

Explainable XGBoost and LSTM Models for Cyberattack Detection in Resource-Constrained IoT Devices

Haryoko, Lilis Dwi Farida, Eli Pujastuti, Irma Rofni Wulandari, Muhammad Koprawi and Afrig Aminudin

Abstract— Internet of Things (IoT) devices are frequently deployed with limited processing capacity, restricted memory, and limited security visibility, which increases their exposure to botnet compromise. This study evaluates explainable XGBoost and LSTM models for binary cyberattack detection using public N-BaIoT traffic data. The proposed workflow starts with public CSV collection, deterministic source-balanced sampling, cleaning of missing and infinite values, device-grouped validation, model training, metric evaluation, SHAP-based explanation analysis, feature ablation, and local CPU resource measurement. A reproducible subset of 101,000 cleaned rows was constructed from 89 CSV sources across nine commercial IoT devices. The experiment used 5-fold StratifiedGroupKFold with IoT device identity as the grouping variable to reduce train-test device leakage. XGBoost achieved $99.52 \pm 0.71\%$ accuracy, $99.58 \pm 0.61\%$ F1-score, and 0.9995 ± 0.0011 ROC-AUC. LSTM achieved $99.46 \pm 1.10\%$ accuracy, $99.54 \pm 0.93\%$ F1-score, and 0.9999 ± 0.0002 ROC-AUC. Histogram Gradient Boosting achieved the highest internal-baseline F1-score ($99.82 \pm 0.29\%$), confirming the strength of tree-based tabular learners on N-BaIoT. XGBoost remains the main explainable model because it provides direct TreeSHAP support, compact training cost, and competitive cross-device performance, whereas LSTM is retained as a sequential comparator for short source-local windows. SHAP stability analysis showed partially stable feature attribution, with a mean top-10 Jaccard similarity of 0.547. Resource results are interpreted as local CPU proxy measurements rather than embedded-hardware deployment validation.

Index Terms— Cyberattack detection, explainable artificial intelligence, Internet of Things, LSTM, resource-constrained devices, SHAP, XGBoost.

I. INTRODUCTION

IoT deployments expand the attack surface of residential, industrial, and smart infrastructure networks. Many devices operate with limited memory, processing capability, and

runtime monitoring, which increases their exposure to botnets such as Mirai and BASHLITE as represented in the N-BaIoT benchmark [1]. An intrusion detector for this setting should therefore satisfy three requirements: strong detection performance, low resource overhead, and interpretable decision evidence.

Recent literature can be grouped into three relevant directions. First, XAI-oriented studies use SHAP and related attribution methods to interpret IoT and IIoT intrusion detectors [2], [3], [4], [5], [6], [7], [8], [9], [10]. Second, edge-oriented studies examine lightweight neural or tree-based architectures for constrained devices [11], [12], [13], [14]. Third, dataset and comparative studies show that reported intrusion-detection performance is highly sensitive to dataset choice, preprocessing, feature selection, and validation design [15], [16], [17], [18], [19], [20], [21]. In this paper, a memory proxy refers to Python-level peak allocated memory observed during inference with `tracemalloc`; it is useful for comparing model implementations under a controlled environment, but it is not equivalent to physical RAM use on an embedded board.

The research gap addressed in this paper is not the absence of accurate IoT intrusion detectors, but the limited number of compact studies that jointly report cross-device validation, resource-aware inference units, explanation stability, feature ablation, additional baselines, and cautious deployment boundaries. Random row-level splitting can inflate performance when traffic from the same device appears in both training and testing partitions. Moreover, published results are often difficult to compare directly because studies use different subsets, class definitions, preprocessing pipelines, and evaluation protocols.

XGBoost and LSTM are compared because they represent two different modeling assumptions. XGBoost is a tree-ensemble model suitable for tabular traffic statistics and has

¹Manuscript received July 25, 2026.

H. Author is with the Department of Information Technology, Faculty of Computer Science, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia; (corresponding author to provide e-mail: haryoko@amikom.ac.id).

L. D. F. Author was with the Department of Informatics Management, Faculty of Computer Science, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia; (e-mail: rida@amikom.ac.id).

E. P. Author is with the Department of Informatics, Faculty of Computer Science, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia; (e-mail: eli@amikom.ac.id).

I. R. W. Author is with the Department of Information Systems, Faculty of Computer Science, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia; (e-mail: rofni@amikom.ac.id).

M. K. Author is with the Department of Computer Engineering, Faculty of Computer Science, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia; (e-mail: koprawi@amikom.ac.id).

A. A. Author is with the Department of Information Systems, Faculty of Computer Science, Universitas Amikom Yogyakarta, Sleman, 55283, Indonesia; (e-mail: afrig@amikom.ac.id).

mature TreeSHAP support for feature attribution [22], [23]. LSTM is a recurrent neural model that can evaluate short source-local sequences when ordering information may be useful [24], [25]. Logistic Regression, Random Forest, Histogram Gradient Boosting, and Multilayer Perceptron are included as internal baselines to contextualize the selected models rather than to claim direct state-of-the-art superiority.

