DEEP LEARNING

Reliability- and Class-Aware Model Selection and Adaptation for Deploying Deep Learning Models on Resource-Constrained Edge Devices

Rutuja Dnyaneshwar Kemble, A. N. Pawar, Ritesh Ranjan

Department of Artificial Intelligence and Machine Learning Engineering, Navsahyadri Education Society's Group of Institutes, Savitribai Phule Pune University, Pune, India

Published November 2026

Abstract

Deploying deep learning models on resource-constrained edge devices—such as smartphones, Raspberry Pi-class boards, microcontrollers, industrial sensors, smart cameras, wearables, and autonomous robots—requires satisfying strict accuracy, latency, memory, energy, and thermal limits simultaneously. Existing optimization methods typically focus on single objectives or rely on theoretical proxy metrics such as parameter count, FLOPs, average latency, or overall classification accuracy. However, theoretical efficiency metrics fail to predict real-world runtime behavior under concurrent workloads and thermal throttling, while aggregate accuracy can mask severe degradation on minority or safety-critical classes. This paper presents a conceptual framework for Reliability-and Class-Aware Model Selection and Adaptation for Edge AI, which selects and adapts among deployment-ready model configurations based on measured hardware performance vectors. By jointly **enforcing probabilistic latency chance constraints (**$P\ (L_i \leq L_{\max}) \geq 1 - \epsilon$**), memory and energy bud-** gets, and minimum accuracy floors for critical classes, the framework aims to improve deployment dependability under changing device conditions. Each candidate configuration is characterized by its architecture, quantization, pruning, distillation, inference runtime, and hardware setup, and is evaluated against empirical measurements of accuracy (overall and per class), latency distribution, memory, energy, storage, and reliability metrics. Configurations that violate constraints are rejected, feasible options are compared on a Pareto frontier, and runtime variant switching is triggered only when expected benefits outweigh switching overhead. The paper reviews existing model compression, adaptive inference, and hardware profiling approaches, identifies key research gaps, and outlines an experimental evaluation plan across six baselines and heterogeneous edge hardware platforms.

Keywords: edge AI; resource-aware inference; probabilistic latency; class-level accuracy; model compression; dynamic model selection; TinyML; deployment reliability

1 Introduction

1.1 Background

The rapid proliferation of internet-connected embedded devices, mobile platforms, and autonomous edge systems has driven substantial interest in efficient deep learning for resource-constrained hardware, commonly referred to as Edge AI, TinyML, or resource-aware inference. While deep neural networks continue to advance in representational capability across computer vision, audio processing, and time-series analysis [12, 14], their computational, memory, and energy requirements frequently exceed what low-resource edge hardware can reliably sustain. Unlike centralized cloud servers or dedicated data-center GPUs, edge devices operate under tight RAM and flash storage

limits, strict battery budgets, passive thermal dissipation limits, and fluctuating CPU workloads caused by concurrent background applications.

1.2 Problem statement

Most existing model compression and deployment methodologies—including quantization, weight pruning, knowledge distillation, lightweight neural architectures, adaptive early-exit inference, and hardware-aware neural architecture search [8–10, 13]—optimize a single primary objective or static trade-off curve. In operational edge deployments, however, multiple hard and proba-

- *Theoretical vs. Real-World Mismatch:* Hardware-agnostic metrics such as parameter count, floating-point operations (FLOPs), multiply-accumulate operations (MACs), model binary size, and compression ratio do not reliably predict actual wall-clock latency or peak RAM consumption [15]. For example, a 40% reduction in FLOPs or a 75% reduction in weight storage does not guarantee proportional speedups or memory savings because data preprocessing, memory access patterns, thread scheduling, cache locality, and inference runtime overhead strongly shape execution.
- *Latency Reliability vs. Average Latency:* A com-pressed model exhibiting an acceptable mean latency (such as 40 ms) may suffer severe tail-latency spikes up to 120 ms under CPU contention, elevated ambient temperature, or memory pressure, thereby violating a strict real-time deadline (such as $L_{\max}$ = 50 ms) [6, 7].
- *Class-Level Accuracy Degradation:* High aggregate classification accuracy frequently conceals substantial performance degradation on minority, rare, or safety-critical classes. For instance, post-training quantization or aggressive pruning may preserve 93% overall accuracy while reducing accuracy on a critical fire-detection class to 75%.

