



Federated Learning-Based Intrusion Detection System for Edge-Enabled IoT

Hassan W. Hilou¹, Wasal S AL-Bash AL-Azzawi², Nawfal H. Warush³, Darin Shafek^{4,5*},
Aghiad Khedr Suleiman⁶

¹ Computer Engineering Techniques, Al-Ma'moon University College, Baghdad 10008, Iraq

² Computer Engineering Technology, Al-Salam University College, Baghdad 10008, Iraq

³ Mechanical Engineering, Al-Nahrain University, Baghdad 10008, Iraq

⁴ Computer Engineering Techniques Department, Al-Ma'moon University College, Baghdad 10008, Iraq

⁵ Computer and Automatic Control Engineering Department, Latakia University, Latakia 9003, Syria

⁶ Cybersecurity and Cloud Computing, Technologies Engineering Department, Al-Ma'moon University College, Baghdad 10008, Iraq

Corresponding Author Email: darin.s.salim@almamonuc.edu.iq

Copyright: ©2026 The authors. This article is published by IETA and is licensed under the CC BY 4.0 license
(<http://creativecommons.org/licenses/by/4.0/>).s

<https://doi.org/10.18280/jesa.590823>

ABSTRACT

Received: 19 June 2026

Revised: 15 August 2026

Accepted: 24 August 2026

Available online: 31 August 2026

Keywords:

federated learning, intrusion detection, Internet of Things, edge computing, few-shot fine-tuning, non-independent and identically distributed data

The increasing adoption of the Internet of Things (IoT) has brought considerable benefits to all aspects of daily life, but IoT deployments remain vulnerable to a wide range of security threats that interrupt device operation and cause significant losses. Traditional intrusion detection systems (IDS) are usually centralized: raw traffic records are transported from the source devices to a central server for classification, which raises data-privacy concerns and consumes backbone bandwidth. This work proposes HFed-IDS, a hierarchical federated intrusion detection system for edge-enabled IoT. Devices train a compact multilayer perceptron (MLP) locally, edge aggregators combine the client updates, and a central server applies federated averaging (FedAvg) to the edge models only. A lightweight few-shot fine-tuning stage then adapts the converged global model to attack categories that are rare or entirely absent from federated training. The system is evaluated on the UNSW-NB15 dataset (82,332 records, ten classes) under a stratified 70/15/15 train/validation/test protocol in which every preprocessing statistic is estimated on the training split alone, and all results are reported as the mean $\pm$ standard deviation over five random seeds. The hierarchical federated model reaches $82.68 \pm 1.46\%$ accuracy and $80.13 \pm 2.14\%$ weighted F1, which is 3.05 percentage points below an identically configured centralized model ($85.73 \pm 0.19\%$) and statistically indistinguishable from flat FedAvg. Because only the 2 edge aggregators contact the server, cumulative backbone traffic over the 30 training rounds falls from 9.23 MB for raw-data upload and 4.48 MB for flat FedAvg to 0.90 MB, reductions of 90.3% and 80.0%, while the additional device-to-edge traffic remains on the local link. In a leave-one-attack-out study, an attack class withheld from federated training is undetectable (0% recall) until fine-tuning, and recovers to between 33% and 73% recall from 50 labelled samples. Per-class metrics are also reported and identify the rarest categories as the principal remaining limitation.

1. INTRODUCTION

The rapid growth of the Internet of Things (IoT) has connected billions of devices, from simple home sensors to complex industrial control units [1]. This large number of devices and connections widens the attack surface available to adversaries [2]. Robust intrusion detection systems (IDS) are therefore essential for protecting digital infrastructure, particularly in environments that handle sensitive data such as the financial and military sectors. Most deployed security systems rely on centralized architectures in which data are sent from every connected device to a data center for processing, analysis, and modelling [3]. Such architectures raise connectivity costs, consume backbone bandwidth, and increase concerns about data privacy [4].