The novelty of this work lies in a reproducible evaluation framework for explainable and resource-aware IoT intrusion detection using N-BaIoT: device-grouped validation, explicit sampling evidence, fold-level class-distribution reporting, SHAP stability analysis, top-k and removal ablation, FPR/FNR error profiling, and local CPU resource benchmarking. The findings show that XGBoost and LSTM provide statistically comparable F1-scores under five held-out-device folds, while Histogram Gradient Boosting is the strongest internal baseline and XGBoost remains the most directly explainable tree-based focus model.

II. RELATED WORK

This section reviews prior studies on IoT intrusion datasets, machine-learning and deep-learning detection models, explainability, and resource-aware evaluation. The discussion identifies the methodological limitations addressed by the present cross-device comparison of XGBoost and LSTM.

A. IoT Intrusion Datasets and Validation Limitations

N-BaIoT was introduced for network-based detection of IoT botnet attacks using traffic from nine commercial IoT devices infected by Mirai and BASHLITE [1]. Its extracted statistical features are related to the Kitsune framework, which summarizes packet streams for online anomaly detection [26]. The dataset is valuable because it contains device-specific IoT botnet behavior, but it is also highly imbalanced in its local full CSV form, with attack rows substantially outnumbering benign rows. This imbalance reinforces the need to report class distributions and error profiles rather than relying on accuracy alone.

Broader IDS surveys and dataset studies show that intrusion-detection results are strongly influenced by feature extraction, class composition, and split strategy [16], [17], [18], [19], [20], [21]. A limitation in many prior evaluations is that random splits may share device- or source-specific patterns across train and test sets. The present work addresses this issue by grouping folds by device identity.

B. Machine Learning and Deep Learning Models for IoT IDS

Classical and ensemble machine-learning models remain competitive in tabular network-intrusion detection because they can exploit engineered traffic statistics with modest training and inference cost [15], [16], [20], [27], [28]. Logistic Regression provides a linear baseline, Random Forest provides bagged tree ensembles, Histogram Gradient Boosting provides efficient histogram-based boosting, and XGBoost provides regularized gradient-boosted trees with strong tabular performance [22].

Deep-learning IDS models, including recurrent neural networks and LSTM variants, are commonly used when

sequential patterns or temporal dependencies are expected [24], [25], [29], [30]. Prior comparative studies indicate that tree-based and neural models can both achieve high IDS performance, but their practical value depends on validation design, resource cost, and interpretability [14], [19], [20]. This motivates a side-by-side comparison between XGBoost and LSTM under the same cross-device folds.

C. Explainability and Resource-Aware Evaluation

XAI has become important in IoT security because high detection accuracy alone may not provide sufficient evidence for operator trust, debugging, or feature-level interpretation [3], [4], [5], [6], [7], [8], [9], [10]. SHAP-based analysis is particularly relevant for tree-based models because TreeSHAP can provide local and global feature-attribution evidence for fitted tree ensembles [23]. However, attribution stability across folds is not guaranteed; therefore, this study evaluates whether top-ranked features remain similar across held-out-device validation folds.

Resource-aware evaluation is equally important because IoT and edge deployments may impose constraints on memory, latency, model size, and update cost [11], [12], [13], [14]. Existing studies do not always define the inference unit precisely, especially when comparing row-based models with sequence-based models. This paper addresses that limitation by reporting both batch inference and batch-size-1 measurements and by distinguishing a single row-level XGBoost decision from a single LSTM sequence decision.

III. METHOD

This section describes the dataset preparation, grouped validation design, model configuration, evaluation metrics, SHAP analysis, feature ablation, and resource measurement protocol used in the revised experiment.

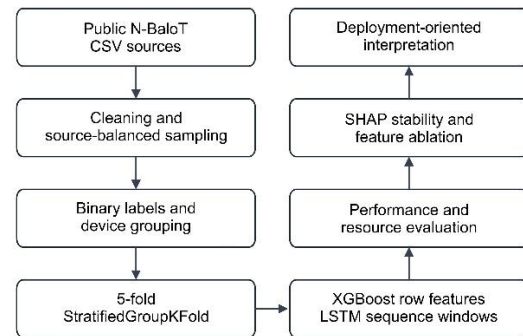


Fig. 1. Experimental workflow for cross-device explainable IoT intrusion detection.

Fig. 1 summarizes the workflow from public data acquisition to deployment-oriented interpretation. The figure is aligned with the methodological subsections: dataset sampling, cross-device validation, model configuration, metric formulation, SHAP and ablation analysis, and resource measurement.

A. Dataset and Sampling Strategy

The experiment used public N-BaIoT CSV files corresponding to the dataset described by Meidan et al. [1]. Each source file contains 115 independent numeric traffic

features. Binary labels were derived from the source path, with benign traffic labeled as 0 and Mirai/BASHLITE or Gafgyt attack traffic labeled as 1.

A deterministic source-balanced subset was used for computational feasibility and source coverage. Up to 5,000 benign rows were loaded per device, and up to 700 rows were loaded from each attack CSV source. Rows were selected from the beginning of each source file to preserve the original source-local ordering required by the LSTM windowing procedure and to make the sampling exactly reproducible. This choice may not represent all later rows in a file; therefore, the experiment is reported as a reproducible source-balanced benchmark rather than a full-dataset prevalence-preserving evaluation.

