

ARTICLE

A Family-Aware Hierarchical XGBoost Framework for Efficient IoT Intrusion Detection

Motab F. Alenezi¹, Fahad M. Alotaibi¹, Badraddin Alturki², Ahmad J. Tayeb²,
Abdulaziz A. Alsulami^{1,*} and Abdullah Alhejaili¹

¹Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

²Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

*Corresponding Author: Abdulaziz A. Alsulami. Email: aaalsulami10@kau.edu.sa

Received: 06 June 2026; Accepted: 13 August 2026; Published: 28 September 2026

ABSTRACT: Machine learning-based intrusion detection for Internet of Things (IoT) networks remains difficult because modern traffic is highly imbalanced and attack behaviors are heterogeneous. Evaluation pipelines can also overestimate performance when preprocessing is performed before train-test separation. We propose a family-aware hierarchical intrusion detection framework for attack-family prediction. The proposed approach first separates normal and attack traffic, then routes attack samples into empirically defined majority and minority attack-family branches, and finally performs branch-specific family classification. Within each cross-validation fold, training-label counts define the majority/minority routing branches, while scaling, weighting, model fitting, stage diagnostics, and metric computation remain fold-local. The final implementation uses XGBoost as the base learner in the hierarchical stages and compares it with flat LightGBM, XGBoost, Random Forest, Extra Trees, and stacking baselines under matched folds and metrics. On the CICIoT2023 30% stratified development split, the proposed approach achieved a Macro-F1 of 0.8380 and Weighted-F1 of 0.9940, performing close to the best flat Random Forest baseline while improving weak rare-family F1 scores for BruteForce and Web. On Edge-IIoTset, where the full processed dataset is used, the proposed approach achieved a Macro-F1 of 0.9456 and Weighted-F1 of 0.9494, outperforming all individual flat baselines and approaching the flat stacking ensemble. The hierarchy had lower inference time than the evaluated flat baselines under the workstation protocol.

KEYWORDS: Intrusion detection; IoT security; attack-family classification; hierarchical learning; XGBoost; class imbalance; runtime evaluation; leakage-controlled evaluation

1 Introduction

The Internet of Things (IoT) has become a major infrastructure layer for automated communication and decision-making in healthcare, transportation, smart cities, and industrial control systems. The Internet of Medical Things (IoMT) [1,2] and connected autonomous vehicle networks, such as Controller Area Network (CAN) buses [3], are examples of critical domains where this integration is rapidly expanding. However, as more different types of edge devices are connected, the attack surface also becomes larger [4,5]. This is especially important because many edge devices have limited computing power and energy. As reported in [6], the number of IoT devices is expected to exceed 31 billion by 2030. This growth supports the economic and technological progress, but it also increases the scale and complexity of cyberthreats. Recent studies

show that IoT networks are facing more types of attacks, which include Denial of Service (DoS), Distributed Denial of Service (DDoS), reconnaissance, spoofing, brute-force, and web-based attacks. These attacks can affect data privacy and the operational reliability of critical infrastructure [7].

Intrusion Detection Systems (IDSs) are an important security layer used to mitigate cyberattacks. They help monitor network traffic to identify suspicious activities. Artificial Intelligence (AI) and Machine Learning (ML) techniques are widely used to implement IDSs. This is because traditional signature-based methods have limitations in detecting polymorphic malware, unknown attack patterns, and zero-day vulnerabilities [8]. Gradient-boosting frameworks such as LightGBM and XGBoost are ensemble learning methods that use multiple decision-tree models to improve prediction performance. They have shown strong performance in analyzing complex and high-dimensional network traffic features [9]. Recent studies have further improved these models by combining them with feature transformation, deep neural networks, and stacked ensembles [2,10,11].

With these advancements, the evaluation of IDSs in IoT environments still faces several challenges. First, many studies report high accuracy on imbalanced data. However, accuracy can hide weak detection of low-frequency attack families. For example, in CICIoT2023, DDoS traffic dominates the attack distribution, while BruteForce and Web attacks represent only a very small fraction of samples. As a result, a model can achieve high overall accuracy while still failing to detect rare but important attack families. In addition, grouping all attacks into broad categories may oversimplify the task, so the reported metrics may no longer reflect attack-family detection. A model that distinguishes only Normal, Majority Attack, and Minority Attack may be operationally useful, but it should not be presented as fine-grained multi-class intrusion detection. Furthermore, reproducibility is weakened when studies do not clearly state whether scaling, feature selection, sampling, weighting, or threshold selection are fitted only on the training fold. Leakage in these steps can inflate reported performance and make comparisons unreliable.

This paper proposes a family-aware hierarchical IDS for network-level IoT traffic. The model is designed for attack-family prediction. Stage 1 separates Normal traffic from Attack traffic. Stage 2 routes attack samples into Majority Attack and Minority Attack branches using a frequency rule computed from each training fold. Stage 3A classifies majority attack families, while Stage 3B classifies minority attack families. The final output is a family label such as Benign, DDoS, DoS, Mirai, Recon, Spoofing, Web, or BruteForce for CICIoT2023, and the corresponding available family labels for Edge-IIoTset. XGBoost is used as the final base learner after controlled experiments with LightGBM, Random Forest, Extra Trees, and other variants. The evaluation reports family-level Macro-F1 and inference time.

Recent IDS studies have examined two-stage classification [12], class-imbalance mitigation [13], and lightweight intrusion detection [14]. The proposed framework derives its routing branches from attack-family frequencies and evaluates family-level predictions using fold-local preprocessing and weighting. It also reports paired fold-level inference-time comparisons.

The major contributions of this paper are summarized as follows:

- A family-aware hierarchical IDS is proposed for attack-family prediction, combining Normal-vs.-Attack detection, empirical Majority-vs.-Minority routing, and branch-specific family classification, with majority and minority attack-family membership determined by a data-derived coverage rule.
- A leakage-controlled and reproducible evaluation protocol is developed, including audited empirical label grouping, fold-local preprocessing and weighting, direct flat-baseline comparisons, and an accompanying replication package¹.

¹<https://zenodo.org/records/20568228>

- A comparative evaluation of family-level Macro-F1 and fold-level inference time against five flat baselines.
- Base-learner-controlled ablations, a frequency-aware comparison, stage diagnostics, and a full CICIOT2023 three-fold evaluation identify where the hierarchy helps and where flat baselines remain competitive.

The remainder of this paper is organized as follows. [Section 2](#) reviews related work. [Section 3](#) describes the proposed methodology. [Section 4](#) reports and discusses the results. Finally, [Section 5](#) concludes the paper and outlines future research directions.

2 Related Work

Research on ML-based IDSs for IoT environments has expanded across classical ensemble methods, deep learning architectures, and federated learning frameworks. Despite strong reported results, several limitations remain unresolved. Class imbalance is often not evaluated at the detailed attack-family level, which can hide poor detection of rare attacks. Some models are also computationally demanding, making them difficult to deploy on resource-constrained edge devices. In addition, some evaluation protocols do not fully prevent data leakage, especially when preprocessing, feature selection, or sampling steps are applied before data splitting. This section reviews recent IDS studies in relation to task granularity, imbalance handling, computational efficiency, and leakage-control evidence, then identifies the research gap addressed by the proposed framework. [Table 1](#) summarizes the main characteristics and reported results of these studies.

Classical ML classifiers and gradient-boosting frameworks are well suited to structured network-traffic data and dominate the IDS literature for tabular IoT datasets. Alharby [\[15\]](#) addressed class imbalance and computational inefficiency on CICIOT2023 using undersampling combined with three feature selection methods, namely Chi-square, PCA, and Random Forest Regressor. They evaluated Decision Tree, Random Forest, Gradient Boosting, Naive Bayes, and K-Nearest Neighbors. A peak accuracy of 99.99% was reported with the Decision Tree model across four traffic classes: Normal, DoS, DDoS, and Mirai. While the reported result is strong, the four-class formulation excludes rarer attack families, and the study does not clearly specify whether feature selection was restricted to the training partition, leaving the leakage status of the evaluation unclear.

Almahaqeri et al. [\[13\]](#) presented a reproducible sequential preprocessing pipeline that combines stratified undersampling with gain-based feature selection for XGBoost, LightGBM, and CatBoost on CICIOT2023. For XGBoost, held-out test accuracy and Macro-F1 were 99.61% and 0.9952, respectively, in the binary task. The corresponding Macro-F1 values were 0.8903 for the 8-class attack-family task and 0.8876 for the 34-class fine-grained task. The lower multiclass Macro-F1 values indicate greater difficulty at finer label granularities. Houichi et al. [\[16\]](#) applied optimized K-Nearest Neighbors with hyperparameter tuning on the same dataset and reported an F1-score of 0.563 for the 8-class task. These findings indicate that high overall accuracy in simplified settings can hide poor performance on minority attack classes, which motivates the need for evaluation protocols that explicitly report per-class performance.

Deep learning methods have also been applied to capture complex temporal and spatial patterns in network traffic. Wang et al. [\[17\]](#) proposed CTWA, which combines a Convolutional Autoencoder (CAE) and Temporal Convolutional Network (TCN) with Weight Alignment techniques to address catastrophic forgetting in incremental learning. The model achieved 96.43% accuracy on CICIOT2023. Gao et al. [\[14\]](#) proposed a lightweight TCN with variance filtering, mutual information, and Pearson-correlation feature selection for Edge-IIoTset. Accuracy was 94.24% with all features and 93.55% with 18 selected features. The

reported TCN result was 93.79% accuracy and 93.13% F1, with a 16.25 KB model, 39.48 MB memory use, and 0.000852 s inference per instance.

Berguiga et al. [18] implemented a CNN-LSTM intrusion detector for IoMT environments on a Raspberry Pi fog node and evaluated binary and multiclass detection on IoTID20 and Edge-IIoTset. On Edge-IIoTset, the reported accuracy was 99.941% for binary classification and 99.939% for multiclass classification. The reported results do not include metrics for individual rare attack families. Salehiyan et al. [10] introduced a hybrid Transformer-GAN-AE framework using GAN-based data balancing, Autoencoder-based denoising, and Transformer classification optimized by an Improved Chimp Optimization Algorithm. Their model achieved 98.63% accuracy on Edge-IIoTset and 98.92% on TON_IoT. Arivukarasi et al. [19] combined a federated autoencoder with a High-performance Vision Transformer and lightweight feature selection. They report 99.84% accuracy and 98.85% F1 on CICIoT2023. The preprocessing procedure for cross-validation is not described in fold-local terms.

Federated learning (FL) has attracted attention as a privacy-preserving alternative to centralized IDS. Anwer et al. [20] integrated a CNN-BiLSTM model within a federated framework for Industrial IoT. Their preprocessing includes correlation- and mutual-information-based feature selection, and they report centralized and federated Edge-IIoTset accuracies of 96.20% and 95.00%, respectively. Cao et al. [21] proposed FedDynST, which combines Graph Convolutional Networks with 1D CNNs and static and dynamic adjacency matrices for DDoS detection in cloud-edge collaborative systems. The study selects 20 traffic features, excludes IP addresses and port numbers, and reports 99.47% global-test accuracy on Edge-IIoTset. The evaluation focuses on DDoS detection. Bilal et al. [22] provided a systematic benchmark of FL aggregation algorithms, including FedAvg, FedProx, and FedNova, with LSTM and Transformer backbones across multiple non-IID client configurations. They reported peak in-domain accuracy of 98.00% on Edge-IIoTset and 97.00% on CICIoT2023 using FedNova with a Transformer. Tanveer et al. [23] combined a federated digital twin with spatiotemporal deep learning and post-quantum cryptographic security for IIoT anomaly detection, reporting 99.89% accuracy, 99.91% precision, 99.87% recall, and an AUC of 0.998 on the RT-IoT2022 dataset. Their evaluation is limited to binary anomaly detection, and per-class or rare-attack performance is not reported. Although FL approaches provide privacy and deployment advantages, they often trade detection performance for decentralization and do not explicitly address minority attack detection under a leakage-controlled evaluation protocol.

Baz [24] proposed a lightweight Counter-Propagation Network (CPN) that combines an unsupervised Growing Hierarchical Self-Organizing Map with a supervised Grossberg-outstar layer for resource-constrained IoT deployment. The model reported 99.99% accuracy across seven public datasets, including CICIoT2023. However, near-perfect accuracy across heterogeneous datasets should be interpreted carefully because high overall accuracy on imbalanced datasets can mask weak minority-class detection. Nasir et al. [25] proposed an edge-aware multi-modal intrusion detection framework for Consumer-Centric Industrial IoT that fuses a GRU for host-log analysis, an LSTM for network-flow modeling, a 1D Transformer for packet-level traffic, and a fine-tuned TinyBERT model for phishing-email detection through a learnable adaptive fusion mechanism. Evaluated across UNSW-NB15, CIC-IDS2018, host-log, and phishing-email datasets, the adaptive fusion configuration outperformed simple-averaging fusion by 1.20 percentage points in accuracy. The authors state that detailed deployment benchmarking on edge hardware remains future work.

