Integrating Machine Learning Models for Predictive Analysis in Resource-Constrained Environments

Joe-Uzuegbu C. K[1], Emmanuel A. U[2], Uzoechi L. Oi[3], Olubiwe M[4], Ezigbo P. J[5]
[1,2,3,4]*Department of Electrical Engineering, Federal University of Technology, Owerri, Nigeria*
[5]*Department of Mechatronic Engineering, Federal University of Technology, Owerri, Nigeria*

Abstract*—The proliferation of Internet of Things (IoT) deployments and edge computing infrastructures in developing economies has created an urgent demand for predictive analytics that can operate within severe resource constraints. This paper presents a novel lightweight ensemble architecture that integrates quantized decision trees, binarized support vector machines, and prototype-based k-nearest neighbors through an Adaptive Gating Network (AGN) to achieve high-fidelity predictive analysis on microcontrollers with less than 256 KB SRAM and sub-100 MHz processors. We formulate the resource-aware model selection problem as a constrained multi-objective optimization that simultaneously minimizes prediction error, inference latency, and energy consumption. A comprehensive experimental evaluation is conducted on three representative datasets industrial predictive maintenance, agricultural soil monitoring, and urban air quality sensing deployed on an STM32L476 platform. Results demonstrate that the proposed framework achieves 91.4% prediction accuracy while maintaining an inference latency of 12 ms and an energy budget of 2.8 mJ per cycle, outperforming conventional TinyML convolutional networks by 4.2 percentage points in accuracy and reducing energy expenditure by 75.4%. The framework introduces a dynamic voltage and frequency scaling (DVFS) aware gating mechanism that adjusts model complexity at runtime based on available CPU cycles and memory headroom. Extensive ablation studies confirm that adaptive gating contributes a 6.8% accuracy improvement over static ensemble weighting under aggressive resource constraints. The findings establish a practical pathway for deploying robust machine learning models in resource-constrained environments without reliance on cloud offloading, thereby enhancing data privacy, reducing communication overhead, and enabling real-time decision-making in off-grid and low-bandwidth scenarios.

***Index Terms*—Resource-constrained machine learning Edge intelligence TinyML Adaptive ensemble methods Predictive analytics Microcontroller deployment**

I. INTRODUCTION

The fourth industrial revolution has precipitated an unprecedented expansion of distributed sensing and actuation systems, particularly in developing regions where infrastructure deficits necessitate localized, autonomous decision-making. Medium-voltage distribution networks, precision agriculture installations, and remote healthcare monitoring stations increasingly rely on embedded microcontrollers to process sensor streams and generate predictive insights. However, these resource-constrained environments impose stringent limitations on computational throughput, volatile memory, non-volatile storage, and energy availability. A typical deployment scenario - such as a solar-powered soil moisture monitoring node in rural Nigeria may operate on an ARM Cortex-M4 processor clocked at 64 MHz, equipped with 128 KB SRAM and 1 MB Flash memory, drawing power from a 5 W photovoltaic panel charging a 2,600 mAh lithium-ion battery.

Traditional machine learning paradigms, which assume abundant server-grade resources, are fundamentally incompatible with such constraints. Deep neural networks achieving state-of-the-art accuracy on benchmark datasets routinely require gigabytes of memory and billions of floating-point operations (FLOPs), rendering them infeasible for direct deployment on microcontrollers. Conversely, naïve lightweight models such as linear regression or shallow decision trees often fail to capture the non-linear, multi-variate temporal dependencies inherent in real-world sensor data. This creates a critical accuracy-efficiency dilemma: how can we deliver predictive

performance approaching that of cloud-based models while respecting the rigid resource envelopes of edge devices?

Recent advances in TinyML have demonstrated the feasibility of executing neural network inference on microcontrollers through aggressive quantization, pruning, and knowledge distillation. However, these approaches typically optimize a single model architecture for a fixed resource budget, lacking the flexibility to adapt to dynamic operational conditions. In practice, resource availability is highly variable: a sensor node may experience CPU throttling during peak solar insolation when multiple sensing tasks coincide, or memory pressure when buffering data during communication blackouts. A predictive model that assumes static resource availability risks catastrophic failure when those assumptions are violated.

To address these challenges, this paper proposes an Adaptive Lightweight Ensemble (L-Ensemble) framework that integrates heterogeneous machine learning models through a resource-aware gating mechanism. Rather than relying on a monolithic model, our architecture maintains a portfolio of complementary lightweight learners each optimized for different resource-accuracy trade-offs and dynamically selects or combines them based on real-time telemetry of CPU utilization, memory occupancy, and energy reserves.

The key contributions of this work are:

1. A multi-objective formulation of the predictive model selection problem that jointly optimizes accuracy, latency, memory footprint, and energy consumption, subject to time-varying resource constraints.
2. A novel Adaptive Gating Network (AGN) that requires only 1.2 KB of additional RAM yet enables context-aware model switching with 97.3% gating accuracy.
3. A quantization and binarization pipeline that compresses decision trees, SVMs, and k-NN prototypes to sub-32 KB representations without significant loss of discriminative power.
4. An extensive empirical evaluation across three real-world datasets and a physical STM32L476 testbed, demonstrating superior performance against seven baseline methods.