Class balancing was not performed after sampling. The revised subset contains 45,000 benign rows and 56,000 attack rows, which is moderately imbalanced but far less attack-dominant than the full local CSV collection. Unweighted binary cross-entropy and logistic objectives were used; consequently, precision, recall, F1-score, FPR, and FNR are reported to expose possible class-asymmetric behavior.

TABLE I
FULL LOCAL N-BaIoT ROWS AND REVISED SOURCE-BALANCED SUBSET

LABEL	FULL LOCAL ROWS	FULL (%)	REVISED SUBSET ROWS	SUBSET (%)
Attack	6,506,674	92.13%	56,000	55.45%
Benign	555,932	7.87%	45,000	44.55%
Total	7,062,606	100.00%	101,000	100.00%

Table I shows that the local full CSV collection is dominated by attack traffic, whereas the revised subset intentionally reduces this dominance. The subset therefore controls source representation, but it should not be interpreted as preserving the natural full-dataset attack prevalence.

B. Cross-Device Validation Design

The experiment used 5-fold StratifiedGroupKFold with IoT device identity as the grouping variable. In each fold, complete devices were held out for testing, and one remaining device was used for validation. This setup is stricter than random row splitting because samples from the same device do not appear simultaneously in training and testing partitions.

For XGBoost and the additional baselines, each inference unit is one 115-feature row. For LSTM, sequences of length 10 were generated within the same source file, and each sequence produces one label for the last row in the window. The ordering is therefore source-local ordering of extracted traffic-statistic rows, not a reconstruction of raw packet timestamps. LSTM results should be interpreted as a short-window sequential comparator under this ordering assumption.

TABLE II
DEVICE-LEVEL FOLD ASSIGNMENT AND TEST-CLASS DISTRIBUTION

FOLD	VALIDATION DEVICE	HELD-OUT TEST DEVICE(S)	TEST BENIGN	TEST ATTACK	TEST ATTACK (%)
1	Danmini_Doorbell	Provision_PT_737E_Security_Camera; SimpleHome_XCS7_1002_WHT_Security_Camera	10,000	14,000	58.33%
2	Ennio_Doorbell	Ecobee_Thermostat; Samsung_SNH_1011	10,000	10,500	51.22%

		_N_Webcam			
3	Ennio_Doorbell	Provision_PT_838_Security_Camera	5,000	7,000	58.33%
4	Provision_PT_838_Security_Camera	Danmini_Doorbell; Ennio_Doorbell	10,000	10,500	51.22%
5	Provision_PT_838_Security_Camera	Philips_B120N10_Baby_Monitor; SimpleHome_XCS7_1003_WHT_Security_Camera	10,000	14,000	58.33%

Table II reports held-out devices and test-class ratios for each fold. The test attack proportion ranges from 51.22% to 58.33%, indicating moderate fold-level variation after source-balanced sampling.

C. Model Configuration and Hyperparameter Selection

Hyperparameter configurations were determined through preliminary validation experiments and computational constraints rather than an exhaustive search. A small validation search was used for XGBoost, with F1-score as the selection criterion, while the compact LSTM architecture was selected by considering validation loss, validation F1-score, and training cost. The resulting configurations were fixed before the final cross-device evaluation and were applied consistently across all folds. Within each fold, model development used only the training and validation devices; the held-out test devices remained excluded from preprocessing, model fitting, and configuration selection until final evaluation.

TABLE III
MODEL HYPERPARAMETERS AND TRAINING CONFIGURATION

MODEL	INFERENCE INPUT	KEY CONFIGURATION
XGBoost	115-feature row	objective=binary:logistic, eval_metric=logloss, tree_method=hist, n_estimators=160, max_depth=4, learning_rate=0.10, subsample=0.90, colsample_bytree=0.90, n_jobs=4.
LSTM	10-row source-local sequence	Input size 115, hidden size 32, dropout 0.20, fully connected hidden units 16, Adam optimizer, learning rate 0.001, BCEWithLogitsLoss, 4 epochs, batch size 512.
Logistic Regression	115-feature row	StandardScaler, solver=lbfgs, max_iter=300, random seed 42.
Random Forest	115-feature row	n_estimators=120, max_features=sqrt, min_samples_leaf=1, n_jobs=4, random seed 42.
Histogram Gradient Boosting	115-feature row	max_iter=160, learning_rate=0.08, max_leaf_nodes=31, random seed 42.
Multilayer Perceptron	115-feature row	StandardScaler, one hidden layer with 64 units, ReLU, Adam, learning rate 0.001, batch size 512, max_iter=40, early stopping with n_iter_no_change=5.

Table III summarizes the fixed configurations used for the two principal models and four comparative baselines. XGBoost employed shallow trees and stochastic row and feature sampling to limit model complexity, whereas the LSTM used a compact recurrent architecture to capture dependencies within each 10-row source-local sequence. Logistic Regression, Random Forest, Histogram Gradient Boosting, and the Multilayer Perceptron represent linear, bagging-based, boosting-based, and neural-network baselines, respectively. Feature standardization was applied only to models whose optimization is sensitive to feature scale, and all preprocessing parameters were estimated from the corresponding training partition. All models were evaluated using the same device-grouped folds. However, XGBoost and the conventional baselines classify individual feature rows, whereas the LSTM classifies 10-row sequences. Therefore, their latency results

should be interpreted according to the inference units defined in Section III.F.

