A Comparative Analysis of Pruning, Quantization, and Compilation for LightGBM-Based Electricity Anomaly Detection on IoT Edge Devices

Soiful Hadi
Diponegoro University, Semarang, Indonesia | Semarang University, Semarang, Indonesia
saiful@usm.ac.id

Wahyul Amien Syafei
Diponegoro University, Semarang, Indonesia
wasyafe@gmail.com (corresponding author)

Adi Wibowo
Diponegoro University, Semarang, Indonesia
bowo.adi@live.undip.ac.id

Wahyu Maulana Hassanudin
Semarang University, Semarang, Indonesia
whyumaulan@gmail.com

Eko Fidia Setiana
Semarang University, Semarang, Indonesia
eko.wongmbanjar@gmail.com

Astrid Novita Putri
Semarang University, Semarang, Indonesia
astrid@usm.ac.id

Received: 22 November 2025 | Revised: 31 December 2025 and 13 January 2026 | Accepted: 17 January 2026

 DOI: https://doi.org/10.48084/etasr.16433

ABSTRACT

Operating machine learning models designed for electricity theft detection on resource-limited Internet of Things (IoT) edge devices involves significant trade-offs among inference speed, power utilization, and detection precision. This paper provides the first detailed comparison of pruning, quantization, and compilation strategies on Raspberry Pi 4 hardware for Light Gradient Boosting Machine (LightGBM)-based anomaly detection. We tested three optimization strategies: pruned models, Open Neural Network Exchange (ONNX) 8-bit integer (INT8) quantization, and Treelite compilation against a baseline native version for 1,000 inference cycles under practical operating conditions. ONNX INT8 quantization produced a 40.66× speedup for real-time inference (0.091 ms), but exhibited significant thermal load (80.67% CPU utilization, +7.3 °C). Treelite offered best-in-class batch processing efficiency (13.41× speedup, 25.99% CPU utilization). All approaches retained 96.98% accuracy and produced an identical confusion matrix to the baseline model. Experiments indicate that the choice of the optimization strategy hinges critically on deployment scenarios, such as real-time streaming versus occasional batch processing. Our results give practical recommendations for implementers of edge-based grid smart-monitoring systems.

Keywords-edge computing; LightGBM optimization; electricity theft detection; model quantization; IoT deployment

I. INTRODUCTION

A machine learning algorithm that achieves high accuracy in detecting electricity theft is rendered ineffective if it cannot function within the 100 ms response interval needed for real-time monitoring of the grid on a Raspberry Pi 4 (costing approximately $75). Electricity theft and consumption irregularities cost utility companies billions of dollars annually and destabilize grid infrastructure [1, 2]. However, the persistent deployment gap between laboratory performance and edge feasibility prevents sophisticated anomaly detection systems from being effectively implemented on resource-constrained Internet of Things (IoT) devices used in smart grids. The integration of IoT sensors and edge computing into smart grid infrastructures has enabled continuous monitoring of electricity consumption patterns across industrial and residential environments [1, 3, 4]. Gradient boosting algorithms such as Light Gradient Boosting Machine (LightGBM) have been extensively compared and analyzed in prior work [5]. Complementary multitiered edge-cloud architectures have also been proposed to enhance anomaly detection responsiveness and scalability in smart meter networks [3]. Recent studies have highlighted the effectiveness of LightGBM-based models in energy forecasting and load analysis, further reinforcing the potential of gradient-boosting algorithms for smart grid intelligence [6]. Yet, the computational demands of real-time anomaly detection on edge devices remain a critical bottleneck [7]. Similar real-time edge Artificial Intelligence (AI) deployments in smart grid and IoT environments have reported latency and resource trade-offs consistent with these challenges [4, 8].

While our prior work demonstrated that LightGBM outperforms Convolutional Neural Network–Long Short-Term Memory (CNN-LSTM) and Convolutional Neural Network–Extreme Gradient Boosting (CNN-XGBoost) architectures for electricity anomaly detection [9], deploying this highly accurate model on edge hardware raises a critical question: which optimization—pruning, quantization, or compilation—is necessary to achieve real-time functionality without sacrificing detection precision?

