Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.083321

ARTICLE

An Adaptive Federated Learning with XGBoost Ensembles for Intrusion Detection in Heterogeneous IoT Networks

Abdulaziz A. Alsulami[1], Qasem Abu Al-Haija[2,*], Rayed Alakhtar[3], Ahmad J. Tayeb[3], Badraddin Alturki[3], Huda Alsobhi[4] and Rayan A. Alsemmeari[3]

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

[2]Department of Cybersecurity, Faculty of Computer & Information Technology, Jordan University of Science and Technology, Irbid, Jordan

[3]Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

[4]Department of Information Technology, Faculty of Computing and Information Technology, Taif University, Taif, Saudi Arabia

*Corresponding Author: Qasem Abu Al-Haija. Email: qsabuhaija@just.edu.jo

Received: 01 April 2026; Accepted: 12 June 2026; Published: 23 July 2026

ABSTRACT: The rapid growth of the Internet of Things (IoT) devices has increased the attack area of modern networks, which makes effective intrusion detection systems (IDSs) essential to detect attacks that target IoT infrastructures. Federated learning is a promising approach for collaborative model training in the absence of centralized raw data. Conventional federated approaches rely on fixed client participation and static training configurations, which ensure symmetric treatment of clients despite heterogeneous local data distributions. This can limit convergence and degrade detection performance in non-IID conditions. This paper proposes an Adaptive Action-Based Federated Learning (AA-FL) framework for decentralized intrusion detection in heterogeneous IoT environments. The framework dynamically adjusts both participating clients and local training workload at each communication round using a Linear Upper Confidence Bound (LinUCB) contextual bandit controller. The proposed Adaptive-FL model is based on XGBoost boosters and uses quality-weighted server-side ensemble aggregation. At the same time, adaptation is guided by a multi-objective reward that balances classification performance, training latency, communication overhead, and computational cost. The framework is evaluated on CIC IoMT 2024 and RT-IoT2022 under realistic non-IID conditions using stratified 5-fold cross-validation and benchmarked against Static-FL, FedAvg-FL, and a centralized XGBoost upper bound. Experimental results demonstrate that Adaptive-FL outperforms all federated baselines across both datasets, achieving Macro-F1 scores of 98.27% on RT-IoT2022 and 94.21% on CIC IoMT 2024, with statistically significant improvements over Static-FL on both datasets. Adaptive-FL maintains superior classification stability while avoiding raw-data centralization. It remains within 0.67 and 0.35 percentage points of the centralized upper bounds on RT-IoT2022 and CIC IoMT 2024, respectively.

KEYWORDS: Federated learning; Internet of Things (IoT); intrusion detection systems (IDS); XGBoost; adaptive federated learning

1 Introduction

The Internet of Things (IoT) has transformed fundamental sectors, with the worldwide number of connected devices predicted to reach 21.1 billion by the end of 2025 and nearly 39 billion by 2030 [1]. Although increased connectivity enables greater productivity, it has also heightened the global attack surface,

presenting serious security risks. Cyberattacks on IoT networks have increased by more than 400% in recent years. Attackers frequently exploit the limited computational capabilities and insecure communication channels of smart devices to deploy disruptive botnets, cause data breaches, and mount Distributed Denial-of-Service (DDoS) attacks [2]. Robust and scalable Intrusion Detection Systems (IDS) are a requirement rather than a preference.

Traditional centralized machine learning and deep learning algorithms achieved detection rates, but required frequent and extensive data transfers to a centralized server. The centralized architecture causes high latency, consumes bandwidth, and raises major privacy concerns for raw data in IoT networks [3,4]. Federated Learning (FL) addresses these problems by enabling edge devices to train models locally and send only model updates, thereby preserving data privacy and distributing the computational load [3,5].

However, the practical realization of standard FL architectures in IoT settings is not straightforward. The IoT data in edge nodes are heterogeneous and non-IID, which means that the data collected from different clients have different distributions and are not statistically independent. High data imbalance breaks typical aggregation methods such as Federated Averaging (FedAvg) and leads to bad classification performance and irregular convergence [6,7]. Tree-based ensemble methods, such as XGBoost, have been very successful and computationally efficient on tabular data typical of network traffic [8,9]. However, their use in decentralized environments brings new aggregation challenges. Recent works have been able to deploy federated XGBoost and Random Forest architectures to achieve high accuracy in threat detection and near real-time anomaly identification [10,11]. However, these traditional federated tree adaptations tend to experience communication bottlenecks and static aggregation approaches.

Recent research has studied Deep Reinforcement Learning (DRL) and Contextual Bandits for adaptive client selection, dynamic decision boundary tuning, and temporal dependency extraction to optimize resource use and improve efficiency in such limited networks [12–15]. However, there is still an important challenge: these methods mostly treat the local learning workload as a fixed hyperparameter, thereby excluding the possibility of dynamically adjusting the actual computational cost, such as tree depth and boosting rounds, based on the network's evolving constraints at each communication round.

To address these limitations, this paper presents an Adaptive Action-Based Federated Learning (AA-FL) framework for decentralized IoT intrusion detection in heterogeneous, non-IID settings. Instead of relying on symmetric client participation and fixed training settings, AA-FL dynamically adjusts both the participating clients and the amount of local training performed at each communication round. The framework includes an adaptive federated (Adaptive-FL) model based on XGBoost boosters and a Static-FL baseline used as a reference for performance comparison.

The framework has three core parts. The first part is that the data are split across clients using a Dirichlet distribution to mimic realistic non-IID behavior and simple entropy statistics are computed to describe how different the clients are. Second, the chosen clients train locally regularized multi-class XGBoost models. Third, a contextual bandit controller based on the Linear Upper Confidence Bound (LinUCB) selects one of six predefined actions (Light, Balanced, Heavy, Diverse, Largest, Random). Each action represents a distinct trade-off between client diversity, model capacity and resource utilization.

Instead of using randomly averaged parameters, the server builds an ensemble of XGBoost boosters trained by the client, and then combines them by the average predicted class probabilities. A multi-objective reward is used to kick-start adaptation, which trades off Macro-F1 with normalized training time, communication overhead and computing cost. This allows the system to improve efficiency without sacrificing classification performance.

The main contributions of this work are summarized as follows:

- We introduce an adaptive federated learning framework for IoT intrusion detection that uses a contextual LinUCB policy, instead of fixed schedules, to select local training configurations and participating clients in each communication round. The framework also includes four practical enhancements, namely quality weighted ensemble aggregation, an EMA-smoothed delta-F1 reward to reduce noise, annealed Dirichlet partitioning to gradually increase data heterogeneity, and a ten-feature context vector that captures training progress, client statistics, and recent model performance.
- In contrast to Federated Averaging, the proposed strategy improves stability under non-IID client training by combining predictions through quality-weighted probability averaging and aggregating client contributions using a server-side ensemble of multi-class XGBoost boosters.
- We define a reward function that balances Macro-F1 and system efficiency. Training latency is measured using wall-clock time per round, while communication and computation costs are estimated using normalized simulation-based metrics derived from action complexity. We evaluate this trade-off under four penalty configurations.
- We conduct ablation studies to evaluate the contributions of the LinUCB controller, quality-weighted ensemble, and best fixed-action baseline (Static-Heavy). We also compare against FedAvg-FL and a centralized baseline, and perform sensitivity analysis on key hyperparameters (β, K, T, α). Statistical significance is validated using one-tailed Wilcoxon signed-rank tests across five folds.
- We adopt a strict evaluation protocol using stratified 5-fold cross-validation, where reward updates rely only on a server-validation split taken from the training fold, and the test fold remains untouched until the final evaluation.

The paper is organized as follows: Section 2 reviews the state-of-the-art work on federated learning, tree-based models for intrusion detection, and adaptive learning for IoT security. Then, in Section 3, we present a motivational example to illustrate the need for adaptive federated learning in various IoT scenarios. In Section 4, the datasets used in the study and their pre-processing are described. Section 5 describes the proposed framework, Adaptive Action-Based Federated Learning (AA-FL). It includes adaptive client selection, ensembling, reward setting, and explaining the experiment setup. We present the results of experiments, comparisons, component studies, sensitivity analyses and statistical proofs in Section 6. Finally, in Section 7, we conclude with a discussion of our results and directions for future work.

2 Related Work

The move from centralized machine learning to decentralized architectures has driven much research into secure resource-constrained IoT contexts. In order to create reliable Intrusion Detection Systems (IDS) for these networks, it is necessary to balance different data distributions, high communication costs and particular computing constraints. The scope of this challenge has been recorded by comprehensive surveys of IDS in IoT, which cover threat categories, detection strategies, deployment models and public benchmark datasets [16,17]. In this section, we examine recent advances in three connecting domains that are critical to this challenge: the deployment of normal and deep Federated Learning (FL) models for network security, the adaptation of tree-based ensemble algorithms for tabular IoT traffic, and the combined use of reinforcement learning and contextual bandits for adaptive resource management in decentralized systems.

Recent research has focused substantially on the convergence of distributed machine learning and Internet of Things (IoT) security. In 2025, this paper [18] proposed a large-scale experiment to determine the appropriate local deep learning model and data volume for implementing intrusion detection systems on resource-constrained IoT devices, utilizing Federated Learning (FL). Using Convolutional Neural Networks (CNN), Deep Neural Networks (DNN), and a hybrid CNN+BiLSTM architecture on the CICIoT2023

dataset, they obtained around 98% accuracy for the CNN and 99.9% for the hybrid model. This work [7] presented FedMSE, a semi-supervised federated learning strategy to identify unknown cyber threats by simulating realistic non-IID settings using Dirichlet distribution. This increased the N-BaIoT (Network-Based Detection of IoT Botnet Attacks Dataset) detection accuracy to 97.30%. To accomplish particular security requirements, Torre et al. [5] proposed a 1D CNN inside a federated system protected by a triple-layer privacy architecture including Differential Privacy, Diffie-Hellman Key Exchange and Homomorphic Encryption. With only a computational cost of 10%, preprocessing with label encoding and MinMaxScaler normalization produced an average accuracy of 97.31% across seven IoT datasets (TON-IoT, IoT-23, BoT-IoT, CIC IoT 2023, CIC IoMT 2024, RT-IoT 2022, and EdgeIIoT). Furthermore, Shalan et al. [19] presented a federated learning framework integrated with blockchain technology to detect intrusions in smart home environments, while Hamad et al. [20] carried out an extensive systematic analysis of FL approaches for IoT environments, which highlighted the unresolved constraints of decentralization.

Tree-based models and complex ensemble architectures have recently seen extensive adaptation to manage tabular network traffic. In 2025, Kouassi et al. [8] proposed a scalable intrusion detection approach that combines a Top-K feature selection framework with heterogeneous ensemble learning models such as XGBoost and Random Forest. Abd Elaziz et al. [9] introduces a novel trust-centric FL framework based on the tab transformer (TTF) model for IDS, attaining 99.94% accuracy on CICIoT2023 dataset across various benchmarks. In 2025, Tawfik et al. [4] presented FedMedSecure, a federated few-shot learning framework that combines Cross Transformers, FEAT(Few-shot Embedding Adaptation with Transformer), Relation Networks, and MAML (Model-Agnostic Meta-Learning). They used semantic clustering (grouping fine-grained labels into 5 macro-classes) and multi-method feature selection to decrease dimensionality to 20 features. The confidence-weighted ensemble achieved 99.9% accuracy on the CIC IoMT 2024 and CIDC2017 datasets by employing Explainable AI (SHAP(SHapley Additive exPlanations)) and differential privacy techniques. FedMalDetect, a lightweight FL-based classification model with ROC-AUC of 0.9485 and PR-AUC of 0.9177 on N-BaIoT, has been proposed by [21] in 2025 to address botnets. In addition, Ceran et al. [22] presented a strategy utilizing Graph Neural Networks (GNN) and XGBoost with late-fusion aggregation to reduce spatial relationships into robust classifications.

Recent research has used hierarchical architectures, Contextual Bandits, and Reinforcement Learning (RL) to address IoT networks' dynamic restrictions and changing environment. In 2025, Islam et al. [3] introduced the Privacy-Preserving Hierarchical Fog Federated Learning (PP-HFFL) architecture to address edge restrictions. They addressed excessive non-IID distributions by aggregating and downsampling 34 malware types into seven macrocategories. The hierarchical framework achieved over 98% accuracy on the RT-IoT 2022 and CIC-IoT 2023 datasets by utilizing a Multi-Layer Perceptron (MLP) with Personalized Federated Learning (PFL) and differential privacy features. Tamuka et al. [12] proposed a self-aware RL framework for real-time intrusion detection in fog networks using hierarchical techniques and dynamic resource management. The approach uses Thompson Sampling to achieve high stability in the presence of concept drift and was compared with standard contextual bandits such as LinUCB. Similarly, Mutambik and Almuqrin [13] proposed a contextual bandit-driven framework which is a cost-sensitive, multi-tier solution for cyber threat detection that greatly reduces the operational triage costs.

Table 1 summarizes the state of the art in Decentralized IoT Intrusion Detection. In summary, while the current literature has made important advances in federated architectures, reinforcement learning, contextual bandits, and hierarchical FL for IoT security, important gaps remain. Most existing FL-based IDS methods either rely on static local training settings, use computationally costly models that are less suitable for tabular or flow-level data, or employ parameter-averaging techniques such as FedAvg, which can degrade under heterogeneous, non-IID conditions. Furthermore, although contextual bandits and RL have

been investigated for threshold adjustment, routing, resource allocation, and basic client-selection decisions, they have rarely been used to jointly adapt client participation and the local model's workload, such as tree depth and boosting rounds, within a federated IDS training loop.

Table 1: Summary of related works in decentralized IoT intrusion detection.