D. Evaluation Metrics and Mathematical Formulation

XGBoost minimizes a regularized additive objective, where f_t denotes the tree added at boosting iteration t and $\Omega(\cdot)$ penalizes model complexity:

$$\mathcal{L}^{(t)} = \sum_i l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t), \quad (1)$$

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2$$

where x_i and y_i denote the feature vector and binary class label of instance i , respectively; $\hat{y}_i^{(t-1)}$ is the prediction obtained before iteration t ; f_t is the tree added at iteration t ; and $l(\cdot)$ is the differentiable loss function. The regularization term $\Omega(f)$ controls model complexity, where T is the number of leaves, w is the vector of leaf weights, and γ and λ are regularization coefficients.

The LSTM classifier was optimized using binary cross-entropy over the predicted attack probability p_i :

$$\text{BCE} = -\frac{1}{N} \sum_i [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \quad (2)$$

where N is the number of evaluated instances, $y_i \in \{0,1\}$ is the ground-truth label, and $p_i \in [0,1]$ is the predicted probability that instance i belongs to the attack class.

Threshold-dependent metrics were computed from true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In this study, TP represents attack instances correctly classified as attacks, TN represents benign instances correctly classified as benign, FP represents benign instances incorrectly classified as attacks, and FN represents attack instances incorrectly classified as benign:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (3)$$

$$\text{Recall} = \frac{TP}{TP + FN},$$

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

$$\text{FPR} = \frac{FP}{FP + TN}, \quad (4)$$

$$\text{FNR} = \frac{FN}{FN + TP}$$

where Accuracy denotes the proportion of all correctly classified instances, Precision denotes the proportion of predicted attacks that are actual attacks, Recall denotes the proportion of actual attacks correctly detected, and F1 is the harmonic mean of Precision and Recall. FPR denotes the proportion of benign instances incorrectly classified as attacks, whereas FNR denotes the proportion of attack instances incorrectly classified as benign.

FPR and FNR were included because false alarms and missed attacks have different operational implications. High FPR can overload administrators with benign events flagged as attacks, whereas high FNR leaves malicious traffic undetected.

E. SHAP Stability and Feature Ablation

TreeSHAP was applied to XGBoost validation-device samples in each fold. The additive SHAP explanation model and fold-pair stability score are:

$$g(z') = \phi_0 + \sum_j \phi_j z'_j, \quad J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (5)$$

where ϕ_j is the attribution of feature j and $J(A, B)$ is the Jaccard similarity between two top-10 SHAP feature sets. The ablation study compared the full 115-feature model, SHAP top-5, top-10, top-20 subsets, and a model excluding the top-10 SHAP features. This design tests whether highly ranked features are compactly predictive and whether correlated N-BaIoT statistics can compensate when top-ranked features are removed.

F. Resource Measurement Protocol

Resource measurements were obtained on a Windows 11 local CPU environment with 8 logical CPU threads and approximately 31.95 GB RAM. Batch inference latency was measured after warm-up prediction using `time.perf_counter`; serialized model size was measured from saved artifacts; and Python-level peak memory proxy was measured using `tracemalloc`. The batch-size-1 benchmark used 1,000 sampled test instances per fold, 50 warm-up calls, and `time.perf_counter_ns`. These measurements are local CPU proxy indicators and do not constitute Raspberry Pi, microcontroller, or embedded IoT hardware deployment results.

IV. RESULTS AND DISCUSSION

This section presents the cross-device detection results, error profiles, resource measurements, SHAP stability, feature-ablation findings, and comparisons with additional baseline models. The results are interpreted with respect to predictive performance, explainability, computational requirements, and the validity boundaries of the experimental design.

A. Cross-Device Detection Performance and Error Profile

Table IV shows that XGBoost and LSTM achieved comparable cross-device detection performance. XGBoost reached $99.58 \pm 0.61\%$ F1-score, while LSTM reached $99.54 \pm 0.93\%$ F1-score. The fold-wise paired comparison is exploratory because only five fold-level observations are available. The F1-score difference between XGBoost and LSTM was small (mean difference 0.00043), and neither the paired t-test ($t(4)=0.093$, $p=0.930$) nor the Wilcoxon signed-rank test ($W=7.0$, $p=1.000$) indicated a clear fold-level advantage.

TABLE IV
FIVE-FOLD CROSS-DEVICE DETECTION PERFORMANCE

MODEL	ACCURACY (%)	PRECISION (%)	RECALL (%)	F1-SCORE (%)	F1 95% CI (%)	ROC-AUC
XGBoost	99.52 ± 0.71	99.83 ± 0.29	99.35 ± 1.22	99.58 ± 0.61	98.82-100.00	0.9995 ± 0.0011
LSTM	99.46 ± 1.10	99.14 ± 1.81	99.96 ± 0.04	99.54 ± 0.93	98.39-100.00	0.9999 ± 0.0002

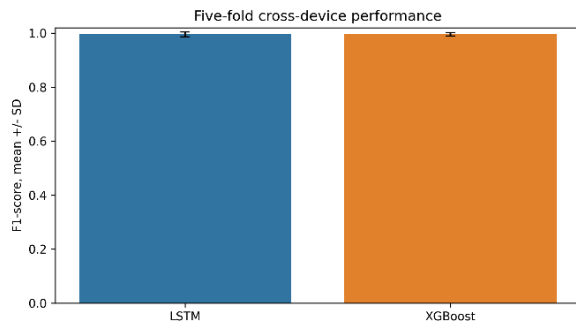


Fig. 2. Five-fold cross-device F1-score comparison with standard deviation

Fig. 2 shows that the performance difference between XGBoost and LSTM is small relative to fold-level variation.