Table 1: Comparison of related IoT IDS studies by task, imbalance handling, leakage-control status, and efficiency reporting. The table distinguishes explicit leakage-control evidence from protocols in which fold-local preprocessing is not described.

Study	Task/Classes	Imbalance Handling and Leakage-Control Status	Efficiency and Main Result
Alharby [15]	CICIoT2023, four balanced attack categories	Undersampling. Feature selection is described before splitting, so fold-local preprocessing is not documented	9.1 s training time and 0.03 s prediction time. 99.99% accuracy after balancing
Baz [24]	TII-SSRC-23, IoT-23, CICIoT2023, multi-class	Imbalance handling not reported. Train/test separation is stated, but fold-local preprocessing is not described	Reports FLOPs, learnable parameters, model size, inference time, and energy consumption. Near-unity accuracy is reported
Houichi et al. [16]	CICIoT2023, binary, 8-, and 34-class	No completed balancing method reported. Split description is internally inconsistent	No per-sample latency or model-size result reported. Eight-class KNN F1: 0.563
Wang et al. [17]	CICIoT2023, incremental multi-class	Imbalance handling not emphasized. A 7:3 split is reported. Fold-local preprocessing is not described	789.58 s runtime for 40 training epochs. 96.43% accuracy
Arivukarasi et al. [19]	CICIoT2023, federated multi-class	Non-IID client imbalance is discussed. Validation separation is described, but a raw-data fold-local protocol is not reported	Edge-device inference, model size, and energy. 99.84% accuracy and 98.85% F1
Almahaqeri et al. [13]	CICIoT2023, binary, 8-, and 34-class	Undersampling and class weights. Explicit leakage-safe split and selection	Training time and per-sample latency. 99.61% binary accuracy and 0.8903 eight-class Macro-F1
Salehiyan et al. [10]	WUSTL-IIoT-2021, Edge-IIoTset, TON_IoT	SMOTE, over- and undersampling. A 70/15/15 split is reported. Fold-local preprocessing is not described	Convergence runtime, not per-sample latency. 98.63% Edge-IIoTset accuracy
Anwer et al. [20]	X-IIoTID, WUSTL-IIoT, Edge-IIoTset, DDoS, Injection, Backdoor, MitM	SMOTE. Train/validation/test split is reported. Fold-local preprocessing is not described	Training time, per-record inference time, and model size. 96.2% centralized Edge-IIoTset accuracy
Gao et al. [14]	Edge-IIoTset, 15 classes	No completed balancing method reported. Fold-local preprocessing is not described	16.25 KB model size, 39.48 MB memory usage, and 0.000852 s/sample inference time. 93.79% accuracy and 93.13% F1
Cao et al. [21]	CICDDoS2019 and Edge-IIoTset, DDoS-specific	Balanced clients and random sampling. Federated 8:1:1 client split is reported. Fold-local preprocessing is not described	Federated-round convergence time, not per-sample latency. 99.47% global-test accuracy

(Continued)

Table 1 (continued)

Study	Task/Classes	Imbalance Handling and Leakage-Control Status	Efficiency and Main Result
Berguiga et al. [18]	IoTID20 and Edge-IIoTset, binary and multiclass	SMOTE. A split is reported, but fold-local preprocessing is not described	On Edge-IIoTset, Raspberry Pi test time was 59.11 s for binary classification and 92.5 s for multiclass classification. The corresponding accuracies were 99.941% and 99.939%
Bilal et al. [22]	Edge-IIoTset, CICIOT2023, TII-SSRC-23, 6/8/5 classes	No rebalancing in the main runs. Scaler fit on the training split	Communication-round efficiency. No per-sample latency. 98% Edge-IIoTset accuracy with FedNova Transformer

Table 2 focuses on four studies that align with the evaluation dimensions of this work. Almahaqeri et al. [13] provide a CICIOT2023 gradient-boosting reference across binary, 8-class, and 34-class tasks. Gao et al. [14] provide Edge-IIoTset evidence for a lightweight model with reported model parameters, memory usage, and per-instance inference time. Baz [24] evaluates a GHSOM-based Counter-Propagation Network on CICIOT2023 and reports resource metrics in an IoT-23 benchmark. The GHSOM-based architecture organizes learned representations hierarchically. It does not use separate branch-specific classifiers for sample routing. Berguiga et al. [18] provide Edge-IIoTset and Raspberry Pi fog-node evidence for binary and multiclass IDS. Together, these studies permit a source-specific comparison of task formulation, data treatment, efficiency reporting, and evaluation context.

Table 2: Comparison of the proposed framework with selected IoT IDS studies in terms of model design, task formulation, data treatment, reported evidence, and validation context.

Property	Almahaqeri et al. [13]	Gao et al. [14]	Baz [24]	Berguiga et al. [18]	Proposed Framework
Model design	Flat XGBoost, LightGBM, and CatBoost classifiers	Temporal Convolutional Network with hybrid feature selection	GHSOM-based Counter-Propagation Network with a Grossberg-outstar layer	CNN-LSTM hybrid in a Raspberry Pi fog architecture	Normal/Attack detection, majority/minority routing, and branch-specific family classifiers
Task reported	CICIOT2023 binary, 8-class, and 34-class tasks	15-class Edge-IIoTset task	Multiclass IDS on TII-SSRC-23, IoT-23, and CICIOT2023	Binary and multiclass IDS on IoTID20 and Edge-IIoTset	Attack-family labels on CICIOT2023 and Edge-IIoTset
Main data treatment	Stratified undersampling, inverse-frequency weights, and gain-based feature selection	Variance filtering followed by mutual-information and Pearson-correlation feature selection	Hierarchical topological representation learning	SMOTE for binary and multiclass balancing	Frequency-derived routing taxonomy and training-fold sample weighting

(Continued)

Table 2 (continued)

Property	Almahaqeri et al. [13]	Gao et al. [14]	Baz [24]	Berguiga et al. [18]	Proposed Framework
Reported evidence	Leakage-safe splitting and selection, training time, and per-sample inference latency	16.25 KB model, 39.48 MB memory, and 0.000852 s per instance	IoT-23 resource comparison: FLOPs, learnable parameters, model size, inference time, and energy consumption	Raspberry Pi fog deployment and binary/multiclass testing durations	Fold-local scaling, weighting, fitting, and diagnostics. Workstation latency, throughput, serialized size, and process RSS
Evaluation and deployment context	Stratified 80/20 split with five-fold cross-validation for hyperparameter optimization	Edge-IIoTset evaluation on a workstation, with edge-oriented deployment motivation	Training/test sub-dataset separation reported. No physical edge-device test reported	Stratified 80/20 split with Raspberry Pi fog-node evaluation	Stratified five-fold evaluation. CICIoT2023 uses a 30% development split and Edge-IIoTset uses the full processed dataset

Recent IoT IDS studies report strong results, but the studies summarized in [Tables 1](#) and [2](#) differ in dataset composition, label definitions, sampling strategies, class-imbalance handling, and leakage-control evidence. Coarse attack grouping can obscure the difficulty of family-level detection, while preprocessing before data splitting can inflate reported performance. This study addresses these gaps by evaluating attack-family classification directly, auditing label transformations, comparing multiple flat baselines under identical folds, and reporting runtime alongside accuracy-oriented metrics. The majority/minority split serves only as an internal routing mechanism, while the final prediction remains an attack-family label.

3 Methodology

This section describes the proposed family-aware hierarchical IDS, the label-audit procedure, the experimental protocol, and the leakage-control safeguards used in the evaluation. The design objective is to preserve the interpretability of a hierarchical IDS while reporting family-level predictions that are more informative than a three-class grouped task.

3.1 Problem Formulation

Let $X \in \mathbb{R}^{n \times d}$ denote the network-flow feature matrix, where n is the number of samples and d is the number of input features. Let y denote the corresponding family-level target label. The prediction task is multi-class attack-family classification. Normal traffic is retained as the normal family, and attack traffic is mapped to family-level labels. For CICIoT2023, fine-grained variants such as DoS-UDP_Flood, DoS-TCP_Flood, and DoS-SYN_Flood are mapped to DoS. DDoS variants are mapped to DDoS, Mirai variants are mapped to Mirai, and related low-frequency labels are mapped to BruteForce, Recon, Spoofing, or Web. Edge-IIoTset uses the available processed family-level labels.

The final prediction objective is formalized in [Eq. \(1\)](#):

$$\hat{y} \in \mathcal{Y}_{family}, \quad (1)$$

Here, $\hat{y}$ is the predicted family label and $\mathcal{Y}_{family}$ is the set containing the normal label and all attack-family labels available in the dataset. The Majority Attack and Minority Attack groups are not final output classes. They are internal routing labels used to train specialized Stage 3 classifiers.

3.2 Empirical Majority and Minority Grouping

The majority/minority split is determined independently within each cross-validation fold from attack-family counts in the training partition. Normal or benign labels are assigned to the Normal group. Attack families are sorted by descending count, and the smallest set whose cumulative count covers at least 50% of training-fold attack samples is assigned to the Majority Attack branch. All remaining attack families are assigned to the Minority Attack branch. The resulting taxonomy is applied unchanged to that fold's test partition.

[Table 3](#) summarizes the source-dataset family counts and the corresponding grouping under the 50% coverage rule. In CICIOT2023, DDoS accounts for approximately 74.56% of attack samples. In Edge-IIoTset, DDoS and Reconnaissance together account for approximately 53.17%. During cross-validation, branch membership was recomputed from each training fold. For both datasets, all five fold-specific groupings yielded the same majority and minority family sets shown in [Table 3](#).

Table 3: Source-dataset family counts and reference grouping under the 50% coverage rule. For each dataset, the five fold-specific groupings were computed separately from training-fold labels and yielded the same family sets shown here. Majority and minority labels are internal routing labels, not final output classes.

Dataset	Routing Group	Attack-Family Labels	Source-Dataset Count	Attack Coverage
CICIOT2023	Normal	Benign	178,379	–
	Majority Attack	DDoS	5,511,797	74.56%
	Minority Attack	DoS, Mirai, Spoofing, Recon, Web, BruteForce	1,880,238	25.44%
Edge-IIoTset	Normal	Normal	24,301	–
	Majority Attack	DDoS, Reconnaissance	70,544	53.17%
	Minority Attack	Web_Injection, Ransomware, Uploading, Backdoor, Password, MITM	62,141	46.83%

The coverage threshold was evaluated at 40%, 50%, 60%, and 70% using the dataset-specific five-fold protocol of the main comparison. Fold-specific branch membership at each threshold was derived from the training labels. On CICIOT2023, DDoS alone exceeded every tested threshold in all folds, so the grouping and Macro-F1 remained unchanged. On Edge-IIoTset, the 40% and 50% thresholds assigned DDoS and Reconnaissance to the majority branch in every fold and produced the same Macro-F1. The 60% threshold also included Web_Injection, and the 70% threshold included Ransomware. [Table 4](#) shows that the 40% and 50% thresholds shared the highest observed Edge-IIoTset Macro-F1 of 0.9456.

Table 4: Sensitivity of the proposed hierarchy to the majority-branch coverage threshold under the dataset-specific main protocol. CICIOT2023 uses the stratified 30% development subset, and Edge-IIoTset uses the full processed dataset. Values are Macro-F1 mean $\pm$ standard deviation across five folds. Bold values indicate the highest mean within each dataset column, including ties.

Coverage Threshold	CICIOT2023	Edge-IIoTset
40%	0.8380 $\pm$ 0.0060	0.9456 $\pm$ 0.0015
50%	0.8380 $\pm$ 0.0060	0.9456 $\pm$ 0.0015
60%	0.8380 $\pm$ 0.0060	0.9397 $\pm$ 0.0018
70%	0.8380 $\pm$ 0.0060	0.9256 $\pm$ 0.0009

3.3 Proposed Family-Aware Hierarchical Model

Fig. 1 shows the proposed IDS architecture. The pipeline contains four predictive stages:

- **Stage 1: Normal-vs.-Attack detection.** The first classifier is trained on all training-fold samples using binary labels Normal and Attack.
- **Stage 2: Majority-vs.-Minority routing.** Samples whose true training label is an attack are used to train a routing classifier that distinguishes Majority Attack from Minority Attack using the grouping computed from that fold's training labels.
- **Stage 3A: Majority-family classification.** A family classifier is trained only on attack families in the majority branch.
- **Stage 3B: Minority-family classification.** A family classifier is trained only on attack families in the minority branch.

At inference time, a sample first passes through Stage 1. Samples predicted as normal exit the hierarchy. Predicted attack samples are routed by Stage 2 to Stage 3A or Stage 3B, and the selected branch outputs the final attack-family label. The final hierarchical decision rule is defined in Eq. (2):

$$\hat{y} = \begin{cases} \text{Normal,} & \text{if Stage 1 predicts Normal,} \\ f_{3A}(x), & \text{if Stage 1 predicts Attack and Stage 2 predicts Majority,} \\ f_{3B}(x), & \text{if Stage 1 predicts Attack and Stage 2 predicts Minority.} \end{cases} \quad (2)$$

Here, x denotes an input sample, f_{3A} and f_{3B} denote the majority- and minority-branch family classifiers, respectively, and $\hat{y}$ denotes the predicted family label.

