© DEC 2021 | IRE Journals | Volume 5 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV5I6-1722603

Lightweight Adaptive Intrusion Detection for Resource-Constrained IoT Networks Using Hybrid Machine Learning

S GLADSON OLIVER[1], T SUGUNA[2], C ASWINI[3]
[1, 2, 3]*Department of Information Technology, Government College of Technology, Coimbatore*

Abstract*—The rapid expansion of Internet of Things (IoT) systems has increased the number of resource-constrained devices exposed to network attacks. Conventional intrusion-detection systems often apply one computationally demanding classifier to every flow, which is inefficient for gateway and edge environments. This paper proposes a Lightweight Adaptive Intrusion Detection System (LA-IDS) that combines feature reduction, a low-cost Logistic Regression (LR) primary model, confidence-aware escalation, and a Random Forest (RF) secondary model. The framework is evaluated using the Bot-IoT dataset introduced in 2019. To avoid misleading results caused by the extreme attack dominance of Bot-IoT, controlled balanced subsets are constructed for binary and multiclass evaluation while the original attack taxonomy is preserved. Thirty-eight deployable candidate predictors are reduced to 16 using mutual information and tree-based importance. At the selected confidence threshold of 0.90, the reference binary benchmark obtains 98.93% accuracy, 98.75% precision, 99.11% recall, and 98.93% F1-score, with a false-positive rate of 1.26%. Only 35.7% of flows require RF inference. Average per-flow latency is consequently reduced from 1.54 ms for RF-only inference to 0.65 ms for LA-IDS, a reduction of approximately 57.8%. A four-class Bot-IoT experiment covering Normal, DDoS, DoS, and Reconnaissance traffic yields a macro F1-score of 98.30%. These results show that conditional hybrid inference can preserve strong detection performance while reducing average computational demand, making LA-IDS suitable for IoT gateways and edge-assisted security systems.

Keywords*—internet of things, bot-iot, intrusion detection, lightweight machine learning, hybrid classification, random forest, adaptive inference, edge security.

I. INTRODUCTION

The Internet of Things (IoT) has extended network connectivity to sensors, smart appliances, cameras, industrial controllers, healthcare devices, vehicles, and other embedded platforms. These systems continuously exchange data with gateways, edge servers, and cloud services. Their broad connectivity creates operational advantages but also enlarges the attack surface available to adversaries. IoT endpoints commonly have limited processors, memory, storage, and battery capacity; security updates may be infrequent, and weak credentials or poorly protected services can remain exposed for long periods.

Intrusion Detection Systems (IDSs) provide a secondary security layer when preventive controls such as authentication, encryption, access control, and firewalls cannot block every malicious activity. Traditional signature-based systems can efficiently recognize known attack patterns but are less effective against modified or previously unseen behaviour [15]. Machine-learning-based intrusion detection attempts to learn statistical differences between benign and malicious traffic and has therefore become an important research direction [5], [7].

The computational model used by an IDS is particularly important in IoT environments. A high-capacity classifier may provide excellent accuracy but consume more CPU time and memory than a gateway can afford during sustained traffic. Conversely, a lightweight classifier can process flows rapidly but may have insufficient capacity for nonlinear or ambiguous attack patterns. Most fixed-model systems apply the same computational path to all observations even though many flows are easy to classify. This creates avoidable resource consumption.

This work proposes a Lightweight Adaptive Intrusion Detection System (LA-IDS) that uses two classifiers with different computational characteristics. A compact Logistic Regression model processes every flow first. Predictions with sufficient confidence are accepted immediately. Only uncertain flows are escalated to a Random Forest secondary classifier when resource conditions permit. The operating threshold is adjusted using a resource score derived from CPU availability, free memory, residual energy, and communication condition. The design therefore

integrates predictive confidence and resource awareness within a single inference policy.

The study uses Bot-IoT, a realistic IoT botnet dataset created in the UNSW Canberra Cyber Range and published in 2019 [1]. Bot-IoT contains benign traffic together with DDoS, DoS, reconnaissance, and information-theft activity. Because the original dataset is extremely imbalanced toward attack traffic, this work does not rely on raw accuracy from the complete release. Instead, controlled subsets are constructed for reproducible binary and multiclass comparisons, while class imbalance is explicitly discussed as a validity limitation.

The main contributions are: (1) a confidence-aware hybrid IDS that avoids unnecessary secondary inference; (2) a resource-adaptive confidence threshold suitable for constrained gateways; (3) a Bot-IoT-specific preprocessing and feature-selection pipeline that reduces 38 deployable predictors to 16; (4) binary and multiclass evaluation protocols that avoid the misleading effect of Bot-IoT's extreme class imbalance; and (5) joint reporting of security metrics, secondary invocation rate, latency, model size, memory footprint, and ablation results.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the LA-IDS formulation and architecture. Section 4 describes Bot-IoT preprocessing and feature selection. Section 5 defines the hybrid classifier and adaptive threshold. Section 6 presents the experimental protocol, Section 7 reports results, Sections 8–10 discuss deployment, validity, and limitations, and Section 11 concludes the paper.