TABLE V
COMBINED HELD-OUT-DEVICE ERROR PROFILE

MODEL	TN	FP	FN	TP	FPR (%)	FNR (%)	OPERATIONAL INTERPRETATION
XGBoost	44,890	110	458	55,542	0.24%	0.82%	Lower false-alarm rate, with more missed attacks than LSTM.
LSTM	44,307	612	23	55,257	1.36%	0.04%	Lower missed-attack rate, with more benign traffic flagged as attack.

Table V clarifies the operational trade-off between the two models. XGBoost produced fewer false positives, which is preferable when false alarms create operational burden. LSTM produced fewer false negatives, which is preferable when missed attacks are more costly than additional alert review.

B. Resource-Aware Evaluation

Table VI distinguishes the inference unit for each model. XGBoost reports latency per row, whereas LSTM reports latency per 10-row sequence that generates one decision for the last row. In batch mode, XGBoost was faster because tree traversal and vectorized prediction were efficiently amortized across many rows. In batch-size-1 mode, LSTM showed lower median and p95 latency than XGBoost in the local CPU benchmark. This reversal is interpreted as an implementation- and call-overhead effect: single-row XGBoost prediction incurs per-call wrapper and tree-traversal overhead, while the compact LSTM state dictionary executes a small recurrent computation for a single sequence. The result does not imply embedded-hardware superiority for LSTM.

TABLE VI
RESOURCE-AWARE COMPARISON WITH EXPLICIT INFERENCE UNITS

MODE L	INFERE NCE UNIT	BATC H LATEN CY	NORMALI ZED INPUT-ROW LATENCY	MEDI AN BATCH -SIZE-1 LATEN CY	P95 BATCH -SIZE-1 LATEN CY	MOD EL SIZE (MB)	PEAK MEMO RY PROX Y (MB)	TRA IN TIM E (S)
XGBo ost	One 115-feature row	0.0042 ± 0.0016 ms/unit	0.0042 ms/input row	1.1345 ± 0.0653 ms	5.9105 ± 0.6584 ms	0.171 ± 0.003	0.313 ± 0.074	19.4 ± 4.34
LSTM	One 10-row sequence; one label for the last row	0.1730 ± 0.0587 ms/unit	0.0173 ms/input row	0.8380 ± 0.0840 ms	2.0802 ± 0.2895 ms	0.077 ± 0.000	0.229 ± 0.020	144.71 ± 42.18

C. SHAP Stability and Feature-Ablation Results

The mean pairwise Jaccard similarity of the top-10 SHAP feature sets was 0.547. This value indicates moderate, not complete, explanation stability across held-out devices. Four features appeared in the top-10 list in all five folds, suggesting repeatedly high-ranked attribution patterns, but correlated N-BaIoT statistics can alter the exact ranking across devices.

TABLE VII
XGBOOST SHAP STABILITY ACROSS FIVE CROSS-DEVICE FOLDS

FEATURE	TOP-10 APPEARANCES	MEAN RANK	MEAN SHAP
HH_jit_L0.1_mean	5	1.00	4.373
HH_jit_L5_mean	5	2.00	1.462
MI_dir_L0.01_weight	5	4.40	0.718
MI_dir_L0.1_weight	5	5.60	0.727
H_L0.1_weight	4	3.75	0.802
MI_dir_L0.01_variance	4	7.25	0.503
HH_jit_L0.01_mean	4	8.50	0.410
H_L0.01_variance	3	6.67	0.491

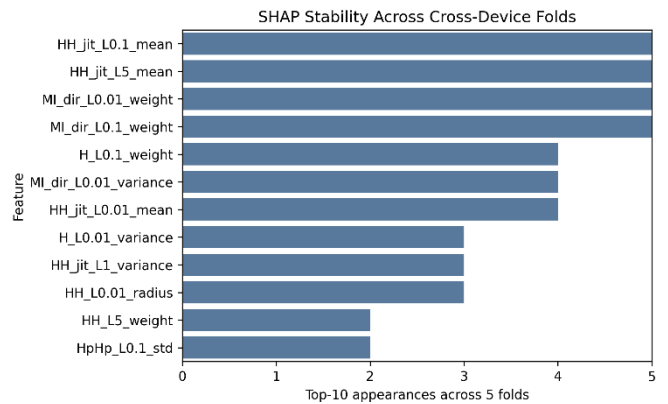


Fig. 3. Frequency of SHAP top-10 feature appearances across the five cross-device folds.

Fig. 3 visualizes the repeated appearance of high-ranked XGBoost features across folds and supports a cautious interpretation of partial explanation stability.