Reference (Year)	Methodology/ Models	Datasets	Key Performance	Primary Differences from Proposed Work
Albanbay et al. [18] (2025)	FL, CNN, DNN, Hybrid CNN+BiLSTM	CICIoT2023	~98% (CNN), ~99% (Hybrid)	Fixed FL schedule with CNN/DNN; client count and local model complexity (tree depth, boosting rounds) are not jointly governed by a learned policy. AA-FL's LinUCB jointly controls both dimensions via six actions at every round.
Nguyen & Beuran [7] (2025)	Semi-supervised FL, MSE Aggregation	N-BaIoT	97.30% Accuracy	Fixed MSE aggregation with static client participation and binary detection only; local training depth and rounds are never adjusted. AA-FL jointly adapts client selection and XGBoost configuration per round via LinUCB with quality-weighted ensemble aggregation.
Torre et al. [5] (2025)	FL, 1D CNN, DP, DK, HE (Triple Privacy)	TON-IoT, IoT-23, BoT-IoT, CIC IoT 2023, CIC IoMT 2024, RT-IoT 2022, EdgeIIoT	97.31% Avg Accuracy	Static CNN-based FL applies the same client set and training workload every round regardless of heterogeneity. AA-FL's LinUCB explicitly and jointly controls tree depth, boosting rounds, and client count at each communication round.
Kouassi et al. [8] (2025)	Top-K Feature Selection, RF, XGBoost	CIC IoMT 2024	90.51% (RF Top-10), 88.98% (XGBoost All)	Centralized ensemble with no federated client selection and no dynamic workload adjustment. AA-FL is fully decentralized, with LinUCB jointly tuning client selection and local XGBoost complexity (depth, rounds) at each round.
Abd Elaziz et al. [9] (2025)	Attention-based Tab Transformer, XGBoost	N-BaIoT, UNSW-NB15, CICIoT2023	99.94% on CICIoT2023	Fixed FL aggregation with high-cost transformer gradients; client participation and local training complexity are not adapted across rounds. AA-FL uses compact XGBoost tree structures with LinUCB jointly controlling client selection and depth/rounds per round.
Tawfik et al. [4] (2025)	Fed Few-Shot, Cross Transformer, MAML, XAI	CIC IoMT 2024, CIDC2017	99.9% (CIC IoMT 2024), 98.5% (CIDC2017)	Static federated setup with fixed Cross Transformer; all clients participate every round with no workload adjustment. AA-FL's LinUCB selects from six actions jointly specifying client count, tree depth, and boosting rounds via a ten-feature context vector each round.
Patel et al. [21] (2025)	Semi-supervised FL, Shallow Neural Network	N-BaIoT, IoT-23, RaDaR	ROC-AUC 0.9485, PR-AUC 0.9177	Fixed model selection with uniform client participation and no multi-objective reward; local depth and rounds are never adapted. AA-FL's reward explicitly penalizes latency, communication, and cost while LinUCB jointly adapts client participation and local XGBoost workload per round.

(Continued)

Table 1 (continued)

Reference (Year)	Methodology/ Models	Datasets	Key Performance	Primary Differences from Proposed Work
Ceran et al. [22] (2025)	Graph Neural Networks (GNN), XGBoost	CICIoT-2023, CICIDS-2017, UNSW-NB15, IoMT-2024	99.51% Accuracy	Centralized late-fusion requiring graph structures; no federated client selection or workload adaptation exists. AA-FL preserves data locality and uses LinUCB to jointly control client selection and local model complexity at every round without graph dependencies.
Shalan et al. [19] (2025)	FL, Blockchain and Knowledge Distillation	N-BaIoT	90% Accuracy	Blockchain-enabled FL with knowledge distillation; client participation and local model configurations remain static across all FL rounds. AA-FL's LinUCB jointly determines both which clients train and their local model complexity at each round.
Islam et al. [3] (2025)	PP-HFFL, MLP, Personalized FL, DP	RT-IoT 2022, CIC-IoT 2023	95.02%–98.75%	Static hierarchical MLP with fixed client selection and training workload; local complexity is not co-optimized across rounds. AA-FL's LinUCB jointly governs client count and XGBoost complexity (depth, boosting rounds) per round with cost-aware adaptation.
Tamuka et al. [12] (2026)	Self-Aware RL (HATS-RL, F-HATS-RL)	Simulated Fog Traffic	0.933 AUROC	Thompson Sampling adapts anomaly thresholds only; client count and local training workload (depth, boosting rounds) are fixed and outside the bandit's action space. AA-FL's LinUCB directly encodes client count, tree depth, and boosting rounds as jointly optimized variables each round.
Mutambik and Almuqrin [13] (2025)	Contextual Bandits (UCB, Thompson), NN, RF	CSE-CIC-IDS2018	78%–99% Accuracy	Contextual bandits control cost thresholds only; local training workload and client participation are fixed and not part of the bandit's decision. AA-FL extends the bandit action space to jointly encode client selection, tree depth, and boosting rounds each round.

The proposed AA-FL framework directly addresses this gap. Using a contextual LinUCB controller, AA-FL dynamically determines not only how many clients participate, but also the local XGBoost training configuration required at each communication round. Coupled with a multi-objective reward function that balances Macro-F1 against normalized latency and resource costs, and a server-side quality-weighted ensemble of XGBoost boosters, the proposed framework provides an adaptive, cost-aware solution for non-IID tabular IoT intrusion detection.

3 Motivational Scenario

Consider we have a network of IoT devices distributed throughout the network. Every device has unique traffic information. Due to privacy concerns and computational expenses, we want to train an intrusion detection model without sending all that data to a single central server. As a result, using FL may reduce the privacy concern. Each device trains a local model and only transmits the model to the server rather than sending data. The server creates a single global model through the combination of all the local models. This is considered as a standard FL approach, but it has a limitation of the assumption that every device uses the symmetric training setup in every round. In fact, some devices are slower, some have more data, and some have more unusual attack types.

As a result, we must employ an Adaptive-FL model that dynamically modifies its configuration at every training cycle. Rather than employing the symmetric setup, the controller selects the configuration that is most likely to balance accuracy and resource efficiency at that particular moment based on the current state of the system (client heterogeneity, round progress, and last round's performance). It selects from six strategies; some give priority to the complexity of the model, while others give priority to the clients that should be included. A LinUCB bandit controller, which leverages past reward history to refine action selection, governs this decision.

Thus, the system learns from experience to select more intelligent training configurations round by round, balancing detection accuracy against time and resource costs, rather than executing FL in the same manner each time.

4 Datasets and Preprocessing

This section describes the datasets used to evaluate the proposed AA-FL framework and the preprocessing procedures used to prepare them for training and testing. Two publicly available IoT network traffic datasets are used in our study: RT-IoT2022 [23] and CIC IoMT 2024 WiFi MQTT [24].

4.1 Dataset Selection Rationale

The choice of these datasets is motivated by three main reasons. First, they span complementary IoT domains such as CIC IoMT 2024 WiFi MQTT capturing healthcare oriented IoMT traffic over Wi-Fi and MQTT communication [24] and RT-IoT2022 reflecting general purpose IoT traffic from non healthcare IoT devices and services in benign and adversarial network conditions [23]. Evaluating the proposed framework on both datasets makes it possible to assess whether its performance is generalizable across several IoT deployment situations rather of being limited to a particular domain. Second, both datasets are recent publicly available and have been used in recent IoT/IoMT intrusion detection studies, which includes federated learning based evaluations, which enables comparison with existing work under similar data conditions. Third, both datasets provide structured tabular features extracted from network traffic, which aligns with the XGBoost based modeling approach and the non-IID federated partitioning strategy adopted in this study.

4.2 Dataset Preprocessing Steps

We performed a number of preprocessing procedures on both datasets to ensure their integrity and reliable evaluation before model training. Fig. 1 provides an overview of the primary procedures used. Beginning with the raw dataset, the procedure goes through a series of cleaning and preparation steps.

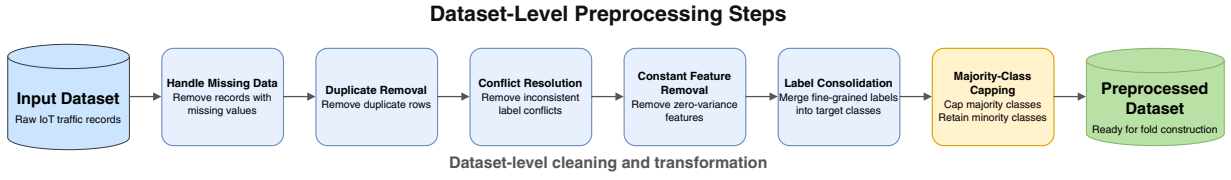


Figure 1: Dataset preprocessing steps applied prior to model training.

4.2.1 Preprocessing Scope

All dataset-level cleaning steps, including missing-value handling, duplicate removal, conflict resolution, constant-feature removal, label consolidation, majority-class capping, and random shuffling, were

applied to the full dataset before fold construction. Only two steps were performed within each fold: (i) feature-column alignment, where the server-validation and test feature matrices were aligned to the columns and ordering of the client-training feature matrix, and (ii) class-weight computation based only on the training fold. This ensures that no information from the validation or test folds influences the training feature schema or class-weight computation.

These preprocessing steps were designed to improve data quality and ensure reliable model evaluation. Missing values were handled prior to subsequent cleaning operations, so that feature-level and record-level checks were performed on complete records. Duplicate and conflicting records were removed to avoid biased learning and inconsistent labels. Constant features were discarded because they do not provide useful information and increase computational overhead. Label consolidation simplified highly granular attack categories, while majority-class capping reduced severe class imbalance.

4.3 CIC IoMT 2024

The CIC IoMT 2024 dataset originally had 7.1 million samples and 46 columns, with the last column representing the attack label. During the preprocessing steps, we discovered 2,645,426 duplicate rows and removed them by comparing the entire feature vector. We also identified 654 samples in 327 groups with identical features assigned conflicting labels and marked them as unreliable. In addition, a feature (Drate) was removed because its values were constant across all samples and did not provide useful information.

For label consolidation, we grouped the original fine-grained attack types into six categories: DDoS (Distributed Denial of Service), DoS (Denial of Service), Recon (Reconnaissance), and ARP (Address Resolution Protocol)_Spoofing, MQTT(Message Queuing Telemetry Transport)_Malformed, and Benign. Since DDoS and DoS had a very large number of samples compared to other classes, we capped them at 200,000 each to prevent them from dominating the training process. After all preprocessing steps, the final dataset contains 704,692 samples with 44 numerical features and 6 classes.

4.4 RT-IoT2022

This dataset has 123,117 samples, each described by 84 statistical flow features and one class label (85 columns in total). The dataset covers several attack scenarios, including DOS_SYN_Hping, DDOS_Slowloris, different NMAP scanning techniques, MQTT publishing activity, ARP poisoning, IoT device traffic, and brute-force SSH (Secure Shell) attempts.

Fortunately, the RT-IoT2022 dataset did not contain exact duplicate rows. We then examined whether identical feature values were associated with different labels. Two such cases (four samples in total) were identified and removed to avoid inconsistencies. Additionally, the features without variation were reviewed and bwd_URG_flag_count was discarded because it had the same value in all samples.

The detailed attack labels were grouped into the following categories: DoS, DDoS, Recon, ARP_Spoofing, MQTT and IoT_Device. In this dataset, the IoT_Device class represents legitimate IoT communications and serves as the normal traffic category. The class Metasploit_Brute_Force_SSH contains only 37 samples (0.03% of the dataset), it was therefore removed due to its very small size. Finally, no majority-class capping was needed. The final RT-IoT2022 dataset contains 123,076 samples with 83 numerical features and 6 consolidated classes.

4.5 Dataset Size Summary

Table 2 summarizes the dataset size changes after each preprocessing stage for both datasets. More than 2.6 million records were removed from the CIC IoMT 2024 dataset as they were duplicate records.

Resolving feature–label conflicts removed only a small number of samples but ensured consistent labeling. Removing the constant feature reduced the number of columns from 46 to 45. The final reduction to 704,692 samples resulted from label consolidation and limiting the two largest classes. However, RT-IoT2022 had only minor adjustments. This is because it had no duplicated records, only four conflicting samples were found and removed, and one constant feature was discarded. After label consolidation and removing the very small brute-force class, the dataset contained 123,076 samples. Unlike CIC IoMT 2024, no class capping was applied.

Table 2: Dataset size changes across preprocessing stages.

Stage	CIC IoMT 2024		RT-IoT2022	
	Samples	Cols.	Samples	Cols.
Original dataset	7,160,831	46	123,117	85
After duplicate removal	4,515,405	46	123,117	85
After conflict removal	4,514,751	46	123,113	85
After constant feature removal	4,514,751	45	123,113	84
After label consolidation and balancing	704,692	45	123,076	84

4.6 Common Preprocessing Strategy

All remaining features in both datasets were numerical after preprocessing, so categorical encoding was not required in the final experiments. Infinite values were replaced with NaN (Not A Number), missing values were filled with zeros, and the data were converted to float32 to reduce memory usage and improve computational efficiency. The samples were then randomly shuffled to remove any ordering effects.

To reduce the impact of class imbalance, class weights were computed using the balanced weighting scheme shown in Eq. (1):

$$w_c = \frac{N}{C \cdot n_c} \tag{1}$$

where N is the total number of samples, C is the number of classes, and n_c is the number of samples in class c. The weights were normalized to have a mean of one and incorporated during model training.

Table 3 shows the final class distributions of both datasets after preprocessing.

Table 3: Final class distribution after preprocessing.

Class	CIC IoMT 2024	RT-IoT2022
DDoS	200,000	534
DoS	200,000	94,659
Recon	90,820	7630
ARP_Spoofing	16,010	7748
MQTT/MQTT_Malformed	5130	4146
Benign/IoT_Device	192,732	8359
Total	704,692	123,076

5 Methodology

This section presents the methodology of the proposed AA-FL framework. The key idea behind AA-FL is that instead of relying on a fixed federated configuration and the system adapts its behavior at every communication round. Specifically, it dynamically adjusts the types of clients participating and how complex local models should be.