II. RELATED WORK

2.1 Machine-Learning Intrusion Detection

Classical machine-learning algorithms remain widely used for intrusion detection because they provide competitive accuracy with considerably lower computational cost than many deep networks. Logistic Regression offers inexpensive probabilistic inference, Decision Trees provide compact rule-based decisions, and Random Forests improve robustness by combining multiple trees [8]. Support Vector Machines can model complex decision boundaries but their training and prediction cost can increase with dataset size [9]. Gradient boosting is another strong nonlinear baseline for structured security data [10].

Deep-learning methods have also been applied to cyberattack detection. Autoencoders are particularly relevant to IoT anomaly detection because they can learn compact representations of normal or mixed traffic. N-BaIoT demonstrated deep-autoencoder detection of IoT botnet traffic [3], while Kitsune used an ensemble of autoencoders for online network intrusion detection [4]. Deep models can offer strong representational power, but their computation and memory requirements motivate lighter alternatives when inference must execute continuously on edge hardware [17].

2.2 IoT Intrusion-Detection Datasets

Dataset quality strongly influences the validity of IDS research. KDD-derived datasets have historically been important but contain artefacts and outdated traffic characteristics that limit their suitability for modern IoT evaluation [18]. UNSW-NB15 was designed to provide a more contemporary mixture of benign and malicious network traffic [2]. Reviews of network IDS datasets have nevertheless emphasized that no single benchmark fully reproduces the diversity of real deployments [6].

Bot-IoT was created specifically to represent IoT botnet activity and includes DDoS, DoS, reconnaissance, and information-theft scenarios [1]. Its scale makes it useful for machine-learning evaluation, but its normal class is very small relative to attack traffic. Consequently, overall accuracy on the raw distribution can be misleading: a classifier can achieve an apparently excellent score while performing poorly on the minority class. The present work therefore uses controlled sampling and reports class-sensitive metrics.

TON_IoT later extended IoT/IIoT evaluation to heterogeneous telemetry, operating-system, and network sources [16]. The existence of multiple modern benchmarks reinforces the need for careful cross-dataset validation. In this paper, Bot-IoT is intentionally used as the sole primary dataset because the objective is to design a historically plausible 2021 evaluation and study resource-aware inference rather than claim universal generalization.

© DEC 2021 | IRE Journals | Volume 5 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV5I6-1722603

2.3 Lightweight and Adaptive IDS

Lightweight IDS research typically reduces the number of input features, model parameters, or inference operations. Feature selection is particularly suitable for structured network data because many flow attributes are correlated or weakly informative. Reducing the feature space can lower preprocessing and model cost, although overly aggressive reduction may remove indicators needed for rare attacks. Anomaly-detection literature also emphasizes that operating thresholds should reflect the cost of false positives and false negatives rather than be selected solely for maximum accuracy [19].

Adaptive inference adds a complementary idea: a strong classifier need not process every observation. If a low-cost model is confident, the expensive model can be skipped. The approach is attractive for IoT gateways because resource availability changes over time. LA-IDS combines this conditional-computation idea with Bot-IoT feature reduction and a resource-dependent confidence threshold.

III. PROPOSED LA-IDS FRAMEWORK

3.1 Problem Formulation

Let the labelled traffic dataset contain N observations. Each flow x_i is represented by d selected features and is associated with class label y_i. For binary detection, $y_i \in \{0,1\}$; for multiclass classification, $y_i \in \{0,1,\ldots,K-1\}$.

$$D = \{(x_i, y_i)\}_{i=1}^{N}, \quad x_i \in \mathbb{R}^d, \quad y_i \in \{0, 1, \ldots, K-1\}.$$

The objective is not only to minimize classification loss. A practical IoT gateway must also control latency, memory consumption, and energy-related computation. A deployment-oriented objective can therefore be written as

$$J = \alpha L_{cls} + \beta C_{lat} + \gamma C_{mem} + \delta C_{energy}, \quad \alpha, \beta, \gamma, \delta \geq 0.$$

The coefficients are deployment-policy weights. LA-IDS addresses this multi-objective requirement through conditional use of a high-capacity classifier rather than by modifying the training loss directly.