3.4 Base Learner and Imbalance Handling

The final proposed model uses XGBoost as the base learner for all hierarchy stages. This choice was made after controlled ablations showed that the hierarchical decomposition is most beneficial with boosting-based learners, especially XGBoost. Flat LightGBM, flat XGBoost, flat Random Forest, flat Extra Trees, and flat stacking are used as baselines.

Class imbalance is handled using training-fold-only sample weights. This choice avoids synthetic traffic generation and is easier to defend in an IDS setting where generated samples may not correspond to valid network behavior. For each fold, class weights are computed only from the training partition. In the proposed hierarchy, weights are computed separately for Stage 1, Stage 2, Stage 3A, and Stage 3B using only the samples available to that stage. The test fold is never used to compute weights.

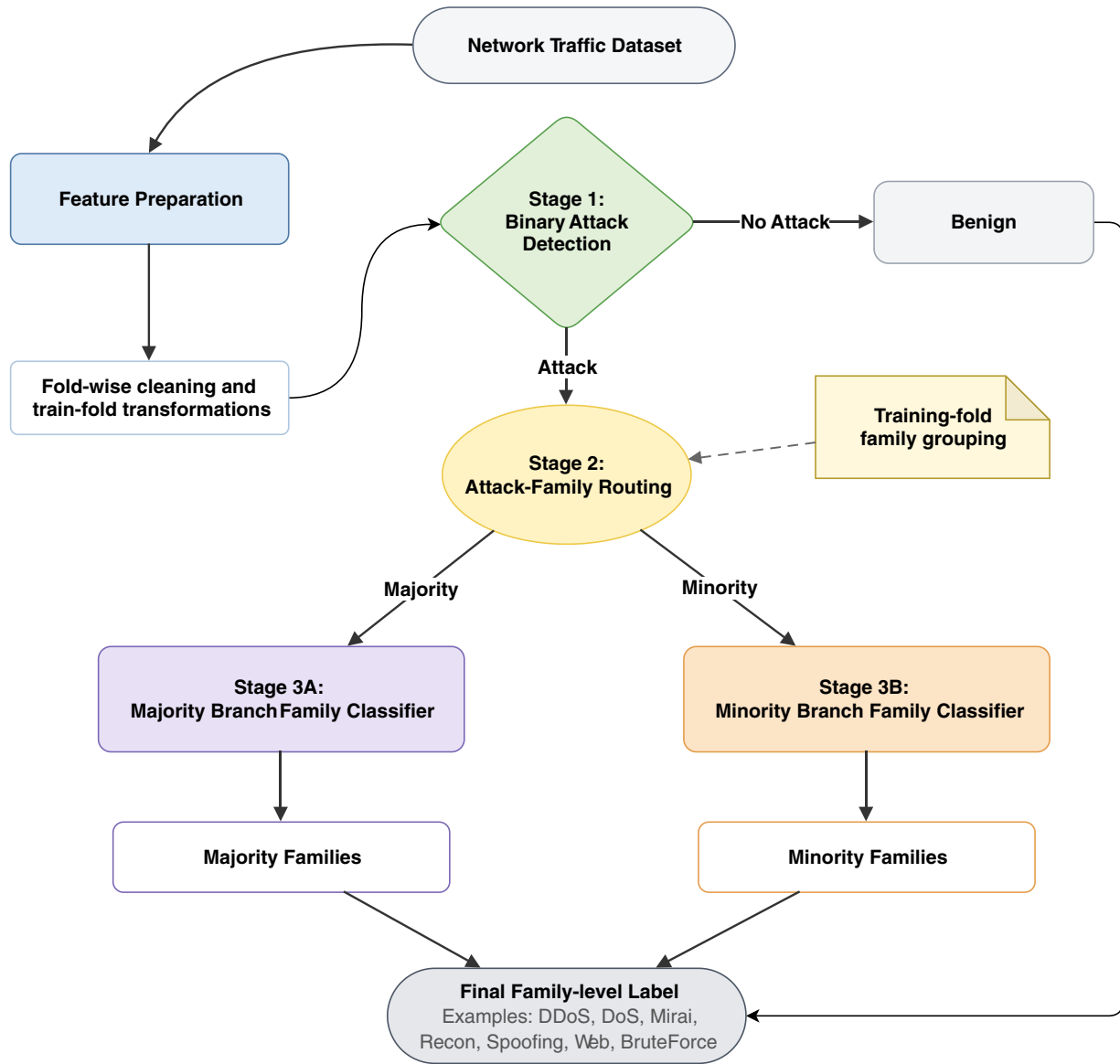


Figure 1: Overview of the proposed family-aware hierarchical IDS. The majority/minority branch is used for routing, while the final output remains an attack-family label.

Table 5 lists the configurations used in the main comparison. The proposed XGBoost hierarchy uses a Stage 1 attack-probability threshold of 0.5 and a minority-branch weight strength of 1.0. Gradient-boosting models train for their specified numbers of estimators without early stopping. The flat stacking ensemble combines Random Forest, Extra Trees, LightGBM, and XGBoost with an internal 3-fold cross-validated logistic-regression meta-learner.

3.5 Datasets and Evaluation Protocol

Two IoT IDS datasets are used: CICIoT2023 [26] and Edge-IIoTset [27]. CICIoT2023 is very large, containing 7,570,414 cleaned family-normalized records. For the main five-fold paper experiment, a stratified 30% development split is used to preserve the family distribution while making repeated baseline comparison computationally feasible across all flat and hierarchical models. This split contains 2,271,123 records and keeps

approximately 30% of each family label, as shown in Fig. 2. An additional three-fold evaluation on the full CICIOT2023 dataset is reported in the results section. Edge-IIoTset is substantially smaller. The full processed dataset of 156,986 records is therefore used in the main experiment without additional sampling.

Table 5: Fixed model configurations for the main five-fold comparison. Hyperparameters are held constant across folds and datasets. Training-fold sample weights are used for the hierarchy stages and non-stacking flat baselines, whereas the flat stacking ensemble is fitted without sample weights.

Model	Key Hyperparameters	Tuning/Early Stopping
Proposed stages (XGBoost binary/multiclass)	n_estimators=300, learning_rate=0.05, max_depth=6, subsample=0.9, colsample_bytree=0.9, Stage 1 attack threshold 0.5, minority-branch weight strength 1.0	Fixed values, no early stopping
Flat XGBoost	n_estimators=300, learning_rate=0.05, max_depth=6, subsample=0.9, colsample_bytree=0.9	Fixed values, no early stopping
Flat LightGBM	n_estimators=500, learning_rate=0.05, num_leaves=31, min_child_samples=20, subsample=0.9, colsample_bytree=0.9	Fixed values, no early stopping
Flat Random Forest	n_estimators=300, max_depth=24, min_samples_leaf=2, max_samples=0.7	Fixed values, not applicable to non-boosting models
Flat Extra Trees	n_estimators=300, max_depth=24, min_samples_leaf=2, max_samples=0.7, bootstrap=True	Fixed values, not applicable to non-boosting models
Flat Stacking Ensemble	Base learners: Random Forest, Extra Trees, LightGBM, and XGBoost. Meta-learner: logistic regression with max_iter=2000	Internal 3-fold CV for meta-learner fitting with n_jobs=1. Base learners do not use early stopping. Sample weights are not applied

All main comparisons use stratified five-fold cross-validation with random seed 42. The same fold assignments are reused for all baselines and proposed variants within a dataset. Min-max scaling is fitted only on the training fold and then applied to the corresponding test fold. Model training, stage-specific weighting, and diagnostic computations are fold-local. The primary metric is Macro-F1 because it gives equal importance to rare and frequent families. Accuracy and Weighted-F1 are also reported, together with per-family precision, recall, and F1.

For a set of C family-level classes, including Normal where present, Macro-F1 is the unweighted mean of the per-class F1 scores, $\text{Macro-F1} = \frac{1}{C} \sum_{c=1}^C F1_c$, where C is the number of classes, c indexes a class, and $F1_c$ is that class's F1 score. Every family, whether frequent or rare, therefore contributes equally regardless of its sample count. Weighted-F1 instead weights each class's F1 score by its support, $\text{Weighted-F1} = \sum_{c=1}^C \frac{n_c}{n} F1_c$, where n_c is the number of test samples in class c and n is the total number of test samples. It reflects performance on the observed class distribution and is less sensitive to rare-family behavior.

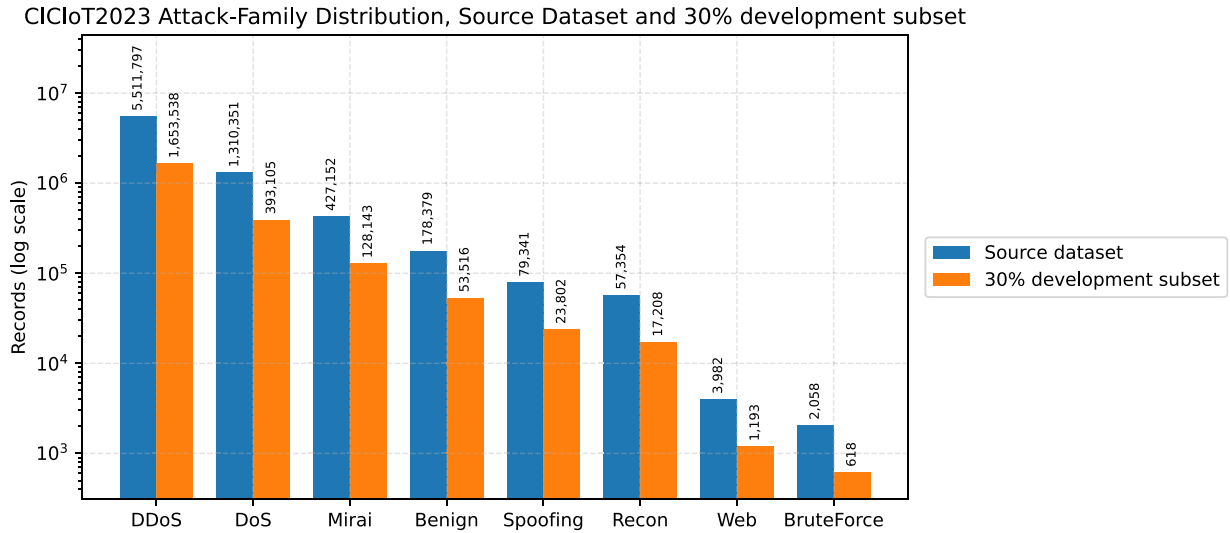


Figure 2: CICIoT2023 family distribution. Blue bars show counts in the source dataset, and orange bars show the stratified 30% family-level development subset used in the main five-fold experiment. Counts are shown on a logarithmic axis because family sizes differ by several orders of magnitude.

3.6 Computational Environment

All training and inference runtimes reported in this study were measured on a Windows 64-bit workstation with an Intel Core i7-11800H CPU at 2.30 GHz, 8 physical cores, 16 logical processors, and 16 GB RAM. All models were evaluated on the same machine using the same fold assignments, preprocessing protocol, and metric definitions. Therefore, the runtime results are valid only for the specific computer used in this experiment. The actual training and inference times should not be interpreted as universal values, since runtimes may be faster or slower on different hardware.

3.7 Baselines, Runtime, and Statistical Testing

Our proposed method is compared with five flat classifiers named LightGBM, XGBoost, Random Forest, Extra Trees, and a stacking ensemble. All baselines are trained directly on the same family-level labels used by the proposed model. Runtime is recorded for each model and fold, which includes training time and inference time. Because the proposed method is intended for IoT IDS deployment, the comparison emphasizes the tradeoff between family-level Macro-F1 and inference cost. Paired fold-level tests compare inference time between the proposed hierarchy and each baseline. The results are reported alongside the five-fold estimates of family-level Macro-F1.

Inference time is measured as wall-clock model-prediction time after fold-specific feature scaling has been applied to the test fold. For attack samples, the hierarchy includes Stage 1, Stage 2, and one branch-specific Stage 3 classifier. Normal samples exit after Stage 1. Feature scaling is excluded because it is shared preprocessing. Table 6 reports the 95% confidence interval of the mean fold-level time difference, Cohen's d_z paired effect size, and the Wilcoxon signed-rank p -value. Paired t -tests provide a parametric assessment of the same fold-level differences. With five folds, a consistent direction across all folds gives an exact two-sided Wilcoxon p -value of 0.0625, which does not cross the conventional 0.05 threshold.

Table 6: Paired fold-level inference-time comparison of the proposed hierarchy and each flat baseline. Δ denotes proposed minus baseline test-fold inference time in seconds, and the 95% confidence interval is for the mean difference across five folds. Negative values favor the proposed hierarchy. Cohen's d_z is the standardized paired difference, and Wilcoxon p -values are exact two-sided values.