5.1 Architecture of the AA-FL Framework

The proposed framework combines an Adaptive-FL mechanism with a Static-FL baseline to allow the performance comparison of both approaches. The architecture of the AA-FL framework is shown in Fig. 2. First, the input dataset is loaded. Then, it is preprocessed. Several λ settings are defined, each configuration corresponds to a different experimental run to assess the trade-off between predictive performance and system efficiency. The dataset is validated by cross-validation, where each fold is divided into training and test subsets. The data in each training fold is further split into client training set and server validation set. Split alignment is performed in a leakage-safe way, using only the client training part, so that the server validation and test sets are the same without leakage of information.

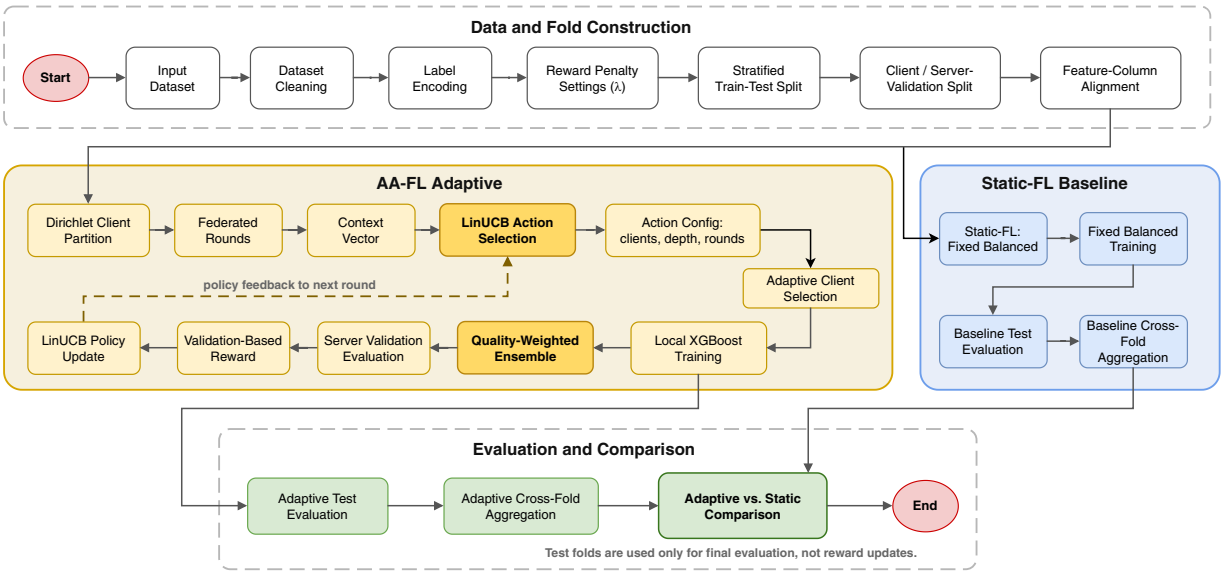


Figure 2: Architecture of the proposed AA-FL framework.

The blue components in Fig. 2 represent the Static-FL baseline. In this setting, a fixed Balanced action is applied in every round with uniform random client selection and no adaptive mechanism. The model is trained using the client training set, evaluated on the test set, and aggregated across folds to compute the overall baseline performance.

The yellow components represent the Adaptive-FL model. The client training data are partitioned among multiple clients using a Dirichlet distribution to simulate non-IID conditions. Training proceeds through multiple federated rounds, where a context vector summarizes the current system state. Based on this context, the LinUCB algorithm selects an action that determines both the client selection strategy and local model configuration. Selected clients perform local XGBoost training, and their trained boosters are incorporated into a server-side ensemble. The ensemble is evaluated on the server validation set, and a multi-objective reward is calculated to update the LinUCB policy. After completing all federated rounds,

the final ensemble is evaluated on the test set, and fold-level results are aggregated to compute the final comparison summary.

5.2 Problem Formulation

The dataset can be defined as shown in Eq. (2):

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}, \tag{2}$$

N is the total number of samples. Each feature vector x_i lies in $\mathbb{R}^d$, where d denotes the number of input features, and each label $y_i \in \{1, 2, \ldots, C\}$ corresponds to one of C classes.

In the federated setting, the dataset is distributed across K clients as follows:

$$\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}_k, \tag{3}$$

where $\mathcal{D}_k$ is the local data stored at client k. Eq. (3) captures the standard federated assumption that learning is performed without centralizing raw data.

5.3 Non-IID Client Partitioning

In practical federated learning settings, data across clients are often different. Some clients may mainly contain certain classes, while others have a more balanced distribution. This condition is known as non-IID data. To simulate this heterogeneity, we use a Dirichlet distribution for data partitioning, as defined in Eq. (4) [25]. The parameter β represents the concentration parameter that controls the level of data imbalance across clients. Smaller values of β produce more imbalanced class distributions (stronger non-IID), whereas larger values result in more balanced data allocations.

$$p_k \sim \text{Dirichlet}(\beta). \tag{4}$$

Instead of using a fixed β value during all training rounds, the Adaptive-FL model gradually changes β over time. As shown in Eq. (5), the model starts with a higher β value to create a more balanced (near-IID) data distribution in the early rounds and gradually moves to a lower β value to simulate stronger data heterogeneity in later rounds:

$$\beta_r = \beta_{\text{start}} + \frac{r-1}{T-1}(\beta_{\text{end}} - \beta_{\text{start}}), \tag{5}$$

where $\beta_{\text{start}} = 1.5$, $\beta_{\text{end}} = 0.3$, and T is the total number of communication rounds. This helps stabilize learning in the early stages while gradually exposing the model to more realistic non-IID conditions. The data partition is updated in each round using the current β_r value and the same client seed to ensure consistent and gradual changes. For comparison, the Static-FL baseline uses a fixed $\beta = 0.5$ during all rounds.

Entropy is computed to measure how diverse the class distribution is at client k,

$$H_k = -\sum_{c=1}^{C} p_{k,c} \log_2(p_{k,c}), \tag{6}$$

where the entropy definition in Eq. (6) [26] assigns higher values to clients with more uniform class mixtures and lower values to clients dominated by few classes [27]. In Adaptive-FL, these heterogeneity statistics are later used as part of the contextual signal for adaptive decision making.

5.4 Local Model Training

Each selected client trains a multi-class XGBoost model by minimizing the regularized objective as shown in Eq. (7) [28].

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{B} \Omega(f_t), \tag{7}$$

where $l(\cdot)$ denotes multi-class log-loss, B denotes the number of boosting rounds, and $\Omega(f_t)$ regularizes the complexity of the boosting trees. The first term enforces predictive fit to local data while the second term discourages overly complex models, which is important under non-IID distributions.

For inference, class probabilities are obtained using Eq. (8) which converts raw scores into a normalized probability distribution over the C classes.

$$P(y = c|x) = \text{softmax}(F(x)), \tag{8}$$

where $F(x)$ is the aggregated score produced by the boosting ensemble.

XGBoost was selected over deep learning because the datasets used in this study contain tabular network traffic features rather than raw images, text, or sequential signals. Each sample consists of fixed statistical features such as packet counts, byte sizes, inter-arrival times, and protocol flags. For this type of structured data, gradient-boosted trees often perform very well while requiring less computation and less training data than deep learning models. This is especially important in federated learning, where each client may only have a limited number of samples after Dirichlet partitioning.

Deep learning models such as CNNs and transformers have shown strong performance in IoT intrusion detection studies [5], but they usually require higher computational resources and generate larger communication overhead when exchanging model updates. They also require more complex hyperparameter tuning across heterogeneous clients. In contrast, XGBoost uses a smaller set of interpretable hyperparameters, such as tree depth and boosting rounds, which can be directly controlled by the LinUCB action space. This makes adaptive workload management easier and more practical in our federated setting. For these reasons, tree-based methods remain widely used in recent tabular IoT intrusion detection research [8,9,21,22], making XGBoost a suitable choice for the proposed framework.

5.5 Adaptive Action Selection via LinUCB

Instead of relying on a fixed federated configuration across communication rounds, Adaptive-FL selects at each round one action from a predefined discrete action set:

$$\mathcal{A} = \{\text{Light, Balanced, Heavy, Diverse, Largest, Random}\}. \tag{9}$$

Eq. (9) defines the available training configurations that the adaptive controller may choose from. Each action specifies (i) the number of participating clients in the current round, (ii) the local XGBoost model complexity, and (iii) the client selection strategy. The exact specifications are summarized in Table 4.

The six-action space was designed to cover the main decisions in federated training without adding unnecessary complexity. The first group controls local training workload. Light, Balanced, and Heavy represent different trade-offs between efficiency and model performance by changing tree depth, boosting rounds, and the number of participating clients. The second group controls client selection. Diverse selects clients with more label diversity, Largest selects clients with more data samples, and Random selects clients randomly to maintain exploration.

Table 4: Action specifications used in Adaptive-FL.

Action	# Clients	Max Depth	Boost Rounds	Client Policy
Light	3	4	60	ClientUCB
Balanced	4	6	120	ClientUCB
Heavy	5	7	160	ClientUCB
Diverse	4	6	180	Highest Entropy
Largest	4	6	180	Largest Dataset
Random	4	6	180	Uniform Random

Together, these actions allow the model to adapt both training workload and client participation. The Random action is important because it provides unbiased exploration during early rounds when the reward estimates are still unstable. Adding more actions would make learning slower and require more communication rounds, which is not suitable for our 15-round setting. The ablation results in Section 6.5 show that no single action performs best in all cases, which supports the need for adaptive action selection.

At communication round t, the adaptive controller observes a context vector x_t that summarizes the current federated learning state. The context vector contains ten features: (1) normalized round progress r/T, (2) mean client entropy $\bar{H}$, (3) client entropy variation σ_H, (4) relative mean client size $\bar{n}/n_{\max}$, (5) previous round reward r_{t-1}, (6) normalized training time from the previous round, (7) ensemble saturation ratio $|E|/M_{\max}$, (8) current EMA-smoothed Macro-F1 $\tilde{f}_t$, (9) F1 improvement rate $\tilde{f}_t - \tilde{f}_{t-1}$, and (10) a constant bias term.

Given this context, LinUCB evaluates each action using the upper confidence bound defined in Eq. (10) [29]:

$$p_a = \theta_a^\top x_t + \alpha\sqrt{x_t^\top A_a^{-1} x_t}, \tag{10}$$

where θ_a is the parameter vector for action a, A_a is the accumulated design matrix, and α controls the exploration strength. The first term estimates the expected reward (exploitation), while the second term measures uncertainty and encourages exploration.

The action selected at round t is determined by [29]:

$$a_t = \arg\max_{a \in \mathcal{A}} p_a, \tag{11}$$

which chooses the action with the highest optimistic reward estimate under the current context. Eq. (11) therefore formalizes the adaptive decision mechanism that governs client participation and model complexity throughout training.

5.6 Ensemble-Based Aggregation

Unlike FedAvg-style parameter aggregation, Adaptive-FL builds a server-side ensemble of client-trained boosters. Each booster is assigned a quality weight equal to its local validation Macro-F1 score, and the global class probability is computed through weighted probability averaging:

$$P(y|x) = \frac{\sum_{m=1}^{M} w_m \, P_m(y|x)}{\sum_{m=1}^{M} w_m}, \tag{12}$$

where $P_m(y|x)$ is the probability output of the m-th booster, M is the current ensemble size, and w_m is the local validation Macro-F1 of the m-th booster's training client. This quality-weighted scheme ensures that boosters trained on more informative or better-distributed local data contribute proportionally more to the global prediction. Eq. (12) provides a stable aggregation rule that avoids parameter misalignment across client models.

Ensemble eviction. When the number of accumulated boosters exceeds the maximum ensemble size $M_{\max}$, the lowest-quality boosters (by weight w_m) are evicted first. This quality-based eviction strategy retains the most informative boosters regardless of their arrival order, unlike FIFO (First In First Out)-based eviction.

Finally, the predicted label is obtained,

$$\hat{y}(x) = \arg \max_{c \in \{1,\ldots,C\}} P(y = c \mid x) \tag{13}$$

where $\hat{y}(x)$ denotes the predicted class label for input x, C is the total number of classes, and $P(y = c \mid x)$ represents the predicted posterior probability of class c. According to Eq. (13), the class with the highest predicted probability is selected as the final decision.

5.7 Multi-Objective Reward

To guide adaptive selection, AA-FL employs a multi-objective reward. Rather than using the raw server-validation Macro-F1 directly, which can be noisy across rounds, the reward is based on the improvement in an exponential moving average (EMA) of the Macro-F1. As defined in Eq. (14), the EMA is updated after each round as:

$$\tilde{f}_t = \alpha_{\text{ema}} f_t + (1 - \alpha_{\text{ema}}) \tilde{f}_{t-1}, \tag{14}$$

where f_t is the raw Macro-F1 on the server validation set at round t and $\alpha_{\text{ema}} = 0.3$ is the smoothing coefficient. The final reward is then computed using Eq. (15):

$$r_t = (\tilde{f}_t - \tilde{f}_{t-1}) - \lambda_{time} T_{norm} - \lambda_{comm} C_{norm} - \lambda_{cost} K_{norm}, \tag{15}$$

where $(\tilde{f}_t - \tilde{f}_{t-1})$, derived from Eq. (14), represents the EMA-smoothed improvement in Macro-F1, T_{norm} is the normalized wall-clock training time per round, C_{norm} is the normalized communication overhead, and K_{norm} is the normalized computational cost. All penalty terms are normalized to $[0, 1]$ to ensure they remain proportional to the performance term. These normalization terms are formally defined in Eq. (16) as:

$$T_{norm} = \frac{t_{round}}{\max_t(t_{round})}, \quad C_{norm} = \frac{\hat{c}_{comm}}{\max_t(\hat{c}_{comm})}, \quad K_{norm} = \frac{\hat{c}_{cost}}{\max_t(\hat{c}_{cost})}, \tag{16}$$

where t_{round} is the total wall-clock time for client training and server evaluation in round t, $\hat{c}_{\text{comm}} = K \cdot B \cdot (D_{\text{depth}} + 1) \cdot \log_2(F)$ is the communication proxy with K clients, B boosting rounds, D_{depth} tree depth, and F feature dimensionality, and $\hat{c}_{\text{cost}} = K \cdot B \cdot (D_{\text{depth}} + 1)$ is the computational cost proxy. The maximum values in Eq. (16) are updated incrementally as training progresses, so normalization is always relative to the most expensive round observed so far.