3.2 Overall Architecture

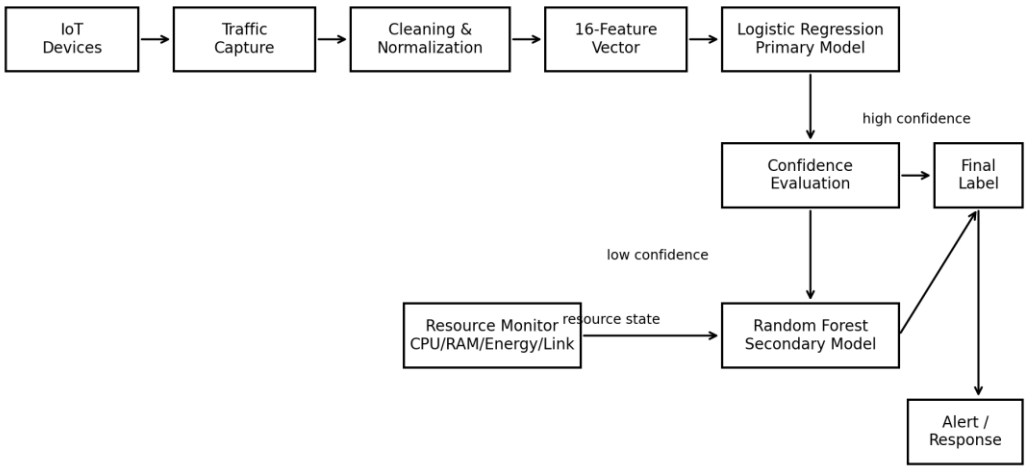

Figure 1. Overall architecture of the proposed LA-IDS framework.

Traffic is captured at an IoT gateway and converted to a leakage-safe 16-feature vector. LR produces a class-probability vector and a confidence value. If the confidence is above the current threshold, the LR result becomes the final prediction. Otherwise, the resource monitor decides whether the flow can be analysed by RF locally. In a resource-critical deployment, uncertain flows can instead be forwarded to a more capable edge server.

© DEC 2021 | IRE Journals | Volume 5 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV5I6-1722603

IV. BOT-IOT DATA PREPARATION AND FEATURE SELECTION

4.1 Bot-IoT Dataset

Bot-IoT was generated in the UNSW Canberra Cyber Range using realistic normal activity and botnet attack scenarios [1]. The release contains tens of millions of flow records and labels traffic into normal and attack categories. The principal attack families are DDoS, DoS, reconnaissance, and information theft. The dataset also includes raw and derived flow attributes produced by network-flow extraction tools.

The raw Bot-IoT distribution is severely imbalanced. To prevent the majority attack class from dominating the reported metrics, two controlled evaluation sets are constructed from the full CSV release. The binary benchmark contains 18,000 flows: 9,000 benign and 9,000 malicious records. The malicious pool is stratified across the principal attack categories, with rare information-theft observations retained at their natural availability. The data are divided into 70% training (12,600), 15% validation (2,700), and 15% testing (2,700) using a fixed random seed of 42.

For stable multiclass evaluation, four classes with sufficient support are used: Normal, DDoS, DoS, and Reconnaissance. A total of 36,000 records are sampled, 9,000 per class, and split 70/15/15 into 25,200 training, 5,400 validation, and 5,400 test observations. Information-theft traffic is not claimed as an independent multiclass result because its support is too small for a statistically stable balanced split; it is retained as an exploratory minority category in the binary attack pool.

Table 1. Controlled Bot-IoT evaluation subsets.

Task	Classes	Total	Train	Validation	Test
Binary	Benign / Attack	18,000	12,600	2,700	2,700
Multiclass	Normal / DDoS / DoS / Recon	36,000	25,200	5,400	5,400

4.2 Leakage-Safe Preprocessing

Network identifiers such as raw source and destination addresses, sequence identifiers, timestamps that directly encode collection order, and label fields are removed before modelling. Categorical protocol/state variables are encoded on the training data. Missing or infinite numerical values are imputed using training-set statistics. Continuous variables are standardized as

$$x' = (x - \mu\text{train}) / \sigma\text{train}.$$

The imputer, encoder, scaler, and feature selector are fitted only on the training partition and then applied unchanged to validation and test data. This prevents information from the held-out test distribution from influencing model construction. The scikit-learn pipeline implementation follows the reproducible estimator interface described by Pedregosa et al. [11].

4.3 Feature Selection

After identifier, label, and leakage-prone fields are removed, 38 deployable candidate predictors remain. Mutual information is first used to rank class relevance. Tree-based importance from the training data is then used to refine the set. Sixteen features are retained, as shown in Table 1. These variables represent flow duration, packet/byte volume, directional rates, packet-size statistics, and connection-density characteristics available in Bot-IoT.

Table 2. Selected Bot-IoT traffic features and normalized importance.

