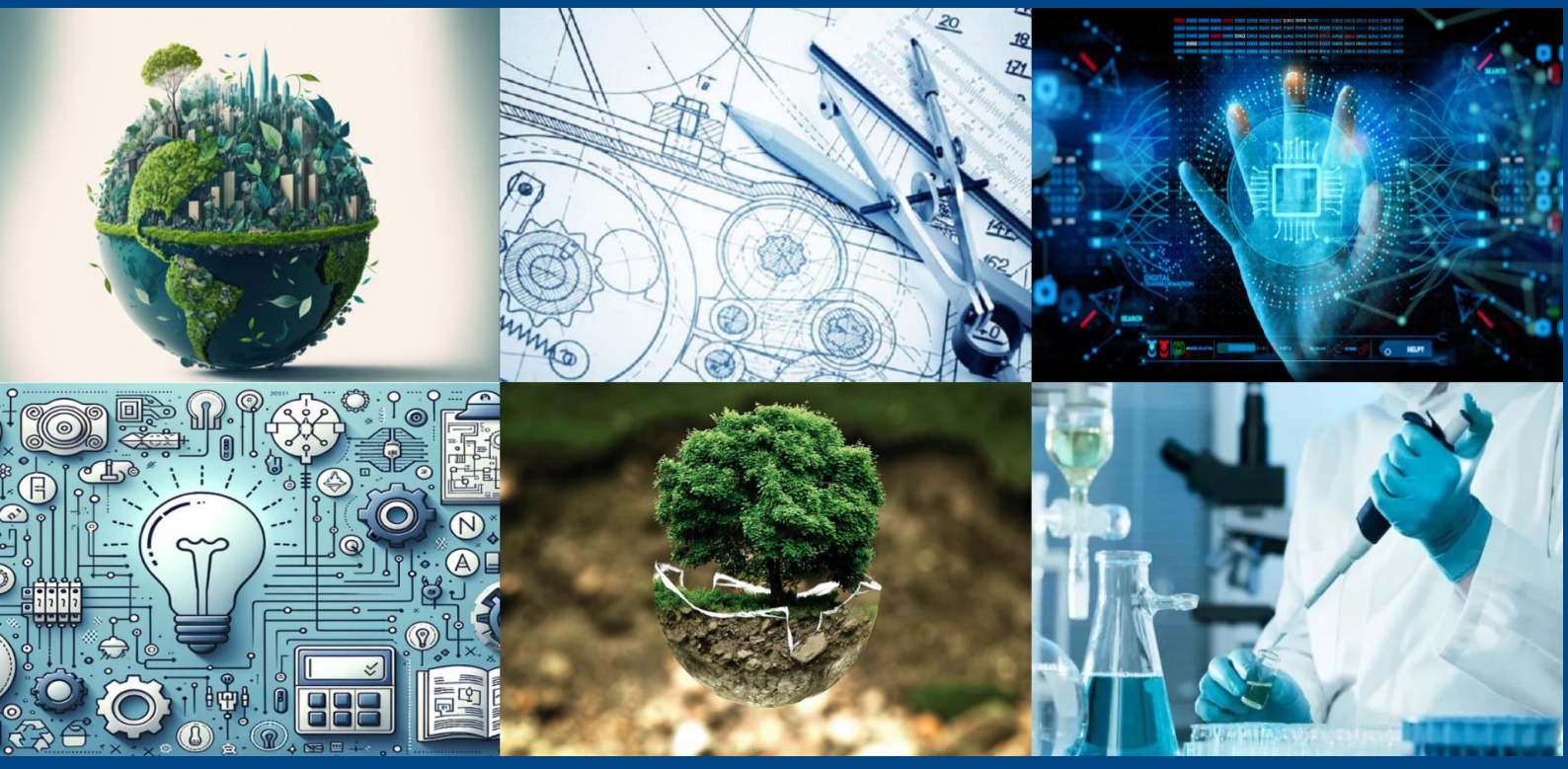
ISSN: 2582-7219

International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.864

Volume 9, Issue 8, August 2026

© 2026 IJMRSET | Volume 9, Issue 8, August 2026| DOI:10.15680/IJMRSET.2026.0908014