Using Eq. (15), the framework balances predictive accuracy with system efficiency. This allows the adaptive policy to account for resource constraints during action selection. A higher reward indicates that the action achieved a strong classification improvement at low cost, which makes it more likely to be selected in future rounds through the LinUCB update in Eq. (10).

To examine the impact of cost sensitivity, we evaluated four configurations of penalty coefficients λ_{time}, λ_{comm}, and λ_{cost}, as summarized in Table 5. The first configuration $(0, 0, 0)$ represents an accuracy-only baseline in which no cost penalties are applied. The remaining configurations progressively increase efficiency awareness by assigning larger weights to time and resource costs.

Table 5: Penalty coefficient configurations used for reward sensitivity analysis.

Setting	λ_{time}	λ_{comm}	λ_{cost}	Description
S1	0.00	0.00	0.00	Accuracy-only baseline.
S2	0.03	0.02	0.02	Mild efficiency regularization that introduces limited cost sensitivity.
S3	0.08	0.05	0.05	Moderate efficiency regularization with stronger cost influence.
S4	0.12	0.08	0.08	Strong efficiency regularization emphasizing reduced resource consumption.

Relative ordering $\lambda_{time} > \lambda_{comm} = \lambda_{cost}$ reflects the practical importance of training latency in federated environments, where multiple communication rounds may significantly impact the scalability of the system. Communication and computational costs are treated with comparable influence in our experimental setup. The selected coefficient ranges ensure that the total penalty in Eq. (15) remains proportionate to the Macro-F1 score (bounded in $[0, 1]$), which prevents the cost terms from dominating the optimization objective while still allowing meaningful efficiency-aware behavior.

5.8 Experimental Configuration

This subsection describes the experimental setup used to evaluate the proposed AA-FL framework. First, it discusses data partitioning throughout the framework. Then it presents the parameter setup used for the adaptive federated ensemble. Finally, it provides the configuration for the Static-FL baseline, which is used as a reference model to compare with the proposed framework's performance.

Fig. 3 illustrates the data partitioning strategy that is adopted in this research to evaluate the proposed AA-FL framework. First, the entire dataset is split by a stratified 5-fold cross-validation, with 80% of the data allocated to the training fold and the remaining 20% to the test fold which is kept entirely isolated during the entire training process. The 80/20 train–test ratio is a natural consequence of the five-fold cross-validation protocol, with each held-out fold containing 20% of the data. We opted for five-fold stratified cross-validation as it provides a reasonable trade-off between computational cost and statistical robustness.

Within each training fold, 15% is reserved as a server validation set used exclusively for computing the multi-objective reward and updating the LinUCB policy in Adaptive-FL, and for convergence tracking in Static-FL. The remaining 85% forms the client training data. For Adaptive-FL, client data are distributed using an annealed Dirichlet schedule from $\beta = 1.5$ to $\beta = 0.3$ across communication rounds, while Static-FL uses a fixed $\beta = 0.5$ partition. Each client further splits its local data into 80% for local XGBoost training and 20% for local validation used only for early stopping. This hierarchical design ensures strict leakage prevention at every stage.

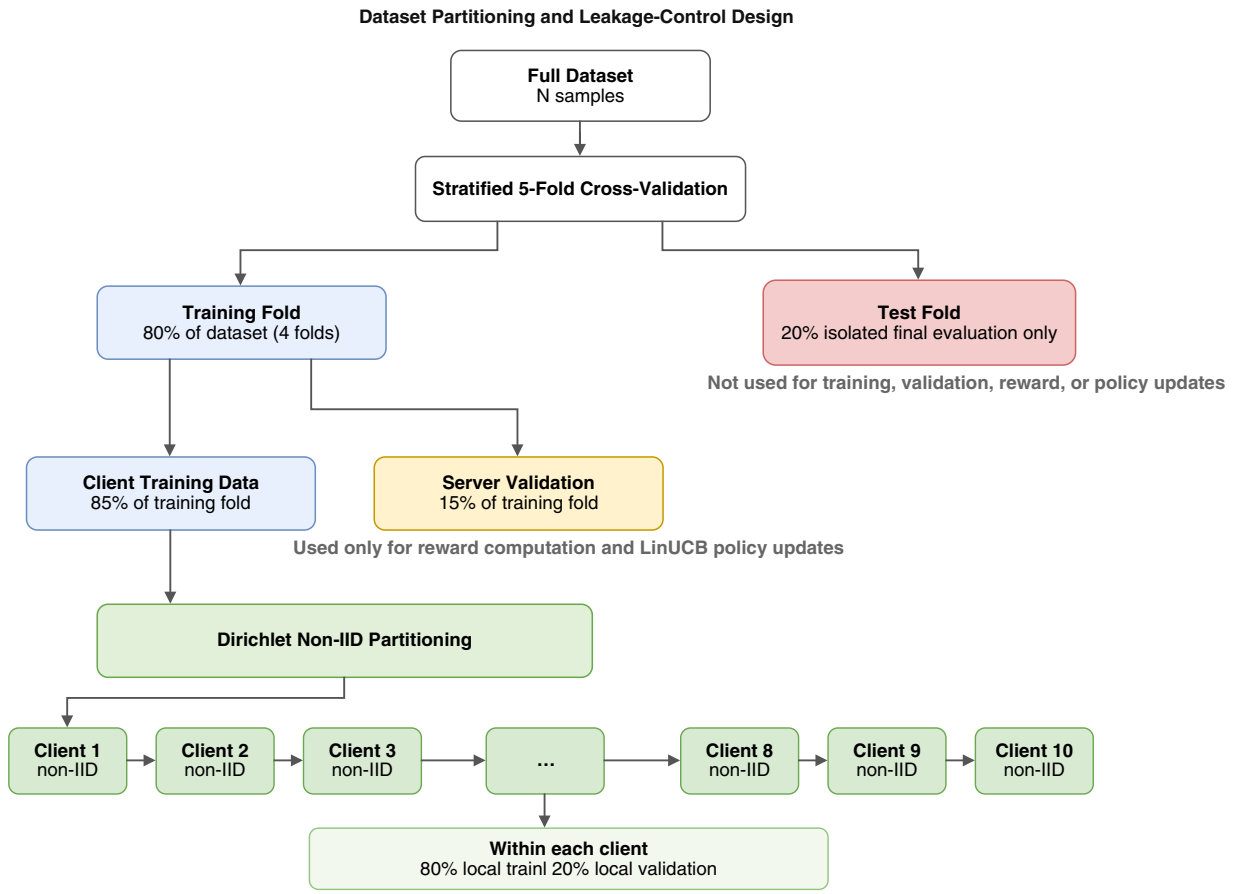


Figure 3: Hierarchical data partitioning strategy used to evaluate the proposed AA-FL framework.

Table 6 summarizes the key parameters used for both Adaptive-FL and Static-FL. The federated simulation is conducted with $K = 10$ clients over 15 communication rounds. For Adaptive-FL, non-IID partitioning follows an annealed Dirichlet schedule from $\beta = 1.5$ to $\beta = 0.3$ across communication rounds, while Static-FL uses a fixed $\beta = 0.5$. The adaptive controller uses LinUCB with an exploration parameter of $\alpha = 2.0$ and ridge regularization of 1.0. Early stopping with 20 rounds is applied based on local validation performance to prevent overfitting at the client level. Performance consistency across folds is reflected by the low standard deviation of Macro-F1 across the five independent test folds: ±0.42% on RT-IoT2022 and ±0.30% on CIC IoMT 2024.

Table 6: Experimental configuration comparison between Adaptive-FL and Static-FL.

Component	Parameter	Adaptive-FL	Static-FL
Cross-validation	Protocol	Stratified 5-Fold	Stratified 5-Fold
Federated setup	Clients (K)	10	10
	Rounds	15	15
Data heterogeneity	Dirichlet β	1.5 → 0.3 over rounds	0.5 fixed
Server validation	Fraction of train fold	15%	15%
Client validation	Local split	20%	20%

(Continued)

Table 6 (continued)

Component	Parameter	Adaptive-FL	Static-FL
Base learner	XGBoost objective	Multi-class softmax	Multi-class softmax
Training control	Early stopping	20 rounds	20 rounds
Action strategy	Training policy	Dynamic LinUCB action selection	Fixed Balanced
Client selection	Policy	Action-dependent	Uniform Random
Adaptive controller	LinUCB exploration (α) Ridge regularization Context vector size	2.0 1.0 10 features	None None None
Ensemble	Aggregation Max ensemble size ($M_{\max}$)	Quality-weighted averaging 60 boosters	Fixed Balanced ensemble 60 boosters
Reward	EMA smoothing (α_{ema})	0.3	None
Reward settings	λ configurations	S1–S4 (Table 5)	None
Evaluation	Metrics	Accuracy, Macro-F1, Weighted-F1	Accuracy, Macro-F1, Weighted-F1
Leakage control	Split alignment	Train-fold only	Train-fold only

Both models are trained using the same XGBoost training pipeline, validation splits, evaluation metrics and leakage control policy, to allow for a fair comparison. The main difference is that Adaptive-FL selects the training actions by LinUCB adaptively and aggregates the clients ensemble by quality-weighted, while Static-FL always selects the fixed Balanced action in each round and selects clients uniformly at random.

6 Results and Discussion

In this section, we present and discuss the experimental results of the proposed AA-FL framework evaluated on two datasets: RT-IoT2022 and CIC IoMT 2024. We analyze the behavior of the Adaptive-FL model under realistic non-IID conditions using stratified 5-fold cross-validation, with Static-FL serving as the reference baseline. We begin by comparing the classification performance of both models in terms of Macro-F1 and accuracy across all penalty configurations. Next, we analyze class-wise behavior using confusion matrices to examine misclassification patterns for each traffic category. After that, we present the action selection distribution of the adaptive controller across all folds for the selected configuration. Finally, we examine the server-validation Macro-F1 convergence curve across communication rounds to assess the stability and learning behavior of the Adaptive-FL model.

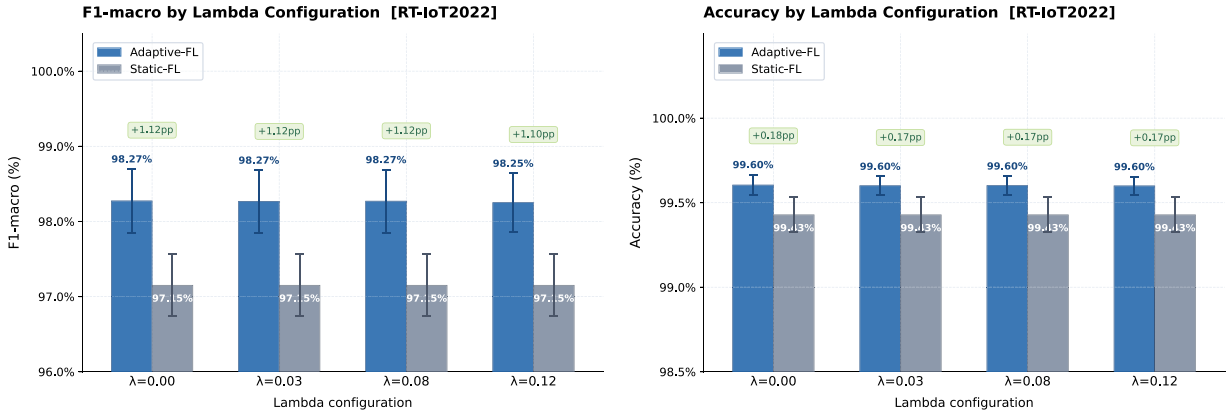
6.1 Results and Discussion on RT-IoT2022

6.1.1 Classification Performance on RT-IoT2022

We tested four penalty configurations, S1–S4, as defined in Table 5, to show the effect of resource efficiency penalties on classification performance. S1 applies no penalties and focuses only on accuracy, while S4 applies the strongest penalties to reduce resource usage.

As shown in Fig. 4, Adaptive-FL outperformed Static-FL across all penalty configurations in both Macro-F1 and accuracy. Adaptive-FL achieved a stable Macro-F1 of 98.27% in S1–S3 (±0.42% across folds), with only a slight drop to 98.25% in S4 under the strongest penalties. In contrast, Static-FL remained at 97.15% (±0.41%) across all settings, resulting in a consistent improvement of approximately 1.12 percentage points for Adaptive-FL (Wilcoxon one-tailed $p = 0.031$, Cohen's $d = 2.14$).

A similar trend can be seen in accuracy where Adaptive-FL maintained 99.60% in all configurations against 99.43% for Static-FL. The smaller error bars for Adaptive-FL in Fig. 4 also suggest more consistent performance across folds. In summary, the penalties in efficiency grow, and Adaptive-FL stays robust, showing it can keep good classification performance with resource limits. On the basis of this trade-off, S2 was selected as the representative configuration for RT-IoT2022, since it achieves the same performance as S1 but with moderate efficiency awareness.



