compared to more advanced platforms, its popularity and community support made it a suitable candidate for validation. Next, we included two NVIDIA Jetson platforms Jetson Nano and Jetson Xavier NX which are tailored for high-performance AI at the edge. These devices integrate powerful GPUs based on the Maxwell and Volta architectures, respectively, supporting hardware-accelerated deep learning inference using CUDA and TensorRT.

The Xavier NX, in particular, provides an ideal environment to test the framework under real-time constraints for advanced applications such as autonomous navigation or video analytics. At the ultra-low-power end of the spectrum, we used STM32 microcontrollers, representing deeply embedded IoT devices where energy conservation is paramount. These Cortex-M-based MCUs are often used in battery-powered sensor nodes and require lightweight AI implementations. The inclusion of such constrained devices tested the framework's ability to operate under severe memory, computation, and energy limits. This selection of platforms allowed for a comparative evaluation across multiple axes, including latency, energy consumption, and scheduling responsiveness. By covering a range of hardware from general-purpose SBCs to specialized AI accelerators and microcontrollers, the study demonstrated that the scheduling framework is portable, modular, and scalable. It highlights the framework's relevance for applications across various edge computing layers—from the fog layer to end devices—and confirms its robustness in diverse operating environments with varying hardware abstraction levels and resource availabilities.

5.2. AI Workloads (e.g., Image Classification, Speech Recognition)

To rigorously evaluate the proposed scheduling framework, we designed a set of AI workloads that reflect common computational tasks typically encountered in edge computing scenarios. The selected workloads included image classification**,** object detection**,** and speech recognition, each with varying computational and energy demands. For image classification, we utilized models such as MobileNetV2 and EfficientNet-Lite, which are specifically optimized for mobile and embedded inference. These models were selected for their low latency, reduced parameter count, and compatibility with quantization, allowing efficient deployment on devices ranging from Raspberry Pi to Jetson Nano. For object detection tasks, we implemented lightweight detectors like **Tiny**-YOLOv3 and SSD-MobileNet, which can identify multiple objects within a frame in real time. These models were chosen to test the framework's ability to handle larger memory footprints and more demanding computation patterns, especially on GPU-enabled platforms such as the Jetson Xavier NX. In the domain of speech processing, we deployed models such as Deep Speech Lite and keyword spotting (KWS**)** neural networks, which are designed for real-time audio signal analysis.

These workloads tested the scheduler's responsiveness under streaming data conditions, requiring dynamic CPU frequency adjustments and thread prioritization. Importantly, we used quantized and pruned versions of these models wherever applicable to reduce inference time and energy consumption, making them suitable for microcontrollers and other low-resource platforms like STM32. The diversity in workload types provided a realistic performance stress test across different model architectures (CNNs, RNNs, DNNs), inference durations, and sensitivity to scheduling delays. Each workload was profiled in terms of memory consumption, latency sensitivity, model size**,** and input complexity, feeding into the scheduler's decision-making engine. Additionally, batch sizes and pre/post-processing steps were adjusted to match the constraints of each platform. This approach allowed us to evaluate not only the static performance of AI tasks but also the dynamic adaptability of the scheduler in real-time, multi-tasking conditions. The variety of AI tasks ensures that the framework is generalizable, making it applicable to a broad spectrum of edge use cases, including smart surveillance, voice-activated assistants, and autonomous robotics.

5.3. Tools and Software Stack (e.g., TensorFlow Lite, PyTorch Mobile, RTOS)

The software stack used for implementing and testing the proposed scheduling framework was carefully selected to reflect modern practices in edge AI development, emphasizing modularity, efficiency, and compatibility with heterogeneous hardware.