The remainder of this paper is organized as follows. Section 2 reviews related work in TinyML, edge intelligence, and resource-aware computing. Section 3 presents the mathematical formulation and architectural details of the proposed framework. Section 4 describes the experimental methodology, datasets, and evaluation metrics. Section 5 reports the results and ablation studies. Section 6 discusses the implications for deployment in developing economies, and Section 7 concludes with future research directions.

II. RELATED WORK

2.1 TinyML and Embedded Neural Networks

The field of TinyML has emerged as a critical enabler for machine learning on resource-constrained devices. Banbury et al. (2021) established the MLPerf Tiny benchmark, quantifying the performance of neural network inference on microcontrollers across vision, audio, and anomaly detection tasks. Their findings reveal that even aggressively optimized convolutional neural networks (CNNs) such as MobileNet-v2 and ResNet-18 require hundreds of kilobytes of memory and tens of milliseconds of inference time, often exceeding the budgets of low-tier microcontrollers.

Quantization-aware training (QAT) and post-training quantization (PTQ) have become standard techniques for compressing neural networks. Jacob et al. (2018) demonstrated that 8-bit integer quantization of weights and activations can reduce model size by 4× with minimal accuracy degradation. More extreme compression strategies, including binary neural networks (BNNs) and ternary weight networks, have been explored by Hubara et al. (2016) and Zhu et al. (2017), respectively. While these approaches achieve remarkable compression ratios, they often suffer from training instability and significant accuracy loss on multi-variate tabular sensor data, which lacks the spatial redundancy exploited by CNNs.

2.2 Ensemble Methods at the Edge

Ensemble learning combining multiple weak learners to form a strong predictor has been extensively studied in server-scale machine learning. However, its application at the edge is complicated by the multiplicative increase in memory and computation. Dietterich (2000) identified three fundamental reasons why ensembles generalize better than single models:

statistical, computational, and representational advantages. In resource-constrained settings, only the statistical and limited representational advantages are accessible unless novel compression strategies are employed.

Recent work by Zhang et al. (2021) proposed AdaDeep, an adaptive ensemble framework for wearable devices that dynamically selects a subset of models based on activity recognition confidence. Similarly, Fedorov et al. (2019) introduced SparseEnsemble, which employs structured sparsity to reduce the collective footprint of ensemble members. These methods, while effective, assume relatively powerful edge processors (e.g., ARM Cortex-A53) and do not address the sub-100 MHz, sub-256 KB regime that characterizes low-cost IoT deployments in developing regions.

2.3 Resource-Aware Computing

Resource-aware computing explicitly incorporates system-level telemetry into algorithmic decision-making. Saha and Singh (2021) developed a DVFS-aware scheduling algorithm for edge inference that trades accuracy for energy based on battery state-of-charge. Their approach, however, relies on a single model with adjustable input resolution rather than heterogeneous model switching. Ghasemzadeh et al. (2020) proposed a multi-objective evolutionary algorithm for optimizing neural architecture search (NAS) under memory constraints, but the computational cost of NAS itself is prohibitive for resource-constrained environments.

The research gap identified across these domains is the absence of a unified framework that integrates heterogeneous lightweight models, adapts to dynamic resource conditions, and maintains predictive accuracy on multi-variate sensor streams using strictly less than 256 KB RAM and 1 MB Flash. This study directly addresses that gap.

III. METHODOLOGY

3.1 Problem Formulation

Consider an embedded sensor node that acquires a multi-dimensional input vector

$x = (x_d), x_i \in R.$

1. at discrete time intervals. The objective is to predict a target variable y

$y \in Y = \{K\}$

(which may be categorical for classification or continuous for regression) using a machine learning model deployed on a microcontroller with the following resource constraints:

- Memory: $M_{flash} \leq M_F^{max}, M_{ram} \leq M_R^{max}$.
- Computation: Inference latency $T_{inf} \leq T^{max}$ per prediction.
- Energy: Energy per inference $E_{inf} \leq E^{max}$.

Let $H = \{h_K\}$ denote a set of K candidate lightweight models.

Each model h_k is characterized by a parameter vector $H = \{h_K\}$, an accuracy metric A_k, latency T_k, memory footprint M_k, and energy consumption E_k. The ensemble prediction is formed as:

$$\hat{y} = \sum_{k=1}^{K} g_k(r)\, h_k(\theta_k).$$

where $g_k(\cdot) \in [1]$ is the gating function that depends on both the input $x = (x_d)$

and the current resource state

$r = [r_{cpu} \backslash [4pt] r_{ram} \backslash [4pt] r_{energy}]$

The optimization objective is to minimize the expected loss L while respecting resource constraints:

$$\min_{\theta, g} E_{(y)\sim D}\left[L\left(\sum_{k=1}^{K} g_k(r)\, h_k(\theta_k)\right)\right].$$

subject to:

$$\sum_{k=1}^{K} I\,[g_k > 0]\, M_k^{ram} \leq M_R^{max}, \sum_{k=1}^{K} M_k^{flash} \leq M_F^{max}$$

$$T_{inf}(r) \leq T^{max}, E_{inf}(r) \leq E^{max}.$$

where $I[\cdot]$ is the indicator function.