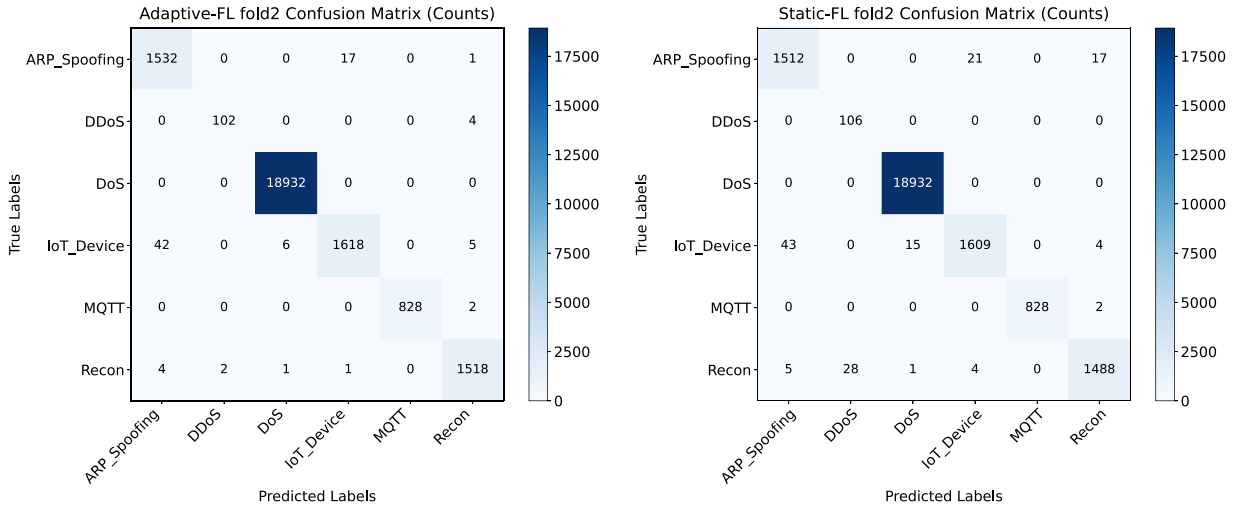
(a) Macro-F1 performance

(b) Accuracy performance

Figure 4: Macro-F1 and accuracy performance across lambda configurations for Adaptive-FL and Static-FL on the RT-IoT2022 dataset. Error bars denote the standard deviation across the five cross-validation folds. Improvements are reported in percentage points (pp).

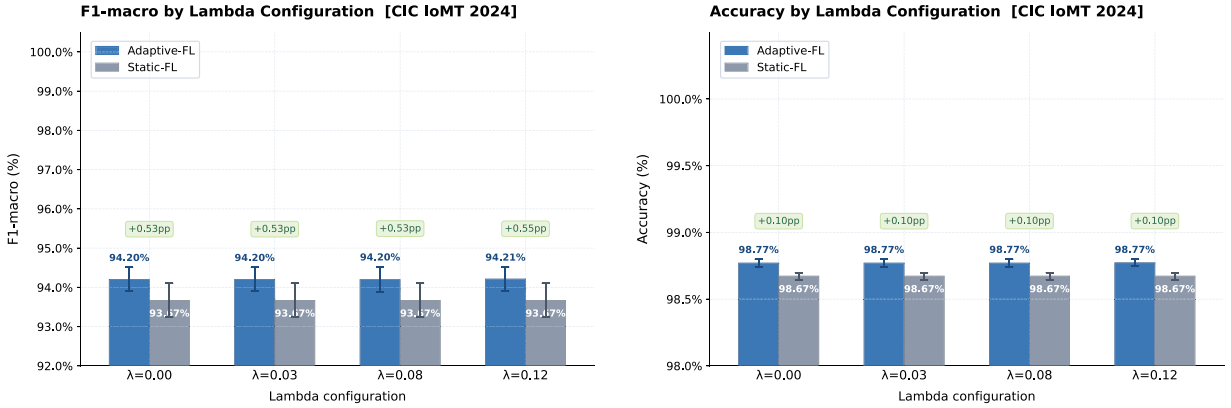
6.1.2 Confusion Matrix Analysis

In addition to Macro-F1 and accuracy, Fig. 5 depicts the confusion matrices of Adaptive-FL and Static-FL on the RT_IoT2022 test set. Both models show high diagonal dominance, indicating strong classification performance across all six classes. However, Adaptive-FL produces fewer misclassifications in several important classes. For ARP_Spoofing, Adaptive-FL correctly classifies 1532 samples compared with 1512 samples for Static-FL, with fewer errors toward IoT_Device and Recon. Adaptive-FL also improves IoT_Device classification by correctly classifying 1618 samples compared with 1609 samples for Static-FL. The largest improvement is observed in the Recon class, where Adaptive-FL correctly identifies 1518 samples compared with 1488 samples by Static-FL, with less confusion with DDoS. Both models achieve the same performance on MQTT and perfect classification for DoS. For DDoS, Static-FL performs slightly better, with 106 correctly classified samples compared with 102 for Adaptive-FL; however, the difference is small because the number of samples is limited. Overall, Adaptive-FL provides more consistent class-level performance, particularly for ARP_Spoofing and Recon, where it reduces misclassifications compared with Static-FL (In Fig. 6, we will discuss results related to CIC IoMT 2024).



(a) Adaptive-FL (b) Static-FL

Figure 5: Confusion matrices for Adaptive-FL and Static-FL on the RT_IoT2022 test set (fold 2).



(a) Macro-F1 performance (b) Accuracy performance

Figure 6: Macro-F1 and accuracy performance across lambda configurations for Adaptive-FL and Static-FL on the CIC IoMT 2024 dataset. Error bars denote the standard deviation across the five cross-validation folds. Improvements are reported in percentage points (pp).

6.1.3 Adaptive Action Behavior on RT-IoT2022

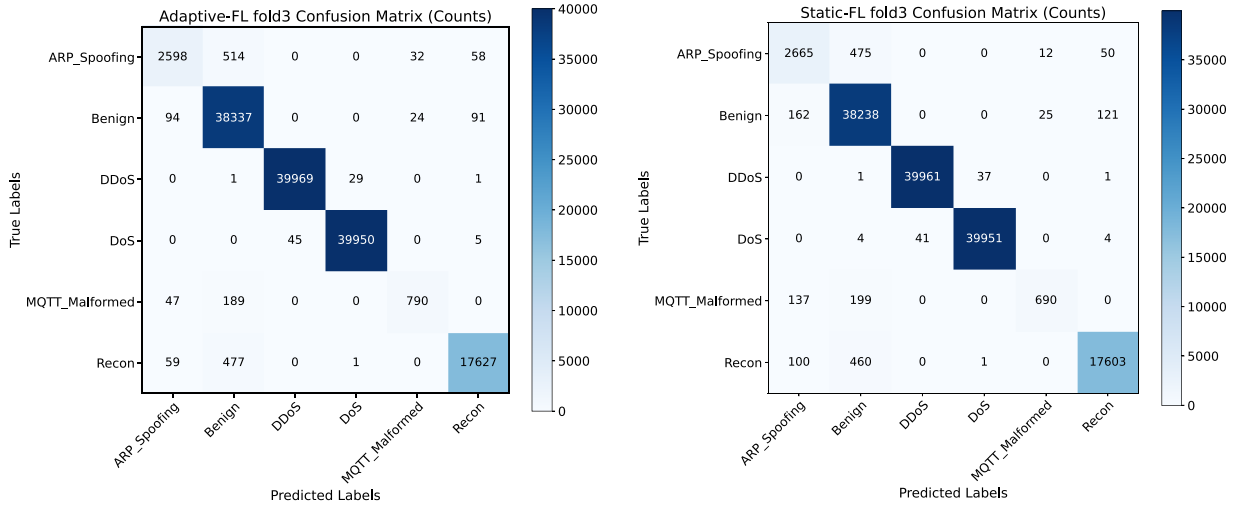
Table 7 shows the overall action selection distribution over all folds for the chosen configuration S2. The results show a relatively even spread over the six candidate strategies. The configurations chosen most often were the Light and Balanced ones (20% each), followed by Heavy (17.3%), Diverse and Random (14.7% each), and Largest (13.3%). The absence of a dominant action suggests that different configurations of client selection and model capacity have similar performance on RT-IoT2022. The adaptive mechanism does not favor any strategy since the classification accuracy is close to saturation and convergence is fast as observed before. Rather, it keeps looking over various configurations without losing stability. This indicates that RT-IoT2022 can be well separated and less sensitive to client heterogeneity or model complexity change in the tested scenarios. The Adaptive-FL model is thus robust without collapsing to a single deterministic policy, which reflects the low structural difficulty of the dataset under non-IID partitioning.

Table 7: Aggregated action selection frequency for RT-IoT2022 (Adaptive-FL, S2) across all folds.

Action	Count	Selection (%)
Light	15	20.0
Balanced	15	20.0
Heavy	13	17.3
Diverse	11	14.7
Random	11	14.7
Largest	10	13.3

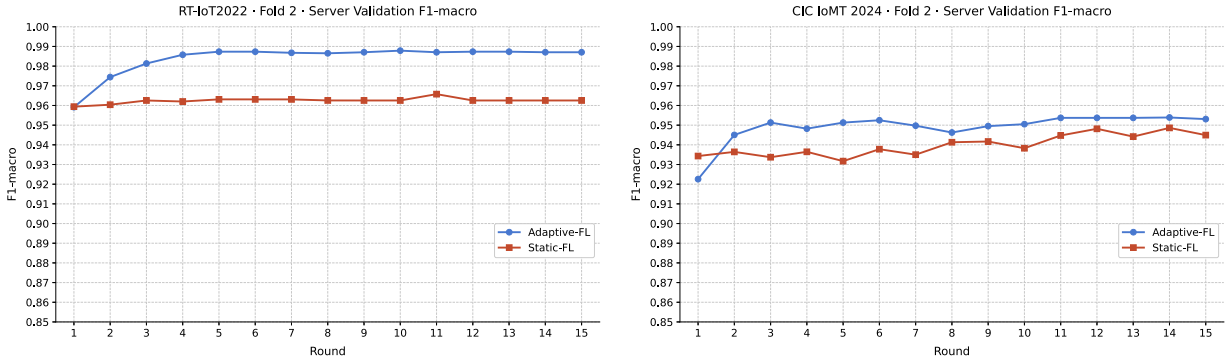
6.1.4 Convergence Behavior on RT-IoT2022

While Fig. 7 shows the confusion matrices for the other dataset, Fig. 8a compares the server-validation Macro-F1 convergence across communication rounds for Adaptive-FL and Static-FL on the RT-IoT2022 dataset (fold 2). Both methods start from similar initial values around 0.959, but their trajectories diverge immediately after round 1. Adaptive-FL rises sharply to 0.974 by round 2, then continues improving steadily to reach approximately 0.986 by round 4. From round 4 onward, the model stabilizes within a narrow band of 0.986–0.988, demonstrating rapid convergence. The gap between the two me thods stabilizes at approximately 0.021–0.025 Macro-F1 points from round 4 onward, demonstrating a consistent advantage for the adaptive strategy on this dataset. Static-FL, by contrast, remains largely flat around 0.960–0.966 throughout training, with only a brief upward fluctuation around round 11 before returning to its prior level. This flat behavior confirms that applying the same fixed training configuration every round is poorly suited to the heterogeneous data distribution of RT-IoT2022.



(a) Adaptive-FL (b) Static-FL

Figure 7: Confusion matrices for Adaptive-FL and Static-FL on the CIC IoMT 2024 test set (fold 3).



(a) RT-IoT2022 (fold 2)

(b) CIC IoMT 2024 (fold 2)

Figure 8: Server-validation Macro-F1 convergence across communication rounds for Adaptive-FL and Static-FL.

6.2 Results and Discussion on CIC IoMT 2024

6.2.1 Classification Performance on CIC IoMT 2024

Adaptive-FL achieves 94.20% Macro-F1 at S1, S2, and S3 (±0.30%–0.32% std), rising marginally to its best result of 94.21% at S4 (±0.30%, as shown in Fig. 6). This indicates that stronger efficiency regularization does not reduce classification performance and may provide a minor benefit on this more complex dataset. In contrast, Static-FL remains at 93.67% (±0.43%) across all settings because it does not adapt to the penalty configuration. As a result, the performance gap between Adaptive-FL (S4) and Static-FL is +0.55 percentage points (Wilcoxon $p = 0.031$, Cohen's $d = 1.26$). Classification accuracy remains stable for both methods: Adaptive-FL achieves 98.77%, while Static-FL stays at 98.67%, a gap of 0.10 percentage points. Error bars represent the standard deviation across the five cross-validation folds. Overall, Adaptive-FL performs better than Static-FL in all settings, and stronger regularization does not reduce its effectiveness on this dataset. Based on these results, S4 is selected as the final configuration for the CIC IoMT 2024 experiments because it achieves the highest Macro-F1 (94.21%).

6.2.2 Confusion Matrix Analysis

This research also reports the confusion matrices for the experiments on the CIC IoMT 2024 dataset. Fig. 7 presents the confusion matrices for Adaptive-FL and Static-FL on the CIC IoMT 2024 test set (fold 3). Both models show strong diagonal dominance, but the misclassification patterns are more pronounced than in RT_IoT2022, reflecting stronger overlap among ARP_Spoofing, MQTT_Malformed, and Benign traffic.

Adaptive-FL performs better on most classes, especially Benign, MQTT_Malformed, and Recon. For Benign traffic, Adaptive-FL correctly classifies 38,337 samples compared with 38,238 for Static-FL and reduces confusion with ARP_Spoofing (94 vs. 162). The largest improvement appears in MQTT_Malformed, where Adaptive-FL correctly classifies 790 samples compared with 690 for Static-FL, reducing confusion with ARP_Spoofing from 137 to 47. Adaptive-FL also slightly improves Recon classification (17,627 vs. 17,603 correct samples).