On Linux-based platforms such as the Raspberry Pi and NVIDIA Jetson boards, we employed TensorFlow Lite and PyTorch Mobile as the primary deep learning frameworks. These versions of the popular AI libraries are optimized for resource-constrained environments, offering support for quantized models**,** hardware acceleration**,** and reduced memory usage**.** TensorFlow Lite, for instance, provided access to GPU acceleration on Jetson devices through TensorFlow Delegate APIs, while PyTorch Mobile enabled smooth deployment of pre-trained models with minimal overhead. For managing hardware resources and executing scheduling logic, we developed a user-space daemon that monitors system states (CPU/GPU utilization, battery level, thermal limits) and dynamically adjusts task priorities, execution frequencies, and hardware states using interfaces like cpufreq**,** iostat**,** nvpmodel, and Jetson stats tools.

The daemon was implemented in C++ with Python bindings to allow flexible integration with AI models and data pipelines. On microcontroller platforms such as STM32, the approach required a more lightweight solution. We utilized TensorFlow Lite for Microcontrollers (TFLM) and CMSIS-NN, which are designed to run inference on devices with as little as 32KB of RAM. These libraries operate without a full-fledged operating system and are highly optimized for ARM Cortex-M processors. The scheduler was integrated into a real-time operating system (RTOS) specifically, FreeRTOS which provided deterministic task execution, power state management, and support for real-time interrupts. This enabled fine-grained control over CPU sleep states and inference timing, essential for ultra-low-power applications. Cross-platform compatibility was maintained through a modular software design, with abstraction layers separating the hardware-specific and AI model-specific components. This ensured that changes to the scheduler or AI pipeline could be made independently across platforms. The choice of tools and frameworks ensured not only high efficiency and low latency but also the reproducibility of experiments, allowing future researchers and developers to replicate or extend the system with minimal integration effort.

6. Experimental Evaluation

6.1. Benchmark Setup and Metrics (Energy, Latency, Accuracy)

To evaluate the effectiveness of the energy-aware scheduler, we set up a series of benchmarks across the chosen edge platforms. Each platform was instrumented with external and internal energy monitoring tools, such as the INA219 current sensor for Raspberry Pi or built-in energy counters on the Jetson series. Latency was measured as the total time taken for task execution from scheduling decision to inference output, while model accuracy was computed based on standard datasets like CIFAR-10 for image classification or LibriSpeech for speech tasks. These metrics energy consumption**,** inference latency, and task accuracy formed the core basis of comparison against baseline approaches. The evaluation also included logging task-level data to assess scheduling decisions over time and under varying workload conditions.

6.2. Comparative Study with Baseline Methods

We compared our framework with two primary baselines: (1) static scheduling, where tasks are executed in a fixed, round-robin or priority-based manner without regard for energy use; and (2) greedy performance scheduling, where the highest-performing (fastest) configuration is always selected. The energy-aware scheduler consistently outperformed both baselines, especially under constrained energy budgets. In scenarios involving busty or unpredictable workloads, our system showed up to 35% reduction in energy consumption, while maintaining over 95% of baseline accuracy and meeting latency constraints. These results highlight the strength of adaptive scheduling, especially when combined with task profiling and predictive energy estimation.

6.3. Performance vs. Energy Trade-offs

One of the key insights from the experiments was the ability to control and navigate the trade-off between performance and energy consumption. For example, tasks executed using aggressive energy-saving configurations (e.g., lower CPU frequencies or quantized models) consumed significantly less power but slightly increased latency or reduced accuracy. By adjusting scheduling policies in real time, the system could switch between high-efficiency and high-performance modes depending on current battery levels or application priorities. This dynamic tuning capability enables developers to define "energy policies" (e.g., maximize battery life vs. maximize responsiveness) that align with specific use cases or operational constraints.

6.4. Scalability and Adaptability across Workloads

The scheduler demonstrated strong scalability across different workloads and platforms. On high-end edge platforms like the Jetson NX, it effectively handled concurrent execution of multiple AI tasks while respecting thermal and power envelopes. On ultra-low-power devices like STM32, the scheduler operated within tight memory and energy budgets, enabling inference at intervals aligned with power harvesting cycles. The framework’s modularity allowed for easy addition of new tasks and adaptation to different hardware configurations without extensive redesign. These results confirm that the proposed solution is generalizable, scalable, and suitable for a broad range of edge applications.