Dataset	Baseline	Mean Δ Infer (s)	95% CI	Cohen's d_z	Wilcoxon p
CICIoT2023	Flat LightGBM	-24.06	[-25.02, -23.10]	-31.22	0.0625
	Flat XGBoost	-2.77	[-3.90, -1.64]	-3.04	0.0625
	Flat Random Forest	-0.90	[-1.04, -0.76]	-8.09	0.0625
	Flat Extra Trees	-2.10	[-2.40, -1.81]	-8.77	0.0625
	Flat Stacking	-33.33	[-37.68, -28.97]	-9.51	0.0625
Edge-IIoTset	Flat LightGBM	-1.98	[-2.08, -1.89]	-25.20	0.0625
	Flat XGBoost	-0.19	[-0.23, -0.15]	-6.28	0.0625
	Flat Random Forest	-0.10	[-0.13, -0.08]	-4.76	0.0625
	Flat Extra Trees	-0.09	[-0.12, -0.07]	-4.66	0.0625
	Flat Stacking	-3.10	[-3.35, -2.85]	-15.58	0.0625

3.8 Leakage-Control Safeguards

The evaluation pipeline enforces the following safeguards. The fold-test set is not used for scaling, sample-weight computation, model fitting, feature importance estimation, or stage diagnostics. Family normalization maps raw dataset labels to attack-family labels. Within each cross-validation fold, training-label counts define the routing taxonomy. Neither step uses test-fold feature values or labels to construct the fold's taxonomy. All model artifacts, fold metrics, classification reports, confusion matrices, runtime logs, feature-importance tables, and stage diagnostics are saved separately for reproducibility. The accompanying replication package contains the code, data, configuration files, and supporting results needed to reproduce the evaluation.

Algorithm 1 separates the evaluation into a one-time setup phase and a fold-local phase. The routing taxonomy, scaling, sample weighting, model fitting, and stage diagnostics are recomputed independently inside each fold using only that fold's training partition.

Algorithm 1: Fold-local family-aware hierarchical training and evaluation

Require: raw dataset D with attack-family labels, fold count K , seed s , majority-coverage threshold τ

Ensure: per-fold metrics, confusion matrices, stage diagnostics, and saved artifacts

- 1: **Setup (once, before cross-validation):**
- 2: Build K stratified folds $\{(train_k, test_k)\}_{k=1}^K$ from D using seed s
- 3: **for** $k = 1$ **to** K **do** ▷ Fold-local phase
- 4: Count labels in $D[train_k]$ and sort attack families by descending frequency
- 5: $G_k \leftarrow$ smallest set of training-fold attack families whose combined count covers $\geq \tau$ of training-fold attack samples
- 6: Assign normal or benign labels to NORMAL
- 7: Assign each attack-family label to MAJORITY if it belongs to G_k , and to MINORITY otherwise
- 8: Fit scaler ϕ_k on $D[train_k]$ only; transform $D[train_k]$ and $D[test_k]$ with ϕ_k
- 9: Compute sample weights w_k from $D[train_k]$ labels only (no test-fold information)
- 10: Fit Stage 1 (Normal vs. Attack) on $D[train_k]$ with weights w_k

(Continued)

Algorithm 1 (continued)

```

11: Fit Stage 2 (Majority vs. Minority) on attack rows of  $D[train_k]$  with stage-local weights
12: Fit Stage 3A (majority-family classifier) on majority-branch rows of  $D[train_k]$ 
13: Fit Stage 3B (minority-family classifier) on minority-branch rows of  $D[train_k]$ 
14: for each  $x \in D[test_k]$  do
15:   if Stage 1 predicts Normal then
16:      $\hat{y} \leftarrow \text{Normal}$ 
17:   else if Stage 2 predicts Majority then
18:      $\hat{y} \leftarrow \text{Stage 3A prediction for } x$ 
19:   else
20:      $\hat{y} \leftarrow \text{Stage 3B prediction for } x$ 
21:   end if
22: end for
23: Compute fold metrics, confusion matrix, and per-stage diagnostics from  $D[test_k]$  predictions only
24: Record training and inference runtime and memory statistics for fold  $k$ 
25: Save fold model bundle, figures, and diagnostic tables for fold  $k$ 
26: end for
27: Aggregate per-fold metrics into mean  $\pm$  standard deviation summaries and write run manifest

```

4 Results and Discussion

This section reports the family-level IDS results and discusses the evidence provided by the main comparison, runtime analysis, stage diagnostics, confusion matrices, feature importance, robustness testing, and ablation studies. The results are interpreted jointly in terms of family-level Macro-F1 and inference time. The results should be interpreted as attack-family classification results, because the final output is a specific family label, while the Majority Attack and Minority Attack groups are used only for internal routing and are not final prediction classes.

4.1 Main Baseline Comparison

Table 7 reports the main five-fold results. On CICIOT2023, the proposed XGBoost hierarchy achieved a Macro-F1 score of 0.8380 ± 0.0060 , Accuracy of 0.9942, and Weighted-F1 of 0.9940. Flat Random Forest achieved the best aggregate Macro-F1 on the CICIOT2023 development split (0.8423 ± 0.0053), with only a small Macro-F1 margin over the proposed model. However, the proposed hierarchy required lower training time and lower inference time, showing a stronger performance-efficiency tradeoff. The proposed model also outperformed flat LightGBM, flat XGBoost, flat Extra Trees, and flat stacking in Macro-F1.

On Edge-IIoTset, the proposed model achieved an Accuracy of 0.9487, a Macro-F1 score of 0.9456 ± 0.0015 , and a Weighted-F1 of 0.9494. These results outperform all individual flat baselines in Macro-F1 and remain close to the flat stacking ensemble, which achieved the highest Accuracy, Macro-F1, and Weighted-F1. Although the proposed model required higher training time than some individual baselines, training is typically performed offline. More importantly for deployment, the proposed model achieved the lowest inference time, indicating stronger operational efficiency during real-time IDS use.

Table 7: Main five-fold family-level comparison on CICIOT2023 and Edge-IIoTset. Bold values indicate the best result within each dataset block.

Dataset	Model	Accuracy	Macro-F1	Weighted-F1	Train s/Fold	Infer s/Fold
CICIOT2023	Flat LightGBM	0.9945	0.7891 ± 0.0067	0.9947	169.1	26.78
	Flat XGBoost	0.9949	0.8029 ± 0.0049	0.9952	309.9	5.49
	Flat Random Forest	0.9954	0.8423 ± 0.0053	0.9953	218.3	3.62
	Flat Extra Trees	0.7434	0.6880 ± 0.0055	0.7697	216.1	4.83
	Flat Stacking	0.9956	0.7780 ± 0.0254	0.9954	2870.6	36.05
	Proposed XGBoost	0.9942	0.8380 ± 0.0060	0.9940	129.4	2.72
Edge-IIoTset	Flat LightGBM	0.9449	0.9399 ± 0.0021	0.9458	14.3	2.26
	Flat XGBoost	0.9436	0.9414 ± 0.0014	0.9446	26.5	0.47
	Flat Random Forest	0.9108	0.9093 ± 0.0008	0.9137	9.7	0.38
	Flat Extra Trees	0.9021	0.9060 ± 0.0016	0.9049	5.3	0.37
	Flat Stacking	0.9529	0.9481 ± 0.0012	0.9529	183.5	3.38
	Proposed XGBoost	0.9487	0.9456 ± 0.0015	0.9494	15.1	0.28

4.2 Per-Family Behavior

The per-family results show why Macro-F1 is more informative than accuracy in these datasets. In CICIOT2023, frequent attack families such as DDoS, DoS, and Mirai are classified almost perfectly by the proposed approach, with F1 scores near 1.0000. The main weakness is the low-frequency minority branch. BruteForce achieved an F1 score of 0.6400 and Web achieved an F1 score of 0.5250. As shown in Table 8, these are the highest BruteForce and Web F1 scores among the evaluated models. This supports the usefulness of the hierarchy for rare-family behavior even though Flat Random Forest remains slightly higher on aggregate CICIOT2023 Macro-F1. Fig. 3 visualizes the corresponding CICIOT2023 per-family precision, recall, and F1 patterns.

Table 8: CICIOT2023 rare-family F1 comparison for BruteForce and Web. Values are five-fold means from the family-level evaluation. Bold values indicate the highest F1 score in each column.

Model	BruteForce F1	Web F1
Flat LightGBM	0.4363	0.2862
Flat XGBoost	0.3732	0.3923
Flat Random Forest	0.6300	0.4648
Flat Extra Trees	0.3438	0.3200
Flat Stacking	0.1814	0.3875
Proposed XGBoost	0.6400	0.5250

BruteForce and Web have limited support even before cross-validation. Table 9 reports their fold-level test-set counts and F1 scores alongside the five-fold means. Each test partition contains only 123–124 BruteForce and 238–239 Web samples on the 30% development split. BruteForce F1 ranges from 0.5761 to 0.6866, while Web F1 ranges from 0.4847 to 0.5565. These small per-fold counts warrant caution when interpreting the five-fold means, particularly for BruteForce.

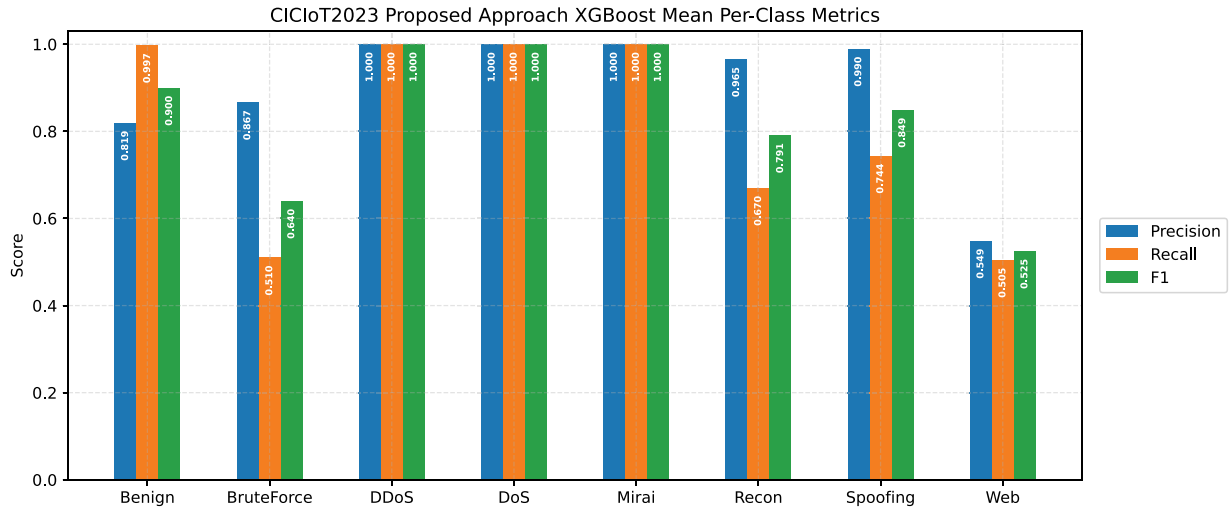


Figure 3: Mean per-family precision, recall, and F1 for the proposed XGBoost hierarchy on CICIoT2023. The rare BruteForce and Web families form the clearest bottleneck.

Table 9: CICIoT2023 fold-level test-set support and F1 for BruteForce and Web under the proposed hierarchy. The five test folds contain 618 BruteForce and 1193 Web samples in total.

Family	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
BruteForce test n	123	123	124	124	124
BruteForce F1	0.6763	0.6497	0.6866	0.6114	0.5761
Web test n	239	239	239	238	238
Web F1	0.4847	0.5022	0.5393	0.5425	0.5565

The aggregate confusion matrix across five folds shows a consequential limitation for the rare families. BruteForce recall is 0.51 (315 of 618 true samples detected) and precision is 0.87 (315 of 364 predicted samples correct). Web recall is 0.50 (602 of 1193) and precision is 0.55 (602 of 1099). Of the missed samples, 92% of BruteForce false negatives (279 of 303) and 87% of Web false negatives (517 of 591) are predicted as Benign. Consequently, roughly half of BruteForce and Web attacks in this evaluation produce no alert. This limitation is not captured fully by Macro-F1 alone. Deployment for these two families would require complementary anomaly-based or rule-based safeguards.

On Edge-IIoTset, the proposed model is more balanced across families. It achieved strong F1 scores for Normal (0.9974), DDoS (0.9668), MITM (0.9987), Backdoor (0.9834), Reconnaissance (0.9600), Uploading (0.9186), Ransomware (0.9196), Web Injection (0.8860), and Password (0.8798). The lower Edge-IIoTset classes are still substantially stronger than the weakest CICIoT2023 rare families, which helps explain why the hierarchy is more competitive on Edge-IIoTset. Fig. 4 summarizes the Edge-IIoTset per-family metric distribution.