1.3 Motivation and objectives

Deploying deep neural networks at the edge without accounting for tail-latency variability and class-specific degradation risks real-time deadline misses and unsafe failures on high-consequence categories. Rather than proposing yet another compression algorithm, practical deployment value lies in constructing a principled decision and runtime adaptation framework that leverages existing model variants and empirical on-device measurements under realistic resource variability.

The primary objectives of this work are:

1. To examine the foundations of resource-aware edge inference and demonstrate why theoretical complexity proxies are insufficient for dependable deployment.
2. To survey and compare existing model compression, adaptive inference, and hardware profiling approaches against deployment-relevant reliability criteria.
3. To formulate a joint constraint-based model configuration selection and runtime adaptation framework incorporating probabilistic latency chance constraints and class-level accuracy floors.
4. To define a structured experimental protocol, baseline comparison plan, limitations, and future research directions.

1.4 Scope and paper organization

This study addresses inference-time configuration selection and runtime adaptation across pre-prepared model variants on heterogeneous edge devices (such as ARM-based single-board com-

puters and mobile smartphones) for a focused inference task. It does not introduce a novel neural network compression operator or claim universal optimality across arbitrary hardware and datasets.

Section II reviews related work across compression, adaptive execution, and hardware profiling. Section III presents a comparative analysis of existing deployment strategies. Section IV identifies core research gaps. Section V details the proposed mathematical problem formulation and adaptation pipeline. Section VI outlines the planned experimental methodology, expected out-comes, and future scope, and Section VII concludes the paper.

2 Literature survey

2.1 Overview of edge inference techniques

Research on efficient deep learning inference for edge devices spans four primary technical families: (1) model compression (quantization, structured and unstructured pruning, and knowledge distillation), (2) lightweight architectures engineered for low compute, (3) adaptive inference (early-exit branches and input-dependent execution paths), and (4) empirical hardware profiling and surrogate latency modeling. Table 1 summarizes the core findings and limitations across these research areas.

2.2 Model compression and lightweight architectures

Quantization reduces numerical precision from 32-bit floating point (FP32) to 16-bit floating point (FP16),

2.3 Adaptive inference and hardware profiling

Adaptive inference mechanisms, such as early-exit convo-lutional networks [13] and reinforcement-learning-driven TinyML controllers [5], dynamically adjust computation based on input difficulty or device state. Concurrently, hardware profiling frameworks such as nn-Meter [15] and tail-latency or thermal-aware schedulers [6, 7] demonstrate that kernel-level execution and thermal throttling strongly govern real-time deadline compliance. Similarly, automated model selection studies in time-series anomaly detection [1–3] highlight that no single static model dominates across changing operating conditions. However, existing methods typically optimize average metrics or single objectives rather than jointly enforcing probabilistic latency bounds and per-class accuracy floors under dynamic resource contention.

3 Comparative analysis of existing approaches

3.1 Deployment strategies and evaluation criteria

Existing deployment strategies in the literature can be grouped into five categories: (1) single-metric static se-lection (choosing one variant by smallest storage, lowest mean latency, or highest overall accuracy), (2) complexity-proxy selection (ranking models by parameter count, FLOPs, MACs, or compression ratio), (3) adaptive or early-exit inference (varying execution depth per input), (4) unconstrained Pareto-based selection (identifying non-dominated trade-offs without per-class accuracy bounds), and (5) static resource-aware selection (matching a fixed model to nominal hardware specifications without runtime adaptation).