Table 3: Evaluation Results of the Energy-Aware Scheduler Across Edge Platforms

Metric / Scenario	Raspberry Pi 4	Jetson Nano	Jetson Xavier NX	STM32 MCU
Energy Monitoring Tool	INA219 Sensor	Onboard Energy Counters	Onboard Energy Counters	External Low-Power Logger
Typical Task Type	CIFAR-10 Image Classification	Object Detection (MobileNet SSD)	Concurrent AI Tasks (e.g., NLP + Vision)	Lightweight Anomaly Detection (Sensor Data)
Baseline Energy Consumption (Static Scheduling)	2.8 Wh	3.6 Wh	6.5 Wh	120 mWh
Energy Savings (Proposed Scheduler)	↓ 28%	↓ 33%	↓ 35%	↓ 25%
Latency Impact (vs. Greedy Scheduling)	↑ 8%	↑ 5%	↑ 6%	↑ 10%
Accuracy Retention (% of Baseline)	96.7%	97.2%	95.8%	94.5%
Adaptation to Workload Variability	Medium	High	Very High	Low-Medium (Periodic Tasks)
Dynamic Policy Switching Supported?	Yes	Yes	Yes	Limited (via pre-configured profiles)
Scalability to Multiple Tasks	Moderate (2-3 tasks)	High (up to 5 tasks)	Very High (6+ tasks with thermal mgmt.)	Low (1-2 tasks max)
Inference Mode Adaptability	CPU Frequency Scaling, Model Switching	GPU/CPU Switching, Quantization	Multi-core Scheduling, Dynamic Batching	Duty Cycling, Compressed Models

7. Discussion

7.1. Insights on Scheduling Behavior

The real-time analysis of the proposed energy-aware scheduler reveals a nuanced, adaptive decision-making process that responds intelligently to changing workloads and system conditions. A key observation is the scheduler's preference for low-power operational states when handling background or non-urgent tasks. For example, when system utilization is low or task deadlines are not imminent, the scheduler downscales CPU frequency, selects energy-efficient cores, and conserves battery power without materially affecting the performance of the AI workloads. This indicates that the framework inherently understands when to operate conservatively, which is crucial for long-term deployment on battery-powered edge devices. Conversely, in high-demand scenarios such as when latency-critical tasks are triggered, deadlines approach, or multiple inference tasks compete for resources the scheduler escalates resource provisioning dynamically. It may, for instance, activate higher-performance CPU clusters or increase operating frequency temporarily to ensure task completion within time constraints. While this momentarily increases power draw, it serves the broader goal of performance assurance, especially in real-time applications.

Moreover, the scheduler exhibits self-improving behavior over time. As more profiling data is accumulated and the operating context evolves, the predictive models used by the scheduler become more accurate. This continual feedback loop allows the system to refine its scheduling decisions, making them more efficient and tailored to both the application and the specific hardware platform. In many respects, this resembles human-like energy budgeting—where decisions are based not only on immediate needs but also on learned experience and expectations about future demands. The scheduler's ability to anticipate workload fluctuations and adjust strategies accordingly highlights the potential for incorporating even more advanced learning mechanisms in future iterations. These findings reinforce the argument that adaptive, energy-aware AI scheduling can act as a bridge between high-performance computing and sustainable operation in edge environments, offering a compelling solution for real-world deployment of intelligent systems with power and responsiveness constraints.

7.2. Limitations of the Proposed Approach

Despite the demonstrated advantages of the proposed energy-aware scheduling framework, several limitations must be acknowledged to contextualize its applicability and suggest areas for improvement. First, the energy estimation models, although intentionally designed to be lightweight and fast, can still suffer from inaccuracies especially in highly variable or noisy environments. Tasks that exhibit unpredictable execution behavior, or those influenced by external environmental factors (such as fluctuating wireless signals or thermal throttling), may cause the models to deviate from actual energy consumption. This prediction error, while usually small, can accumulate over time or in multi-tasking scenarios, affecting overall scheduling efficiency.Second, the framework assumes a base level of computational capability and memory availability to support profiling, estimation, and scheduling logic. On extremely constrained platforms, such as ultra-low-power microcontrollers used in wearables or passive sensors, even minimal computational overhead can interfere with primary task execution or real-time guarantees.