To address these challenges, federated learning (FL) has emerged [5]. FL allows multiple clients to train a shared model locally on their own data without transferring that data to a central server [6]. The most widely used FL algorithm is federated averaging (FedAvg), in which the global parameters are obtained by aggregating the locally updated weights [7]. Keeping raw records on the device removes the need to transmit them and therefore reduces their exposure in transit, but it does not by itself provide a formal privacy guarantee; the threat model assumed in this work is stated explicitly in Section 3.8. FL also introduces its own difficulties, including the volume of communication required between the central server and thousands of connected devices, and limited flexibility in adapting to threats that appear at an individual node [8].

Deep learning models also require large datasets to track dynamic changes in traffic patterns [9]. When those data are divided into small per-client shards, such models may generalize poorly [10]. It is therefore necessary to develop a system that can operate under small-sample conditions while keeping raw records on the device.

This paper proposes HFed-IDS, a hierarchical federated intrusion detection system for edge-enabled IoT. Inserting edge aggregators between the IoT devices and the central server reduces the number of model exchanges that traverse the backbone link, while a post-aggregation few-shot fine-tuning stage adapts the converged global model at the device level using a small labelled support set. It should be stated at the outset that this adaptation stage is ordinary supervised fine-tuning: no distribution over tasks is sampled, no episodes are constructed, and no outer-loop objective is optimized. The method is therefore not meta-learning in the sense of Finn et al. [11], and the name used in the previous version of this manuscript, FFS-IDS, has been changed accordingly.

The contributions of this work are the following. (1) A client-edge-cloud hierarchical federated intrusion detection architecture is implemented and evaluated on UNSW-NB15 under a fully specified train/validation/test protocol whose preprocessing statistics are estimated on the training split alone. (2) The proposed system is compared with three baselines that share the same network architecture, the same data partition, and the same held-out test set: a centralized model, flat FedAvg, and hierarchical FedAvg without fine-tuning. (3) Communication cost is accounted cumulatively over all training rounds and is separated into device-to-edge and edge-to-cloud traffic, instead of being quoted for a single model transfer. (4) Per-class precision, recall, and F1 are reported together with a confusion matrix, and the averaging convention is stated explicitly, so that the effect of class imbalance is visible. (5) A leave-one-attack-out study quantifies how well the system adapts to an attack category that was never observed during federated training. (6) Every experiment is repeated over five random seeds and reported as a mean with a standard deviation.

2. RELATED WORK

In recent years, the integration of FL with IoT security has attracted significant attention. Early work focused on basic aggregation methods such as FedAvg, which enable decentralized training but suffer from a high communication cost [12]. More recent studies have moved towards hierarchical architectures. Ismael [13] introduced the client-edge-cloud hierarchical FedAvg scheme and showed that inserting edge aggregators between the devices and the cloud reduces the number of model exchanges that cross the backbone link, at the cost of additional traffic on the local device-to-edge links. Lazzarini et al. [14] applied a related hierarchical design with an attention-based aggregation rule to heterogeneous IoT networks, and Liu et al. [15] discussed edge-computing deployments for IoT more generally. The architecture used in the present work follows the client-edge-cloud formulation of [13]. Other approaches target data heterogeneity. FedKF uses a Kalman filter for weight estimation [16], and FD-IDS applies knowledge distillation [17]; both address non-independent and identically distributed (non-IID) data and device heterogeneity. Moustafa et al. [18] used device-type-specific behavioural models in a federated

setup to detect IoT malware. To address the challenge of limited labelled data, Moustafa et al. [19] introduced a prototypical-network few-shot learning classifier and reported over 99% accuracy on a binary formulation with very few training samples per class. Munirathinam [20] combined FL with a long short-term memory (LSTM) network to detect false data injection attacks in distributed solar farms, and reported improved detection relative to a centralized model built on the same LSTM architecture. Their evaluation targets a power-system dataset rather than general IoT network traffic, so the comparison with the present work is indicative rather than direct.

Two related lines of work should be distinguished from the approach taken here. The first is meta-learning, in which an outer loop optimizes an initialization across sampled tasks so that a few inner gradient steps suffice on a new task [11]. The second is personalized FL, in which a shared global model is specialized for each client [21]. The adaptation stage used in this paper belongs to the second family in its objective and is implemented as plain supervised fine-tuning of the last layers of the converged global model; no meta-objective is optimized at any point.