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 9.864| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

An Adaptive AI/ML Model Framework for Energy Conservation: A Hardware-Aware Runtime Switching Approach

Manju B S, Akshay M, Chandan Hegde

Student, Department of MCA, Surana College, Bangalore, Karnataka, India

Student, Department of MCA, Surana College, Bangalore, Karnataka, India

Assistant Professor, Department of MCA, Surana College, Bangalore, Karnataka, India

ABSTRACT: Edge AI deployment is constrained by limited power, memory, and thermal budgets, since conventional inference systems rely on one fixed model regardless of device conditions. This paper proposes a hardware-aware adaptive inference framework, built on a modular four-layer architecture, that switches between ResNet18 and a lightweight Tiny CNN using real-time battery, CPU, and thermal telemetry. Experiments show ResNet18 gives high prediction capability at 13.27–14.149 ms steady-state latency, while the Tiny CNN cuts latency to under 0.4 ms with only 314 parameters, confirming runtime energy-aware model switching is feasible for edge devices, though logic-branch and telemetry gaps remain for future refinement.

KEYWORDS: Edge AI, Energy Conservation, Adaptive Model Switching, TinyML, Internet of Things (IoT)

I. INTRODUCTION

A. Background

There has been a tremendous growth in the number and kind of Edge Artificial Intelligence (AI) applications recently such as autonomous vehicles, health and fitness monitoring, industrial automation, security and surveillance which encompasses numerous applications under the Internet of Things (IoT), per se with its large variety of use cases. Edge AI enables real time inference as opposed to cloud based AI. Nevertheless, running deep learning models at inference on edge devices throws a number of trials for developers primarily due to the strict limitations that generally exist with an array of edge devices. These limitations consist of accessible processing power, memory, energy and thermal ability. The large, high capacity deep learning models that we use when utilising the ResNet18 to achieve a high level of accuracy for prediction-related tasks come at their price: they are extremely complex to run on the device with respect to how power consumption and other aspects of resource utilisation need to be considered, both in terms of model storage as well as temporary memory utilised whilst executing calculations during model usage. They can also lead to high energy consumption and long delay for time sensitive applications, in addition to putting the device at risk of overheating if used over an extended period.

B. Problem Statement

Most of the current solutions utilize fixed-size models for edge computing. These models are not adapted to the current state of the host device (e.g. battery level, current load on CPU, temperature) and therefore lead to suboptimal usage of resources, delay, energy consumption and, in extreme cases, even thermal throttling. As a result, there is a huge potential for adaptive inference frameworks, which for a given task choose the best model from a library of models for the current state of the host device.

C. Scope and Objectives

The purpose of this work is to bridge the required performance of AI on the one hand and the restrictions of hardware of embedded systems on the other hand. The main goals of this work are:

- Develop an adaptive inference framework to support cross-layer, real-time telemetry-driven orchestration of software and hardware.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 9.864| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- Design and implement a runtime energy-aware model switching mechanism that changes the computational load of the model on the basis of the real-time health metrics of the device.

II. LITERATURE SURVEY

The TinyML research to train and scale Neural Networks (NNs) to run on tiny microcontrollers with very little RAM has seen great progress recently. The existing literature mostly deals with static optimization techniques for the NN itself, like weight quantization, parameter pruning and knowledge distillation. However, all these optimizations are typically done as pre-deployment optimization and do not consider the environment the system is running in or the specific characteristics of the physical device running the software.

A more recent set of neural network architectures, Dynamic Neural Networks (DNNs) and Early Exit Architectures, for instance, focus on reducing the amount of redundant computation in forward propagation for complex data. Such architectures are able to adapt their behavior, for instance by early exiting from the network as soon as a certain threshold of confidence has been reached, but transitions between the different parts of the architecture are generally not influenced by hardware-specific constraints, such as the amount of remaining battery life or the current silicon temperature. Thus, they are not energy-aware in a holistic sense.