Static-FL performs better only on ARP_Spoofing, where it correctly classifies 2665 samples compared with 2598 for Adaptive-FL. DDoS and DoS are classified almost perfectly by both models. Overall, Adaptive-FL provides more consistent class-level performance on CIC IoMT 2024, particularly for the more difficult MQTT_Malformed and Benign classes.

6.2.3 Adaptive Action Behavior on CIC IoMT 2024

Table 8 shows how the adaptive controller selected actions across all folds under configuration S4. Light and Balanced were chosen most often at 20% each, followed by Diverse at 17.3% and Random at 16.0%, while Heavy and Largest were each selected 13.3% of the time.

Table 8: Aggregated action selection frequency for CIC IoMT 2024 (Adaptive-FL, S4) across all folds.

Action	Count	Selection (%)
Light	15	20.0
Balanced	15	20.0
Diverse	13	17.3
Random	12	16.0
Heavy	10	13.3
Largest	10	13.3

The Diverse and Random actions were selected more frequently on CIC IoMT 2024 than on RT-IoT2022, together accounting for 33.3% of all selections. This is due to the greater similarity between attack classes and the more complex data distributions present in CIC IoMT 2024. To handle this complexity, the controller allocated more rounds to configurations that bring a wider variety of client data into the ensemble, which is consistent with the more challenging classification results observed in the confusion matrix analysis.

The controller is exploring all possible strategies during the entire training phase and no single action dominates the selection distribution. This steady exploration helps the ensemble improve gradually over time, which is consistent with the stable convergence curve observed on CIC IoMT 2024, rather than relying on a single configuration throughout.

6.2.4 Convergence Behavior on CIC IoMT 2024

Fig. 8b compares the server-validation Macro-F1 convergence across communication rounds for Adaptive-FL and Static-FL on the CIC IoMT 2024 dataset (fold 2). Static-FL starts slightly higher in round 1 (0.934 vs 0.923), which reflects the Adaptive-FL warmup phase where the LinUCB controller has not yet accumulated sufficient reward history. From round 2 onwards, Adaptive-FL takes a consistent lead, rising to 0.945 while Static-FL remains at 0.937. Adaptive-FL continues to improve, reaching above 0.950 by round 3 and stabilizing in the 0.946–0.954 range for the remainder of training. Static-FL fluctuates in a lower band of 0.932–0.949 throughout. Adaptive-FL wins 14 of 15 rounds and ends at 0.953 compared to 0.945 for Static-FL. This contrasts with RT-IoT2022 where the separation between the two models is larger and more immediate.

6.3 Statistical Significance Analysis

To verify that the observed improvements of Adaptive-FL over Static-FL are not due to random variation across folds, we applied a one-tailed Wilcoxon signed-rank test (H_1: Adaptive-FL > Static-FL) to the five per-fold Macro-F1 scores. Fig. 9 plots the per-fold F1 values for both methods on both datasets, with dashed horizontal lines indicating the cross-fold means and vertical connectors linking each paired fold. Table 9 reports the formal test results. On both RT-IoT2022 and CIC IoMT 2024, the test yields $p = 0.031$, which is statistically significant at the $\alpha = 0.05$ level. Because the test uses only five paired observations, $p = 0.031$ is the smallest attainable one-tailed Wilcoxon signed-rank p-value; it occurs when all five paired differences favor the same method. Therefore, this boundary value should be interpreted as the strongest possible

evidence under the five-fold design, rather than compared directly with smaller p-values that can arise from studies with larger sample sizes. The corresponding Cohen's d values of 2.14 and 1.26 indicate large effect sizes. Together, these results provide evidence that the advantage of Adaptive-FL is systematic rather than coincidental.

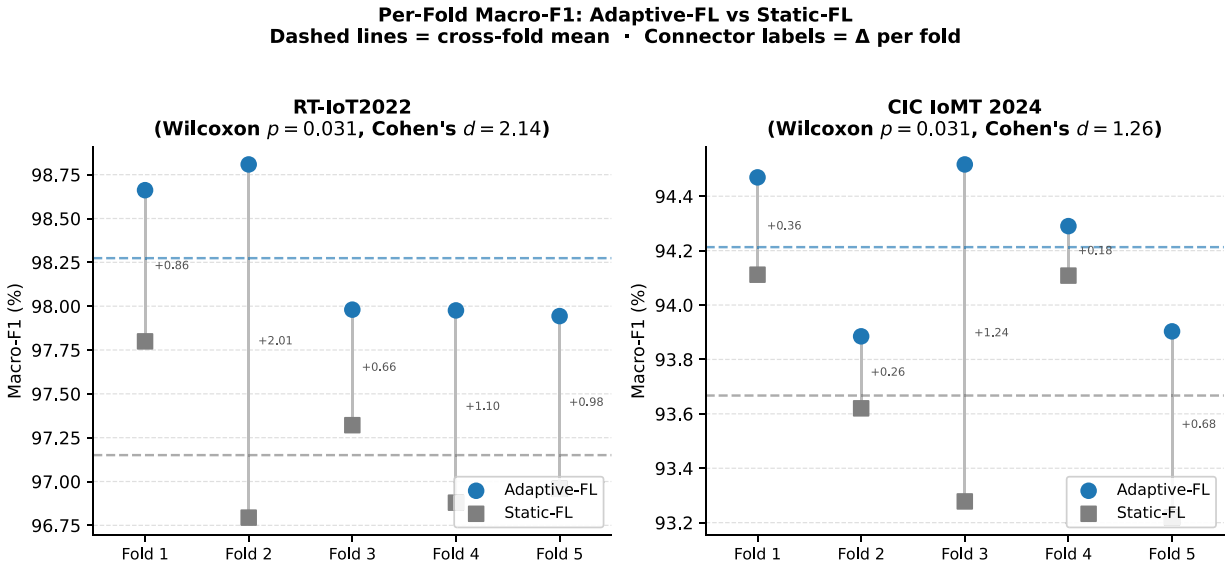


Figure 9: Per-fold Macro-F1 comparison of Adaptive-FL and Static-FL across both datasets.

Table 9: Wilcoxon signed-rank test results (Adaptive-FL vs. Static-FL, one-tailed, H_1: Adaptive-FL > Static-FL).

Dataset	Method	F1 mean	F1 std	ΔF1 (pp)	p-Value	Cohen's d
RT-IoT2022	Adaptive-FL	98.27%	±0.42%	+1.12	0.031*	2.14
RT-IoT2022	Static-FL	97.15%	±0.41%			
IoMT2024	Adaptive-FL	94.21%	±0.30%	+0.55	0.031*	1.26
IoMT2024	Static-FL	93.67%	±0.43%			

Note: *significant at $\alpha = 0.05$; $p = 0.031$ is the minimum achievable with five paired observations.

6.4 Extended Baseline Comparison

To position Adaptive-FL against a broader set of reference points, we include two additional baselines evaluated under identical stratified 5-fold cross-validation conditions: (1) **FedAvg-FL**, which resets the ensemble every round and uses equal-weight probability averaging of the current-round boosters only, closely approximating the aggregate-and-replace spirit of FedAvg for tree-based models; and (2) **Centralized-XGB**, which trains a single XGBoost model on the full (unfederated) client training data in each fold, serving as a privacy-agnostic upper bound. Fig. 10 visualizes all methods; Table 10 gives the numerical results.

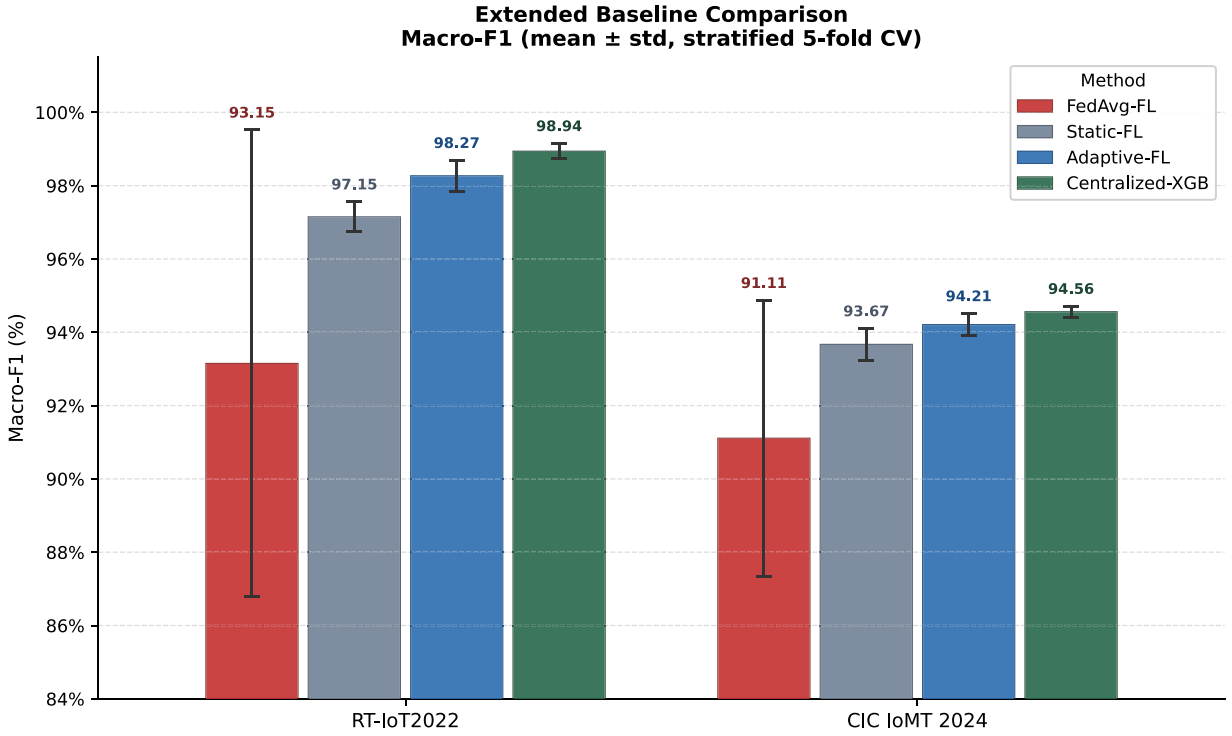


Figure 10: Macro-F1 comparison of all methods on RT-IoT2022 and CIC IoMT 2024 (mean ± standard deviation across 5 folds).

Table 10: Extended baseline comparison (Macro-F1, mean ± std across 5 folds).

Dataset	Method	Acc (%)	Macro-F1 (%)	Δ vs. Static-FL	Note
RT-IoT2022	Centralized-XGB	99.77 ± 0.03	98.94 ± 0.21	+1.79 pp	Privacy-agnostic upper bound
RT-IoT2022	Adaptive-FL (Ours)	99.60 ± 0.06	98.27 ± 0.42	+1.12 pp*	Federated, privacy-preserving
RT-IoT2022	Static-FL	99.43 ± 0.10	97.15 ± 0.41	–	Federated baseline
RT-IoT2022	FedAvg-FL	99.28 ± 0.12	93.15 ± 6.37	−4.00 pp	Highly unstable
IoMT2024	Centralized-XGB	98.82 ± 0.02	94.56 ± 0.15	+0.89 pp	Privacy-agnostic upper bound
IoMT2024	Adaptive-FL (Ours)	98.77 ± 0.03	94.21 ± 0.30	+0.55 pp*	Federated, privacy-preserving
IoMT2024	Static-FL	98.67 ± 0.03	93.67 ± 0.43	–	Federated baseline
IoMT2024	FedAvg-FL	98.32 ± 0.46	91.11 ± 3.76	−2.55 pp	Highly unstable

Note: *significant at $p = 0.031$ (Wilcoxon signed-rank, Table 9).

The results highlight two key findings. First, FedAvg-FL performs substantially below all other federated methods, with Macro-F1 dropping to 81.9% on one RT-IoT2022 fold and 84.5% on one IoMT2024 fold. Its instability is also evident from standard deviations of 6.37% on RT-IoT2022 and 3.76% on IoMT2024, confirming that round-level ensemble reset without quality-weighted accumulation is poorly suited to non-IID tree-based federation. Second, the performance gap between Adaptive-FL and the centralized upper bound remains small at only 0.67 percentage points on RT-IoT2022, where Adaptive-FL achieves 98.27% compared to 98.94%, and 0.35 percentage points on CIC IoMT 2024, where Adaptive-FL achieves 94.21% compared to 94.56%. This shows that adaptive federated training recovers nearly all centralized performance while preserving complete data locality.

6.5 Component Ablation Study

To identify which design choices drive the gains of Adaptive-FL, we evaluate three ablation variants under the best-λ configuration for each dataset (S2 for RT-IoT2022, S4 for IoMT2024):

- **Adaptive-NoLinUCB:** Replaces the LinUCB controller with uniform random action selection. Isolates the contribution of learned action selection.
- **Static-Heavy:** Uses the Heavy action every round (depth=7, 160 rounds, 5 clients). Tests whether the single best fixed configuration matches full adaptation.
- **Adaptive-EqualWeight:** Uses LinUCB but replaces quality-weighted ensemble aggregation with equal-weight averaging. Isolates the contribution of quality weighting.

Results are reported in Table 11.

Table 11: Component ablation study (Macro-F1, mean ± std across 5 folds).