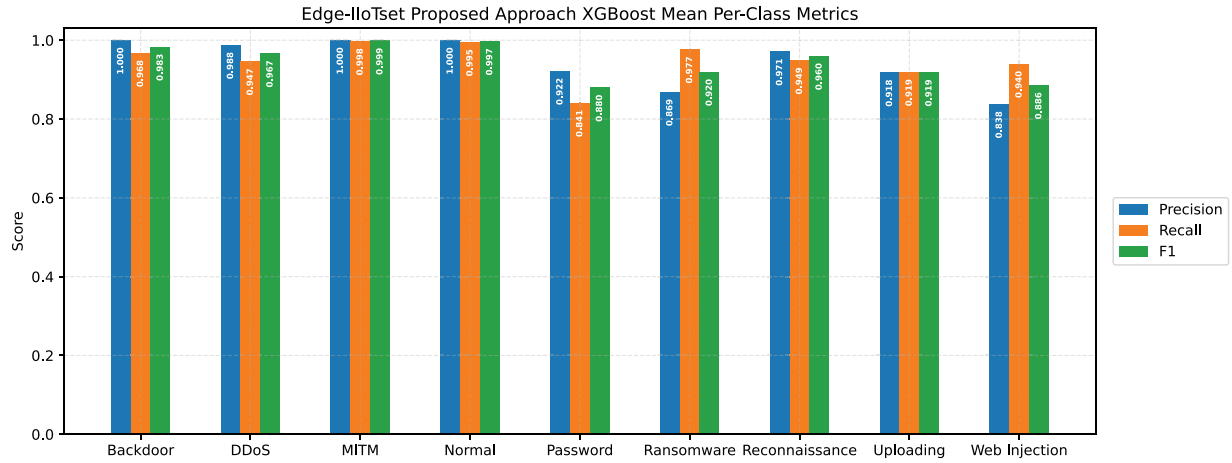


Figure 4: Mean per-family precision, recall, and F1 for the proposed XGBoost hierarchy on Edge-IIoTset.

4.3 Stage Diagnostics

Table 10 reports the internal stage diagnostics for the proposed model. On CICIOT2023, Stage 2 and Stage 3A are almost perfect, while Stage 3B reaches only 0.7963 Macro-F1. This confirms that the main bottleneck is not DDoS, DoS, or Mirai detection; it is the minority-family classifier that must distinguish smaller and more heterogeneous families such as BruteForce, Recon, Spoofing, and Web. On Edge-IIoTset, Stage 3B reaches 0.9554 Macro-F1, indicating that the minority branch is much more learnable in that dataset. Figs. 5 and 6 provide visual summaries of these conditional stage diagnostics.

Table 10: Final end-to-end family-level performance and conditional stage diagnostics for the proposed XGBoost hierarchy. Values are reported as mean $\pm$ standard deviation across the five cross-validation folds. Bold values identify the final end-to-end family-level Macro-F1 for each dataset. The stage rows report conditional diagnostics on the indicated sample subsets.

Dataset	Evaluation Level	Test Samples Used	Macro-F1
CICIOT2023	Final end-to-end family prediction	All held-out samples	0.8380 $\pm$ 0.0060
	Stage 1 Normal vs. Attack	All held-out samples	0.9485 $\pm$ 0.0012
	Stage 2 Majority vs. Minority	True attack samples only	0.9998 $\pm$ 0.0000
	Stage 3A Majority Family Classification	True majority-branch attack samples only	1.0000 $\pm$ 0.0000
	Stage 3B Minority Family Classification	True minority-branch attack samples only	0.7963 $\pm$ 0.0077
Edge-IIoTset	Final end-to-end family prediction	All held-out samples	0.9456 $\pm$ 0.0015
	Stage 1 Normal vs. Attack	All held-out samples	0.9985 $\pm$ 0.0003
	Stage 2 Majority vs. Minority	True attack samples only	0.9665 $\pm$ 0.0010
	Stage 3A Majority Family Classification	True majority-branch attack samples only	0.9963 $\pm$ 0.0004
	Stage 3B Minority Family Classification	True minority-branch attack samples only	0.9554 $\pm$ 0.0016

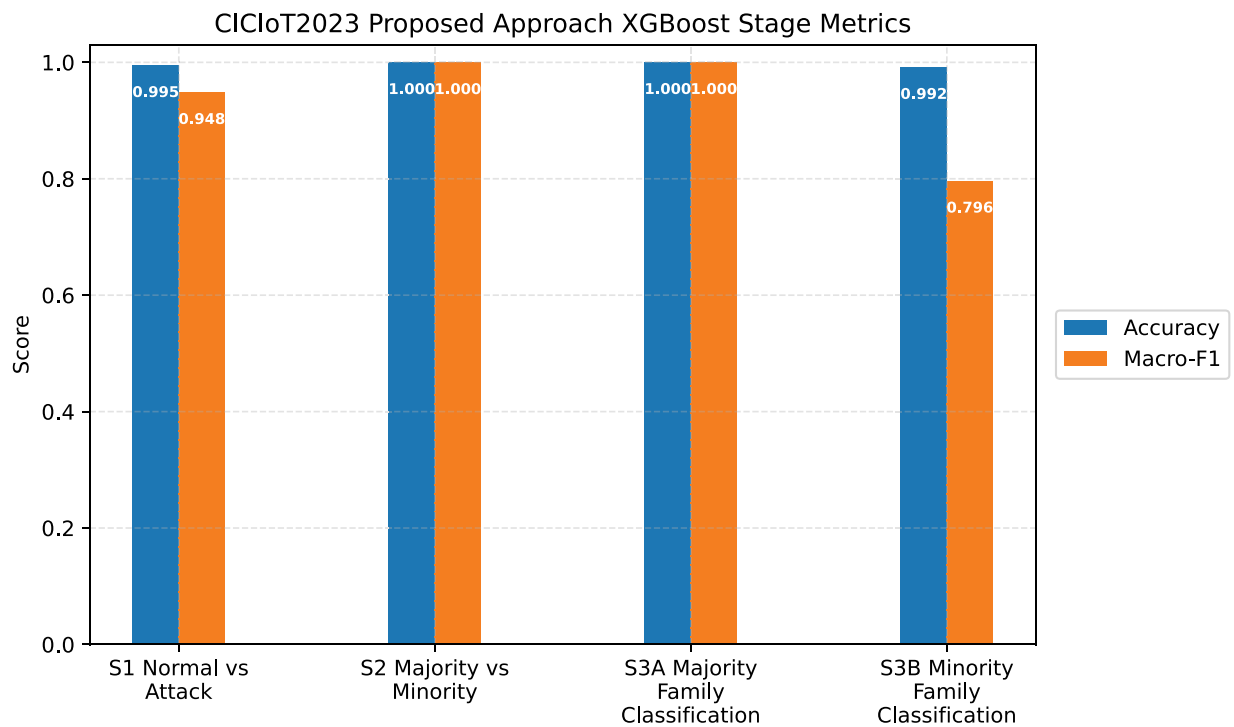


Figure 5: Stage diagnostics for the proposed XGBoost hierarchy on CICIoT2023. Performance is limited mainly by the Stage 3B minority-family branch.

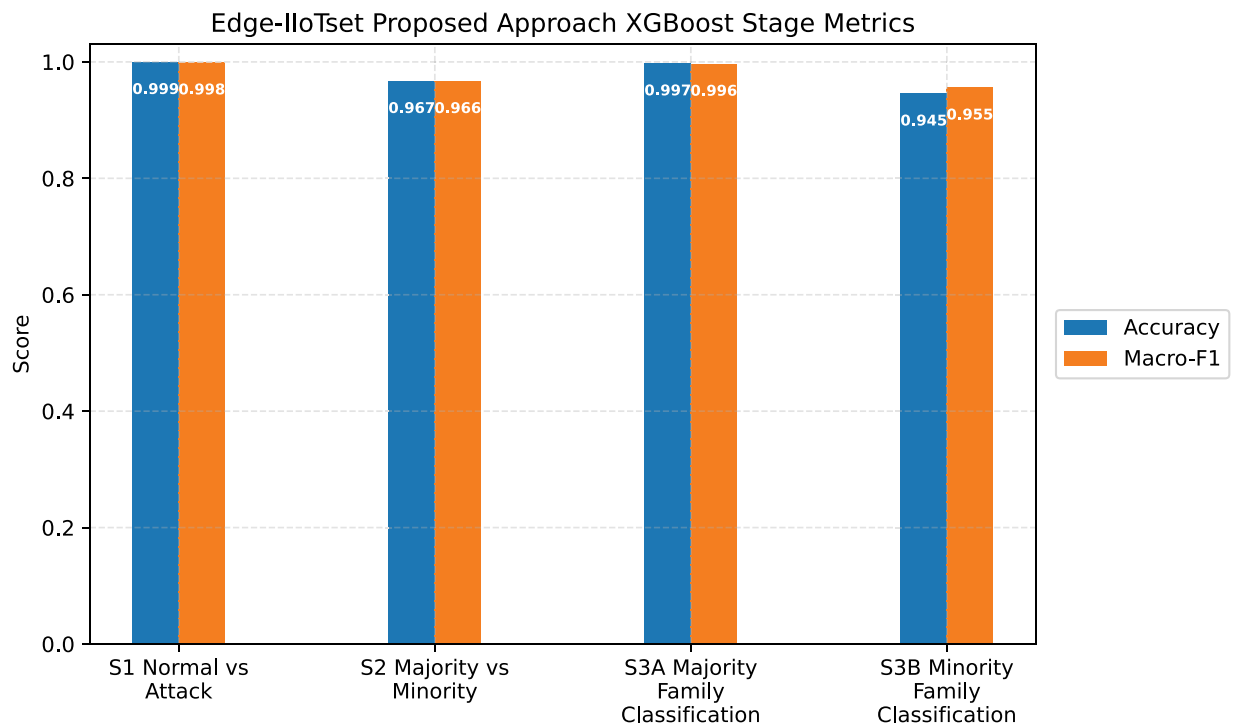


Figure 6: Stage diagnostics for the proposed XGBoost hierarchy on Edge-IIoTset. The hierarchy remains strong across the four stages.

4.4 Confusion Matrix and Feature-Importance Analysis

Figs. 7 and 8 provide row-normalized aggregate confusion matrices across the five folds. Each row shows the distribution of predicted labels conditional on one true family, which makes rare-family errors visible despite the large difference in class frequencies. CICIoT2023 errors are concentrated in low-frequency minority families, while the high-volume DDoS branch has few errors. Edge-IIoTset has a more balanced error profile, although confusions remain among related attack families.

CICIoT2023 Proposed Approach XGBoost 5-Fold CV Confusion Matrix (Row-Normalized)

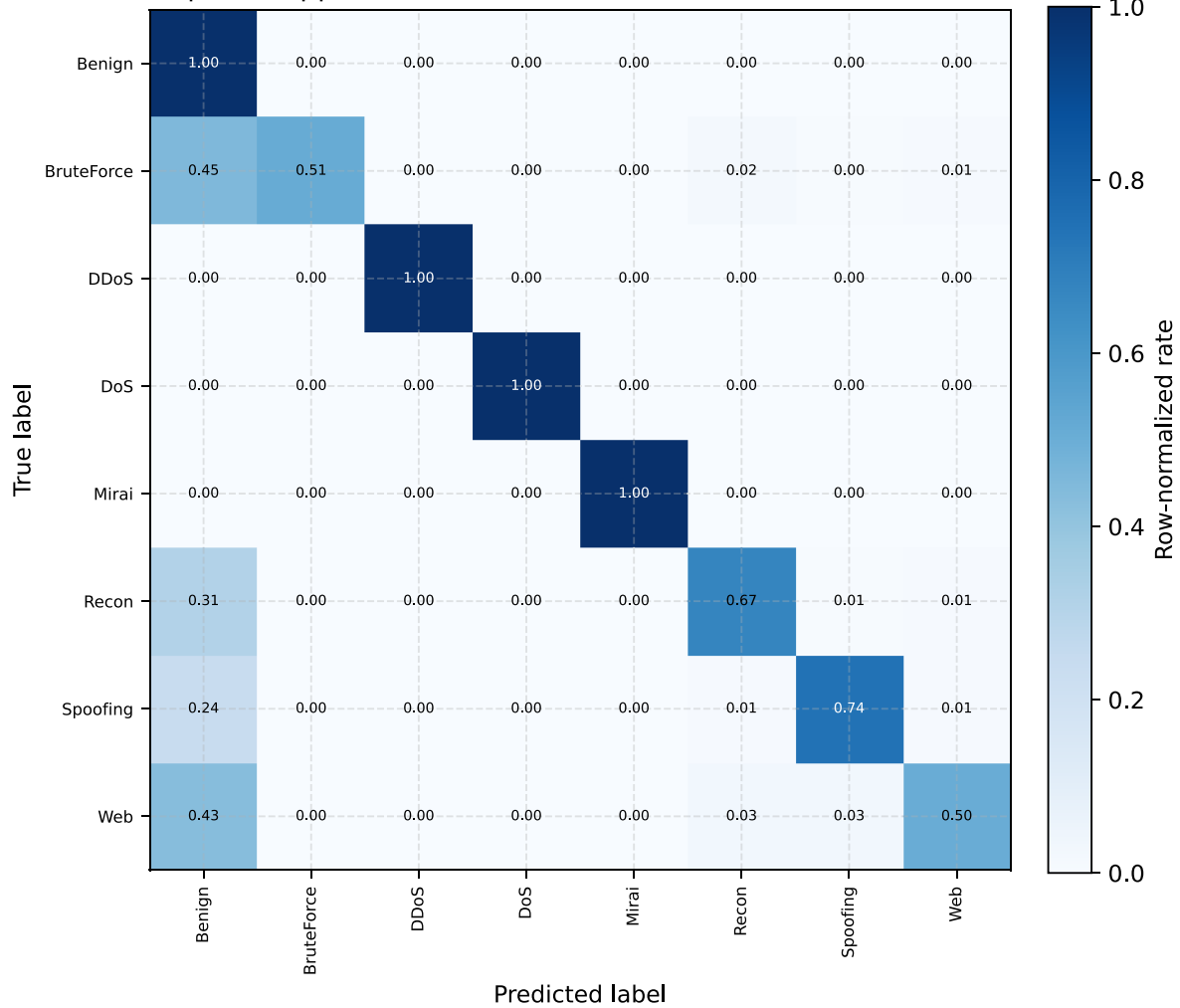


Figure 7: Row-normalized confusion matrix for the proposed XGBoost hierarchy on CICIoT2023 across all folds. Values are normalized rates from 0 to 1 within each true class.