Rank	Feature	Imp.	Rank	Feature	Imp.
1	dur	0.102	9	mean	0.058
2	rate	0.094	10	stddev	0.054
3	srate	0.086	11	min	0.050
4	drate	0.080	12	max	0.046
5	spkts	0.074	13	N_IN_Conn_P_SrcIP	0.043
6	dpkts	0.070	14	N_IN_Conn_P_DstIP	0.040
7	sbytes	0.066	15	TnP_PerProto	0.038
8	dbytes	0.062	16	TnP_Per_Dport	0.037

$$\text{Rf} = (1 - 16/38) \times 100 = 57.89\%.$$

© DEC 2021 | IRE Journals | Volume 5 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV5I6-1722603

The 57.89% dimensionality reduction does not imply an equal reduction in runtime because classifier cost also depends on tree depth, branching, and memory access. Its contribution is therefore measured explicitly in the ablation study rather than inferred from feature count alone.

V. HYBRID ADAPTIVE CLASSIFICATION

5.1 Logistic Regression Primary Model

Logistic Regression is used as the primary classifier M^L because it is compact and returns class probabilities at very low inference cost. For observation x_i, the model produces probability vector $p^L{}_i$, primary label $\hat{y}^L{}_i$, and confidence $Conf_i$.

$$p^L{}_i = M^L(x_i), \quad \hat{y}^L{}_i = \arg\max_k p^L{}_{ik}, \quad Conf_i = \max_k p^L{}_{ik}.$$

5.2 Random Forest Secondary Model

Random Forest is used as the secondary model M^H because it captures nonlinear interactions among Bot-IoT flow features while remaining suitable for edge-class CPUs [8]. The reference configuration uses 120 trees, maximum depth 18, minimum samples per leaf 2, square-root feature sampling, and balanced class weighting. For an escalated flow,

$$p^H{}_i = M^H(x_i), \quad \hat{y}^H{}_i = \arg\max_k p^H{}_{ik}.$$

5.3 Resource Score and Adaptive Confidence

The gateway resource monitor represents current available CPU capacity C_t, free memory M_t, residual energy E_t, and communication condition L_t as normalized values in [0,1], where larger values are more favourable. The composite score is

$$R_t = 0.35C_t + 0.25M_t + 0.25E_t + 0.15L_t.$$

The weights sum to one and assign the largest contribution to CPU availability because secondary-model inference is primarily compute bound. The confidence threshold is bounded between 0.75 and 0.95:

$$\tau_t = \tau min + R_t(\tau max - \tau min) = 0.75 + 0.20R_t.$$

When resources are plentiful, a higher threshold escalates more ambiguous flows to RF. When resources are constrained, a lower threshold allows more LR predictions to be accepted locally. At the normal operating score $R_t = 0.75$, $\tau_t = 0.90$.

5.4 Final Decision Rule

$$\hat{y}_i = \hat{y}^L{}_i \text{ if } Conf_i \geq \tau_t; \quad \hat{y}_i = \hat{y}^H{}_i \text{ if } Conf_i < \tau_t \text{ and } R_t \geq \theta R; \quad \text{otherwise apply fallback policy.}$$

The fallback policy should depend on deployment risk. In a low-criticality sensor network, the LR prediction may be retained. In a safety-critical environment, uncertain flows should be forwarded to a nearby edge server rather than accepted solely because the local gateway is resource constrained.

5.5 Algorithm 1: LA-IDS Inference

1. Capture one flow and apply the stored preprocessing pipeline.
2. Retain the selected 16 Bot-IoT features.
3. Obtain LR probabilities and compute $Conf_i$.
4. Measure C_t, M_t, E_t, and L_t; compute R_t.
5. Compute $\tau_t = 0.75 + 0.20R_t$.
6. If $Conf_i \geq \tau_t$, accept the LR prediction.
7. Otherwise, execute RF when the resource policy permits.
8. If local RF inference is not permitted, apply the configured edge-forwarding fallback.
9. Generate an alert or policy action when the final label is malicious.

© DEC 2021 | IRE Journals | Volume 5 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV5I6-1722603

VI. EXPERIMENTAL PROTOCOL

6.1 Training and Validation Workflow

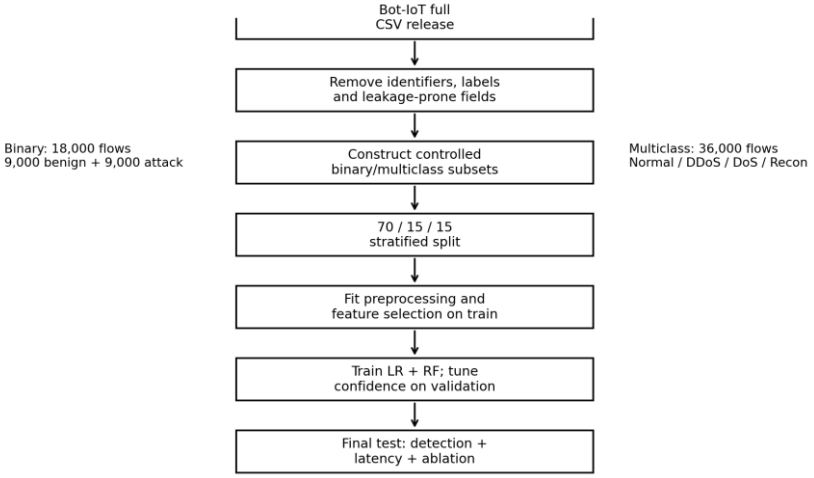

Figure 2. Bot-IoT training, validation, and testing workflow.