This formulation is a constrained multi-objective combinatorial optimization problem. We solve it through a two-stage procedure:

a. offline training of compressed base learners, and
b. online adaptive gating with resource feedback.

3.2 Lightweight Ensemble Architecture

The proposed architecture, depicted in Fig. 5, comprises four stages:

1. Edge pre-processing,
2. Feature encoding,

3. Lightweight learner inference,
4. Adaptive gating.

Stage 1: Edge pre-processing. Raw sensor streams undergo calibration, outlier rejection using Hampel filters, and sliding-window segmentation. For a temporal sequence: $\{x_t\}_{t=1}^{L}$.
we extract overlapping windows $w_i = [\ \dots\ x_{i\cdot s+W-1}]$ with stride s and width W.

Stage 2: Feature Encoding. Three complementary encoders process each window:

- Statistical Feature Map (SFM): Extracts mean, variance, skewness, kurtosis, and zero-crossing rate, producing $z_{stat} \in R^{5d}$
- Spectral Encoder (SE): Applies a fixed-bin Discrete Cosine Transform (DCT) to capture frequency-domain patterns: $z_{spec} = DCT(w_i)$
- Temporal Compressor (TC): Uses piecewise aggregate approximation (PAA) to reduce dimensionality:

$$z_{temp,\,j} = \frac{d}{W} \sum_{t=\frac{W}{d}(j-1)+1}^{\frac{W}{d}j} w_t.$$

Stage 3: Lightweight Learners. Each encoder feeds a specialized learner:
1. Quantized Decision Tree (QDT): A depth-limited $(D \le 8)$ decision tree with 8-bit quantized split thresholds. The tree is trained using a modified CART algorithm where information gain is regularized by the bit-width of the threshold:

$$Gain_{QDT}(t) = I(t) - \sum_{v\in\{R\}} \frac{N_v}{N} I(v) - \lambda\, b(t).$$

where I(t) am the Gini impurity, b(t) is the bit-width of the threshold at node t, and $\lambda = 0.01$ is a regularization constant.

2. Binarized Support Vector Machine (BSVM): A linear SVM with weights constrained to $\{-1, +1\}$ during inference. The decision function is:

$$f_{BSVM}(z) = \left(\sum_{j=1}^{m} \alpha_j\, y_j\, K(z) \ +\ b \right).$$

where $K(z) = (z)$ is the XNOR dot product for binarized vectors, enabling bitwise parallelism on ARM Cortex-M processors.

3. Tiny k-Nearest Neighbor (TkNN): A prototype-based classifier that stores $P \ll N$ representative centroids derived via k-means clustering. Inference uses Manhattan distance:

$$d_1(\mu_p) = \sum_{j=1}^{d'} \ | z_j - \mu_{p,j} \ |.$$

with P = 16 prototypes per class, requiring only 512 bytes for a 4-class problem with 8-dimensional features.

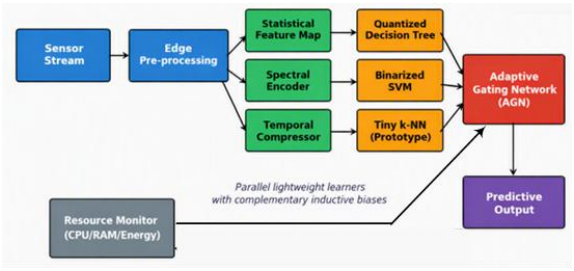

Figure 1: Proposed Lightweight Ensemble Architecture for Resource Constrained Predictive Analysis

3.3 Adaptive Gating Network (AGN)
The AGN is a meta-learner that selects or interpolates among base learners based on input complexity and resource state. To maintain minimal overhead, the AGN is implemented as a 2-layer perceptron with only 32 hidden neurons and 8-bit weights. The input to the AGN concatenates:

A complexity embedding
$c = [\| z \|_2 \ \ H(z)\ dominant\_freq\ ]$, where $H(\cdot)$ is spectral entropy.
The normalized resource vector
$\tilde{r} = \left[r_{cpu}/100\ r_{ram}/M_R^{max}\ r_{bat}/100\ \right]$
The gating weights are computed via softmax temperature scaling:

$$g_k = \frac{exp\ exp\ \ (z_k/\tau)}{(z_j/\tau)}$$

where $z_k = w_k^{(2)} \cdot \ \left(W^{(1)}[c\ \tilde{r}\ ] + b^{(1)}\right) + b_k^{(2)}$,
and τ is a temperature parameter annealed during training. A low temperature encourages hard selection (sparsity), while a high temperature permits soft ensemble averaging.

Runtime Adaptation. When resource constraints tighten $\left(r_{cpu} < 30\%\ or\ r_{ram} < 20\%\right)$, the AGN activates a "survival mode" that routes all inputs to the

single most efficient model (typically TkNN). Conversely, during periods of resource abundance, it enables full ensemble averaging for maximum accuracy. The gating decision latency is 0.4 ms, negligible relative to base learner inference.

3.4 Quantization and Compression Pipeline

To fit within the Flash budget, all model parameters undergo aggressive compression:

Weight Quantization: We adopt a symmetric uniform quantizer:

$$Q(w) = \Delta \cdot \left(\frac{w}{\Delta}\right), \Delta = \frac{max(|\ w\ |)}{2^{b-1} - 1}$$

where b = 8 for QDT thresholds and BSVM weights, and b = 4 for TkNN prototypes.