The issues associated with edge deployment for tree-based models such as LightGBM are distinct. Tree-based ensemble models have previously been applied to electricity theft detection on smart meter data, demonstrating strong performance in identifying non-technical losses [10, 11]. In contrast to Deep Neural Networks (DNNs) that utilize GPU acceleration, Gradient-Boosted Decision Tree (GBDT) models utilize histogram-based gradient boosting [12] and depend on sequential CPU-based inference with irregular memory access [13], which poses unique optimization challenges for resource-constrained IoT devices. This model must operate on power-efficient hardware (limited RAM, passive cooling) and address domain-specific challenges, such as significant class imbalance where false negatives are particularly harmful. Recent advances in Process-In-Memory (PIM) edge AI chips for electrical anomaly detection have highlighted similar trade-offs between energy efficiency and inference reliability [8, 14].

State-of-the-art optimization literature offers pragmatic advice to practitioners with edge deployment choices. Model compression methods like pruning, quantization, and knowledge distillation have been rigorously investigated for DNNs [15, 16], but their potential for tree-based models is relatively understudied. GBDT model optimization studies mostly focus on algorithmic accuracy gains in high-end computing scenarios, assess Database Management System (DBMS)-style compilation frameworks without IoT-specific requirements such as thermal management and power, and analyze a single optimization technique like tree pruning or quantization without comparison [17]. Edge machine learning deployment studies have mostly been geared toward computer vision applications with CNNs [16] and time-series anomaly detection in IoT multivariate data using graph-based models [8].

The distinct nature of ensemble tree models, including their larger memory footprint and irregular inference-time memory access patterns, poses optimization challenges on edge devices [17]. No previous work comprehensively compares pruning, quantization, and compilation schemes under realistic edge operating scenarios with a full suite of deployment-sensitive metrics measured: inference latency, memory footprint, CPU utilization, thermal characteristics, and accuracy retention across an unequally distributed dataset. To address this gap, we provide the inaugural systematic empirical comparison of pruning, quantization, and compilation for LightGBM-based electricity anomaly detection on Raspberry Pi 4 devices. Beginning with the model baseline, we develop three optimized variants: (1) an importance-guided pruned model, (2) an 8-bit integer (INT8)-quantized Open Neural Network Exchange (ONNX) version, and (3) a Treelite-compiled model. Each configuration is tested in actual operating settings to assess latency, memory usage, CPU utilization, and thermal performance. This research addresses inference delay, resource trade-offs, and the ideal compromise for smart grid installations, offering the first thorough, multi-metric comparison of LightGBM on edge devices.

The key contributions of this paper are:

1. The first comparative analysis of pruning, quantization, and compilation for LightGBM on IoT edge devices.
2. A comprehensive performance evaluation across latency, memory, CPU utilization, thermal characteristics, and accuracy metrics under realistic operating conditions.
3. Empirical insights into trade-offs between optimization strategies for imbalanced, time-dependent anomaly detection.

Unlike prior edge-deployment studies that typically evaluate a single acceleration method or focus on DNNs, this work provides a side-by-side, multi-metric comparison of three practical optimization routes for a tree-ensemble (LightGBM) on commodity IoT hardware. Beyond latency, we quantify RAM footprint, CPU utilization, and thermal impact under two

realistic deployment scenarios (streaming single-sample inference and periodic batch analysis), and we report whether optimization affects both classification outcomes and probability calibration. The resulting trade-off analysis offers actionable guidance for smart-grid implementers selecting an optimization strategy.

II. METHODOLOGY

The evaluation of LightGBM optimization techniques was conducted through a five-phase framework depicted in Figure 1: (1) data acquisition and preprocessing involving field collection, rule-based labeling, and domain-knowledge augmentation; (2) baseline model training with 10-run, 5-fold cross-validation; (3) three optimization methods: pruning, ONNX INT8 quantization, and Treelite compilation; (4) dual-scenario benchmarking on Raspberry Pi 4 to capture performance under realistic smart grid deployment conditions: real-time single-sample inference simulating immediate sensor stream processing, and batch processing representing periodic gateway analysis of accumulated readings; and (5) statistical analysis using t-tests, McNemar's test, and Cohen's d effect sizes.