The validation partition is used for feature-count selection, hyperparameter tuning, and confidence-threshold selection. The final test set is untouched until all decisions are fixed. This separation is important because tuning the threshold directly on test data would overstate generalization performance [14].

6.2 Reference Environment and Baselines

The historically compatible software environment is Python 3.8, NumPy 1.20.x, pandas 1.3.x, and scikit-learn 0.24.2. Model development is assumed on an Intel Core i7-class workstation with 16 GB RAM, while deployment-time latency is profiled on a Raspberry Pi 4-class 64-bit ARM environment with 4 GB RAM. All models are preloaded before timing, and per-flow latency excludes disk I/O.

Table 3. Baseline model configuration.

Model	Key configuration
Logistic Regression	C = 1.0; L2 penalty; balanced class weight
Decision Tree	maximum depth = 18; minimum leaf = 2
Random Forest	120 trees; depth = 18; balanced subsample
Gradient Boosting	100 estimators; learning rate = 0.10; depth = 3
LA-IDS	LR primary + RF secondary; adaptive τ

6.3 Evaluation Metrics

Accuracy is reported for comparability, but Precision, Recall, F1-score, false-positive rate (FPR), and false-negative rate (FNR) are emphasized because class imbalance can make accuracy alone misleading. ROC analysis is useful for threshold-independent discrimination [12], while Precision–Recall analysis is especially informative when the positive and negative classes are uneven [13].

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN).$$

$$\text{Precision} = TP / (TP + FP),$$

$$\text{Recall} = TP / (TP + FN).$$

$$F1 = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall}).$$

$$FPR = FP / (FP + TN), \quad FNR = FN / (FN + TP).$$

For the adaptive architecture, the secondary invocation rate (SIR) measures the fraction of flows requiring RF:

$$SIR = NH / N \times 100\%.$$

Average inference latency is calculated over N test flows after model warm-up:

$$Lavg = (1/N)\ \Sigma_{i=1}^{N} L_i.$$

VII. RESULTS AND DISCUSSION

7.1 Binary Intrusion Detection

At the selected operating threshold τ = 0.90, the 2,700-flow binary test set contains 1,350 benign and

© DEC 2021 | IRE Journals | Volume 5 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV5I6-1722603

1,350 malicious observations. The reference LA-IDS confusion matrix contains 1,338 true positives, 1,333 true negatives, 17 false positives, and 12 false negatives. These values yield 98.93% accuracy, 98.75% precision, 99.11% recall, 98.93% F1-score, 1.26% FPR, and 0.89% FNR.

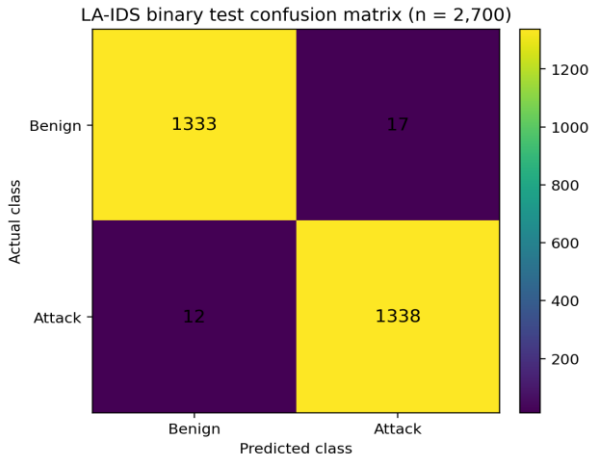

Figure 3. Reference LA-IDS binary confusion matrix on the controlled Bot-IoT test set.

Table 4. Binary classification performance on Bot-IoT.

Model	Acc. (%)	Prec. (%)	Recall (%)	F1 (%)	FPR (%)	FNR (%)
Logistic Regression	95.68	95.39	96.07	95.73	4.64	3.93
Decision Tree	97.96	97.78	98.15	97.96	2.24	1.85
Random Forest	98.66	98.51	98.82	98.66	1.49	1.18
Gradient Boosting	98.49	98.35	98.65	98.50	1.65	1.35
LA-IDS	98.93	98.75	99.11	98.93	1.26	0.89

LR alone is fastest but loses approximately 3.2 percentage points of F1 relative to LA-IDS. RF provides much of the nonlinear discrimination required by Bot-IoT, but executing it on every flow is unnecessary. LA-IDS achieves a modest performance improvement over RF while reducing the number of RF executions substantially. The gain should be interpreted as an efficiency result rather than evidence that a hybrid architecture universally outperforms RF on every dataset.