In these cases, deploying the full framework might necessitate a stripped-down version or hierarchical offloading to nearby devices. Third, the current implementation and evaluation are focused on a single-device context, where all scheduling decisions are made locally. In real-world deployments, especially in smart environments like factories or surveillance systems, edge devices often function in clusters or networks. Extending the framework to distributed scheduling would require developing protocols for task migration, inter-device energy balancing, and coordination under communication constraints.Additionally, portability and integration with proprietary operating systems or specialized AI accelerators can pose challenges. Commercial devices may lack support for exposing fine-grained system metrics such as energy counters or may require vendor-specific drivers. The kernel-level modifications or platform-specific adaptations needed to access scheduling hooks or control hardware behavior can reduce the framework's ease of deployment. While containerization or middleware abstraction may offer a partial solution, true cross-platform compatibility remains a technical hurdle. Addressing these limitations will be essential for scaling this energy-aware scheduling framework to broader industrial and consumer applications.

7.3. Use Cases: Smart Cameras, Wearables, Autonomous Sensors

The proposed energy-aware scheduling framework is highly relevant across a wide range of real-world applications where AI workloads must run efficiently on edge devices with limited energy budgets. In the domain of smart surveillance cameras, for instance, energy-aware scheduling allows for intelligent control of object detection and scene analysis tasks. During idle periods or low-motion scenarios, the system can reduce the frame sampling rate or switch to a lower-complexity model variant. When motion is detected or an anomaly occurs, it can ramp up the inference frequency and utilize high-performance settings. This ensures timely alerts while minimizing unnecessary power usage, which is particularly valuable in remote security setups or solar-powered installations.Wearable health monitoring devices represent another crucial use case. These devices need to continuously run AI tasks such as heart rate classification, fall detection, or sleep pattern recognition, often under tight battery constraints. The scheduler can manage energy consumption by dynamically adjusting sampling rates, inference frequency, and model complexity based on user activity or battery state. For example, it may execute lightweight heart rate analysis during sedentary periods and escalate to more complex signal processing only during exercise or abnormal events. This selective adaptation ensures 24/7 functionality without frequent recharging, enhancing user convenience and trust in the device.

In autonomous sensor networks used in agriculture, structural health monitoring, or environmental sensing, energy-aware scheduling plays a vital role in extending device lifespan. These sensors often operate in harsh or isolated conditions with limited opportunities for maintenance. The scheduler enables smart inference triggers based on context—such as soil moisture levels crossing a threshold or vibration patterns indicating structural strain—thereby conserving energy until action is needed. Furthermore, in scenarios involving energy harvesting (e.g., solar-powered nodes), the system can adapt task execution to energy availability, creating a self-sustaining intelligence loop. These diverse applications highlight the versatility and importance of adaptive scheduling in making edge AI both practical and scalable across mission-critical domains.

8. Conclusion and Future Work

In conclusion, this work presented a comprehensive energy-aware AI scheduling framework tailored for resource-constrained edge devices, addressing the growing need for sustainable and efficient AI execution at the edge of the network. By integrating task profiling, energy estimation, and real-time dynamic scheduling, the proposed system successfully balances performance, accuracy, and power consumption, achieving energy savings of up to 35% across various hardware platforms and AI workloads with minimal compromise in latency or inference quality. These findings underscore the practicality of context-aware, adaptive scheduling as a vital enabler for deploying intelligent applications in power-sensitive edge environments. Looking toward future enhancements, the integration of reinforcement learning offers a promising path to further improve the scheduler's adaptability, allowing it to evolve optimal policies based on feedback over time, particularly beneficial in environments with repetitive tasks or fluctuating resource availability. Moreover, federated learning opens avenues for decentralized intelligence, enabling collaborative model refinement across devices without sharing raw data, thereby preserving privacy while improving overall system performance.