TABLE VIII
EXTENDED XGBOOST SHAP TOP-K FEATURE ABLATION

VARIANT	FEATURES	ACCURACY (%)	F1-SCORE (%)	ROC-AUC	TRAIN TIME (S)
Full 115 features	115	99.52 ± 0.71	99.58 ± 0.61	0.9995 ± 0.0011	19.46 ± 4.34
SHAP top-5 only	5	98.36 ± 1.75	98.57 ± 1.48	0.9990 ± 0.0013	1.45 ± 0.24
SHAP top-10 only	10	98.81 ± 2.12	98.99 ± 1.79	0.9992 ± 0.0015	5.34 ± 3.50
SHAP top-20 only	20	99.47 ± 0.74	99.54 ± 0.64	0.9995 ± 0.0010	2.23 ± 0.37
Without SHAP top-10	105	99.89 ± 0.20	99.90 ± 0.17	1.0000 ± 0.0000	10.59 ± 4.37

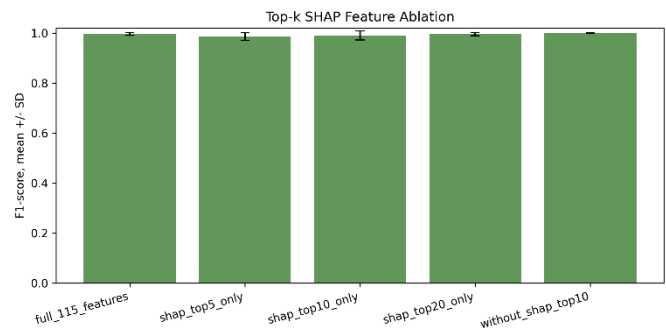


Fig. 4. Extended top-k SHAP feature ablation under cross-device validation.

The top-20 SHAP subset retained near-full performance with $99.54 \pm 0.64\%$ F1-score. In contrast, removing the top-10

SHAP features did not reduce performance and reached $99.90 \pm 0.17\%$ F1-score, indicating substantial redundancy among related N-BaIoT traffic statistics. Thus, SHAP values are interpreted as model-level attribution evidence rather than causal proof of attack generation mechanisms.

D. Additional Baseline Comparison

Table IX contextualizes the main XGBoost-LSTM comparison against additional internal baselines evaluated under the same cross-device folds. Histogram Gradient Boosting achieved the highest mean F1-score, which is plausible because histogram-based boosting is highly effective for structured tabular features and can exploit threshold-like splits in the N-BaIoT statistical feature space. Nevertheless, XGBoost remains the primary explainable model in this paper because it combines competitive performance, mature TreeSHAP support, and established deployment tooling. The baseline results therefore strengthen the conclusion that tree-based models are especially competitive for N-BaIoT, rather than supporting a claim that XGBoost is the single best classifier.

TABLE IX
ADDITIONAL INTERNAL BASELINE BENCHMARK UNDER CROSS-DEVICE
VALIDATION

MODEL	ACCURACY (%)	F1-SCORE (%)	ROC-AUC	TRAIN TIME (S)
Logistic Regression	98.97 ± 1.52	99.12 ± 1.26	0.9924 ± 0.0160	2.85 ± 0.93
Random Forest	99.63 ± 0.80	99.69 ± 0.68	1.0000 ± 0.0000	10.79 ± 1.55
Histogram Gradient Boosting	99.79 ± 0.34	99.82 ± 0.29	1.0000 ± 0.0000	14.26 ± 1.71
Multilayer Perceptron	98.82 ± 1.27	99.00 ± 1.06	0.9938 ± 0.0104	10.41 ± 2.60
XGBoost	99.52 ± 0.71	99.58 ± 0.61	0.9995 ± 0.0011	19.46 ± 4.34
LSTM	99.46 ± 1.10	99.54 ± 0.93	0.9999 ± 0.0002	144.71 ± 42.18

E. Limitations and Validity Boundaries

Several validity boundaries remain. First, the experiment uses a deterministic source-balanced subset rather than the full N-BaIoT dataset; this improves computational feasibility and source coverage but does not preserve full-dataset attack prevalence. Second, the LSTM sequence design uses source-local row ordering of extracted statistical features and should not be interpreted as raw packet-level temporal modeling. Third, statistical testing with five folds is exploratory, so confidence intervals and fold-level variability are reported alongside p-values. Fourth, resource measurements are local CPU proxy measurements and do not replace evaluation on Raspberry Pi-class hardware, microcontrollers, or production edge gateways. Fifth, N-BaIoT mainly represents Mirai and BASHLITE/Gafgyt traffic; therefore, the results may not generalize to emerging botnets, encrypted IoT traffic, adversarial evasion, or live local network conditions.