A. Dataset Collection and Preprocessing

1) Data Acquisition

The dataset was obtained over a four-month period from IoT energy monitors installed in various institutional and residential settings in Central Java, Indonesia. This deployment produced 10,075 records, each consisting of six electrical characteristics: voltage (V), current (A), active power (W), cumulative energy consumption (kWh), frequency (Hz), and power factor sampled at five-minute intervals. Anomaly labels were assigned using a rule-based methodology that combines (i) current and energy limits reported for Indonesian Perusahaan Listrik Negara (PLN) residential customers [18], and (ii) the low-voltage service-connection standard issued by PLN, which specifies maximum current and 2-hour energy limits of 2.05 A / 0.9 kWh (450 VA), 4.09 A / 1.8 kWh (900 VA), and 5.90 A / 2.6 kWh (1300 VA) [19]. The detailed anomaly-labeling rules follow the procedure in [18], whereas the contracted power classes and associated Miniature Circuit Breaker (MCB) ratings are taken from [19].

2) Augmentation and Validation

Domain-knowledge-driven augmentation expanded the training set to 20,000 samples (15% anomalies: overload, contextual, power factor) [11]. Performance metrics were evaluated solely on the original field-collected data to avoid synthetic bias.

3) Model Training and Validation

A LightGBM model was selected as the baseline algorithm for its high efficiency and scalability [12, 20]. The model was configured with constrained hyperparameters to balance accuracy and edge-deployment efficiency: tree structure (n_estimator = 25, max_depth = 2, num_leaves = 15, min_data_in_leaf = 50), learning parameters (learning_rate = 0.2, min_split_gain = 0.1), regularization (lambda_l1 = 0.5, lambda_l2 = 0.5), and boosting_type = gbdt with random_state = 42. Model robustness was evaluated using a 10-run, 5-fold stratified cross-validation protocol [5]. After validation, a final baseline model was trained on the full augmented training set and serialized as lightgbm_model.pkl (670.27 KB). The model achieved 96.98% accuracy, 0.9636 Receiver Operating Characteristic–Area Under the Curve (ROC-AUC), and 0.89 F1-score on the held-out test set, which serves as the baseline for all subsequent comparisons.

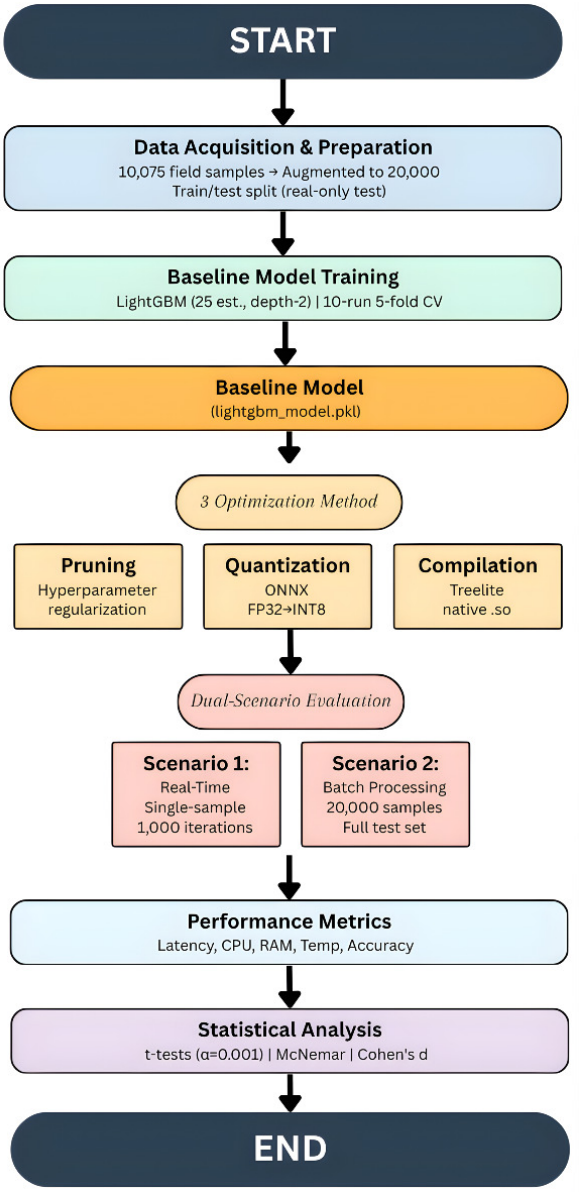

Fig. 1. Systematic methodology framework.