3. METHODOLOGY

This section presents the methodology of the proposed HFed-IDS. The method combines FedAvg over a client-edge-cloud topology, which trains a compact artificial neural network as a global model, with a lightweight few-shot fine-tuning stage that specializes that global model at the edge.

FedAvg is an algorithm based on federated stochastic gradient descent (FedSGD) [22]. FedSGD is the baseline of FedAvg: a randomly selected client computes a single-batch gradient in each round, and the averaged gradient is sent to the server, which aggregates it and updates the global weights. FedAvg generalizes FedSGD in that each client performs several local update steps and returns the resulting weights rather than gradients [23]. Through this process, multiple clients train collectively on their own data without uploading raw records to a server, which is the principal motivation for FL.

3.1 Problem formulation

The evaluation uses the UNSW-NB15 dataset [24], which is represented as $D = \{(x_i, y_i)\}_{i=1}^N$.

Where: $x \in \mathbb{R}^d$: Features vector. $y \in \{1, 2, \dots, C\}$: Attack labels, which are shown in Table 1.

3.2 Dataset and experimental protocol

The experiments use the 82,332-record partition of UNSW-NB15 [24–26] whose class distribution is given in Table 1. The previous version of this manuscript did not state how this partition was divided, so the protocol is defined explicitly here. The records are split once, by stratified random sampling, into three disjoint subsets: 70% for training (57,632 records), 15% for validation (12,350 records), and 15% for testing (12,350 records). The per-class sizes of the three subsets are listed in Table 1. Federated clients are formed exclusively from the training split. The validation split is used to monitor convergence and to select the number of local epochs and the number of fine-tuning steps; the test split is used once to

produce the numbers reported in Section 4, and never for any tuning decision.

Table 1. Class distribution of the UNSW-NB15 partition used in this study and the sizes of the stratified training, validation, and test splits

Attack Category	Total	Training	Validation	Test
Analysis	677	474	101	102
Backdoor	583	408	88	87
DoS	4,089	2,862	614	613
Exploits	11,132	7,792	1,670	1,670
Fuzzers	6,062	4,243	910	909
Generic	18,871	13,210	2,830	2,831
Normal	37,000	25,900	5,550	5,550
Reconnaissance	3,496	2,447	524	525
Shellcode	378	265	56	57
Worms	44	31	7	6
Total	82,332	57,632	12,350	12,350

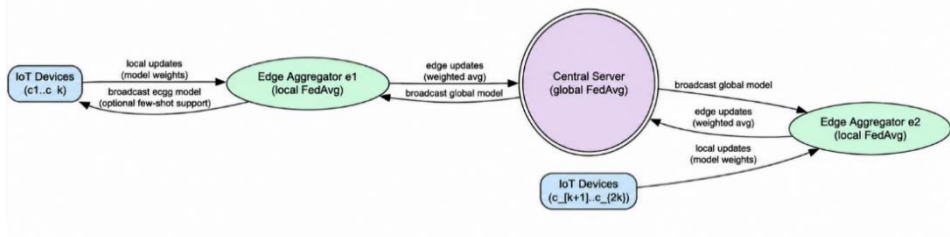


Figure 1. The system architecture

All preprocessing operators, namely the label encoder for the categorical features, the mean imputer, and the standard scaler, are fitted on the training split alone and then applied unchanged to the validation and test splits, so that no statistic derived from evaluation data can influence training. Categorical values that appear only in an evaluation split are mapped to a reserved 'unseen' code instead of triggering a re-fit. Support sets used for few-shot fine-tuning are also drawn exclusively from the training split. The training split is distributed over 10 clients using a Dirichlet label-skew partition with concentration parameter $\alpha = 0.5$, which is the standard procedure for generating non-independent and identically distributed (non-IID) federated data: for every class, the proportion allocated to each client is drawn from a Dirichlet distribution, so both the class mixture and the shard size differ across clients. The clients are assigned in equal numbers to 2 edge aggregators. The full parameter set is listed in Section 4.1, and every experiment is repeated with the five random seeds 0 to 4, which control the split, the partition, and the weight initialization. The network contains a set of edge aggregators $\mathcal{E} = \{e_1, e_2, \dots, e_M\}$ and clients (IoT devices) $\mathcal{C} = \{c_1, c_2, \dots, c_K\}$. Each client c_k has its own local dataset D_k of size n_k , which contains its own data, and it is different from the other clients. The total number of clients used in experiments is $K = 10$. The federated objective is to learn a set of classifier parameters θ that minimizes the empirical risk across all clients without sending any raw data to the datacenter. For a new or existing client that holds only a small labelled set, called the support set, $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{K_s} : K_s \ll n_k$, the few-shot adaptation objective is to compute adapted parameters θ' by a few steps of internal gradient steps so that q performs well on the client's held-out records, called the query set.