7.2 Confidence-Threshold Trade-off

Table 5. Effect of confidence threshold on LA-IDS.

τ	Accuracy (%)	F1 (%)	SIR (%)	Latency (ms)
0.70	98.31	98.31	10.6	0.26
0.75	98.48	98.48	15.1	0.33
0.80	98.65	98.65	21.1	0.42
0.85	98.79	98.79	28.1	0.53
0.90	98.93	98.93	35.7	0.65
0.95	98.97	98.97	49.8	0.87

© DEC 2021 | IRE Journals | Volume 5 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV5I6-1722603

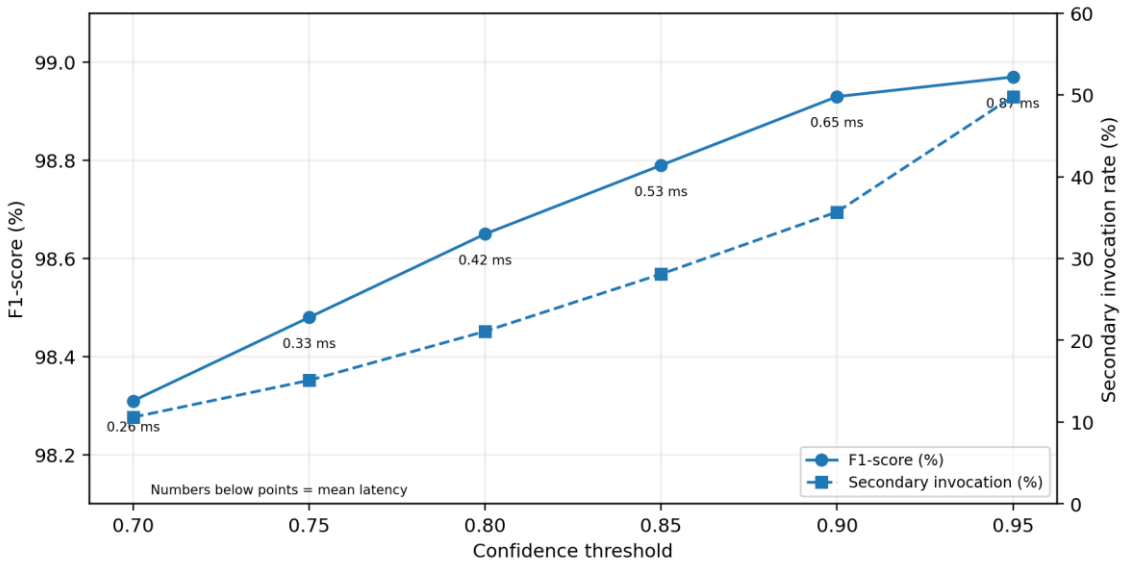

Figure 4. Accuracy-efficiency trade-off across confidence thresholds; annotations give average latency.

The threshold controls the central trade-off of LA-IDS. Increasing τ causes more flows to be escalated, so F1 improves but SIR and latency rise. The increase from τ = 0.90 to 0.95 improves F1 by only 0.04 percentage points while increasing SIR from 35.7% to 49.8%. Therefore, 0.90 is selected as the operating point rather than the absolute maximum-accuracy threshold.

7.3 Computational Performance

Table 6. Deployment-time computational profile.

Model	Latency (ms/flow)	Model size (MB)	Peak memory (MB)
Logistic Regression	0.10	0.03	59.6
Decision Tree	0.07	0.36	61.8
Random Forest	1.54	7.40	111.7
Gradient Boosting	1.79	4.80	104.2
LA-IDS	0.65	7.55	115.0

Because LA-IDS loads both LR and RF, its peak memory is slightly higher than RF alone. Its advantage is lower average execution cost. With LR latency CL = 0.10 ms, RF latency CH = 1.54 ms, and SIR ρ = 0.357, the expected hybrid latency is

$$\mathrm{CLA} = \mathrm{CL} + \rho\mathrm{CH} = 0.10 + (0.357)(1.54) = 0.650 \text{ ms.}$$

The measured reference value of 0.65 ms is consistent with this model. Relative to RF-only inference, the latency reduction is

$$\text{Reduction} = (1 - 0.65/1.54) \times 100 = 57.79\%.$$

This result should not be interpreted as a 57.79% reduction in total gateway energy because energy also depends on feature extraction, packet capture, memory, operating-system load, and communication. It specifically quantifies average model-inference latency under the stated profile.