V. CONCLUSION

This paper presented a revised explainable and resource-aware evaluation of XGBoost and LSTM models for IoT cyberattack detection using public N-BaIoT data. Under 5-fold held-out-device validation, XGBoost achieved $99.52 \pm 0.71\%$

accuracy, $99.58 \pm 0.61\%$ F1-score, and 0.9995 ± 0.0011 ROC-AUC, while LSTM achieved $99.46 \pm 1.10\%$ accuracy, $99.54 \pm 0.93\%$ F1-score, and 0.9999 ± 0.0002 ROC-AUC. XGBoost produced a lower combined FPR (0.24%) than LSTM (1.36%), whereas LSTM produced a lower FNR (0.04%) than XGBoost (0.82%). Histogram Gradient Boosting achieved the highest internal-baseline F1-score ($99.82 \pm 0.29\%$), confirming the strength of tree-based learners on N-BaIoT tabular features. SHAP analysis showed moderate attribution stability, and top-k ablation showed that a top-20 SHAP subset retained near-full performance. The results support XGBoost as a strong explainable and throughput-oriented detector and LSTM as a competitive short-window sequential comparator. Future work should evaluate full-dataset training, multiclass labels, live local IoT traffic, repeated group splits, and real edge hardware deployment.

DATA AVAILABILITY

The experiment used the public N-BaIoT dataset described by Meidan et al. [1]. The analysis scripts, generated summaries, figures, trained fold models, and reproducibility metadata are available from the corresponding author upon reasonable request. No private or human-subject data were used.

ACKNOWLEDGMENT

The authors acknowledge the UCI Machine Learning Repository and the creators of the N-BaIoT dataset for making the IoT botnet traffic data publicly available for research.

REFERENCES

- [1] Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul. 2018, doi: 10.1109/MPRV.2018.03367731.
- [2] Y. Hu, K. Xiao, L. Luo, and L. Chen, "An XGBoost-Based Intrusion Detection Framework with Interpretability Analysis for IoT Networks," *Appl. Sci.*, vol. 16, no. 2, p. 980, Jan. 2026, doi: 10.3390/app16020980.
- [3] S. Bin hulayyil, S. Li, and N. Saxena, "Explainable AI-based intrusion detection in IoT systems," *Internet of Things*, vol. 31, p. 101589, May 2025, doi: 10.1016/j.iot.2025.101589.
- [4] Y. Wang, M. A. Azad, M. Zafar, and A. Gul, "Enhancing AI transparency in IoT intrusion detection using explainable AI techniques," *Internet of Things*, vol. 33, p. 101714, Sep. 2025, doi: 10.1016/j.iot.2025.101714.
- [5] F. O. Albasheer and M. Agarwal, "OptiXID: An optimized and eXplainable AI framework for intrusion detection in IoT networks," *Cluster Comput.*, vol. 28, no. 16, p. 1060, Dec. 2025, doi: 10.1007/s10586-025-05777-0.
- [6] M. A. Hossain, S. Saif, and M. S. Islam, "A novel federated learning approach for IoT botnet intrusion detection using SHAP-based knowledge distillation," *Complex Intell. Syst.*, vol. 11, no. 10, p. 422, Oct. 2025, doi: 10.1007/s40747-025-02001-9.
- [7] K. Rajkumar and S. M. Shalinie, "SHAP-based intrusion detection in IoT networks using quantum neural networks on IonQ hardware," *J. Parallel Distrib. Comput.*, vol. 204, p. 105133, Oct. 2025, doi: 10.1016/j.jpdc.2025.105133.
- [8] M. S. Farooqui et al., "AutoSHARC: Feedback Driven Explainable Intrusion Detection with SHAP-Guided Post-Hoc Retraining for QoS Sensitive IoT Networks," *Comput. Model. Eng. Sci.*, vol. 145, no. 3, pp. 4395–4439, 2025, doi: 10.32604/cmescs.2025.072023.
- [9] D. P. Sharma, L. Xue, X. Sun, X. Lin, and P. Xiong, "Enhancing Adversarial Robustness of IoT Intrusion Detection via SHAP-Based Attribution Fingerprinting," in *2025 IEEE 24th International*

- Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, IEEE, Nov. 2025, pp. 1080–1091. doi: 10.1109/Trustcom66490.2025.00125.
- [10] L. A. C. Ahakonye, C. I. Nwakanma, J. M. Lee, and D.-S. Kim, "Machine Learning Explainability for Intrusion Detection in the Industrial Internet of Things," *IEEE Internet Things Mag.*, vol. 7, no. 3, pp. 68–74, May 2024, doi: 10.1109/IOTM.001.2300171.
- [11] Y. Pavan Kumar S, S. Mishra, and R. K. Singh, "TrustEdge: SAINT-Driven Intrusion Detection for Resource-Constrained IoT Devices," *IEEE Access*, vol. 14, pp. 84187–84195, 2026, doi: 10.1109/ACCESS.2026.3686083.
- [12] Ö. B. Dinler, "Task-scaled spiking neural networks with multi-stage feature selection for resource-constrained iot intrusion detection," *Cluster Comput.*, vol. 29, no. 3, p. 198, May 2026, doi: 10.1007/s10586-026-06064-2.
- [13] H. Benaddi, M. Jouhari, N. Laamech, A. Motii, and K. Ibrahim, "Lightweight Intrusion Detection in IoT via SHAP-Guided Feature Pruning and Knowledge-Distilled Kronecker Networks," in *2025 8th International Conference on Advanced Communication Technologies and Networking (CommNet)*, IEEE, Dec. 2025, pp. 1–6. doi: 10.1109/CommNet68224.2025.11288887.
- [14] A. K. Tripathy, A. K. Mishra, and R. Panda, "Performance Comparison and Analysis of Machine Learning and Deep Learning Models for Network Intrusion Detection In IoT-Edge Frameworks," in *2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE)*, IEEE, Feb. 2024, pp. 1–6. doi: 10.1109/ic-ETITE58242.2024.10493276.
- [15] J. García, J. Entrena, and Á. Alesanco, "Empirical evaluation of feature selection methods for machine learning based intrusion detection in IoT scenarios," *Internet of Things*, vol. 28, p. 101367, Dec. 2024, doi: 10.1016/j.iot.2024.101367.
- [16] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.
- [17] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022, doi: 10.1109/ACCESS.2022.3165809.
- [18] R. Doshi, N. Aphorpe, and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, May 2018, pp. 29–35. doi: 10.1109/SPW.2018.00013.
- [19] M. A. Ferrag, L. Maglaras, S. Moschyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.
- [20] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: Datasets and comparative study," *Comput. Networks*, vol. 188, p. 107840, Apr. 2021, doi: 10.1016/j.comnet.2021.107840.
- [21] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019, doi: 10.1109/JIOT.2018.2871719.
- [22] T. Chen and C. Guestrin, "XGBoost," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
- [23] S. M. Lundberg *et al.*, "From local explanations to global understanding with explainable AI for trees," *Nat. Mach. Intell.*, vol. 2, no. 1, pp. 56–67, Jan. 2020, doi: 10.1038/s42256-019-0138-9.
- [24] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.
- [25] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.
- [26] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," in *Proceedings 2018 Network and Distributed System Security Symposium*, Reston, VA: Internet Society, 2018. doi: 10.14722/ndss.2018.23204.
- [27] B. M. Kouassi, A. B. Ballo, K. J. Ayikpa, D. Mamadou, and M. Z. J. Coulibaly, "Top-K Feature Selection for IoT Intrusion Detection: Contributions of XGBoost, LightGBM, and Random Forest," *Futur. Internet*, vol. 17, no. 11, p. 529, Nov. 2025, doi: 10.3390/fi17110529.
- [28] S. Walling and S. Lodh, "Enhancing IoT intrusion detection through machine learning with AN-SFS: a novel approach to high performing adaptive feature selection," *Discov. Internet Things*, vol. 4, no. 1, p. 16, Oct. 2024, doi: 10.1007/s43926-024-00074-5.
- [29] F. Ebrahimi, R. Javidan, R. Akbari, and Y. Hosseini, "Intrusion detection in the internet of things using convolutional neural networks: an explainable AI approach," *Cybersecurity*, vol. 8, no. 1, p. 66, Sep. 2025, doi: 10.1186/s42400-025-00369-2.
- [30] X. Chen, M. Liu, Z. Wang, and Y. Wang, "Explainable Deep Learning-Based Feature Selection and Intrusion Detection Method on the Internet of Things," *Sensors*, vol. 24, no. 16, p. 5223, Aug. 2024, doi: 10.3390/s24165223.

Haryoko obtained his Bachelor's degree in Information Systems from Universitas Amikom Yogyakarta in 2010. He obtained his Master's degree in Computer Science from the Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, in 2016. He is currently a Lecturer in the Information Technology Study Program, Faculty of Computer Science, Universitas Amikom Yogyakarta, Indonesia. His research interests include computer networks, the Internet of Things, network security, image processing, and machine learning.

Lilis Dwi Farida is a lecturer at the Faculty of Computer Science, Amikom Yogyakarta University. His research in the field of Human Computer Interaction focuses on UI/UX. Not only in this field, his research is also in the field of Information Systems and Data Science.

Eli Pujastuti is an Assistant Professor and the Head of the Informatics Study Program, Faculty of Computer Science, Universitas Amikom Yogyakarta, Indonesia. Her research interests include human-computer interaction, user-interface evaluation, digital entrepreneurship, and educational technology.

Irma Rofni Wulandari is a lecturer in the Information Systems Study Program, Faculty of Computer Science, Universitas Amikom Yogyakarta, Indonesia. Her research interests include information systems, decision-support systems, data analytics, and Human Computer Interaction (HCI).

Muhammad Kopravi is a lecturer in the Computer Engineering Study Program, Faculty of Computer Science, Universitas Amikom Yogyakarta, Indonesia. He received his Bachelor of Computer Science from STMIK Amikom Yogyakarta and his Master of Engineering (M.Eng.) in Information Technology from the Graduate Program in Electrical Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia. He actively teaches courses in software engineering and cybersecurity. His research interests include software engineering, web security, cybersecurity, digital forensics, the Internet of Things, embedded systems, and information technology governance. He has extensive experience in software development and information technology consulting, including the development of information systems for government and organizational environments. He has also been involved as an information technology consultant and assessor in the areas of digital transformation, cybersecurity, and IT governance.

Afrig Aminuddin obtained his Bachelor's degree in Informatics Engineering from Universitas Amikom Yogyakarta in 2014. He obtained his Master's degree in Information Technology from Universitas Gadjah Mada in 2017. He obtained his Ph.D. degree in Computer Science from Universiti Malaysia Pahang in 2025. Since 2017, he has been a Lecturer at Universitas Amikom Yogyakarta. His research interests include image processing, image watermarking, image authentication, and machine learning.