3.3 System architecture

A hierarchical architecture is proposed for an IoT network organized as IoT devices, edge aggregators, and a central server. Its objective is to combine hierarchical edge aggregation with a post-aggregation few-shot fine-tuning stage performed at the edge. Figure 1 shows the system architecture. Based on the system architecture shown in Figure 1, clients train locally and send model parameter updates to their assigned edge aggregator. Each edge aggregator computes the sample-weighted average of its clients' updates and repeats this local exchange 2 times before forwarding the aggregated edge model to the central server. The server then computes the sample-weighted average of the edge models (FedAvg) and broadcasts the new global model. The distinguishing element of this design is the intermediate aggregation stage at the edge, which is what removes traffic from the backbone link. After federated convergence, a client that holds a small labelled support set can additionally fine-tune the last layers of the global model locally. One property of the scheme should be stated plainly: with full client participation, sample-weighted averaging, and a single edge round per cloud round, hierarchical aggregation is numerically identical to flat FedAvg. The two schemes differ in accuracy only because 2 edge rounds are performed per cloud round, and the benefit claimed here is a communication benefit rather than an accuracy benefit.

3.4 Data preprocessing

Categorical features and class labels are converted to a numerical form with a label encoder fitted on the training split only, which maps each attack type to a numerical representation:

$$y_i^{enc} = Label_{Encoder(y_i)} \quad (1)$$

The next preprocessing step is imputation, which replaces missing values with a summary statistic, here the mean. Datasets typically contain observations with missing values for one or more variables. The main options for handling missing data are [27]: (1) Exclude observations (rows) that contain any missing values, but this process may result in the loss of information that could be important and useful to the model during the training process. (2) Exclude any variable (column) that contains missing values, but this procedure may also result in the loss of predictive variables that significantly affect model performance. (3) Fill in missing entries with estimated values. This process represents the best procedure for filling in missing values because it avoids the loss of information that could affect performance results. The proposed methodology uses the `simpleImputer()` function to fill in missing values with the arithmetic mean of the values of the variable in which the missing value appears [28]. Assume that $\tilde{x}_i$ denotes the imputed vector, so:

$$x_{i,j} = \begin{cases} x_{i,j} & \text{if } x_{i,j} \text{ observed} \\ \mu_j & \text{if } x_{i,j} \text{ missing} \end{cases} \quad (2)$$

where,

$x_{i,j}$: Represents the value of attribute j in row i .

μ_j : Mean of values in j attribute.

Standardization is the process of scaling data so that all values fall within a specified range, with the mean being 0 and the standard deviation being 1 [29]. This technique is widely used in data preprocessing to ensure that data points across different variables can be compared on the same scale to prevent variables with larger numerical ranges from dominating variables with smaller ranges. The formula for standardization is as follows:

$$x_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \quad (3)$$

where, σ_j is the standard deviation of values in j attribute.

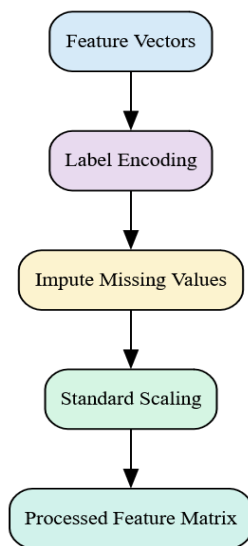


Figure 2. The data preprocessing and feature standardization pipeline

Figure 2 shows the data preprocessing and feature

standardization pipeline.