B. Model Optimization

From the validated baseline model, three optimized variants were constructed to study distinct approaches for enhancing deployment efficiency on the edge device: (1) a pruned model, (2) an INT8-quantized model, and (3) a compiled model.

1) Native (Baseline) and Pruned Models

The baseline configuration consists of the trained LightGBM model serialized in Python's standard pickle format (lightgbm_model.pkl). Inference is performed using the official LightGBM Python module. The pruned model was created through aggressive regularization during training: num_leaves reduced from 15 to 8 and min_data_in_leaf increased from 50 to 100, achieving a 97.3% reduction in model size (18.11 KB).

2) Quantized Model (ONNX INT8)

When creating a quantized variant, the model underwent a two-step conversion. First, the trained LightGBM model was exported to the ONNX format with standard 32-bit floating-point precision (FP32). Subsequently, dynamic quantization was applied through the ONNX Runtime toolbox, converting numerical weights from FP32 to compact INT8 representations. This conversion employs a linear quantization scheme, which maps a real-valued weight r (FP32) to a quantized integer q (INT8) using a scale S and a zero-point Z:

$$r = S \cdot (q - Z) \quad (1)$$

The resulting INT8 model theoretically offers a 4× reduction in weight storage, although the overall serialized model size was reduced by 34% due to ONNX graph overhead (Table I).

3) Compiled Model (Treelite)

To minimize software overhead, the baseline model was built into a native shared library (.so) using Treelite [21], a specialized compiler for tree-based models. Treelite transforms the whole decision tree ensemble into highly efficient C code. This code was then compiled for the target ARM architecture using GCC with the aggressive -O3 optimization level and options enabling NEON vectorization. The resulting compiled library may be called directly from Python, bypassing the interpreter for the fundamental prediction logic and allowing the model to operate as efficient, precompiled machine code.

C. Benchmarking Protocol

A systematic benchmarking methodology was run on the Raspberry Pi 4 to evaluate the deployment performance of the three model versions.

1) Metrics and Tools

The evaluation focused on six main metrics. Inference latency (L , in ms) was assessed using Python's time.perf_counter(), and throughput was calculated as its inverse, scaled to seconds:

$$\text{Throughput} = \frac{1000}{L_{mean_{(ms)}}} \quad (2)$$

Model size (KB) was recorded from the serialized file. RAM usage (MB) and CPU utilization (%) were measured using the psutil library. CPU temperature (°C) was recorded with the vcgencmd tool. Finally, F1-score and accuracy were recalculated on the held-out test set to assess any performance decrease after optimization:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

$$\text{F1} - \text{score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}} \quad (4)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (5)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (6)$$

These metrics are defined based on the components of the confusion matrix: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

2) Benchmarking Scenarios

Benchmarks were conducted under two separate scenarios to simulate various edge deployment use cases. Prior to all experiments, the system was stabilized for five minutes to achieve thermal equilibrium. All experiments were conducted consecutively under uniform ambient conditions, with cool-down intervals between tests to avert thermal carryover. Benchmarks were conducted under two scenarios:

- Real-time single-sample inference: This scenario simulates real-time anomaly detection for streaming sensor data. A single prediction was repeated 1,000 times, preceded by a 10-iteration warm-up phase (results discarded), to measure the minimum achievable latency. Key metrics gathered were the mean latency and standard deviation.
- Batch processing. This scenario simulates the periodic bulk analysis of accumulated sensor readings. The full 20,000-sample evaluation set was processed as a single batch to reflect gateway operations with buffered data. Key metrics gathered were total processing time, average time per sample, and average CPU utilization.

3) Statistical Analysis

Statistical significance was examined using independent t-tests for latency distributions and McNemar's test for prediction disagreements. A rigorous significance level of $\alpha = 0.001$ was employed to adjust for multiple comparisons. Effect sizes for latency improvements were determined using Cohen's d.

III. RESULTS AND DISCUSSION

This section provides detailed benchmark results comparing the baseline LightGBM model with three optimized variants: a pruned model, an ONNX INT8 quantized model, and a Treelite compiled model. Comprehensive benchmark results across all optimization variants are presented in Table I.