Edge-IIoTset Proposed Approach XGBoost 5-Fold CV Confusion Matrix (Row-Normalized)

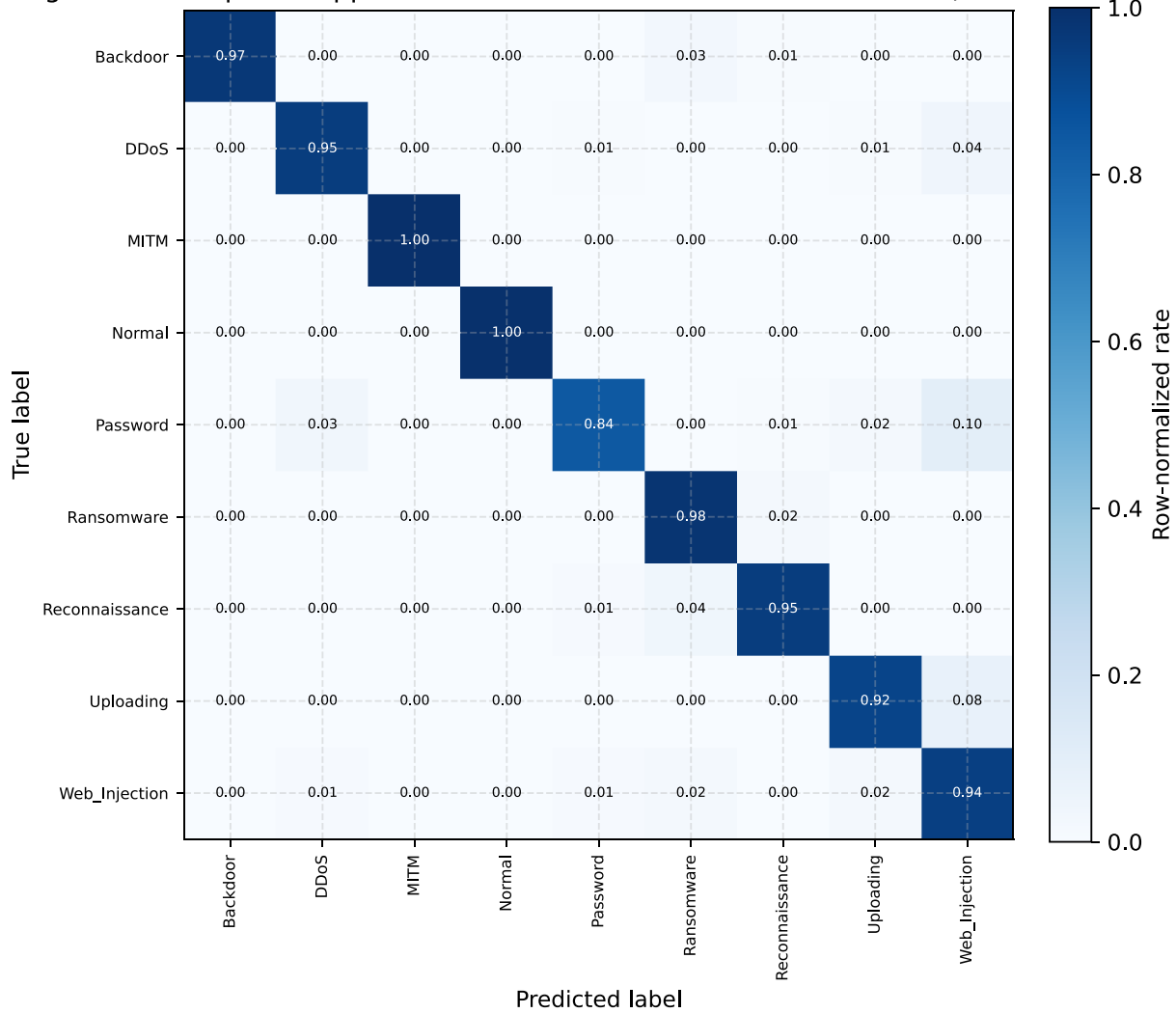


Figure 8: Row-normalized confusion matrix for the proposed XGBoost hierarchy on Edge-IIoTset across all folds. Values are normalized rates from 0 to 1 within each true class.

Feature-importance analysis in Figs. 9 and 10 provides additional interpretability. The most influential features differ across datasets, showing that the proposed framework does not rely on a single fixed signature. This is important for IoT IDS deployment because different IoT environments expose different protocol, flow, and packet-level characteristics.

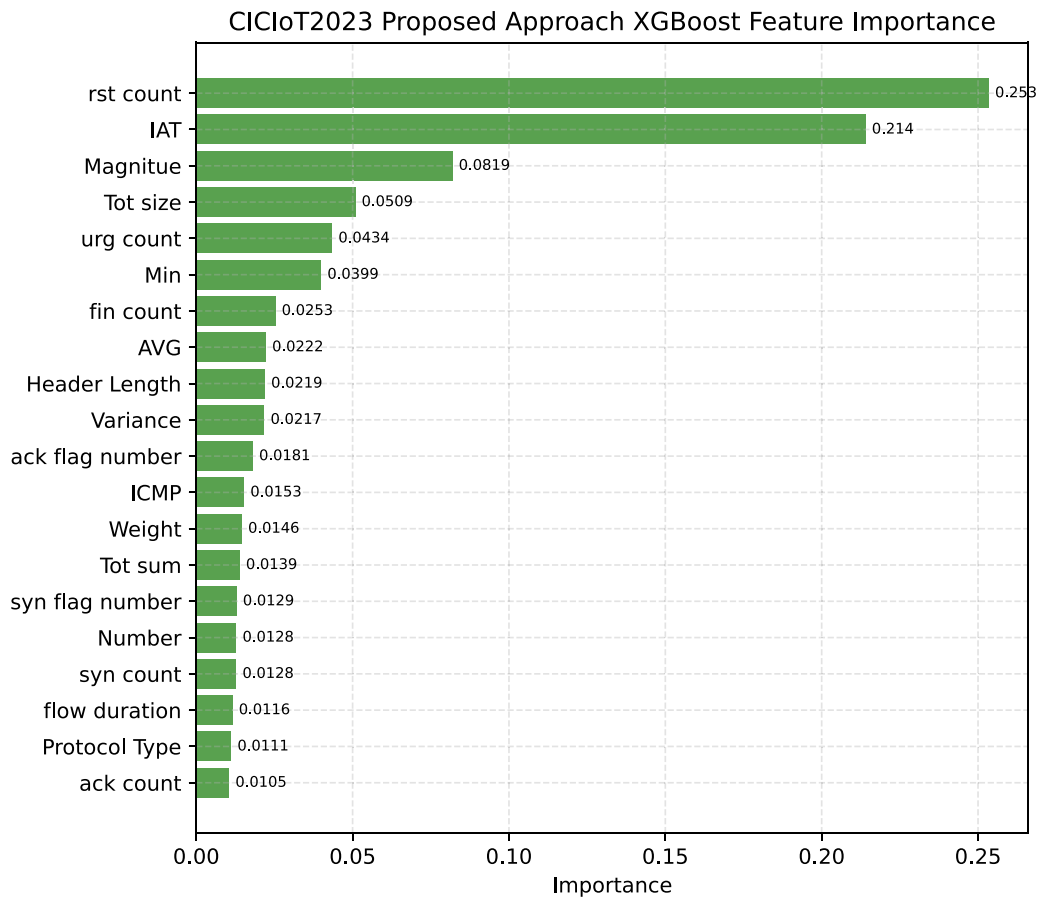


Figure 9: Feature-importance summary for the proposed XGBoost hierarchy on CICIoT2023. Importance values are aggregated from the fitted fold models. Feature names are reported exactly as they appear in the source CICIoT2023 dataset, including its original spelling (e.g., “Magnitue”), to preserve traceability to the raw data.

4.5 Performance-Efficiency Tradeoff

Because the proposed framework targets IoT IDS deployment, the main comparison should be interpreted jointly in terms of family-level Macro-F1 and inference cost. [Figs. 11](#) and [12](#) visualize this tradeoff for each dataset, where points closer to the upper-left have higher Macro-F1 and lower inference time. [Table 11](#) shows how the proposed XGBoost hierarchy compares with each flat baseline. The Macro-F1 difference is calculated as Proposed minus Baseline. A positive value means that the proposed approach achieved a higher Macro-F1. The inference-time difference is also calculated as Proposed minus Baseline. A negative value means that the proposed approach was faster. The inference-time p -value is calculated using paired inference times across the folds and is used only to support the runtime-efficiency claim.

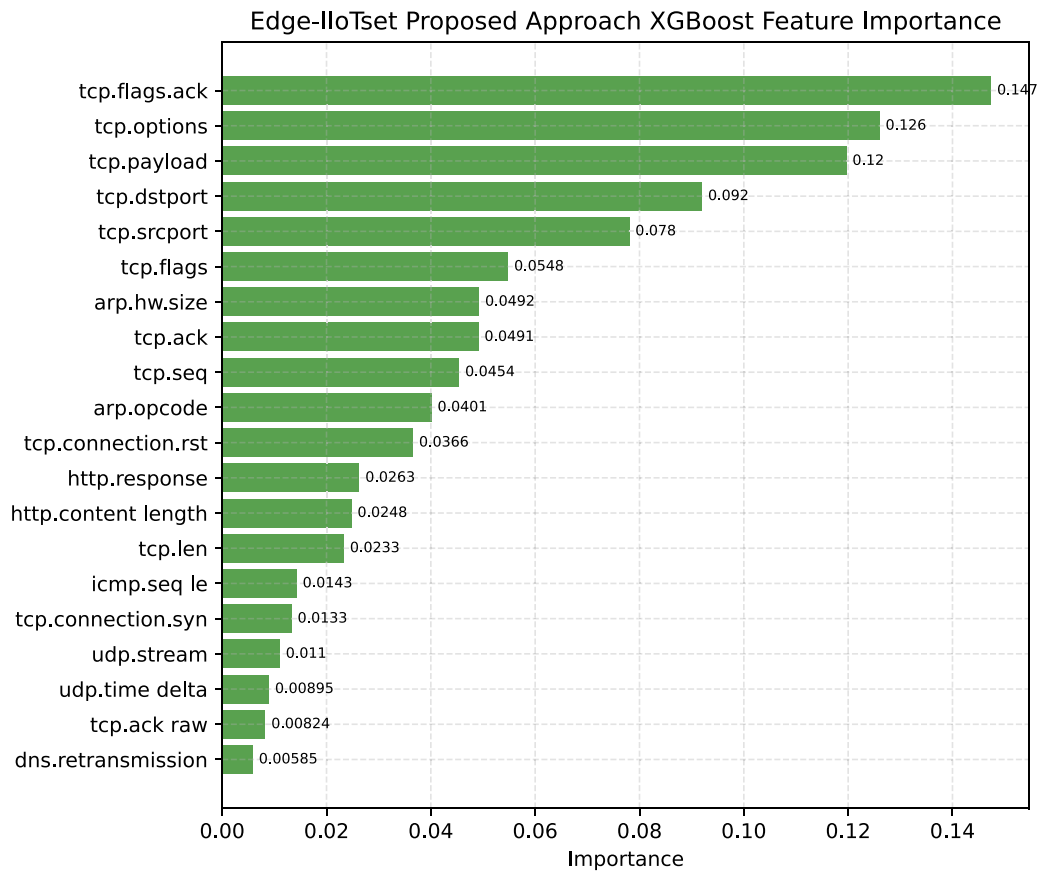


Figure 10: Feature-importance summary for the proposed XGBoost hierarchy on Edge-IIoTset.