In summary, true energy-aware computing must form feedback loops between software and hardware. Such cross-layer communication for runtime adaptability already exists. In order to achieve accurate image classification on the edge with adaptive switching between the TinyML models ResNet18, VGG16, MNIST, Lena, etc., for high accuracy inference when sufficient resources are available, and for reliable but lower accuracy inference using tiny TinyML models when resources are failing, a combination of static and dynamic TinyML methods are required.

Research work like A Study on Adaptive Inference for Real-time Video Analytics on the Edge [1] focuses on making trade-off between accuracy and energy for various scenarios through dynamic resolution and model scaling for video analytics. Research work like Thermal-aware Neural Architecture Search [2] incorporates thermal constraints into the process of Neural Architecture Search (NAS) for designing models that can be optimized for certain thermal envelopes. Research work like Federated Learning with Hardware-aware Scheduling for IoT Devices [3] schedules training and inferencing tasks in IoT devices in a manner that takes into account the battery health and processing capabilities of individual devices to prevent node dropout during training.

Adaptive on-device learning for personal healthcare on the edge saves energy during sedentary wearer periods by adapting between models of different complexity based on wearer activity level [4]. Lightweight transformer models for various IoT applications have also been optimized to run on edge devices, e.g. by attention-head pruning to dynamically prune the model to fit into the available RAM of the device [5].

In summary, current TinyML research enables neural networks on microcontrollers using static techniques (e.g. weight quantization, parameter pruning) for networks larger than the available memory, but these techniques are pre-deployment and not suitable for runtime adaptability. Early exit architectures reduce the number of required computations for complex data but do not consider hardware-specific constraints such as battery life or silicon temperature. Runtime adaptability, therefore, requires a feedback loop between software and hardware, where the software monitors hardware constraints and, in case of resource scarcity, switches from a high-accuracy model (e.g. ResNet18) to a low-accuracy model (e.g. MNIST MobileNet).

III. METHODOLOGY / APPROACH

A. Proposed System Design

Fig. 1 depicts the proposed system architecture. The framework presents four core layers, and the design is modular to ensure easy modification and maintenance in the future.

© 2026 IJMRSET | Volume 9, Issue 8, August 2026| DOI:10.15680/IJMRSET.2026.0908014

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 9.864| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

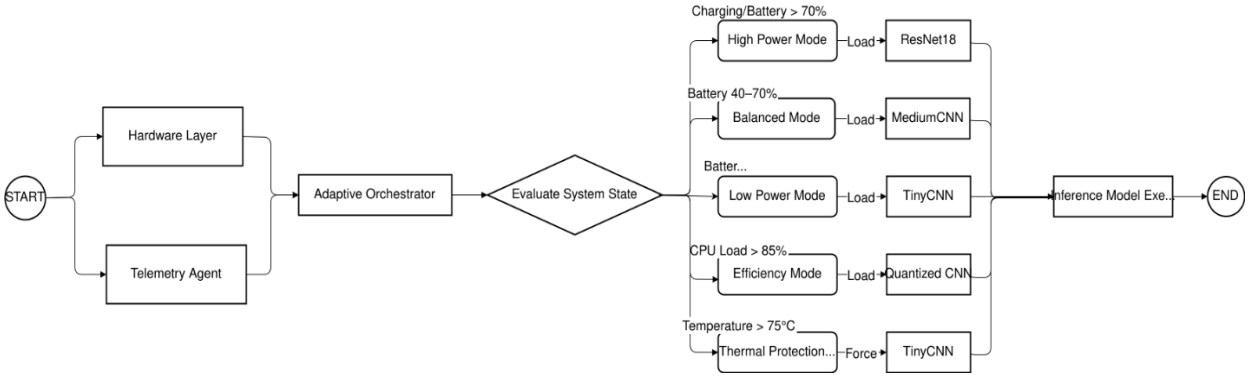


Fig. 1. Proposed Model Architecture

The system is organized into a modular four-layer architecture:

- **Layer 1: Hardware Layer:** The physical foundation consisting of the CPU/NPU, Lithium-Ion battery, and on-board thermal sensors.
- **Layer 2: Telemetry Agent:** Monitors hardware health by polling battery capacity, CPU utilization, and native OS thermal probes.
- **Layer 3: Adaptive Orchestrator:** The decision engine that evaluates telemetry against thresholds. It uses a Memory Management Daemon to ensure model transitions are atomic and clear system RAM. Logic is prioritized as follows:
  - **Emergency (Thermal):** >75°C triggers Tiny CNN.
  - **Traffic (Efficiency):** >85% CPU triggers Quantized CNN.
  - **Stamina (Power):** High Power (>70% battery) uses ResNet18; Balanced (40–70%) uses Medium CNN; Low Power (<40%) uses Tiny CNN.
- **Layer 4: Model Execution Pool:** A suite of models including ResNet18 for peak intelligence and Tiny CNN for operational survival.

B. Implementation

The framework is implemented in Python, leveraging psutil for hardware telemetry and torch for deep learning execution. The implementation relies on three core algorithms:

- Monitoring: Continuously polls system sensors to update telemetry history.
- Selection: Applies the logic hierarchy (Priority: Temperature > CPU Load > Battery Level).
- Execution: Loads the selected model, performs inference, and records metrics like latency and parameter counts.

This work is fully developed using Python for both deep learning and hardware monitoring. The AI operations core is implemented using PyTorch (torch); torchvision is used to load the pre-trained ResNet18 model for deep learning-based operations; and psutil is used to monitor real-time telemetry of the system, including CPU usage and battery status. The orchestrator uses standard Python modules such as enum for the discrete operating modes, dataclasses for structured data, and collections for the system's historical metrics (e.g. last recorded values). The environment can be set up through a standard installation of the required packages using pip.

IV. RESULTS & DISCUSSION

A. Experimental Results

For performance evaluation of the proposed framework, an edge computing platform was implemented and tested with the adaptive AI inference framework by employing a variety of CNNs (i.e., ResNet18 and a Tiny CNN with very few parameters), adapting them for application at the edge by continuously monitoring system parameters such as (i) CPU usage, (ii) battery level and (iii) device temperature, thereby selecting the optimal operating mode (High Power, Low Power or Efficiency) for offloading AI computations through an adaptive AI orchestrator. ResNet18-based CNN was executed in High Power mode to obtain both cold-start and steady-state latency, whereas the Tiny CNN-based model was executed in both Low Power and Efficiency modes. In both cases, average latency (ms) was recorded across

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 9.864| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

multiple iterations, and the number of model parameters was measured and compared. The results are presented in Table 1.

Table 1. Latency and Parameter Disparity in Experiment

Model	Operating Mode	Latency (ms)	Parameters
ResNet18	High Power (Cold Load)	25.000	11,181,642
ResNet18	High Power (Steady State)	13.270–14.149	11,181,642
Tiny CNN	Low Power	0.393	314
Tiny CNN	Efficiency	0.269	314

Testing across various scenarios showed that the system can successfully transition between states. Results from the experiment show that the suggested adaptive AI inference framework can effectively trade off inference performance and computational efficiency by intelligently selecting a suitable CNN model according to the system's operating conditions. ResNet18 provides a high prediction capability while keeping an inference latency of 25 ms during cold start, which further improves to 13.27–14.149 ms once the system reaches a steady state. ResNet18 has 11.18 million parameters, is highly computational, but has the highest model capacity when system resources are readily available.

In contrast, in Low Power and Efficiency modes, the framework moves to the lightweight Tiny CNN, which contains only 314 parameters. This drastic decrease in model size leads to very low inference latencies of 0.393 ms and 0.269 ms, making it extremely appropriate for battery-constrained or thermally limited edge devices.