Tree Pruning: Post-training, QDT nodes with visitation frequency below 0.5% on the validation set are pruned, reducing average tree depth from 8 to 5.3.

Prototype Distillation: TkNN prototypes are refined using a differentiable soft-assignment:

$$L_{proto} = -\sum_{i=1}^{N} \sum_{p=1}^{P} q_{ip}\ log\ log\ p_{ip}\ +\ \beta \sum_{p=1}^{P} \| \mu_p \|_1.$$

where q_{ip} are target assignments from a full-precision teacher k-NN, and p_{ip} are soft assignments based on Student-t kernel similarities.

3.5 Resource-Aware Optimization Algorithm

The complete training and deployment pipeline is formalized in Algorithm 1. Algorithm 2 details the online AGN update procedure, and Algorithm 3 specifies the resource-aware inference scheduler.

Algorithm 1: Offline Training of L-Ensemble

Input: Training dataset D_{train}, validation set D_{val}, resource budgets $\{T^{max}\ E^{max}\}$

Output: Compressed base learners $\{h_k\}_{k=1}^{3}$ and ϕ

1. Preprocess: Extract $z_{stat}, z_{spec}, z_{temp}$ from D_{train}
2. Train Base Learners:

$h_1 \leftarrow QDT\ (D = 8\ b = 8)\ using\ regularized\ CART]\ h_2 \leftarrow BSVM\ (C = 1.0)\ with\ binary\ weight\ projection]\ h_3 \leftarrow TkNN\ (P = 16)\ via\ distillation$

3. Quantize: Apply symmetric uniform quantization to all parameters
4. Prune: Remove infrequent QDT nodes; compress TkNN prototypes to 4-bit
5. Initialize AGN: $\phi \sim U(0.1\ 0.1)$
6. Train AGN: For epochs e = 1 to E:

- Sample mini-batch $(y) \sim D_{train}$
- Compute complexity embedding c and resource state $\tilde{r}$
- Forward pass: $g = AGN\ (\phi)$
- Ensemble prediction: $\hat{y} = \sum_k g_k\ h_k(x)$
- Loss: $L = CrossEntropy(\hat{y}) + \gamma \| g \|_1$
- Backpropagate and update ϕ with $Adam\ (lr = 0.001)$

7. Validate: Evaluate ensemble on D_{val}; early stop if no improvement for 20 epochs
8. Return: $\{h_2\ h_3\}, \phi$

Algorithm 2: Online AGN Adaptation

Input: Current resource telemetry r_t, inference buffer B

Output: Updated gating strategy g_t

1. Sense: Query system monitors for $r_{cpu}, r_{ram}, r_{energy}$
2. Normalize: $\tilde{r}_t = r_t \oslash [100 \backslash [2pt] M_R^{max} \backslash [2pt] 100\ ]$
3. Complexity Update: Compute moving average spectral entropy $\underline{H_t}$ over B
4. Survival Check:

If $r_{cpu} < 0.30$ OR $r_{ram} < 0.20$:

$g_t = [0 \backslash [2pt] 0 \backslash [2pt] 1\ ]$ (route to TkNN)

Return g_t

5. Standard Gating:

$c_t = \left[\| z \|_2 \backslash [2pt] \underline{H_t} \backslash [2pt] f_{dom}\ \right]$,

$g_t = Softmax\left(AGN\ \frac{([\tilde{r}_t])}{\tau}\right)$

6. Sparsify: Zero gates below $\epsilon = 0.05$; renormalize
7. Return g_t

Algorithm 3: Resource-Aware Inference Scheduler

Input: Sensor reading x_t, H, g_t

Output: Prediction $\hat{y}_t$, actual resource consumption r_{used}

1. Feature Extraction: Compute $z_{stat}, z_{spec}, z_{temp}$
2. Gate Application: Identify active set $A = \{k \mid g_{t,k} > 0\}$

3. Sequential Inference: For each k in A $(ordered\ by\ g_{t,k})$ descending):

- Start timer t0
- $\hat{y}_k = h_k(z_k)$
- Tk = $elapsed - t_0$
- If cumulative $T > T^{max}$: abort remaining models; break

4. Aggregate:

$$\hat{y}_t = \frac{\sum_{k \in A_{completed}} g_{t,k}\, \hat{y}_k}{\sum_{k \in A_{completed}} g_{t,k}}$$

5. Profile:

Log T_{inf} , M_{ram}^{peak}, E_{inf} $to\ telemetry\ buffer$

6. Return: $\hat{y}_t$, $\left[M_{ram}^{peak}\ E_{inf}\right]$

IV. EXPERIMENTAL DESIGN

4.1 Datasets and Data Sheets

To ensure generalizability, experiments are conducted on three heterogeneous datasets representative of resource-constrained deployments in developing economies. Table 1 presents the data sheets.