A. Statistical Significance

Statistical analysis confirmed all findings. For single-sample inference, t-tests showed that latency disparities were highly significant (all $p < 10^{-40}$), with large practical effect sizes for ONNX INT8 (Cohen's $d = 18.4$) and Treelite ($d = 14.2$) versus the native model. In batch processing, latency differences also remained significant ($p < 0.001$), although ONNX and Treelite performance were statistically comparable ($p = 0.089$), highlighting scenario-dependent selection. Critically, McNemar's test confirmed zero prediction discrepancies between any optimized model and the baseline ($x^2 = 0, p = 1.0$), verifying lossless classification outcomes.

TABLE I. PERFORMANCE COMPARISON ACROSS ALL OPTIMIZATIONS

Category	Metric	Native	Pruned	ONNX INT8	Treelite
Model characteristics	Model size (KB)	670.27	18.11 (↓97.3%)	442.68 (↓34.0%)	324.27 (↓51.6%)
Single-sample inference	Latency (ms)	3.70	3.59 (1.03×)	0.091 (40.66×)	0.255 (14.51×)
	Throughput (inf/s)	270	278	10,989	3,922
Batch processing	Latency (ms)	3.70	3.59 (1.03×)	0.288 (12.85×)	0.276 (13.41×)
	Throughput (inf/s)	270	278	3,472	3,623
Resource utilization (batch)	Peak RAM (MB)	146.09	145.63	94.99 (↓35%)	82.71 (↓43%)
	CPU utilization (%)	28.18	27.59	80.67	25.99
	Temperature change (°C)	+2.4	+2.9	+7.3	+2.9
Accuracy metrics	Accuracy (%)	96.98	96.98	96.98	96.98
	Precision (C1)	0.97	0.97	0.97	0.97
	Recall (C1)	0.82	0.82	0.82	0.82
	F1-score	0.89	0.89	0.89	0.89
	ROC-AUC	0.9636	0.9636	0.9099	0.9636
	MCC	0.8779	0.8779	0.8779	0.8779

Note: ↓ indicates reduction; × indicates speedup factor; C1 = anomaly class, MCC = Matthews Correlation Coefficient.

B. *Inference Speed Analysis*

Performance was assessed under two separate operational conditions: real-time single-sample inference and periodic batch processing, as depicted in Figure 2. In the real-time, single-sample context, ONNX INT8 achieved the most substantial performance enhancement. The mean latency reached 0.091 ms, representing a 40.66× acceleration compared to the native Python baseline of 3.70 ms, as determined by the formula $S = L_{baseline}/L_{optimize}$. This corresponds to a throughput of 10,989 inferences per second, facilitating high-frequency stream processing.

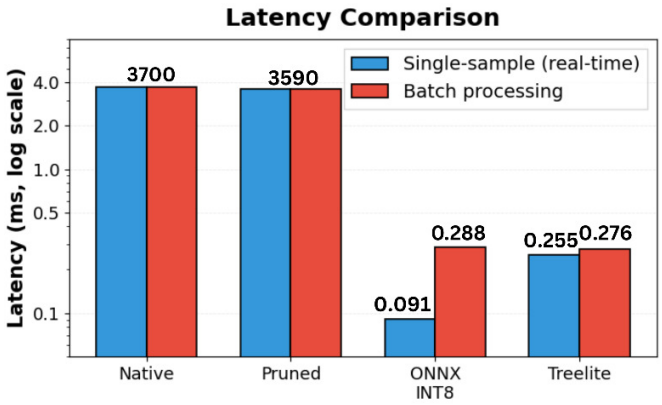

Fig. 2. Inference latency comparison across optimization methods.

The acceleration is attributed to the combination of 8-bit integer arithmetic (via ARM NEON SIMD instructions) and the optimized execution graph of the ONNX Runtime. Treelite also delivered a considerable 14.51× speedup (0.255 ms), mostly by reducing Python interpreter overhead. The Pruned model offered no appreciable speedup (1.03×), demonstrating that inference time is controlled by tree traversal logic rather than model size. In the batch processing scenario, the performance characteristics shifted. Treelite emerged as the most efficient, attaining a 13.41× speedup (0.276 ms/sample), marginally surpassing ONNX INT8 (12.85×). This convergence is attributed to the amortization of fixed execution overheads and, more critically, to thermal throttling observed in the ONNX INT8 model under sustained high-CPU workloads. These results challenge the assumption that a single optimization approach is universally superior, emphasizing that workload characteristics are key determinants of performance.