The results obtained through testing the framework's various functions show a significant trade-off between the considered models. While ResNet18 is mainly customised for accuracy, maximising model capacity, the Tiny CNN is instead optimised for speed, energy consumption and memory allocation, in order to be a light model suited to the edge. The adaptive orchestrator exploits these characteristics by selecting the best model for a specific task in real time, taking into account the current state of the device on which the task is executed. In this way, the proposed framework combines the strengths of the different models to obtain a balanced execution in terms of performance, response time and resource consumption, without resorting to a fixed single neural network optimal only for a specific execution scenario. This makes the framework and its adaptive model-selection strategy well suited for a wide range of resource-constrained edge AI applications.

B. Summary of Findings

The current implementation has several “aspirational” components, according to a rigorous review:

- **Logic Branch Failures:** The code frequently skipped the “Balanced” state, going straight to High Power mode despite having 55% battery left.
- **Initialisation Spikes:** Efficiency mode may be triggered by the temporary CPU load of loading a model, in place of sustained bottlenecks.
- **Simulation Constraints:** Due to the live sensor integration being hardcoded to a default 40.0°C, success of the thermal protection was determined through manual overrides.
- **Model Pool Shortcomings:** “Medium CNN” and “Quantized” modes were basically placeholders, re-utilizing ResNet18 and Tiny CNN.

V. CONCLUSION

In this work, we proposed an adaptive AI/ML framework that selects the best deep learning model on-device for the currently observed hardware conditions, such as battery level, CPU utilization and temperature. In contrast to common approaches which use a single, fixed model, the proposed system dynamically switches between different models that offer a trade-off between power consumption and acceptable prediction accuracy.

The experimental results show that the proposed framework can adjust according to hardware dynamics by using lightweight models under low battery, high CPU usage and thermal stress conditions. The approach saves computation overhead, decreases inference latency and maintains longer operation on a low-power device, making it especially suited for edge devices such as IoT systems, wearable devices, smartphones and portable healthcare equipment.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 9.864| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

The implementation also raised some concerns to be addressed: during testing, some operating modes selected different models than originally intended because the model-switching logic is still under development. These observations can be used to better tune the adaptive decision engine and increase the overall reliability of the framework.

All in all, this work shows that runtime energy-aware model switching is a viable solution for intelligent edge computing. With further enhancements, the proposed framework can contribute to the development of smarter, more energy-efficient AI systems capable of delivering reliable performance while adapting to dynamic hardware conditions.

REFERENCES

[1]. Boldo, Michele, et al. "Domain-adaptive online active learning for real-time intelligent video analytics on edge devices." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 43.11 (2024): 4105-4116.
[2]. Jeon, Seunghyeok, et al. "Phoenix: Thermal-Aware On-Device Inference of Multi-Instance DNNs for Mobile Video Applications." ACM Transactions on Embedded Computing Systems 25.2 (2026): 1-23.
[3]. Puppala, Sai, et al. "A comprehensive survey of federated learning for edge AI: Recent trends and future directions." (2025).
[4]. Lee, Simon A., et al. "Towards on-device foundation models for raw wearable signals." NeurIPS 2025 Workshop on Learning from Time Series for Health. 2025.
[5]. Tao, Xiaoling, et al. "LCT-IDS: A Lightweight Transformer-Based Architecture for IoT Intrusion Detection." IEEE Internet of Things Journal (2026).
[6]. Chen, Tianqi, et al. "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning." Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2018, pp. 578–594.
[7]. Han, Song, Huizi Mao, and William J. Dally. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding." International Conference on Learning Representations (ICLR), 2016.
[8]. Li, Y., et al. "Unveiling Energy Efficiency in Deep Learning: Measurement, Prediction, and Scoring across Edge Devices." arXiv, arXiv:2310.18329, 2023.
[9]. Lin, Ji, Wei Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. "MCUNet: Tiny Deep Learning on IoT Devices." Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 11711–11722.
[10]. Scardapane, Simone, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. "Group Sparse Regularization for Deep Neural Networks." Neurocomputing, vol. 241, 2017, pp. 81–89.
[11]. Tambe, A., et al. "Prediction of Fine-Grained Early Exits for Computation- and Energy-Efficient Inference." Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, no. 7, 2023, pp. 8657–8665.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com