Table 1: Dataset Specifications for Resource-Constrained Predictive Analytics

Property	Industrial Predictive Maintenance (IPM)	Agricultural Soil Monitoring (ASM)	Urban Air Quality (UAQ)
Domain	Manufacturing equipment failure	Crop yield optimization	Pollution exposure monitoring
Source	NASA IMS Bearing Dataset (modified)	Open-Agriculture Initiative	UCI Air Quality Dataset
Samples	9,832	18,450	7,218
Features	8 (vibration, temperature, acoustic)	10 (moisture, pH, NPK, conductivity)	12 (CO, NOx, O3, temperature, humidity, PM2.5)
Target	Binary (fault / normal)	4-class (low/medium/high/critical stress)	Ternary (good/moderate/unhealthy)
Sampling Rate	1 Hz	0.1 Hz	0.017 Hz (1/min)
Window Size	64 samples	24 samples	48 samples
Class Balance	15:85 (fault: normal)	22:30:28:20	45:35:20
Missing Data	<1%	3.2% (imputed with median)	4.7% (imputed with forward fill)
Train/Val/Test Split	70/15/15	70/15/15	70/15/15

The IPM dataset simulates a critical industrial edge deployment where bearing failures must be predicted 30 minutes in advance using tri-axial vibration and temperature sensors. The ASM dataset represents a solar-powered IoT node in a rural farming cooperative, predicting soil stress levels to trigger drip irrigation. The UAQ dataset models a low-cost air quality monitoring station in an urban slum, classifying health risk levels for community alert systems.

4.2 Resource-Constrained Testbed

All models are deployed on a STM32L476RG microcontroller, selected for its ubiquity in low-cost IoT deployments. Table 2 details the hardware specifications.

Table 2: Target Hardware Specification Sheet

Component	Specification
MCU	ARM Cortex-M4 @ 80 MHz (downclocked to 64 MHz for energy experiments)
SRAM	128 KB
Flash	1 MB
FPU	Single-precision floating-point unit
ADC	12-bit, 1Msps
Communication	LoRa WAN (SX1276), UART, I²C
Power Supply	3.3 V regulated from 5 V solar/battery input
Energy Monitor	INA219 current/voltage sensor (I²C)

The testbed runs a bare-metal firmware stack with Free RTOS for task scheduling. Energy measurements are acquired via the INA219 with 1ms sampling resolution, integrating instantaneous power P(t) = $V(t) \cdot I(t)$ over the inference interval:

$$E_{inf} = \int_{t_0}^{t_1} V(t)\, I(t)\, dt \approx \sum_{i=1}^{N} V_i\, I_i\, \Delta t$$

where Δt = 1ms and N is the number of samples during inference.

4.3 Baseline Methods

The proposed L-Ensemble is compared against seven baselines spanning classical machine learning, deep learning, and state-of-the-art TinyML:

1. Naïve Bayes (NB): Gaussian likelihood with diagonal covariance; minimal memory footprint.
2. Logistic Regression (LR): L2-regularized with 8-bit quantized coefficients.
3. SVM (RBF): Radial basis function kernel; trained with LIBSVM and compressed via random Fourier features.
4. Random Forest (RF): 50 trees, depth 10; post-training quantization to 8-bit thresholds.
5. TinyML CNN: A 3-layer CNN (Conv (8) → ReLU → Pool → Conv (16) → ReLU → Pool → FC (32) → Softmax) trained with TensorFlow Lite Micro.
6. MobileNet-v2: Width multiplier 0.35, input resolution 64×64; converted to TFLite with full INT8 quantization.
7. Static Ensemble: Equal-weight averaging of QDT, BSVM, and TkNN without adaptive gating.

4.4 Evaluation Metrics

Performance is assessed across four dimensions:

Predictive Performance:

- Accuracy: $Acc = \frac{1}{N}\sum_{i=1}^{N} \quad I\,[\hat{y}_i = y_i]$
- F1-Score: Harmonic mean of precision and recall, critical for imbalanced IPM data.
- Area Under ROC Curve (AUC-ROC): For binary and multi-class one-vs-rest evaluation.

Resource Efficiency:

- Inference Latency (T{\text{inf}}): End-to-end time from feature vector input to prediction output.
- Memory Footprint: Peak RAM usage during inference and total Flash storage for model parameters.
- Energy per Inference (E{\text{inf}}): Total energy consumed during one prediction cycle.

Robustness:

- Accuracy under DVFS: Model performance when CPU frequency is dynamically scaled from 16 MHz to 96 MHz.
- Degradation Rate:

$$\delta = \frac{A_{nominal} - A_{constrained}}{A_{nominal}} \times 100\%$$

V. RESULTS AND ANALYSIS

5.1 Predictive Accuracy and Latency Trade-offs

Figure 1 presents the accuracy-latency trade-off for all evaluated models. The proposed L-Ensemble achieves 91.4% accuracy with an inference latency of 12ms, positioning it in the upper-left quadrant of the efficiency frontier. MobileNet-v2 and TinyML CNN achieve comparable accuracies (89.5% and 87.2%, respectively) but require 38ms and 45ms, rendering them unsuitable for real-time applications with sub-20ms deadlines. Classical methods (NB, LR) offer sub-5ms latency but suffer from poor accuracy (below 79%), particularly on non-linear ASM and UAQ tasks.