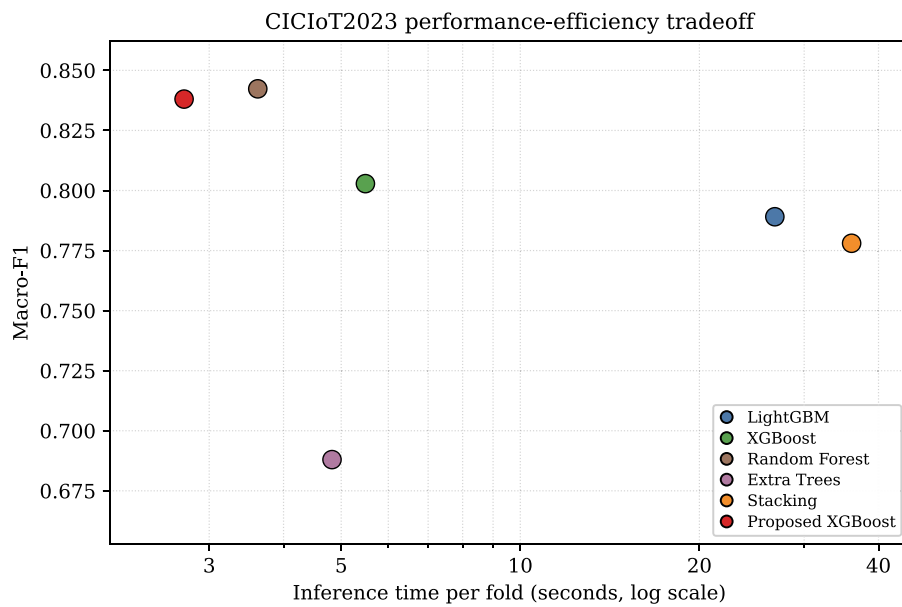


Figure 11: Performance-efficiency Pareto plot on CICIoT2023. The x -axis reports inference time per fold on a logarithmic scale, and the y -axis reports Macro-F1. The legend identifies model colors; points closer to the upper-left indicate a better performance-efficiency tradeoff.

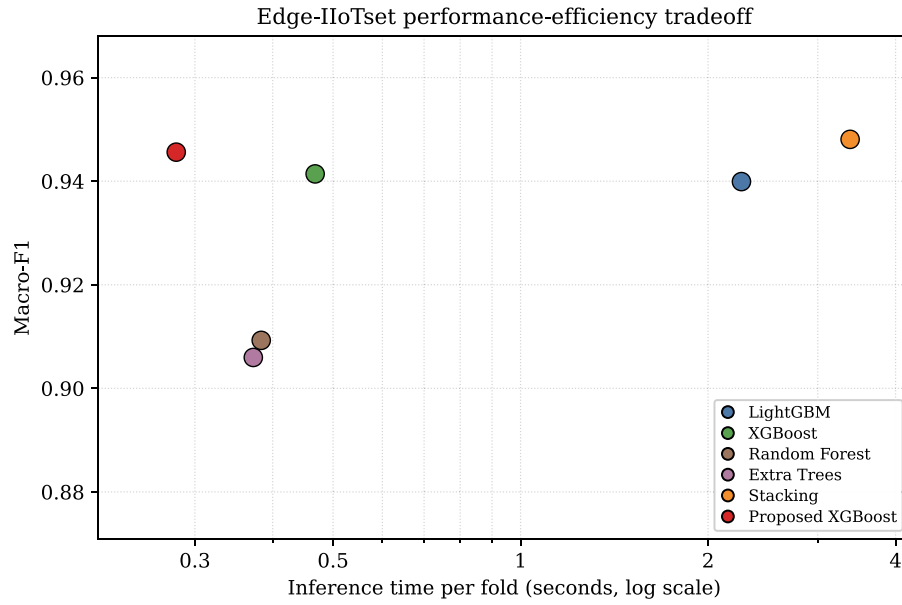


Figure 12: Performance-efficiency Pareto plot on Edge-IIoTset. The x -axis reports inference time per fold on a logarithmic scale, and the y -axis reports Macro-F1. The legend identifies model colors; points closer to the upper-left indicate a better performance-efficiency tradeoff.

Table 11: Performance-efficiency tradeoff between the proposed XGBoost hierarchy and flat baselines. Speedup is computed as Baseline inference time divided by Proposed inference time. Inference-time p -values are computed using paired fold-level t -tests; negative Δ infer values indicate faster inference for the proposed approach. In the Outcome column, F1 $\uparrow$ /F1 $\downarrow$ indicates higher/lower Macro-F1 for the proposed approach, and Infer $\downarrow^*$ indicates significantly lower inference time ($p < 0.05$).

Dataset	Baseline	Δ Macro-F1	Δ infer s/Fold	Speedup	Infer p	Outcome
CICIoT2023	Flat LightGBM	+0.0490	-24.06	9.84 $\times$	2.52×10^{-7}	F1 $\uparrow$, Infer $\downarrow^*$
	Flat XGBoost	+0.0352	-2.77	2.02 $\times$	2.45×10^{-3}	F1 $\uparrow$, Infer $\downarrow^*$
	Flat Random Forest	-0.0043	-0.90	1.33 $\times$	5.50×10^{-5}	F1 $\downarrow$, Infer $\downarrow^*$
	Flat Extra Trees	+0.1501	-2.10	1.77 $\times$	3.98×10^{-5}	F1 $\uparrow$, Infer $\downarrow^*$
	Flat Stacking	+0.0600	-33.32	13.24 $\times$	2.89×10^{-5}	F1 $\uparrow$, Infer $\downarrow^*$
Edge-IIoTset	Flat LightGBM	+0.0057	-1.98	8.08 $\times$	5.94×10^{-7}	F1 $\uparrow$, Infer $\downarrow^*$
	Flat XGBoost	+0.0042	-0.19	1.67 $\times$	1.49×10^{-4}	F1 $\uparrow$, Infer $\downarrow^*$
	Flat Random Forest	+0.0363	-0.10	1.37 $\times$	4.43×10^{-4}	F1 $\uparrow$, Infer $\downarrow^*$
	Flat Extra Trees	+0.0396	-0.09	1.33 $\times$	4.80×10^{-4}	F1 $\uparrow$, Infer $\downarrow^*$
	Flat Stacking	-0.0025	-3.10	12.07 $\times$	4.05×10^{-6}	F1 $\downarrow$, Infer $\downarrow^*$

On CICIoT2023, paired t -tests indicate lower inference time for the proposed approach than for every baseline. The proposed approach also improves Macro-F1 over flat LightGBM, flat XGBoost, flat Extra Trees, and flat stacking. Flat Random Forest remains slightly higher in Macro-F1 and has higher inference time. On Edge-IIoTset, the proposed approach improves Macro-F1 over all individual flat baselines and provides the fastest inference time, while flat stacking remains slightly higher in Macro-F1 and is slower at inference. Together, these results characterize the relationship between family-level Macro-F1 and inference time for the evaluated models.

4.6 Inference Efficiency and Resource Use

Table 12 summarizes five-fold workstation measurements of inference latency, throughput, serialized model size, and peak process resident-set size (RSS). Serialized size represents the complete fitted model bundle, including stage classifiers, encoders, and preprocessing objects, saved in the same format for every model. Peak RSS is the maximum resident memory recorded during training or inference. It includes the loaded dataset and other process state, so it is not an isolated measure of model memory. The proposed XGBoost hierarchy has the lowest latency and highest throughput on both datasets. Its serialized size is comparable to flat XGBoost and substantially smaller than the other flat baselines.

Table 12: Five-fold mean inference-efficiency and resource-use measurements for the proposed XGBoost hierarchy and flat baselines. CICIOT2023 uses the 30% development split, and Edge-IIoTset uses the full processed dataset. Latency and throughput are derived from each fold's inference time and test-set size. Serialized model size and peak process resident-set size (RSS) are reported in decimal MB.

Dataset	Model	Latency (μ s)	Throughput (/s)	Size (MB)	Train RSS (MB)	Infer RSS (MB)
CICIOT2023	Proposed XGBoost	6.40	156,480	7.18	1534.6	863.5
	Flat LightGBM	64.38	15,542	13.95	1229.4	649.7
	Flat XGBoost	13.43	74,476	7.73	1046.4	599.0
	Flat Random Forest	9.70	105,361	493.04	1506.1	1033.6
	Flat Extra Trees	12.66	80,055	595.38	1419.9	1096.7
	Flat Stacking	76.79	13,071	885.29	2130.8	1618.8
Edge-IIoTset	Proposed XGBoost	9.03	111,613	5.61	485.3	445.0
	Flat LightGBM	63.05	15,866	13.03	597.9	559.9
	Flat XGBoost	14.16	70,764	5.48	578.2	542.6
	Flat Random Forest	11.91	84,342	119.36	626.6	626.7
	Flat Extra Trees	12.54	79,954	142.38	621.0	622.9
	Flat Stacking	97.68	10,239	281.02	922.9	798.0

4.7 Full CICIOT2023 Evaluation

Table 13 reports a three-fold comparison on the full 7,570,414-record CICIOT2023 dataset, including the proposed XGBoost hierarchy and flat XGBoost, Random Forest, and stacking baselines. The proposed hierarchy achieved a Macro-F1 of 0.8406, above flat XGBoost's 0.7928, with similar accuracy (0.9943 and 0.9946, respectively). Flat Random Forest achieved the highest Macro-F1 at 0.8585, while flat stacking achieved 0.8092. The development-split main comparison in Table 7 shows the same ordering, with flat Random Forest at 0.8423 and the proposed XGBoost hierarchy at 0.8380. The base-learner-controlled ablation in Table 14 also shows that the hierarchical Random Forest variant did not improve on flat Random Forest (0.8378 vs. 0.8423).

Table 13: Full-dataset CICIOT2023 three-fold comparison. Values are mean $\pm$ standard deviation across stratified folds.

Model	Accuracy	Macro-F1	Weighted-F1
Flat Random Forest	0.9958 $\pm$ 0.0001	0.8585 $\pm$ 0.0049	0.9958 $\pm$ 0.0001
Proposed XGBoost	0.9943 $\pm$ 0.0004	0.8406 $\pm$ 0.0029	0.9941 $\pm$ 0.0004
Flat Stacking Ensemble	0.9959 $\pm$ 0.0001	0.8092 $\pm$ 0.0575	0.9958 $\pm$ 0.0002
Flat XGBoost	0.9946 $\pm$ 0.0001	0.7928 $\pm$ 0.0010	0.9953 $\pm$ 0.0001

Table 14: Base-learner-controlled ablation. Positive deltas indicate that the hierarchical formulation improves over the corresponding flat learner.

Dataset	Base Learner	Flat Macro-F1	Hierarchical Macro-F1	Delta
CICIoT2023	LightGBM	0.7891	0.7917	+0.0026
	XGBoost	0.8029	0.8380	+0.0352
	Random Forest	0.8423	0.8378	-0.0045
	Extra Trees	0.6880	0.6611	-0.0269
Edge-IIoTset	LightGBM	0.9399	0.9426	+0.0027
	XGBoost	0.9414	0.9456	+0.0042
	Random Forest	0.9093	0.9159	+0.0066
	Extra Trees	0.9060	0.9140	+0.0081

4.8 Ablation Studies

The base-learner-controlled ablation is summarized in Table 14. On Edge-IIoTset, the hierarchical formulation improved Macro-F1 for all tested base learners. On CICIoT2023, the hierarchy improved LightGBM and XGBoost, with the strongest gain for XGBoost (+0.0352 Macro-F1), but it did not improve Random Forest or Extra Trees. This result is important because it prevents overclaiming: the benefit comes from the interaction between the hierarchical decomposition and a suitable base learner, not from hierarchy alone.

A frequency-aware alternative divided attack families into head, mid, and tail branches. Table 15 compares it with the proposed two-branch XGBoost hierarchy using the same five-fold assignments within each dataset. Its Macro-F1 was 0.8391 on CICIoT2023, compared with 0.8380 for the proposed hierarchy, and 0.9251 on Edge-IIoTset, compared with 0.9456. The alternative therefore did not provide a consistent improvement across the two datasets.

Table 15: Five-fold Macro-F1 comparison between the proposed two-branch XGBoost hierarchy and the frequency-aware head/mid/tail alternative. Values are mean $\pm$ standard deviation under the dataset-specific main protocols.

Dataset	Proposed Hierarchy	Frequency-Aware Alternative
CICIoT2023	0.8380 $\pm$ 0.0060	0.8391 $\pm$ 0.0039
Edge-IIoTset	0.9456 $\pm$ 0.0015	0.9251 $\pm$ 0.0009

4.9 Interpretation and Limitations

Within the hierarchy, Stage 3B is trained only on minority-branch samples, with its output space restricted to minority families and its class weights recomputed within that subset. This narrower classification problem limits competition from dominant classes during family prediction and may contribute to the higher BruteForce and Web F1 scores reported in Table 8. The effectiveness of this branch also depends on correct upstream routing and sufficient minority-branch data. Errors from Stage 1 or Stage 2 cannot be corrected downstream, and the limited BruteForce and Web counts constrain their absolute F1 scores.