The framework's applicability can be extended beyond individual devices to heterogeneous multi-device edge networks, where cooperative scheduling must account for communication overheads, task offloading, synchronization, and decentralized energy management. Such an extension not only introduces new challenges in terms of coordination and consistency but also presents substantial opportunities for performance scaling, resilience, and load distribution across diverse devices such as microcontrollers, AI-enabled single-board computers, and low-power IoT nodes. Ultimately, this research lays the groundwork for a more intelligent, adaptive, and energy-conscious edge AI infrastructure that is capable of supporting the next generation of ubiquitous computing applications in smart cities, autonomous systems, and industrial IoT. As edge computing continues to grow in importance, our approach contributes a viable strategy for optimizing energy usage without sacrificing responsiveness or intelligence, and sets the stage for more advanced scheduling architectures that integrate learning, collaboration, and system-level awareness to deliver sustainable AI at scale.

References

[1] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, and J. Mars, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[2] A. Mittal and A. Verma, "Dynamic energy-aware scheduling for real-time systems on DVS-enabled processors," *Proceedings of the International Conference on Embedded Software and Systems*, pp. 417–426, 2010.

[3] S. Banerjee, K. Bhardwaj, and T. Mitra, "Power-aware deployment and scheduling of embedded deep neural networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–23, 2019.

[4] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)*, pp. 365–376, 2011.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *International Conference on Learning Representations (ICLR)*, 2016.

[6] Z. Wu, D. An, W. Wu, and X. Li, "An adaptive energy-efficient scheduling mechanism for real-time tasks on multicore embedded systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1026–1037, 2018.

[7] X. Zhang, Z. Lin, and H. Li, "A survey on energy-efficient scheduling for real-time systems," *Journal of Systems Architecture*, vol. 90, pp. 71–84, 2018.

[8] Y. Xiao, D. Jin, and Y. Yang, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.

[9] A. Ignatov et al., "AI benchmark: Running deep neural networks on Android smartphones," *European Conference on Computer Vision (ECCV) Workshops*, pp. 0–15, 2018.

[10] M. Samragh, J. K. Kim, and F. Koushanfar, "Collaborative privacy-preserving deep learning with unmanned aerial vehicles," *Proceedings of the 17th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 1–13, 2019.

[11] F. Zhou, Y. Huang, Q. Wu, and W. Yang, "Energy-efficient task offloading and resource allocation for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11127–11140, 2018.

[12] T. Zhang, S. Ye, Y. Wang, and S. Hu, "Efficient scheduling of deep learning workloads on edge devices," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4430–4441, 2021.

[13] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014.

[14] H. Li, Z. Chen, and H. Zhang, "A survey on energy-efficient computing and resource management in edge AI," *ACM Computing Surveys (CSUR)*, vol. 54, no. 10, pp. 1–36, 2022.

[15] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design & Test of Computers*, vol. 18, no. 2, pp. 62–74, 2001.

[16] Animesh Kumar, "Redefining Finance: The Influence of Artificial Intelligence (AI) and Machine Learning (ML)", Transactions on Engineering and Computing Sciences, 12(4), 59-69. 2024.

[17] Vegineni, Gopi Chand, and Bhagath Chandra Chowdari Marella. "Integrating AI-Powered Dashboards in State Government Programs for Real-Time Decision Support." AI-Enabled Sustainable Innovations in Education and Business, edited by Ali Sorayyaei Azar, et al., IGI Global, 2025, pp. 251-276. https://doi.org/10.4018/979-8-3373-3952-8.ch011

[18] Palakurti, A., & Kodi, D. (2025). "Building intelligent systems with Python: An AI and ML journey for social good". In Advancing social equity through accessible green innovation (pp. 1–16). IGI Global.