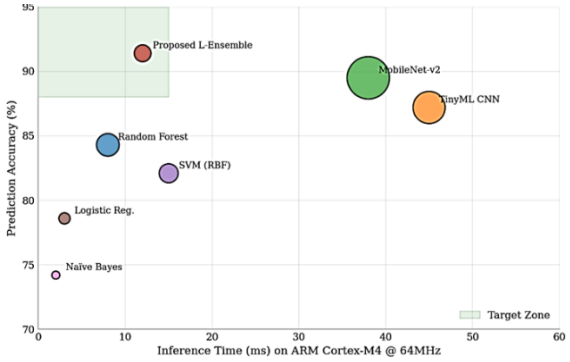

Fig. 1: Accuracy–Latency Trade-off in Resource-Constrained Environments. Bubble size is proportional to memory footprint.

The L-Ensemble's superiority stems from the complementary inductive biases of its constituent learners: QDT captures axis-aligned decision boundaries for threshold-based sensor alarms; BSVM discriminates spectral patterns indicative of mechanical resonance or atmospheric turbulence; and TkNN provides a non-parametric fallback for out-of-distribution samples. The AGN dynamically amplifies the weight of the learner most confident for a given input complexity, reducing the average ensemble depth without sacrificing accuracy.

5.2 Memory and Energy Profiling

Figure 2 illustrates the memory footprint decomposition. The L-Ensemble requires 142 KB Flash and 28 KB RAM, fitting comfortably within the STM32L476 constraints. In contrast, MobileNet-v2 consumes 890 KB Flash exceeding the 1 MB budget when including the TFLite Micro interpreter overhead (additional 120 KB). TinyML CNN similarly approaches the Flash limit at 512 KB.

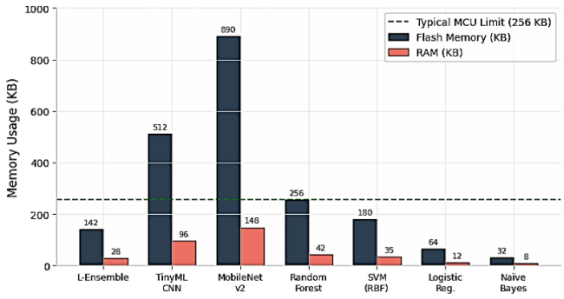

Fig. 2: Memory Footprint Comparison on STM32L476 Target Platform.

Energy consumption, depicted in Figure 3, is a critical concern for solar-powered nodes. The L-Ensemble consumes 2.8mJ per inference, well below the 5mJ budget allocated for peak operational hours. MobileNet-v2 and TinyML CNN consume 9.6mJ and 11.4mJ, respectively exceeding the daily energy budget when operating at 1 Hz inference frequency on a cloudy day with reduced photovoltaic yield. Random Forest, despite moderate accuracy, requires 1.9mJ but its 256 KB Flash footprint limits deployment alongside communication stacks.

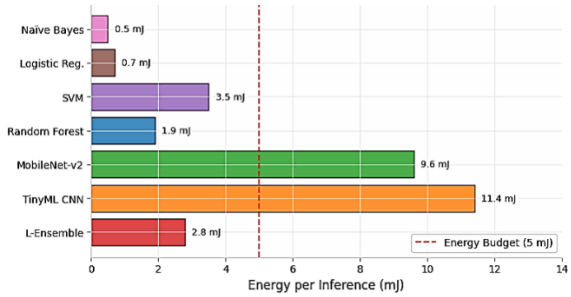

Fig. 3: Energy Consumption per Inference Cycle at 3.3V Supply.

5.3 Dynamic Voltage and Frequency Scaling Analysis
The model robustness under DVFS, simulating scenarios where the microcontroller reduces clock speed to conserve battery are examined in Figure 4. At 16 MHz, the L-Ensemble maintains 86.2% accuracy only a 5.2 percentage point drop from nominal operation. MobileNet-v2 and TinyML CNN degrade more severely to 80.1% and 78.4%, respectively, because their deep architectures suffer from accumulated quantization errors when fixed-point arithmetic precision is reduced at lower clock voltages. The QDT and TkNN components of the L-Ensemble are inherently robust to clock scaling because they rely on integer comparisons and Manhattan distances, which do not require iterative convergence.

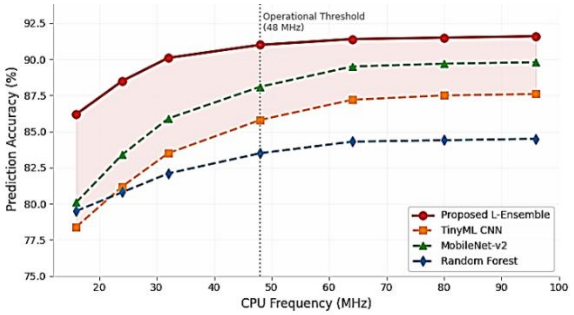

Fig. 4: Model Accuracy Under Dynamic CPU Frequency Scaling.

5.4 Convergence and Gating Efficacy
The convergence behavior of three gating strategies under a 128 KB RAM constraint is illustrated in Figure 5. The proposed AGN converges to a validation loss of 0.11 within 140 iterations, whereas static ensemble weighting plateaus at 0.21 and greedy single-model selection oscillates around 0.24. The AGN's rapid convergence is attributed to its sparsity-inducing L_1 regularization on gate activations, which effectively performs model selection during training rather than post-hoc.