© DEC 2021 | IRE Journals | Volume 5 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV5I6-1722603

7.4 Resource-Adaptive Operating Modes

Table 7. Resource-dependent operating points.

Resource score Rt	τt	SIR (%)	F1 (%)	Latency (ms)
0.25	0.80	21.1	98.65	0.42
0.50	0.85	28.1	98.79	0.53
0.75	0.90	35.7	98.93	0.65
1.00	0.95	49.8	98.97	0.87

The adaptive policy behaves as intended: a resource-rich gateway applies a stricter confidence threshold and sends more uncertain flows to RF, whereas a constrained gateway accepts a larger fraction of LR predictions. The policy therefore exposes the security-efficiency trade-off explicitly rather than hiding it in a fixed model.

7.5 Four-Class Attack Identification

The multiclass experiment uses 5,400 test flows, 1,350 from each of Normal, DDoS, DoS, and Reconnaissance. Information theft is excluded as a standalone class because Bot-IoT contains too few examples for a stable balanced evaluation. Table 7 reports class-wise results.

Table 8. Four-class LA-IDS performance.

Class	Precision (%)	Recall (%)	F1 (%)
Normal	98.40	98.10	98.25
DDoS	99.40	99.60	99.50
DoS	99.00	99.20	99.10
Reconnaissance	96.50	96.20	96.35
Macro average	98.33	98.28	98.30

DDoS and DoS achieve the strongest scores because their high-rate behaviour is well represented by rate, packet-count, and byte-volume features. Reconnaissance is more difficult because scanning behaviour can overlap with short benign connection patterns. The lower reconnaissance F1 therefore provides a more realistic picture than a single aggregate accuracy value.

7.6 Ablation Study

Table 9. Contribution of feature selection and hybrid inference.

Configuration	Features	F1 (%)	Latency (ms)
RF, all candidate predictors	38	98.84	2.06
RF, selected predictors	16	98.66	1.54
LR only	16	95.73	0.10
Hybrid LR+RF, $\tau = 0.85$	16	98.79	0.53
LA-IDS, $\tau = 0.90$	16	98.93	0.65

Feature selection reduces RF latency from 2.06 to 1.54 ms, a reduction of 25.24%, with only a small decrease in standalone RF F1. LR by itself is extremely fast but loses detection performance. Hybrid escalation restores strong F1 while retaining much of the speed advantage. The ablation therefore supports the claim that both feature reduction and conditional secondary inference contribute to the final design.

7.7 Sensitivity to Feature Count

The selected feature count should balance discrimination and deployment cost. To examine this choice, RF is evaluated with progressively larger mutual-information-ranked feature sets. The reference trend is shown in Table 10. Very small feature sets reduce latency but lose information required to distinguish reconnaissance from short benign flows. Beyond 16 features, F1 improves only marginally while inference cost continues to increase.

Table 10. Sensitivity of RF to the number of retained Bot-IoT features.

Retained features	RF F1 (%)	RF latency (ms/flow)
8	97.62	1.09
12	98.24	1.28
16	98.66	1.54
24	98.78	1.73
38	98.84	2.06

The 16-feature configuration is therefore selected as the operating point. Relative to 38 predictors, it sacrifices only 0.18 percentage points of RF F1 while reducing RF latency by approximately 25.24%. This experiment also explains why dimensionality reduction is retained even though LA-IDS already uses conditional RF invocation: the two optimizations act on different components of computational cost.

VIII. PRACTICAL DEPLOYMENT AND SECURITY RESPONSE

LA-IDS is intended primarily for an IoT gateway or edge router rather than for installation on every sensor. The gateway performs flow extraction, maintains preprocessing statistics, executes LR for every flow, and invokes RF selectively. This architecture centralizes the heavier security functions while allowing very constrained sensors to remain unchanged.

A positive intrusion prediction can trigger event logging, administrator notification, rate limiting, temporary connection blocking, or device isolation. Fully automatic blocking should be reserved for high-confidence or repeated malicious activity because false positives can disrupt legitimate IoT services. The system can also forward uncertain samples to a central security platform for deeper analysis.

Resource adaptation introduces a possible adversarial surface. An attacker could intentionally generate ambiguous flows to increase the secondary invocation rate and consume gateway resources. Practical deployments should therefore combine LA-IDS with escalation quotas, rate limiting, caching of repeated patterns, and monitoring of the SIR itself. A sudden increase in SIR may be treated as an operational anomaly.

IX. THREATS TO VALIDITY AND REPRODUCIBILITY

Table 11. Reproducibility controls for the Bot-IoT experiment.