To assess operational dependability, deployment

1. *Probabilistic Latency Reliability:* Meeting real-time deadlines consistently ($P\ (L_i \leq L_{\max}) \geq 1 - \epsilon$) rather than relying solely on mean latency.
2. *Class-Level Accuracy Protection:* Preserving minimum accuracy floors on minority or safety-critical classes (such as fire detection).
3. *Resource and Thermal Awareness:* Respecting peak RAM limits, battery energy budgets, concurrent CPU load, and thermal throttling.
4. *Empirical Grounding:* Relying on measured target-device execution profiles rather than hardware-agnostic complexity proxies.
5. *System and Switching Overhead:* Accounting for offline profiling cost, controller complexity, and runtime

Table 1. Summary of reviewed literature across edge AI optimization families

Research area	Core techniques	Key findings and limitations
Quantization and pruning [8, 9, 11]	Precision reduction (FP32, FP16, INT8, mixed precision) and structured or unstructured weight pruning.	Substantially shrinks model storage footprint, but wall-clock speedups depend heavily on runtime/ hardware kernel support and can disproportionately degrade minority or safety-critical classes.
Distillation, lightweight nets, and adaptive inference [10, 12, 13]	Student-teacher knowledge distillation, compact mobile architectures, early-exit networks, and input-dependent compute paths.	Improves average computational efficiency across typical inputs, but static model choices or purely input-driven adaptation do not guarantee tail-latency deadlines under background device stress.
Hardware profiling and surrogate modeling [5–7, 15]	Empirical on-device benchmarking, kernel-level latency prediction, Gaussian-process regression, Bayesian optimization, and cross-device transfer.	Captures realistic hardware execution behavior and reduces exhaustive profiling cost, though initial measurement overhead, thermal dynamics, and cross-device transfer accuracy require careful validation.

3.2 Trade-off comparison

Table 2 contrasts conventional baseline strategies with the proposed reliability- and class-aware selection framework.

No single static model variant is optimal across all runtime states. For example, in a three-class smart camera application (person, vehicle, fire), an aggressively quantized INT8 variant may satisfy tight latency and RAM budgets yet degrade fire-detection accuracy to 82%, whereas an uncompressed FP32 variant preserves fire accuracy but violates latency deadlines and memory ceilings under load. Selecting a feasible configuration that satisfies both critical-class floors and chance constraints—and switching variants only when net gains exceed switching overhead—directly addresses this operational trade-off.

4 Research gap identification

4.1 Limitations in prior literature

Synthesizing prior work reveals three recurring limitations:

- *Over-Reliance on Average Latency:* Reporting mean inference time obscures heavy-tailed latency distributions (P_{95}, P_{99}) and deadline-miss rates triggered by OS background tasks, memory contention, and thermal throttling.
- *Over-Reliance on Aggregate Accuracy:* Evaluating models exclusively via overall accuracy conceals severe class-level degradation, miscalibration, and vulnerability on rare or safety-critical classes following compression.
- *Proxy and Short-Run Assumptions:* Assuming that FLOPs, parameter counts, or brief idle-device bench-marks accurately reflect sustained execution on heterogeneous edge hardware leads to brittle deployments.

4.2 Identified research gap and significance

An open empirical question remains: *whether jointly enforcing probabilistic latency constraints, memory/energy/storage budgets, overall accuracy floors, and per-class accuracy floors—combined with runtime state monitoring and cost-aware switching—yields measurably more dependable edge deployments than selection methods based on average latency, model size, or aggregate accuracy alone.* Bridging the divide between laboratory compression metrics and sustained edge reliability is vital for safety-critical domains where missed deadlines or false negatives on rare events carry severe consequences.

5 Proposed framework and problem formulation

5.1 Candidate configuration and performance vector

(Conceptual framework proposed for study; experimental validation is pending.)