C. *Accuracy Preservation and Resource Trade-Offs*

A crucial discovery is that all optimization strategies, including INT8 quantization and pruning, were entirely lossless in terms of final classification. All versions created an identical confusion matrix (16,924 TN, 76 FP, 527 FN, 2,473 TP), resulting in the same 96.98% accuracy and 0.89 F1-score. However, a considerable disparity was detected in the resource utilization metrics.

While the native, pruned, and Treelite models all maintained an ROC-AUC of 0.9636, the ONNX INT8 model's score was lowered to 0.9099. This suggests that while the final decision boundary was kept, the INT8 quantization process worsened the calibration of the probability estimates. This is a critical consideration for applications requiring risk scoring, though it is inconsequential for this paper's binary classification goal. Both Treelite and ONNX exhibited greater memory efficiency, lowering the process RAM footprint by 43% (82.71 MB) and 35% (94.99 MB), respectively, compared with the original baseline (146.09 MB). The most significant trade-off was noticed in CPU and thermal performance during batch processing, as depicted in Figure 3. The ONNX model, despite its speed, saturated the CPU at 80.67% utilization, creating a large +7.3°C temperature increase. This tendency is attributable to the aggressive use of SIMD instructions. In sharp contrast, Treelite remained highly efficient, averaging 25.99% CPU utilization with a slight +2.9°C thermal impact, which is practically comparable to the native baseline.

This has immediate consequences for energy-constrained deployments. Based on a 5.1 W idle draw, the ONNX model's high load suggests a ~41% increase in power usage during inference above the baseline. Conversely, Treelite's thermal and CPU efficiency suggest that it is the optimum solution for battery-powered or passively cooled devices requiring sustained operation.

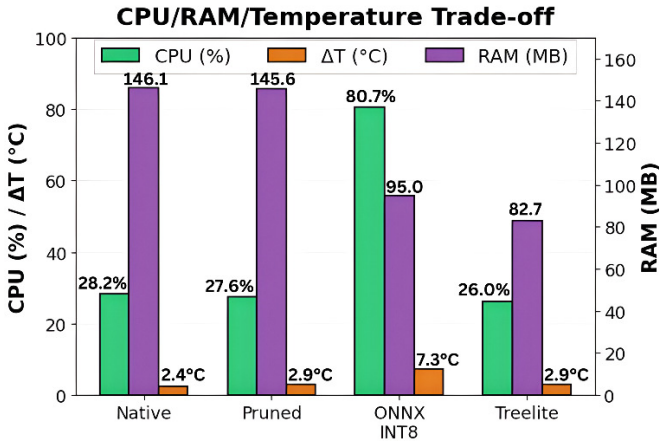

Fig. 3. Resource utilization and thermal impact of optimization methods.

D. Optimal Deployment

Our findings indicate that the optimal strategy is application-specific, as visualized in the multi-dimensional trade-off plot in Figure 4. ONNX INT8 excels for real-time streaming (0.091 ms latency) despite high CPU usage; Treelite achieves an optimal balance for sustained batch processing (13.41× speedup, 25.99% CPU); pruning suits storage-critical deployments (97.3% reduction); native/Treelite maintain calibrated probabilities (ROC-AUC 0.9636) for risk-scoring applications.

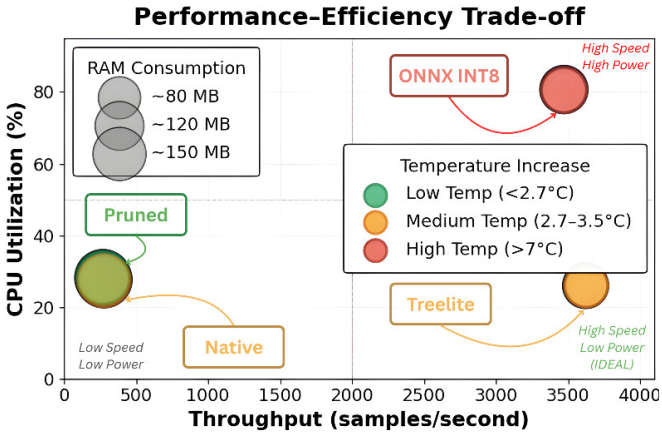

Fig. 4. Performance trade-off summary (batch processing).