3.5 Model architecture

The model is a compact multilayer perceptron (MLP) [30] network $f_\theta(x)$ with a small parameter count, chosen so that it can be trained and transmitted efficiently on edge devices. The forward mapping is shown in the following Eq. (4):

$$f_\theta(x) = W_3\sigma(W_2\sigma(W_1x + b_1) + b_2) + b_3 \quad (4)$$

where,

W_i : The learned linear layers.

σ : The RELU activation function.

The proposed model consists of an input layer, two hidden layers, and an output layer. The first hidden layer contains 30 neurons and the second contains 16 neurons, and dropout with probability 0.2 is applied after each hidden layer to limit overfitting. With 42 input features and ten output classes, the network holds 1,956 trainable parameters, which is 7.64 KB when serialized as 32-bit floating-point values. This small parameter count is what makes the model inexpensive to communicate and suitable for deployment on edge devices. Figure 3 shows the model block diagram, while Figure 4 shows the full detailed system architecture. The identical architecture is used for every model compared in Section 4, including the centralized baseline, so that the comparison isolates the training scheme.

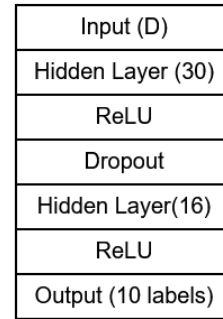


Figure 3. The model block diagram

3.6 Federated learning mechanism

At each global round t , a subset of clients performs local training, starting from the global parameters θ^t . When local training completes, the clients return the updated parameters θ_k^{t+1} . The server then computes the sample-weighted average θ^{t+1} for all θ_k^{t+1} .

$$\theta^{t+1} = \text{Weighted Average}(\theta_k^{t+1}) \quad (5)$$

When using edge aggregators, the server calculates the average of the edge models instead of all the client models.

Assume that the edge e aggregates clients in set $\mathcal{C}_e$ with total samples $n_e = \sum_{k \in \mathcal{C}_e} n_k$. The edge state θ_e is computed as shown in Eq. (6):

$$\theta_e = \text{Weighted Average}(\theta_k) \quad (6)$$

and the server update is:

$$\theta^{t+1} = \text{Weighted Average}(\theta_e) \quad (7)$$

$\sum_k \theta_k \sum_e \theta_e$ hierarchical edge aggregation reduces central communication by reducing the number of messages that traverse the backbone link to the data center. Within one cloud round, each of the 10 clients exchanges its update with its edge aggregator once per edge round, so the device-to-edge traffic grows with the number of clients and with the number of edge

rounds, whereas only the 2 edge aggregators exchange a model with the data center, so the backbone traffic grows with the much smaller number of edges. The cumulative cost of a complete training run, separated by link type, is derived and quantified in Section 4.7.

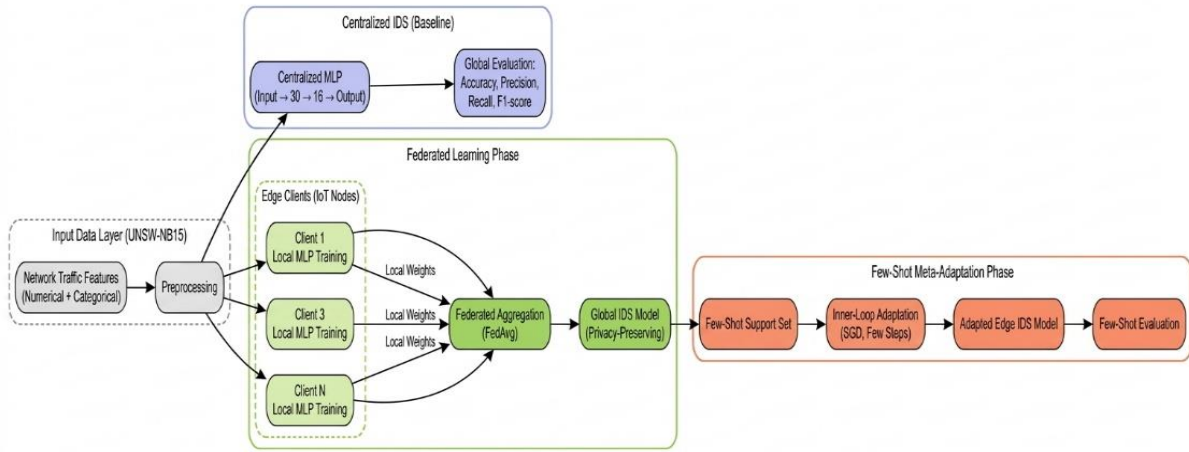


Figure 4. The full detailed system architecture

Figure 5 shows the FL mechanism with hierarchical edge aggregation. In the example of Figure 5, the number of model exchanges on the backbone link to the server is reduced from four to two.