Given a finite library of pre-built deployment variants for a target edge task and hardware platform, each candidate configuration C_i is defined as a six-tuple of design and execution choices:

$$C_i = (A_i,\ Q_i,\ P_i,\ D_i,\ R_i,\ H_i) \tag{1}$$

where A_i denotes the neural network architecture, Q_i represents numerical quantization precision (e.g., FP32, FP16, INT8, or mixed precision), P_i denotes the pruning strategy, D_i denotes knowledge distillation or fine-tuning configuration, R_i specifies the inference runtime engine (e.g., TensorFlow Lite or ONNX Runtime), and H_i captures the hardware execution setup (such as CPU thread count, batch size, and input resolution).

Rather than relying on synthetic proxy scores, each candidate C_i is profiled directly on the target hardware under both idle (normal) and stressed (CPU/memory/thermal load) conditions to obtain an empirical performance vector v_i:

$$v_i = (a_i,\ l_i,\ m_i,\ e_i,\ s_i,\ r_i) \tag{2}$$

Table 2. Comparative analysis of edge model selection and deployment approaches

Deployment approach	Advantages	Limitations	Category
Smallest-model / average-latency / highest-accuracy selection	Simple to implement; provides a clear, deterministic single-criterion ranking.	Ignores tail-latency spikes under contention, violates RAM/energy budgets, or allows severe accuracy drops on critical/minority classes.	Single-metric baselines
Pareto and static resource-aware selection	Explicitly exposes multi-objective trade-offs or matches baseline device hardware specifications.	Unconstrained Pareto frontiers permit unsafe critical-class errors; static selections cannot adapt to runtime workload or thermal shifts.	Multi-objective / static baselines
Proposed reliability- and class-aware selection	Jointly enforces probabilistic latency, RAM/energy/storage limits, and critical-class accuracy floors with controlled runtime adaptation.	Introduces controller complexity, empirical profiling overhead, runtime switching cost, and task-specific configuration scope.	Proposed framework

where a_i encompasses overall accuracy and class-wise accuracy metrics, l_i characterizes the empirical inference latency distribution, m_i measures peak runtime RAM consumption, e_i captures per-inference energy con-sumption, s_i denotes persistent storage footprint, and r_i records runtime reliability indicators including deadline-miss rate, latency variance, and stability under thermal and workload stress.

5.2 Joint deployment constraints

A candidate configuration C_i is deemed feasible if and only if its measured performance vector satisfies all system resource budgets, probabilistic real-time deadlines, and accuracy floors simultaneously:

$$P\left(L_i \le L_{\max}\right) \ge 1 - \epsilon,\ M_i \le M_{\max},\ E_i \le E_{\max},\ S_i \le S_{\max} \tag{3}$$

$$A_{\text{overall},i} \ge A_{\min},\ A_{c,i} \ge A_{c,\min} \forall\ c \in C_{\text{crit}} \tag{4}$$

where L_i is the random variable representing inference latency, $L_{\max}$ is the application deadline, $\epsilon \in (0, 1)$ is the maximum tolerable deadline-miss probability, $M_{\max}$, $E_{\max}$, and $S_{\max}$ denote peak memory, energy, and storage budgets, $A_{\min}$ is the minimum acceptable overall accuracy, and $A_{c,\min}$ specifies the minimum required accuracy floor for each safety-critical or minority class $c \in C_{\text{crit}}$.

5.3 Decision and runtime adaptation pipeline

The proposed selection and adaptation workflow operates through nine structured stages:

1. *Define Constraints:* Specify application-grounded thresholds ($L_{\max}$, ϵ, $M_{\max}$, $E_{\max}$, $S_{\max}$, $A_{\min}$, $A_{c,\min}$).
2. *Build Candidate Variants:* Generate deployment-ready model artifacts spanning diverse combinations of (A_i, Q_i, P_i, D_i).
3. *Check Runtime Compatibility:* Verify operator support and execution compatibility across target run-times R_i and hardware settings H_i.
4. *Empirical Profiling:* Measure accuracy (overall and per class), latency distributions, peak RAM, energy, and storage under both normal and stressed device states.