E. Limitations

This study is subject to several limitations. The evaluation was confined to a single hardware platform (Raspberry Pi 4). Furthermore, training relied on augmented data (though validation used only real samples), and the tests were short-term, without assessing long-term stability.

IV. CONCLUSION

This paper presented a systematic comparison of pruning, Open Neural Network Exchange (ONNX) 8-bit integer (INT8) quantization, and Treelite compilation for deploying a Light Gradient Boosting Machine (LightGBM)-based electricity anomaly detection model on a Raspberry Pi 4 edge device. Using two realistic deployment scenarios (single-sample streaming inference and periodic batch processing), the experiments showed that ONNX INT8 provided the lowest single-sample latency (0.091 ms, 40.66× speedup), whereas Treelite delivered the best sustained batch efficiency (0.276 ms/sample, 13.41× speedup) with low CPU utilization. Pruning achieved the largest storage reduction (97.3%) but did not materially improve inference latency, indicating that traversal overhead dominates for shallow tree ensembles.

All optimized variants preserved the baseline classification outcomes on the held-out test set (96.98% accuracy and an identical confusion matrix), confirming that these deployment optimizations can be applied without sacrificing detection performance. However, the results also highlight an important deployment trade-off: ONNX INT8 achieved the highest throughput at the cost of substantially higher CPU load and a larger temperature increase during sustained execution, whereas Treelite provided a more thermally stable profile that is better suited to passively cooled or energy-constrained nodes. For applications requiring well-calibrated probability scores (risk scoring), the native and Treelite variants are preferable due to their higher Receiver Operating Characteristic–Area Under the Curve (ROC-AUC).

The main contribution of this work is a comprehensive multi-technique, multi-metric study of LightGBM optimization on commodity Internet of Things (IoT) hardware that includes thermal behavior alongside standard accuracy and latency measurements. Future work will extend the evaluation to additional edge platforms and explore hybrid pipelines (e.g., pruning combined with compilation), as well as long-term stability testing under continuous field operation.

DATA AVAILABILITY

The dataset used in this study was collected from IoT energy monitors installed in institutional and residential sites. Because the measurements are linked to real-world usage contexts and may contain sensitive information, the raw dataset cannot be publicly released. De-identified or aggregated data and the benchmarking scripts are available from the corresponding author upon reasonable request, subject to institutional approval.

ACKNOWLEDGMENT

The authors would like to thank Diponegoro University and Semarang University for providing experimental resources to carry out this research.

REFERENCES

[1] B. Wang *et al.*, "An IoT-Enabled Stochastic Operation Management Framework for Smart Grids," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1025–1034, Jan. 2023, https://doi.org/10.1109/TITS.2022.3183327.

[2] A. Muzumdar, C. Modi, and C. Vyjayanthi, "Designing a blockchain-enabled privacy-preserving energy theft detection system for smart grid neighborhood area network," *Electric Power Systems Research*, vol. 207, June 2022, Art. no. 107884, https://doi.org/10.1016/j.epsr.2022.107884.

[3] D. Utomo and P.-A. Hsiung, "A Multitiered Solution for Anomaly Detection in Edge Computing for Smart Meters," *Sensors*, vol. 20, no. 18, Sept. 2020, Art. no. 5159, https://doi.org/10.3390/s20185159.

[4] M. J. C. S. Reis and C. Serôdio, "Edge AI for Real-Time Anomaly Detection in Smart Homes," *Future Internet*, vol. 17, no. 4, Apr. 2025, Art. no. 179, https://doi.org/10.3390/fi17040179.

[5] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1937–1967, Mar. 2021, https://doi.org/10.1007/s10462-020-09896-5.

[6] K.-T. Nguyen, T.-N. Tran, and H.-T. Nguyen, "Research on the Influence of Hyperparameters on the LightGBM Model in Load Forecasting," *Engineering, Technology & Applied Science Research*, vol. 14, no. 5, pp. 17005–17010, Oct. 2024, https://doi.org/10.48084/etasr.8266.

[7] K. M. Hosny, A. Magdi, A. Salah, O. El-Komy, and N. A. Lashin, "Internet of things applications using Raspberry-Pi: a survey," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 1, pp. 902–910, Feb. 2023, https://doi.org/10.11591/ijece.v13i1.pp902-910.