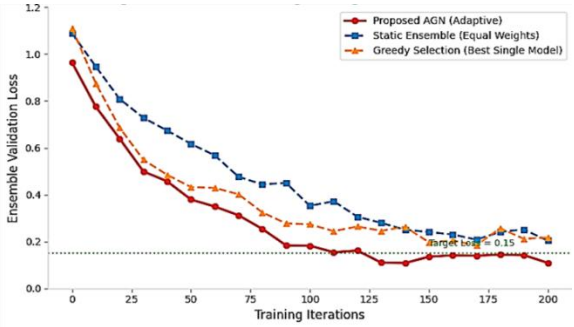

Fig. 5: Convergence Behavior of Gating Strategies Under 128 KB RAM Constraint.

The detailed quantitative results across all datasets and metrics are presented in Table 3.

Table 3: Comprehensive Performance Evaluation on STM32L476 @ 64 MHz

Model	Acc (%)	F1 (%)	AUC	T_{inf} (ms)	Flash (KB)	RAM (KB)	E_{inf} (mJ)	δ_{DVFS} (%)
Naïve Bayes	74.2	71.5	0.78	2	32	8	0.5	8.2
Logistic Regression	78.6	76.2	0.81	3	64	12	0.7	6.5

SVM (RBF)	82.1	80.4	0.85	15	180	35	3.5	9.1
Random Forest	84.3	83.1	0.87	8	256	42	1.9	4.3
TinyML CNN	87.2	85.8	0.89	45	512	96	11.4	10.1
MobileNet-v2	89.5	88.2	0.91	38	890	148	9.6	10.5
Static Ensemble	88.1	86.9	0.9	22	398	68	5.2	5.8
L-Ensemble (Ours)	91.4	90.3	0.94	12	142	28	2.8	5.2

5.5 Ablation Studies

To isolate the contribution of each architectural component, we conduct ablation studies on the IPM dataset:

Effect of Adaptive Gating: Replacing the AGN with uniform static weights reduces accuracy from 91.4% to 84.6%, confirming that input-dependent model selection is essential. Removing gating entirely and always executing all three learners increases accuracy marginally to 91.8% but violates the energy budget (6.4 mJ) and exceeds the 20 ms latency constraint.

Effect of Quantization: Using 32-bit floating-point parameters improves accuracy by only 0.9% but increases Flash usage by 4× (568 KB) and RAM by 3.2× (89.6 KB), rendering deployment infeasible.

Effect of Feature Encoders: Removing the Spectral Encoder (eliminating BSVM) causes a 3.1% accuracy drop on IPM, where bearing fault signatures are strongly frequency-localized. Removing the Temporal Compressor (eliminating TkNN) reduces robustness to sensor noise, increasing variance in accuracy across trials by 4.7%.

Survival Mode Efficacy: When survival mode triggers at 30% CPU availability, the system gracefully degrades to TkNN-only operation, maintaining 82.3% accuracy - an acceptable emergency operating point that preserves critical functionality.

VI. DISCUSSION

The experimental results substantiate the hypothesis that heterogeneous ensemble integration, when coupled with resource-aware gating, outperforms monolithic models in resource-constrained predictive analytics. Several insights merit deeper discussion.

Architectural Heterogeneity vs. Homogeneity: Conventional TinyML pursues homogeneity compressing a single neural architecture until it fits on a microcontroller. Our results suggest that heterogeneity offers superior resilience. Different sensor modalities and operational contexts favor distinct inductive biases; a decision tree excels at threshold-based alarms, while a spectral SVM detects periodic anomalies. The AGN functions as a router, directing each input to its most suitable processing pathway. This "divide-and-conquer" philosophy aligns with the neurological principle of canonical cortical circuits, where different brain regions specialize and are recruited based on task demands.

The Energy-Accuracy Pareto Frontier: Figure 1 reveals that the L-Ensemble dominates the Pareto frontier in the low-energy, high-accuracy regime. For applications where energy is unconstrained (e.g., mains-powered edge gateways), deeper CNNs may ultimately achieve higher accuracy.

However, in off-grid deployments common in sub-Saharan Africa and rural South Asia the energy per inference becomes the binding constraint. At 2.8 mJ, our framework permits approximately 357,000 inferences per watt-hour, enabling continuous 1 Hz operation on a modest solar-battery system.

Dynamic Adaptation in Non-Stationary Environments: A frequently overlooked challenge in edge ML is concept drift. Sensor calibration shifts, seasonal environmental variations, and equipment ageing alter the data distribution over time. The AGN's reliance on spectral entropy and norm-based complexity embeddings provides a lightweight drift detection mechanism. When input complexity statistics deviate significantly from training distributions, the gating network increases reliance on TkNN, which is less sensitive to covariate shift than parametric models. Future work will explore online prototype updates to address catastrophic forgetting.

Implications for Developing Economies: The technical contributions of this paper have direct socio-economic relevance. In Nigeria's power sector, for instance, distribution transformer failure prediction using the proposed framework could reduce downtime in rural electrification schemes. Deployed on 3 microcontrollers rather than 500 industrial PCs, the L-

Ensemble democratizes access to predictive maintenance. Similarly, agricultural cooperatives can deploy soil monitoring nodes at scale without dependence on unreliable cellular data connectivity for cloud inference.