5. *Reject Constraint Violators:* Filter out any configuration C_i violating (3) or (4).
6. *Identify Pareto-Optimal Feasible Options:* Construct the non-dominated Pareto frontier over the remaining feasible configurations.
7. *Select Initial Configuration:* Choose the active variant from the feasible Pareto set according to application priorities.
8. *Monitor Runtime State:* Continuously track observed inference latencies, CPU utilization, memory pressure, and thermal state during operation.
9. *Cost-Aware Variant Switching:* Trigger a runtime transition to an alternative feasible configuration C_j only if the predicted reliability or accuracy benefit exceeds the measured model-switching overhead (memory reload latency and energy cost), preventing oscillatory thrashing.

5.4 Expected operational benefits

By grounding configuration decisions in empirical device measurements rather than FLOPs or parameter counts, the framework avoids selecting variants that suffer hidden runtime bottlenecks. Explicit per-class ac-curacy floors prevent compressed models from silently compromising safety-critical classes, while the probabilistic chance constraint $P\,(L_i \le L_{\max}) \ge 1 - \epsilon$ directly curbs tail-latency deadline misses under contention. Furthermore, reusing existing model variants and runtimes avoids custom kernel development, and hysteresis-aware switching ensures runtime stability.

6 Experimental plan, future scope, and applications

6.1 Planned experimental protocol and baselines

(Planned evaluation methodology; empirical measurements are pending.)

To validate the conceptual framework rigorously, future experimental evaluation is structured around a focused inference task—such as image classification, object detection, audio-event detection, keyword spotting,

The proposed framework will be benchmarked against six baseline selection strategies:

1. *Smallest-Model Selection:* Selects the variant with minimum storage footprint S_i.
2. *Fastest Average-Latency Selection:* Selects the variant with lowest mean latency $\mathrm{E}[L_i]$.
3. *Highest Overall-Accuracy Selection:* Selects the variant maximizing $A_{\mathrm{overall},i}$.
4. *Unconstrained Pareto Selection:* Selects from the accuracy-latency Pareto front without enforcing per-class accuracy floors $A_{c,\min}$.
5. *Average-Latency Reliability Baseline:* Enforces $\mathrm{E}[L_i] \le L_{\max}$ rather than the tail-aware chance constraint $P\,(L_i \le L_{\max}) \ge 1 - \epsilon$.
6. *Static Resource-Aware Selection:* Selects a single feasible variant using idle-device measurements without runtime monitoring or adaptation.

Evaluation metrics will encompass overall and per-class accuracy, latency reliability (deadline satisfaction rate, P_{95} and P_{99} tail latencies, latency variance, and miss rate), resource consumption (peak RAM, storage, and per-inference energy/power), adaptation behavior (switching frequency, switching latency/energy overhead, decision accuracy, and post-switch stability), and robustness under synthetic CPU contention, memory pressure, sustained thermal stress, and input perturbations.

6.2 Future research directions

Two primary extensions are identified to enhance scalability and autonomy:

- *Profiling-Aware and Cross-Device Surrogate Selection:* Exhaustively profiling large configuration spaces on every target device incurs high measurement cost. Integrating adaptive sampling, Gaussian-process surrogate models, or Bayesian optimization—and transferring latency/energy predictors across hardware platforms with calibrated safety margins—can substantially reduce profiling overhead.
- *Uncertainty-, Energy-, and Thermal-Aware Adaptation:* Incorporating predictive uncertainty estimates alongside real-time battery state-of-charge, charging status, and hardware thermal throttling sensors as first-class controller inputs will enable proactive adaptation before hard thermal or energy limits are breached.

6.3 Potential applications

The proposed framework is directly applicable to latency-sensitive and safety-critical edge deployments, including smart surveillance cameras (e.g., real-time fire and hazard detection), industrial visual defect and anomaly inspection, embedded keyword spotting and acoustic event monitoring, wearable human-activity recognition [12], and autonomous mobile robotics.