In the five-fold main comparison, the strongest competing model on each dataset illustrates the accuracy-efficiency trade-off reported in Tables 7 and 11. On CICIoT2023, flat Random Forest exceeded the hierarchy's Macro-F1 by 0.0043, with values of 0.8423 and 0.8380, while its mean fold-level inference time was 3.62 s compared with 2.72 s for the hierarchy. On Edge-IIoTset, flat stacking exceeded the hierarchy's

Macro-F1 by 0.0025, with values of 0.9481 and 0.9456, while its mean inference time was 3.38 s compared with 0.28 s. The appropriate model therefore depends on whether the operating objective places greater weight on aggregate Macro-F1 or inference time.

The latency and throughput measurements in Table 12 describe batch prediction on scaled test folds under the reported workstation configuration. They exclude packet capture, flow construction, feature extraction, and communication overhead. The results characterize relative prediction cost under the evaluated hardware and software conditions, but they do not establish end-to-end real-time performance. The serialized model bundles were 7.18 MB for CICIOT2023 and 5.61 MB for Edge-IIoTset, smaller than the Random Forest, Extra Trees, and stacking bundles. Peak RSS is a process-level workstation measurement that includes the dataset and runtime state. Resource-constrained deployment requires direct measurement on the target device and traffic pipeline.

The frequency-derived grouping allows branch membership to follow the attack distribution in each training fold, so the taxonomy can adapt to datasets with different dominant families. A taxonomy learned from one distribution may change after a shift in labeled traffic. In Table 4, CICIOT2023 performance was unchanged from 40% to 70% coverage, while Edge-IIoTset reached the same highest observed Macro-F1 at 40% and 50%. Threshold selection may therefore need to be repeated for other data distributions.

The evaluation used offline stratified cross-validation on CICIOT2023 and Edge-IIoTset. Streaming operation, chronological validation, concept drift, and zero-day or previously unseen attack families were not evaluated. Future evaluation should combine temporal or streaming protocols with edge-hardware measurements and additional datasets. Fold-local feature pruning or dimensionality reduction could reduce input cost [28]. Tree-specific compression and low-precision tree inference are further options for reducing storage and prediction cost. Dynamic quantization has been studied for neural IDS models [29]. Comparable low-precision methods for tree ensembles require separate evaluation. Federated and continual learning are also relevant extensions when client heterogeneity, communication cost, privacy, and changing traffic distributions are evaluated explicitly.

5 Conclusions

This paper presented a family-aware hierarchical XGBoost framework for IoT intrusion detection. The majority/minority groups guide internal routing, while the final prediction remains an attack-family label. The method applies an audited empirical grouping rule, fold-local scaling and sample weighting, stage-specific training, per-stage diagnostics, and direct comparison with strong flat baselines.

The results show that the proposed approach is best characterized as an efficient and leakage-controlled family-level IDS. On CICIOT2023, it achieved Macro-F1 of 0.8380 on the stratified 30% development split, close to the best flat Random Forest result of 0.8423, while improving rare-family F1 for BruteForce and Web. On Edge-IIoTset, it achieved Macro-F1 of 0.9456, outperforming all individual flat baselines and approaching the flat stacking ensemble. Under the five-fold workstation protocol, the hierarchy had lower inference time than every evaluated flat baseline. In the full CICIOT2023 comparison, it improved on flat XGBoost by approximately 0.048 Macro-F1, while Random Forest achieved the highest Macro-F1 (0.8585 vs. 0.8406).

The study also identifies where the approach remains limited. CICIOT2023 minority-family classification, especially BruteForce and Web, remains the main bottleneck. Therefore, the contribution should not be interpreted as a universal replacement for all flat classifiers. Instead, it provides a reproducible hierarchical IDS design that improves the corresponding flat XGBoost learner, gives competitive performance against strong baselines, offers clear stage-level diagnostics, and has lower observed inference time under the reported workstation protocol. These conclusions are limited to the CICIOT2023 and Edge-IIoTset protocols

and workstation conditions examined here. Future work should validate the framework under temporal and resource-constrained deployment conditions, with particular attention to rare-family detection.

Acknowledgement: Not applicable.

Funding Statement: This Project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia, under grant no. (IPP: 1315-611-2025). The authors therefore acknowledge DSR with thanks for its technical and financial support.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Motab F. Alenezi and Fahad M. Alotaibi; methodology, Motab F. Alenezi and Fahad M. Alotaibi; software, Motab F. Alenezi; validation, Motab F. Alenezi, Fahad M. Alotaibi, Badraddin Alturki, Ahmad J. Tayeb, Abdulaziz A. Alsulami and Abdullah Alhejaili; formal analysis, Motab F. Alenezi and Ahmad J. Tayeb; investigation, Motab F. Alenezi; resources, Motab F. Alenezi; data curation, Motab F. Alenezi; writing—original draft preparation, Motab F. Alenezi, Fahad M. Alotaibi, Badraddin Alturki, Ahmad J. Tayeb, Abdulaziz A. Alsulami and Abdullah Alhejaili; writing—review and editing, Motab F. Alenezi, Fahad M. Alotaibi, Badraddin Alturki, Ahmad J. Tayeb, Abdulaziz A. Alsulami and Abdullah Alhejaili; visualization, Motab F. Alenezi; supervision, Abdulaziz A. Alsulami and Abdullah Alhejaili; project administration, Motab F. Alenezi and Fahad M. Alotaibi; funding acquisition, Motab F. Alenezi, Fahad M. Alotaibi and Abdullah Alhejaili. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used in this study are publicly available. Edge-IIoTset is available at Ref. [27]. CICIOT2023 is available at Ref. [26]. The replication package is available through Zenodo at <https://zenodo.org/records/20568228>. It contains the code, data, configuration files, and supporting results needed to reproduce the evaluation.

Ethics Approval: Not applicable. This study did not involve human participants or animals.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Avula C, Bachala S. A deep feature ensemble framework for intrusion detection in Internet of Medical Things. *Eng Technol Appl Sci Res*. 2025;15(5):26783–91. doi:10.48084/etasr.12242.
2. Alalwany E, Alsharif B, Alotaibi Y, Alfahaid A, Mahgoub I, Ilyas M. Stacking ensemble deep learning for real-time intrusion detection in IoMT environments. *Sensors*. 2025;25(3):624. doi:10.3390/s25030624.
3. Oladimeji D, Jinad R, Rasheed A, Baza M. CANGuard: an enhanced approach to the detection of anomalies in CAN-enabled vehicles. *Sensors*. 2025;25(1):278.
4. Villafranca A, Thant KM, Tasic I, Cano MD. AI-enabled IoT intrusion detection: unified conceptual framework and research roadmap. *Mach. Learn Knowl Extr*. 2025;7(4):115.
5. Siwakoti Y, Bhurtel M, Rawat D, Oest A, Johnson R. Advances in IoT security: vulnerabilities, enabled criminal services, attacks, and countermeasures. *IEEE Internet Things J*. 2023;10(12):11224–39.
6. Statista. Number of Internet of Things (IoT) connected devices worldwide from 2022 to 2034. Statista; 2024 [cited 2026 Apr 11]. Available from: <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>.
7. Alimi OA. Data-driven learning models for Internet of Things security: emerging trends, applications, challenges and future directions. *Technologies*. 2025;13(5):176.
8. Menon VU, Kumaravelu VB, Kumar VC, Rammohan A, Chinnadurai S, Venkatesan R. AI-Powered IoT: a survey on integrating artificial intelligence with IoT for enhanced security, efficiency, and smart applications. *IEEE Access*. 2025;13:50305–25.
9. Kouassi BM, Ballo AB, Ayikpa KJ, Mamadou D, Coulibaly MZJ. Top-K feature selection for IoT intrusion detection: contributions of XGBoost, LightGBM, and random forest. *Future Internet*. 2025;17(11):529.

10. Salehiyan A, Moghaddam PS, Kaveh M. An optimized transformer-GAN-AE for intrusion detection in edge and IIoT systems: experimental insights from WUSTL-IIoT-2021, EdgeIIoTset, and TON_IoT datasets. *Future Internet*. 2025;17(7):279.
11. Yang YM, Chang KC, Luo JN. Hybrid neural network-based intrusion detection system: leveraging LightGBM and MobileNetV2 for IoT security. *Symmetry*. 2025;17(3):314.
12. Thirasak K, Chuaphanngam T, Chainarong D, Fugkeaw S. TF2ML: threat filtering with two-stage machine learning for efficient provenance-aware threat detection and response. *IEEE Open J Comput Soc*. 2025;6:1751–62. doi:10.1109/ojcs.2025.3618157.
13. Almahaqeri SA, Almourish MH, Nasser AA, Alghawli ASA, Elsayed AAK, Alhejoj AN. An optimized gradient boosting framework for IoT intrusion detection: a comprehensive evaluation on the CICIOT2023 dataset. *Sci Rep*. 2026;16(1):16909. doi:10.1038/s41598-026-47399-5.
14. Gao W, Wang M, Pei Y, Li F, Wang C. A lightweight multi-classification intrusion detection model for edge IoT networks. *Electronics*. 2026;15(5):938. doi:10.3390/electronics15050938.
15. Alharby M. Evaluating machine learning approaches for multiple attack classification with improved computational efficiency in IoT networks. *Sci Rep*. 2025;15(1):39914.
16. Houichi M, Jaidi F, Bouhoula A. Enhancing smart city security: an intrusion detection system using machine learning methods with the UNB CIC IoT 2023 dataset. *IET Smart Cities*. 2025;7:e70014.
17. Wang H, Yang Y, Tan P. CTWA: a novel incremental deep learning-based intrusion detection method for the Internet of Things. *Artif Intell Rev*. 2025;58(12):374. doi:10.1007/s10462-025-11358-9.
18. Berguiga A, Harchay A, Massaoudi A. HIDS-IoMT: a deep learning-based intelligent intrusion detection system for the Internet of Medical Things. *IEEE Access*. 2025;13:32863–82.
19. Arivukarasi M, Gopalan SH, Gnanamurugan S, Dhanapal R, Ramachandran A, Manikandan A. Federated autoencoder IDS for IoT: a fed-ANIDS approach using CICIOT2023. *Syst Soft Comput*. 2026;8:200482.
20. Anwer RW, Abrar M, Ullah M, Salam A, Ullah F. Advanced intrusion detection in the industrial Internet of Things using federated learning and LSTM models. *Ad Hoc Netw*. 2025;178:103991. doi:10.2139/ssrn.5137152.
21. Cao Z, Liu B, Gao D, Zhou D, Han X, Cao J. A dynamic spatiotemporal deep learning solution for cloud-edge collaborative industrial control system distributed denial of service attack detection. *Electronics*. 2025;14(9):1843. doi:10.3390/electronics14091843.
22. Bilal MA, Ul Islam I, Idrees S, Qasim M, Junaid Khan M, Khan J. Dataset-centric evaluation of federated intrusion detection models in IoT networks. *Sci Rep*. 2026;16(1):2683.
23. Tanveer MU, Hassan SR, Belhaouari SB. Federated digital twin with spatiotemporal learning and postquantum security for robust IIoT anomaly detection. *IEEE Internet Things J*. 2026;13(12):28031–49. doi:10.1109/jiot.2026.3680990.
24. Baz M. Lightweight IDS for IoT based on counter-propagation networks. *IEEE Access*. 2025;13(5):147086–111.
25. Nasir N, Firdous A, Waseem SS, Hassan SR, Ihsan M, Siddiqui IF. Edge-aware multi-modal intrusion detection for consumer-centric industrial internet-of-things devices using deep adaptive fusion, early access. *IEEE Trans Consum Electron*. 2026. doi:10.1109/TCE.2026.3674715. Early access.
26. Himadri07. CICIOT2023: a modern dataset for intelligent IoT threat detection. Kaggle; 2023 [cited 2026 Aug 12]. Available from: <https://www.kaggle.com/datasets/himadri07/ciciot2023>.
27. Pradhan S. Edge-IIoTset-dataset. Kaggle; 2023 [cited 2026 Aug 12]. Available from: <https://www.kaggle.com/datasets/sibasispradhan/edge-iiotset-dataset>.
28. Carter J, Mancoridis S, Galinkin E. Fast, lightweight IoT anomaly detection using feature pruning and PCA. In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*; 2022 Apr 25–29; Virtual Event. p. 133–8.
29. Wang Z, Chen H, Yang S, Luo X, Li D, Wang J. A lightweight intrusion detection method for IoT based on deep learning and dynamic quantization. *PeerJ Comput Sci*. 2023;9(19):e1569. doi:10.7717/peerj-cs.1569.