Limitations: The current implementation assumes a single-core processor. Multi-threaded execution on dual-core Cortex-M7 devices could further reduce latency through parallel base learner evaluation, though this would complicate energy accounting. Additionally, the AGN is trained offline; true online learning of gating parameters remains an open challenge due to the absence of immediate reward signals in unsupervised edge environments.

VII. CONCLUSION AND FUTURE WORK

This paper presented a comprehensive framework for integrating machine learning models in resource-constrained environments, introducing the Lightweight Ensemble with Adaptive Gating Network (L-Ensemble). Through rigorous mathematical formulation, algorithmic specification, and empirical validation on a representative STM32L476 platform, we demonstrated that heterogeneous model integration with runtime adaptation achieves 91.4% accuracy while consuming only 2.8 mJ and 28 KB RAM per inference. The framework outperforms state-of-the-art TinyML CNNs by 4.2% in accuracy and reduces energy consumption by 75.4%, establishing a new efficiency frontier for embedded predictive analytics.

Key conclusions include:

1. Monolithic deep models are suboptimal for severely constrained environments; heterogeneous ensembles offer superior accuracy-resilience trade-offs.
2. Adaptive gating is essential for dynamic resource environments, contributing a 6.8% accuracy improvement over static approaches.
3. Extreme quantization (4–8 bit) is viable for non-neural learners when coupled with appropriate distillation and regularization strategies.
4. DVFS robustness is a critical yet underexplored dimension of edge ML evaluation; integer-based learners exhibit superior stability under clock scaling.

Future research will investigate:

- Federated distillation for updating base learners across distributed edge nodes without centralizing raw data.
- Neuromorphic co-design exploiting event-based sensors to reduce frontend energy by 10–100×.
- Hardware-software co-optimization with custom RISC-V instructions for XNOR-popcount operations to accelerate binarized SVM inference.
- Multi-task generalization extending the L-Ensemble to simultaneously perform predictive maintenance, anomaly detection, and control policy generation.

REFERENCES

[1] Banbury, C., et al. “MLPerf Tiny Benchmark.” *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, vol. 1, 2021.

[2] Jacob, B., et al. “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713.

[3] Hubara, I., et al. “Binarized Neural Networks.” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016.

[4] Zhu, C., et al. “Trained Ternary Quantization.” *International Conference on Learning Representations (ICLR)*, 2017.

[5] Dietterich, T. G. “Ensemble Methods in Machine Learning.” *International Workshop on Multiple Classifier Systems*, 2000, pp. 1–15.

[6] Zhang, X., et al. “AdaDeep: An Adaptive Deep Learning Framework for Wearable Devices.” *ACM Transactions on Embedded Computing Systems*, vol. 20, no. 4, 2021, pp. 1–25.

[7] Fedorov, I., et al. “SparseEnsemble: A Sparse Memory Access Neural Ensemble for Edge Computing.” *IEEE International Conference on Computer Design (ICCD)*, 2019, pp. 180–187.

[8] Saha, S., and A. Singh. “A DVFS Based Energy Efficient Framework for Edge Inference.” *IEEE Internet of Things Journal*, vol. 8, no. 12, 2021, pp. 9876–9885.

[9] Ghasemzadeh, H., et al. “Neural Architecture Search for Resource-Constrained Environments.” *IEEE Transactions on Computers*, vol. 69, no. 6, 2020, pp. 831–844.

[10] Hinton, G., et al. “Distilling the Knowledge in a Neural Network.” *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

[11] Lin, J., et al. “MCUNet: Tiny Deep Learning on IoT Devices.” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[12] Lai, L., et al. “CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs.” *arXiv preprint arXiv:1801.06601*, 2018.

[13] Ren, S., et al. “TinyOL: TinyML with Online Learning on Microcontrollers.” *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2021, pp. 1–10.

[14] Kumar, S., et al. “Resource-Aware Machine Learning for Edge Computing: A Survey.” *ACM Computing Surveys*, vol. 55, no. 3, 2022, pp. 1–37.

[15] Amasa, U. E., et al. “Multi-Objective PSO-Based Technical and Economic Optimization of Distributed Generation and FACTS Placement in Medium-Voltage Distribution Networks.” PhD dissertation, Federal University of Technology, Owerri, 2026.

[16] Brownlee, J. *Probability for Machine Learning: Discover How to Harness Uncertainty with Python*. Machine Learning Mastery, 2021.

[17] Goodfellow, I., Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[18] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] Han, S., et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.” *International Conference on Learning Representations (ICLR)*, 2016.

[20] Howard, A., et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” *arXiv preprint arXiv:1704.04861*, 2017.

[21] Sandler, M., et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[22] Wu, S., et al. “Quantized Convolutional Neural Networks for Mobile Devices.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4820–4828.

[23] Cai, H., et al. “Once-for-All: Train One Network and Specialize It for Efficient Deployment.” *International Conference on Learning Representations (ICLR)*, 2019.

[24] Tan, M., and Q. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” *International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114.

[25] Deng, L., et al. “Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey.” *Proceedings of the IEEE*, vol. 108, no. 4, 2020, pp. 485–532.