7 Conclusion

Reliability- and class-aware model selection reframes Edge AI deployment around empirical hardware measurements, probabilistic deadline satisfaction, runtime resource variability, and critical-class accuracy protection. While theoretical complexity proxies, mean latency, and aggregate accuracy remain useful initial indicators, they prove insufficient under concurrent work-loads, thermal throttling, and severe class imbalance. By filtering candidate model variants through joint chance constraints and per-class accuracy floors and adapting only when net gains justify switching overhead, the pro-posed framework offers a structured path toward dependable edge inference. At the same time, practical trade-offs—including controller complexity, profiling overhead, switching cost, fine-grained energy and thermal measurement challenges, and cross-hardware generalization limits—must be evaluated transparently across hardware, software, statistical, and operational validity dimensions in upcoming experimental studies.

References

[1] Abdelmaksoud M, Ding S, Morozov A, Abedjan Z. RAMSES: Robust and adaptive model selection for time-series anomaly detection algorithms [preprint]. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2508.18106

[2] Zhou X, Brif C, Lourentzou I. mTSBench: Benchmarking multivariate time series anomaly detection and model selection at scale [preprint]. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2506.21550

[3] Sylligardos E, Paparrizos J, Palpanas T, Senellart P, Boniol P. MSAD: A deep dive into model selection for time series anomaly detection [preprint]. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2508.00862

[4] Goh J, Adepu S, Junejo KN, Mathur A. A dataset to support research in the design of secure water treatment systems. In: *Critical Information Infrastructures Security*; 2017. p. 88-99. https://doi.org/10.1007/978-3-319-71368-7_8

[5] Alaklabi S, Alharbi S. DRL-TinyEdge: Energy- and latency-aware deep reinforcement learning for adaptive TinyML at the 6G edge. *Discover Internet of Things*. 2025;5:87. https://doi.org/10.1007/s43926-025-00187-5

[6] Vijayalakshmi M, et al. Uncertainty-calibrated thermal-aware scheduling for deadline-constrained edge AI inference. *International Journal of Computational and Experimental Science and Engineering*. 2025;11(3):586-593. https://doi.org/10.22399/ijcesen.3354

[7] Eddine KS, Bagaa M, Ouahouah S, Ouameur MA, Ksentini A. Tail-latency aware scheduler for inference workloads. In: *Proceedings of the 2025 International Wireless Communications and Mobile Computing Conference (IWCMC)*; 2025. https://doi.org/10.1109/IWCMC65282.2025.11059653

[8] Kim J, Chang S, Kwak N. PQK: Model compression via pruning, quantization, and knowledge distillation [preprint]. *arXiv*. 2021. https://doi.org/10.48550/arXiv.2106.14681

[9] Francy S, Singh R. Edge AI: Evaluation of model compression techniques for convolutional neural networks [preprint]. TechRxiv. 2024. https://doi.org/10.36227/techrxiv.172923684.57054990/v1

[10] Lin K, Lin Z, Wang S. *Model compression for IoT edge devices based on knowledge distillation*. International Journal of Innovative Research in Computer Science and Technology. 2025.

[11] Hamza A, Tuama A, Moubark AM. FEM-based hybrid compression framework with pipeline implementation for efficient deep neural network deployment. *Journal of Engineering Research*. 2025. https://doi.org/10.1016/j.jer.2025.03.015

[12] Rawas S, Samala AD, Criollo-C S. SmartWear: Real-time Edge AI for personalized health monitoring in chronic disease management. *Discover Internet of Things*. 2025;5:60. https://doi.org/10.1007/s43926-025-00151-3

[13] Fernandez N, Valdes I, Van Vaerenbergh S, de la Iglesia I, Arratibel J. A comparative study of CNN optimization methods for Edge AI: Exploring the role of early-exit architectures [preprint]. *arXiv*. 2025.

[14] Jang S, Morabito R. Edge-first language model inference: Models, metrics, and trade-offs [preprint]. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2505.16508

[15] Zhang LL, Han S, Wei J, Zheng N, Cao T, Yang Y, et al. nn-Meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices. In: *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*; 2021. p. 81-93. https://doi.org/10.1145/3458864.3467882