Dataset	Method	Macro-F1 (%)	Δ vs. Adaptive-FL	Wilcoxon p vs. Static-FL
RT-IoT2022	Adaptive-FL (Full)	98.27 ± 0.42	–	0.031*
RT-IoT2022	Adaptive-EqualWeight	98.24 ± 0.42	−0.03 pp	0.031*
RT-IoT2022	Adaptive-NoLinUCB	98.07 ± 0.49	−0.20 pp	0.031*
RT-IoT2022	Static-Heavy	97.33 ± 0.67	−0.94 pp	0.313 (n.s.)
RT-IoT2022	Static-FL	97.15 ± 0.41	−1.12 pp	–
IoMT2024	Adaptive-FL (Full)	94.21 ± 0.30	–	0.031*
IoMT2024	Adaptive-NoLinUCB	94.22 ± 0.31	+0.01 pp	0.031*
IoMT2024	Adaptive-EqualWeight	94.13 ± 0.34	−0.08 pp	0.031*
IoMT2024	Static-Heavy	94.32 ± 0.54	+0.11 pp	0.063 (n.s.)
IoMT2024	Static-FL	93.67 ± 0.43	−0.55 pp	–

Note: *significant at $\alpha = 0.05$; n.s. = not significant.

Three observations emerge. First, removing LinUCB (Adaptive-NoLinUCB) reduces Macro-F1 by 0.20 pp on RT-IoT2022, confirming that learned action selection provides a measurable benefit on this dataset. On IoMT2024, the gap is negligible (<0.01 pp), which aligns with the more uniform action distribution observed on that dataset (Section 6.2.3). Second, Static-Heavy (the strongest fixed-action configuration) is not statistically better than Static-FL on either dataset ($p = 0.313$ and $p = 0.063$), whereas full Adaptive-FL is significant on both. This demonstrates that the benefit of AA-FL comes from *adaptive* configuration choice, not simply from higher model capacity. Third, quality-weighted ensemble aggregation contributes 0.03 pp on RT-IoT2022 and 0.08 pp on IoMT2024; the larger effect on IoMT2024 is consistent with its more pronounced class imbalance and client heterogeneity.

6.6 Sensitivity Analysis

We conduct a one-at-a-time sensitivity analysis on RT-IoT2022 (S2 λ configuration) to assess how Adaptive-FL responds to changes in key hyperparameters. RT-IoT2022 is used as the sensitivity testbed because its smaller size (123K vs. 705K samples) allows multiple re-runs within a reasonable computational budget; the main experiment confirmed consistent trends on both datasets. For each parameter, the default value is held fixed while one parameter is varied; Table 12 summarises the results.

Table 12: One-at-a-time sensitivity analysis (RT-IoT2022, S2 λ, Macro-F1 mean ± std).

Parameter	Value	Macro-F1 (%)	Note
β_{end}	0.1	98.08 ± 0.58	Strong heterogeneity
	0.3 (default)	98.27 ± 0.41	
	0.5	98.12 ± 0.54	
	0.9	98.23 ± 0.44	Near-IID
K (clients)	5	98.76 ± 0.23	Fewer, larger shards
	10 (default)	98.24 ± 0.37	
	20	97.32 ± 0.42	More diluted shards
T (rounds)	5	97.73 ± 0.85	Insufficient
	10	98.31 ± 0.32	
	15 (default)	98.25 ± 0.37	
	20	98.21 ± 0.32	
α (LinUCB)	0.5	98.25 ± 0.43	Low exploration
	1.0	98.27 ± 0.41	
	2.0 (default)	98.25 ± 0.37	
	3.0	98.19 ± 0.42	High exploration

The model is robust across all four axes. The F1 range across all non-default α values is only 0.08 pp, confirming that the controller is insensitive to the exploration–exploitation balance once at least a few rounds of exploration have occurred. The β_{end} range is 0.19 pp; the best value (0.3) corresponds to the selected default. The model degrades slightly with $K = 20$ clients (−0.92 pp vs. $K = 10$), likely because smaller per-client shards increase noise in the local F1 estimates used for weighting. Finally, 5 rounds is insufficient (−0.52 pp) but performance plateaus after 10 rounds, justifying the default of 15.

6.7 Efficiency and Scalability Analysis

Table 13 reports the mean wall-clock training time per round and the absolute communication and computational cost proxies for each action, measured from the experimental logs. Light is the cheapest action (≈6.7 s on RT-IoT2022, comm proxy 5899) while Heavy is the most expensive (≈27.5 s, proxy 41,949). The adaptive controller implicitly balances these costs through the multi-objective reward: by selecting Light and Balanced most frequently (20% each; Table 7), it keeps the average per-round training time well below the Heavy ceiling while maintaining high classification performance. As shown in Fig. 11, the proposed Adaptive-FL framework exhibits stable performance across different values of β, K, T, and α, demonstrating robustness to hyperparameter selection and consistently outperforming the Static-FL baseline.

Table 13: Per-action efficiency metrics (mean wall-clock training time and cost proxies from experimental logs).

Dataset	Action	Clients	Depth	Rounds	Train Time (s)	Comm Proxy
RT-IoT2022	Light	3	4	60	6.7	5899
RT-IoT2022	Balanced	4	6	120	17.4	22,023
RT-IoT2022	Diverse	4	6	180	18.2	33,035
RT-IoT2022	Random	4	6	180	24.0	33,035

(Continued)

Table 13 (continued)

Dataset	Action	Clients	Depth	Rounds	Train Time (s)	Comm Proxy
RT-IoT2022	Largest	4	6	180	37.9	33,035
RT-IoT2022	Heavy	5	7	160	27.5	41,949
IoMT2024	Light	3	4	60	22.8	4914
IoMT2024	Balanced	4	6	120	55.2	18,344
IoMT2024	Diverse	4	6	180	77.3	27,516
IoMT2024	Random	4	6	180	82.4	27,516
IoMT2024	Largest	4	6	180	121.5	27,516
IoMT2024	Heavy	5	7	160	410.2	34,940

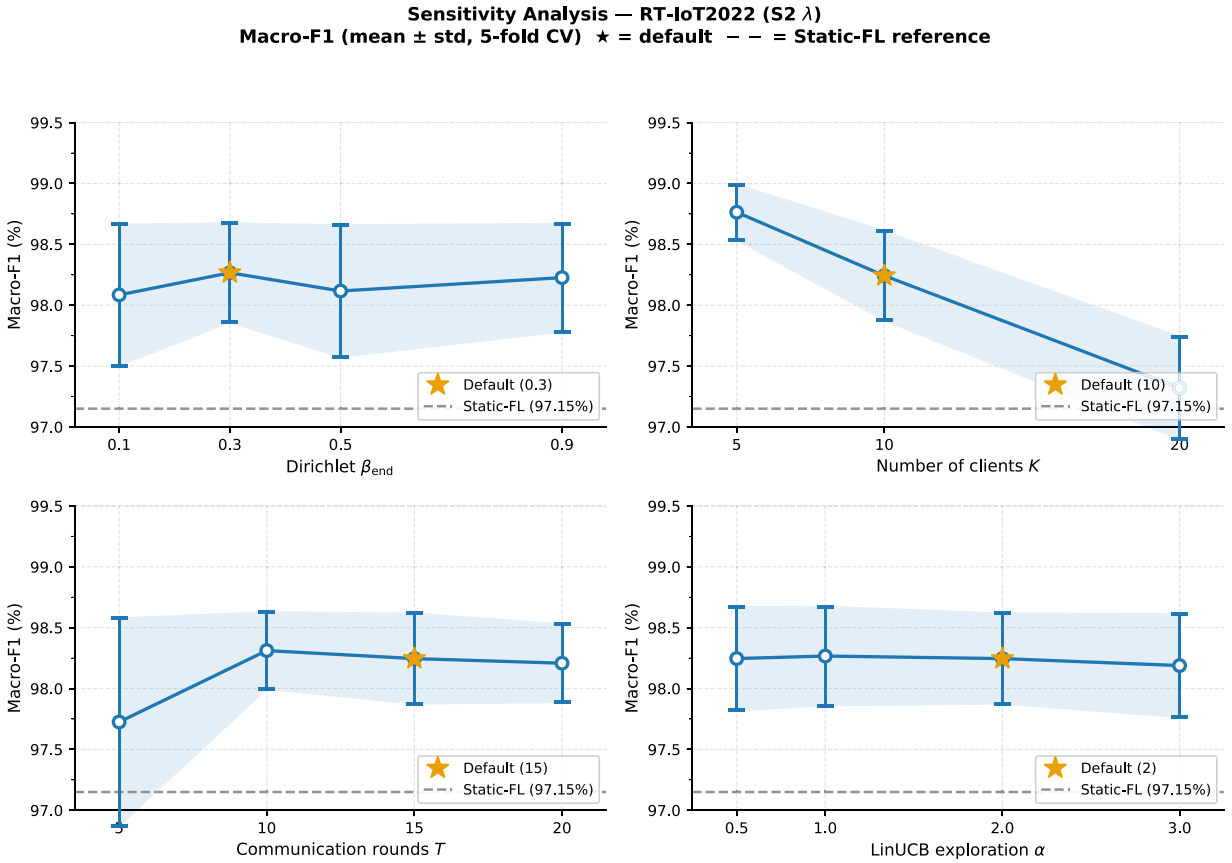


Figure 11: One-at-a-time sensitivity analysis on RT-IoT2022 (S2 λ configuration). Each panel varies one parameter while holding the others at their default values. Error bars show std across 5 folds. The gold star marks the default value; the dashed grey line marks Static-FL performance (97.15%). The model is robust to all four parameters within practical ranges.

Inference cost. At inference time, the ensemble contains up to $M_{\max} = 60$ XGBoost boosters. Each booster prediction is a single forward pass through a depth-≤7 tree with at most 160 leaves. On both datasets, ensemble inference on the server validation set (up to 5000 samples) completed in under 0.5 s. The inference

cost scales linearly with ensemble size and is independent of the number of federated clients, making it suitable for gateway or fog-node deployment.

Communication note. The communication proxy is defined as $\hat{c}_{comm} = K \cdot B \cdot (D_{depth} + 1) \cdot \log_2(F)$, where K is the number of participating clients, B is the number of boosting rounds, D_{depth} is the XGBoost maximum tree depth, and F is the feature dimensionality. This proxy is used only as a relative measure for reward normalization, not as an absolute byte count. Actual model transfer costs depend on the XGBoost serialization format; in practice, each booster transmits only its tree structure (not raw data), which is substantially smaller than transferring gradient tensors in neural-network FL.

6.8 Comparing with Existing Research

In this section, we further validate the performance of our proposed Adaptive-FL by comparing it with existing FL models proposed by researchers. To ensure a fair comparison, we focused on research that used the RT-IoT2022 and CIC IoMT 2024 datasets. Fig. 12 compares the efficiency of the six actions in terms of training time and communication cost. The results show that the Light and Balanced actions are the most resource-efficient, explaining why they are selected most frequently by the adaptive controller.

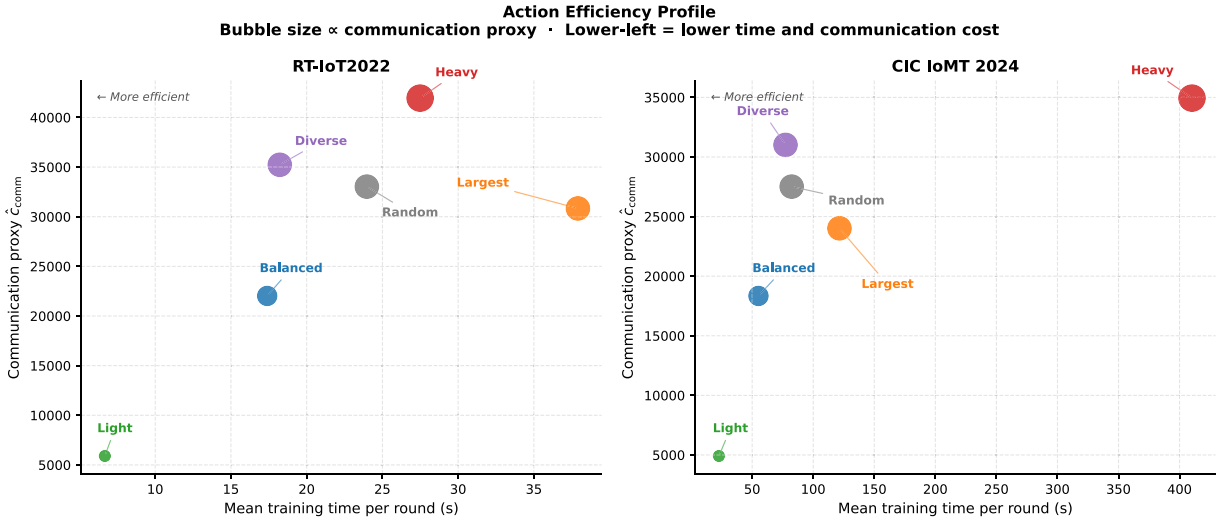


Figure 12: Action efficiency profile: mean training time per round (x-axis) vs. communication proxy (y-axis) for each action on RT-IoT2022 (left) and CIC IoMT 2024 (right). Bubble size is proportional to the communication proxy. Actions in the lower-left quadrant (Light, Balanced) are most efficient; the adaptive controller implicitly favors them by selecting them most frequently (40% combined, Tables 7 and 8).

For the CIC IoMT 2024 dataset, Torre et al. [5] achieved 95.63% classification accuracy in their proposed model, our Adaptive-FL model reached a classification accuracy of 98.77%. Although their reported F1-score was 95.16%, it was not stated whether it is Weighted-F1 or Macro-F1. We report Macro-F1 because it treats all classes equally. Regarding the RT-IoT2022 dataset, Torre et al. [5] reported a classification accuracy of 99.47%, while our model achieved 99.60%. On the F1-score, we achieved 98.27% on Macro-F1 while they achieved 97.31%.