[19] Mohanarajesh, Kommineni (2024). Generative Models with Privacy Guarantees: Enhancing Data Utility while Minimizing Risk of Sensitive Data Exposure. International Journal of Intelligent Systems and Applications in Engineering 12 (23):1036-1044.

[20] Puvvada, R. K. (2025). Enterprise Revenue Analytics and Reporting in SAP S/4HANA Cloud. European Journal of Science, Innovation and Technology, 5(3), 25-40.

[21] B. C. C. Marella, "Data Synergy: Architecting Solutions for Growth and Innovation," International Journal of Innovative Research in Computer and Communication Engineering, vol. 11, no. 9, pp. 10551–10560, Sep. 2023.

[22] A Novel AI-Blockchain-Edge Framework for Fast and Secure Transient Stability Assessment in Smart Grids, Sree Lakshmi Vineetha Bitragunta, International Journal for Multidisciplinary Research (IJFMR), Volume 6, Issue 6, November-December 2024, PP-1-11.

[23] Padmaja Pulivarthy. (2024/12/3). Harnessing Serverless Computing for Agile Cloud Application Development," FMDB Transactionson Sustainable Computing Systems. 2,(4), 201-210, FMDB.
[24] Marella, B.C.C., & Kodi, D. (2025). "Fraud Resilience: Innovating Enterprise Models for Risk Mitigation". Journal of Information Systems Engineering and Management, 10(12s), 683–695.
[25] Mohanarajesh Kommineni, Swathi Chundru, Praveen Kumar Maroju, P Selvakumar. (2025). Ethical Implications of AI in Sustainable Development Pedagogy, Rethinking the Pedagogy of Sustainable Development in the AI Era, 17-36, IGI Global Scientific Publishing.
[26] Aragani, Venu Madhav and Maroju, Praveen Kumar and Mudunuri, Lakshmi Narasimha Raju, Efficient Distributed Training through Gradient Compression with Sparsification and Quantization Techniques (September 29, 2021). Available at SSRN: https://ssrn.com/abstract=5022841 or http://dx.doi.org/10.2139/ssrn.5022841
[27] Pulivarthy, P. (2024). Optimizing Large Scale Distributed Data Systems Using Intelligent Load Balancing Algorithms. AVE Trends in Intelligent Computing Systems, 1(4), 219–230.
[28] Muniraju Hullurappa, Sudheer Panyaram, "Quantum Computing for Equitable Green Innovation Unlocking Sustainable Solutions," in Advancing Social Equity Through Accessible Green Innovation, IGI Global, USA, pp. 387- 402, 2025.
[29] MRM Reethu, LNR Mudunuri, S Banala,(2024) "Exploring the Big Five Personality Traits of Employees in Corporates," in FMDB Transactions on Sustainable Management Letters 2 (1), 1-13
[30] Aragani, Venu Madhav and Maroju, Praveen Kumar and Mudunuri, Lakshmi Narasimha Raju, "Efficient Distributed Training through Gradient Compression with Sparsification and Quantization Techniques" (September 29, 2021). Available at SSRN: https://ssrn.com/abstract=5022841 or http://dx.doi.org/10.2139/ssrn.5022841.
[31] Puvvada, R. K. "Optimizing Financial Data Integrity with SAP BTP: The Future of Cloud-Based Financial Solutions." European Journal of Computer Science and Information Technology 13.31 (2025): 101-123.
[32] INNOVATIVE DESIGN OF REFINING MUSCULAR INTERFACES FOR IMPLANTABLE POWER SYSTEMS, Sree Lakshmi Vineetha Bitragunta ,International Journal of Core Engineering & Management, Volume-6, Issue-12, 2021,PP-436-445.
[33] Kirti Vasdev. (2019). "GIS in Disaster Management: Real-Time Mapping and Risk Assessment". International Journal on Science and Technology, 10(1), 1–8. https://doi.org/10.5281/zenodo.14288561