[8] H. Guo, Z. Zhou, D. Zhao, and W. Gaaloul, "EGNN: Energy-efficient anomaly detection for IoT multivariate time series data using graph neural network," *Future Generation Computer Systems*, vol. 151, pp. 45–56, Feb. 2024, https://doi.org/10.1016/j.future.2023.09.028.

[9] S. Hadi, W. Amien Syafei, and A. Wibowo, "Ambient Intelligence to Detect Misuse of Electricity Consumption Based on IoT Using Blockchain Technology," *IEEE Access*, vol. 13, pp. 80371–80394, 2025, https://doi.org/10.1109/ACCESS.2025.3561677.

[10] S. Y. Appiah, E. K. Akowuah, V. C. Ikpo, and A. Dede, "Extremely randomised trees machine learning model for electricity theft detection," *Machine Learning with Applications*, vol. 12, June 2023, Art. no. 100458, https://doi.org/10.1016/j.mlwa.2023.100458.

[11] S. Ronaghi, A. Ferrero, S. Salicone, and H. V. Jetti, "Data-Driven Anomaly Detection for Smart Energy Meters," in *2024 IEEE 14th International Workshop on Applied Measurements for Power Systems*, Caserta, Italy, 2024, pp. 1–6, https://doi.org/10.1109/AMPS62611.2024.10706664.

[12] G. Ke *et al.*, "LightGBM: a highly efficient gradient boosting decision tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 3149–3157.

[13] M. A. Mohsin, S. N. Shahrouzi, and D. G. Perera, "Composing Efficient Computational Models for Real-Time Processing on Next-Generation Edge-Computing Platforms," *IEEE Access*, vol. 12, pp. 24905–24932, 2024, https://doi.org/10.1109/ACCESS.2024.3365652.

[14] J. Jin, X. Qiu, and C. Lu, "Edge Artificial Intelligence for Electrical Anomaly Detection Based on Process-In-Memory Chip," *Electronics*, vol. 13, no. 21, Oct. 2024, Art. no. 4255, https://doi.org/10.3390/electronics13214255.

[15] R. D. Rachmanto, Z. Sukma, A. N. L. Nabhaan, A. Setyanto, T. Jiang, and I. K. Kim, "Characterizing Deep Learning Model Compression with Post-Training Quantization on Accelerated Edge Devices," in *2024 IEEE International Conference on Edge Computing and Communications*, Shenzhen, China, 2024, pp. 110–120, https://doi.org/10.1109/EDGE62653.2024.00023.

[16] J. Lee, M. Yu, Y. Kwon, and T. Kim, "Quantune: Post-training quantization of convolutional neural networks using extreme gradient boosting for fast deployment," *Future Generation Computer Systems*, vol. 132, pp. 124–135, July 2022, https://doi.org/10.1016/j.future.2022.02.005.

[17] F. Alkhoury, S. Buschjäger, and P. Welke, "Splitting stump forests: tree ensemble compression for edge devices (extended version)," *Machine Learning*, vol. 114, no. 10, Aug. 2025, Art. no. 219, https://doi.org/10.1007/s10994-025-06866-2.

[18] A. P. Taruna, G. Arisona, D. Irwanto, A. B. Bestari, and W. Juniawan, "Electricity Theft Detection Using Machine Learning in Traditional Meter Postpaid Residential Customers: A Case Study on State Electricity Company (PLN) Indonesia," *IEEE Access*, vol. 13, pp. 7167–7191, 2025, https://doi.org/10.1109/ACCESS.2025.3526764.

[19] *Buku 2: Standar Konstruksi Sambungan Tenaga Listrik*, Keputusan Direksi PT PLN (Persero) No. 474.K/DIR/2010, PT PLN (Persero), Jakarta, Indonesia, 2010.

[20] R. Yao, N. Wang, Z. Liu, P. Chen, D. Ma, and X. Sheng, "Intrusion detection system in the Smart Distribution Network: A feature engineering based AE-LightGBM approach," *Energy Reports*, vol. 7, pp. 353–361, Nov. 2021, https://doi.org/10.1016/j.egyr.2021.10.024.

[21] "Treelite : model compiler for decision tree ensembles." Treelite. https://treelite.readthedocs.io/en/2.4.0/.