Kouassi et al. [8] evaluated their proposed model on the CIC IoMT 2024 dataset and reported 90.51% and 90.50% on classification accuracy and F1-score, respectively. Our proposed model clearly outperforms their results.

Tawfik et al. [4] achieved 99.9% in classification accuracy and F1-score on CICIoMT2024. However, in their evaluation, they did not specify what type of F1-score was used. Our proposed model is evaluated using Macro-F1 under stratified cross-validation, providing a more meaningful measure of per-class detection performance. Furthermore, they employed a static federated training setup with fixed client participation and uniform training configurations across all rounds. In contrast, our model uses XGBoost boosters governed by a LinUCB contextual bandit controller that dynamically adapts both client participation and local training configurations at each communication round, enabling more efficient and responsive federated training under heterogeneous non-IID conditions.

On the RT-IoT2022 dataset, Islam et al. [3] achieved 99.17% classification accuracy and 95.27% F1-score without Differential Privacy using 10 fog clients. With Differential Privacy ($\varepsilon = 5.0$), accuracy slightly decreased to 98.46%; however, the authors did not report F1-score under DP settings. Our proposed model demonstrates superior performance across both metrics.

Table 14 summarizes the comparison between the proposed Adaptive-FL model and existing FL models. Overall, our model achieves competitive or better performance across both datasets. In addition, it dynamically adjusts client participation and local training at each communication round using a LinUCB contextual bandit controller. Furthermore, the proposed Adaptive-FL model uses Macro-F1 as the primary evaluation metric, which gives equal weight to all classes and provides a more reliable measure of detection performance under class imbalance.

Table 14: Comparison of proposed Adaptive-FL with existing FL models.

Reference	Dataset	Accuracy (%)	F1-Score (%)	F1 Type
Torre et al. [5]	CIC IoMT 2024	95.63	95.16	Not Specified
Adaptive-FL (Ours)	CIC IoMT 2024	98.77	94.21	Macro-F1
Torre et al. [5]	RT-IoT2022	99.47	97.31	Not Specified
Adaptive-FL (Ours)	RT-IoT2022	99.60	98.27	Macro-F1
Kouassi et al. [8]	CIC IoMT 2024	90.51	90.50	Not Specified
Adaptive-FL (Ours)	CIC IoMT 2024	98.77	94.21	Macro-F1
Tawfik et al. [4]	CIC IoMT 2024	99.90	99.90	Not Specified
Adaptive-FL (Ours)	CIC IoMT 2024	98.77	94.21	Macro-F1
Islam et al. [3]	RT-IoT2022	99.17	95.27	Not Specified
Islam et al. (DP, $\varepsilon = 5.0$) [3]	RT-IoT2022	98.46	–	–
Adaptive-FL (Ours)	RT-IoT2022	99.60	98.27	Macro-F1

6.9 Limitations

The current evaluation is subject to several limitations that should be addressed in future work. First, the experiments are limited to two public IoT network traffic datasets. Although RT-IoT2022 and CIC IoMT 2024 cover complementary IoT domains, further validation on additional IoT (Internet of Things) and IIoT (Industrial Internet of Things) datasets is required to assess broader generalization. Second, the experimental configuration explores a finite range of federated settings. In particular, the sensitivity analysis in Section 6.6 evaluates three client counts (5, 10, and 20) and four communication-round settings (5, 10, 15, and 20) on RT-IoT2022. Therefore, the behavior of the proposed framework under substantially larger client populations, highly dynamic participation, and more extensive action-space exploration remains to be investigated. Third, communication overhead and computational cost are estimated using proxy functions rather than measured on physical IoT devices. Real deployment on resource-constrained hardware may reveal additional

bottlenecks in latency, memory, energy, and communication. Fourth, the baseline comparison is limited to Static-FL, FedAvg-FL, and a centralized upper bound; to the best of our knowledge, no published methods currently jointly optimize both client selection and local training workload in a tree-based federated setting.

7 Conclusions

In conclusion, this study proposed an AA-FL framework for intrusion detection in heterogeneous IoT environments. Unlike traditional federated learning approaches that rely on fixed client participation and static training configurations with parameter averaging, the proposed framework dynamically adapts its behavior at each communication round. Specifically, it decides which clients should participate, how complex their local models should be, and how much training to perform locally. These decisions are guided by a LinUCB contextual bandit controller that learns from past performance and selects the most suitable strategy at each round. The framework incorporates four enhancements: quality-weighted ensemble aggregation, EMA-smoothed delta-F1 reward, annealed Dirichlet heterogeneity, and a ten-feature context vector. Client contributions are aggregated through a server-side XGBoost ensemble using quality-weighted probability averaging rather than weight averaging, which improves stability under non-IID data distributions. A Static-FL baseline was included as a reference to isolate the contribution of the adaptive mechanism by keeping all other components identical and varying only the client selection and action configuration strategy. Extended experiments with FedAvg-FL and a centralized upper bound confirm that the round-level ensemble reset in FedAvg is highly unstable under non-IID conditions, with standard deviations exceeding 3.7% on both datasets. In contrast, Adaptive-FL achieves a Macro-F1 of 98.27% on RT-IoT2022 and 94.21% on CIC IoMT 2024, falling within 0.67 and 0.35 percentage points of the centralized upper bounds of 98.94% and 94.56%, respectively, without transferring any raw data. A component ablation study shows that the LinUCB controller contributes 0.20 pp on RT-IoT2022 and quality-weighted aggregation contributes 0.03 to 0.08 pp, with both ablation variants still outperforming Static-FL on both datasets. The proposed Adaptive-FL model also delivers statistically significant improvements over Static-FL on both datasets while operating under decentralized and privacy-preserving constraints. These results confirm that the adaptive mechanism provides a measurable benefit on RT-IoT2022 and recovers near-centralized performance on CIC IoMT 2024 without sharing raw data. Future work can extend the framework to support heterogeneous local clients with different hardware capabilities and to use model types beyond XGBoost. In addition, other adaptive controllers besides LinUCB could be explored to better respond to changing network conditions.

Acknowledgement: The authors acknowledge DSR at King Abdulaziz University with thanks for its technical and financial support.

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).

Author Contributions: Conceptualization, Abdulaziz A. Alsulami, Badraddin Alturki, and Ahmad J. Tayeb; methodology, Abdulaziz A. Alsulami and Qasem Abu Al-Haija; software, Abdulaziz A. Alsulami; Validation, Qasem Abu Al-Haija, Rayed Alakhtar, Huda Alsobhi, Rayan A. Alsemmeari, and Ahmad J. Tayeb; formal analysis, Ahmad J. Tayeb; investigation, Badraddin Alturki; Resources, Abdulaziz A. Alsulami; Data curation, Abdulaziz A. Alsulami; writing—original draft preparation, Abdulaziz A. Alsulami, Qasem Abu Al-Haija, Rayed Alakhtar, Huda Alsobhi, Badraddin Alturki, Rayan A. Alsemmeari, and Ahmad J. Tayeb; writing—review and editing, Qasem Abu Al-Haija, Rayed Alakhtar, Huda Alsobhi, Rayan A. Alsemmeari, and Badraddin Alturki; visualization, Badraddin Alturki; supervision, Abdulaziz A. Alsulami; project administration, Badraddin Alturki and Ahmad J. Tayeb; funding acquisition, Abdulaziz A. Alsulami. 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. The CIC IoMT 2024 WiFi MQTT dataset is available at https://www.unb.ca/cic/datasets/iomt-dataset-2024.html and is formally published in [24]. The RT-IoT2022 dataset is available through the UCI Machine Learning Repository at https://archive.ics.uci.edu/dataset/942/rt-iot2022 and is formally published in [23].

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflict of interests.

References

1. IoT Analytics. State of IoT 2025: number of connected IoT devices growing 14% to 21.1 billion globally. IoT Analytics GmbH; 2025 [cited 2026 Jun 11]. Available from: https://iot-analytics.com/number-connected-iot-devices/.
2. Zscaler ThreatLabz. 2023 enterprise IoT and OT threat report. Zscaler, Inc.; 2023 [cited 2026 Jun 11]. Available from: https://www.zscaler.com/resources/2023-threatlabz-enterprise-iot-ot-threat-report.
3. Islam MM, Abdullah WM, Saha BN. Privacy-preserving hierarchical fog federated learning (PP-HFFL) for IoT intrusion detection. Sensors. 2025;25(23):7296. doi:10.20944/preprints202510.1115.v1.
4. Tawfik M, Abu-Ein AA, Noaman HM, Abdelhaliem AH, Fathi IS. FedMedSecure: federated few-shot learning with cross-attention mechanisms and explainable AI for collaborative healthcare cybersecurity. Sci Rep. 2025;15:40050. doi:10.21203/rs.3.rs-7208692/v1.
5. Torre D, Chennamaneni A, Jo J, Vyas G, Sabrsula B. Toward enhancing privacy preservation of a federated learning CNN intrusion detection system in IoT: method and empirical study. ACM Trans Softw Eng Methodol. 2025;34(2):53. doi:10.1145/3695998.
6. Buyuktanir B, Gozde SA, Yildiz K. Federated learning in intrusion detection: advancements, applications, and future directions. Cluster Comput. 2025;28(7):473. doi:10.1007/s10586-025-05325-w.
7. Nguyen VT, Beuran R. FedMSE: semi-supervised federated learning approach for IoT network intrusion detection. Comput Secur. 2025;151:104337.
8. 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;17:529.
9. Abd Elaziz M, Fares IA, Dahou A, Shrahili MM. Federated learning framework for IoT intrusion detection using tab transformer and nature-inspired hyperparameter optimization. Front Big Data. 2025;8:1526480. doi:10.3389/fdata.2025.1526480.
10. Shaker BN. Federated learning for early detection of advanced persistent threats in IoT networks. AlKadhim J Comput Sci. 2025;3(4):100–8. doi:10.61710/kjcs.v3i4.140.
11. Rampone G, Ivaniv T, Rampone S. A hybrid federated learning framework for privacy-preserving near-real-time intrusion detection in IoT environments. Electronics. 2025;14(7):1430. doi:10.3390/electronics14071430.
12. Tamuka N, Mathonsi TE, Olwal TO, Maswikaneng S, Muchenje T, Tshilongamulenzhe TM. A comparative analysis of self-aware reinforcement learning models for real-time intrusion detection in fog networks. Future Internet. 2026;18(2):100. doi:10.3390/fi18020100.
13. Mutambik I, Almuqrin A. Balancing efficiency and efficacy: A contextual bandit-driven framework for multi-tier cyber threat detection. Appl Sci. 2025;15(11):6362.
14. Kaur A. Intrusion detection approach for industrial internet of things traffic using deep recurrent reinforcement learning assisted federated learning. IEEE Trans Artific Intel. 2025; 6(1):37–50. doi:10.1109/TAI.2024.3443787.
15. Al-Haija QA, Tamimi SA. A state-of-the-art survey of adversarial reinforcement learning for IoT intrusion detection. Comput Mater Contin. 2026;87(1):2. doi:10.32604/cmc.2025.073540.
16. Aydin B, Aydin H, Gormus S. Intrusion detection systems in IoT: a detailed review of threat categories, detection strategies, and future technologies. J Inf Secur Appl. 2025;95:104291.

17. Khraisat A, Alazab A. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecurity. 2021;4(1):18. doi:10.1186/s42400-021-00077-7.
18. Albanbay N, Tursynbek Y, Graffi K, Uskenbayeva R, Kalpeyeva Z, Abilkaiyr Z, et al. Federated learning-based intrusion detection in IoT networks: performance evaluation and data scaling study. J Sens Actuator Netw. 2025;14(4):78.
19. Shalan M, Hasan MR, Bai Y, Li J. Enhancing smart home security: blockchain-enabled federated learning with knowledge distillation for intrusion detection. Smart Cities. 2025;8(1):35.
20. Hamad NA, Bakar KA, Qamar F, Jubair AM, Mohamed RR, Mohamed MA. Systematic analysis of federated learning approaches for intrusion detection in the Internet of Things environment. IEEE Access. 2025;13:95410–44. doi:10.1109/access.2025.3574672.
21. Patel H, Amalapuram SK, Kumar S, Tamma BR. IoT-FedMalDetect: federated learning based malware detection for IoT edge devices. In: Proceedings of the 2025 IEEE Conference on Communications and Network Security (CNS); 2025 Sep 8–11; Avignon, France. p. 1–9.
22. Ceran O, Özdoğan E, Uysal M. Leveraging graph neural networks for IoT attack detection. Sakarya Univ J Comput Inf Sci. 2025;8(2):223–44. doi:10.35377/saucis...1663435.
23. Sharmila B, Nagapadma R. Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset. Cybersecurity. 2023;6(1):41. doi:10.1186/s42400-023-00178-5.
24. Dadkhah S, Neto ECP, Ferreira R, Molokwu RC, Sadeghi S, Ghorbani AA. CICIoMT2024: a benchmark dataset for multi-protocol security assessment in IoMT. Internet Things. 2024;28:101351.
25. Yurochkin M, Agarwal M, Ghosh S, Greenewald K, Hoang N, Khazaeni Y. Bayesian nonparametric federated learning of neural networks. In: Proceedings of the 36th International Conference on Machine Learning; 2019 Jun 9–15; Long Beach, CA, USA. p. 7252–61.
26. Shannon CE. A mathematical theory of communication. Bell System Tech J. 1948;27(3):379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.
27. Orlandi FC, Dos Anjos JC, Leithardt VR, Santana JFDP, Geyer CF. Entropy to mitigate non-IID data problem on federated learning for the edge intelligence environment. IEEE Access. 2023;11:78845–57. doi:10.1109/access.2023.3298704.
28. Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016 Aug 13–17; San Francisco, CA, USA. p. 785–94.
29. Li L, Chu W, Langford J, Schapire RE. A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web; 2010 Apr 26–30; Raleigh North, CA, USA. p. 661–70.