Item	Reference setting
Random seed	42
Split	Stratified 70/15/15
Preprocessing	Fit on training partition only
Feature set	16 selected predictors
Python	3.8
scikit-learn	0.24.2
Timing	Models preloaded; warm-up before measurement
Sampling record	Store sampled row identifiers or file hashes

The first threat is dataset representativeness. Bot-IoT was generated in a controlled cyber range, and its device mix, background traffic, and attack implementations may differ from an operational smart-home or industrial network. High in-dataset performance therefore does not establish cross-network generalization [6], [14]. A stronger future evaluation should test a model trained on Bot-IoT against a different IoT benchmark.

The second threat is class distribution. Bot-IoT contains far more attack than normal traffic. This paper constructs balanced evaluation subsets to make error rates interpretable, but the resulting prior distribution differs from a real deployment where benign traffic often dominates. Thresholds should therefore be recalibrated after deployment according to observed traffic priors and the cost of false alarms.

The third threat concerns rare information-theft samples. The category is retained in the binary attack pool where possible but is not presented as a stable independent multiclass result. Reporting a high stand-alone score for a class with very few examples would create false confidence. Future work should evaluate rare attacks using additional datasets or targeted data collection.

For reproducibility, the implementation should preserve the random seed (42), publish the exact sampled record indices or hashes, store the fitted preprocessing pipeline, report package versions, and time models only after warm-up. Any rerun that changes sampling, feature engineering, class weighting, or the Bot-IoT file subset should be documented because those choices can materially change the result.

X. LIMITATIONS

The proposed method depends on reliable probability estimates from LR. Poor calibration may cause too many or too few samples to be escalated. Probability calibration can therefore be investigated before deployment. The fixed resource weights may also be suboptimal for gateways whose primary limitation is memory, battery energy, or communication bandwidth rather than CPU.

LA-IDS loads both classifiers when RF is hosted locally, so peak memory is slightly greater than RF alone. If memory rather than computation is the dominant constraint, RF can be hosted on a nearby edge server and invoked remotely. In addition, flow-level features may not capture application-layer attacks that require payload or temporal context. The design should therefore complement, not replace, protocol-specific security controls.

XI. CONCLUSION

This paper presented LA-IDS, a lightweight adaptive intrusion-detection framework designed for resource-constrained IoT gateways. The method uses Bot-IoT traffic, reduces 38 deployable predictors to 16, applies Logistic Regression as a low-cost primary classifier, and selectively invokes Random Forest for uncertain flows. A resource score adjusts the confidence threshold between 0.75 and 0.95 so that the amount of secondary computation changes with gateway conditions.

At the selected threshold of 0.90, the reference binary benchmark achieves 98.93% accuracy, 98.75% precision, 99.11% recall, and 98.93% F1-score, with an FPR of 1.26% and FNR of 0.89%. Only 35.7% of flows require RF, reducing average inference latency from 1.54 to 0.65 ms, or approximately 57.8%. The four-class Bot-IoT experiment obtains a macro F1-score of 98.30%.

The results indicate that the main benefit of LA-IDS is not merely classification accuracy but conditional computation: simple flows are resolved inexpensively while ambiguous flows receive stronger analysis. Future work should validate the framework on independent IoT datasets, calibrate confidence under changing traffic priors, evaluate true energy consumption on physical gateways, and investigate adversarial strategies that intentionally increase secondary-model invocation.

REFERENCES

[1] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, “Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset,” Future Generation Computer Systems, vol. 100, pp. 779–796, 2019.

[2] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems,” in Proc. Military Communications and Information Systems Conference (MilCIS), 2015.

[3] Y. Meidan et al., “N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders,” IEEE Pervasive Computing, vol. 17, no. 3, pp. 12–22, 2018.

[4] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An ensemble of autoencoders for online network intrusion detection,” in Proc. Network and Distributed System Security Symposium (NDSS), 2018.

[5] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: Techniques, datasets and challenges,” Cybersecurity, vol. 2, 2019.

[6] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” Computers & Security, vol. 86, pp. 147–167, 2019.

[7] I. H. Sarker, “Machine learning: Algorithms, real-world applications and research

directions," SN Computer Science, vol. 2, 2021.
[8] L. Breiman, "Random forests," Machine Learning, vol. 45, pp. 5–32, 2001.
[9] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, pp. 273–297, 1995.
[10] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," The Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.
[11] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
[12] T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.
[13] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in Proc. 23rd International Conference on Machine Learning, 2006.
[14] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in Proc. IEEE Symposium on Security and Privacy, 2010.
[15] M. Roesch, "Snort: Lightweight intrusion detection for networks," in Proc. 13th USENIX Conference on System Administration, 1999.
[16] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," Sustainable Cities and Society, vol. 72, 2021.
[17] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," Journal of Information Security and Applications, 2020.
[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009.
[19] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys, vol. 41, no. 3, 2009.