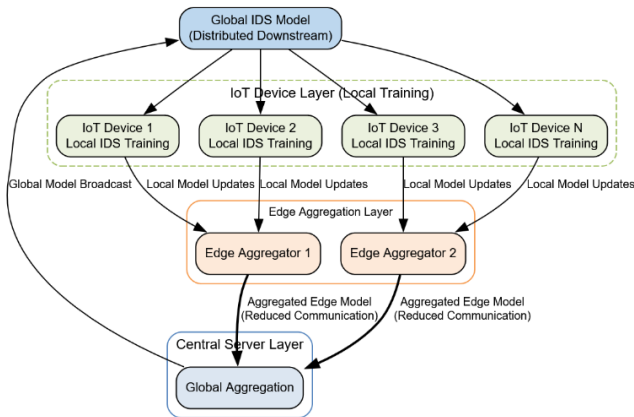


Figure 5. The federated learning (FL) mechanism with hierarchical edge aggregation

3.7 Few-shot fine-tuning for adaptation

The stage described below is supervised fine-tuning of a

converged global model on a small labelled support set. It is deliberately not described as meta-learning: no distribution over tasks is sampled, no episodes are constructed, and no outer-loop objective is optimized. The term few-shot refers only to the size of the support set.

Given global parameters θ , the inner update for a client is shown in the following Eq. (8):

$$\theta' = \theta - \alpha \nabla_{\theta} \mathcal{L}_S(\theta) \quad (8)$$

where,

$\mathcal{L}_S$: The loss on support set.

α : The inner learning rate.

∇_{θ} : The gradient of the loss function with respect to the model parameters θ .

This update is applied to the last two linear layers of the network, with the earlier layers frozen, for 500 full-batch steps at learning rate 0.01. Restricting the update in this way was necessary in practice: updating the whole network on a handful of samples caused catastrophic forgetting of the classes that were already well modelled, while updating only the output layer proved too restrictive to encode a previously unseen class. Both variants were compared on the validation split, and the two-layer variant was selected. Figure 6 shows the few-shot adaptation flow for federated intrusion detection.

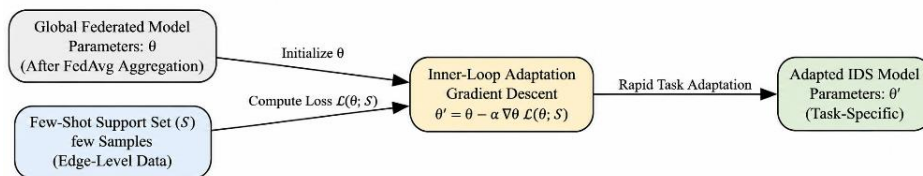


Figure 6. The few-shot adaptation flow for federated intrusion detection

Adaptation is performed after federated convergence. The global model supplies a strong initialization, so specialization for an individual client requires only a small number of labelled samples and avoids a full retraining cycle. As quantified in Sections 4.5 and 4.6, this adaptation trades overall accuracy for a substantial gain in recall on rare and previously unseen attack classes. It is therefore presented as a mechanism for extending coverage to such classes, not as a way of improving aggregate accuracy.

3.8 Evaluation metrics and threat model

The confusion matrix is the basic tool used to assess the quality of a classifier [31]. Figure 7 shows the confusion matrix in general.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Figure 7. The confusion matrix in general

The confusion matrix is used to derive accuracy, precision, recall, and F1 score. Because the dataset is strongly imbalanced, the averaging convention matters and is therefore stated explicitly: weighted averages are computed over the ten classes with weights proportional to class support, whereas macro averages give every class the same weight regardless of its frequency. Both are reported throughout Section 4. Weighted averages are dominated by the Normal and Generic classes and consequently track accuracy closely, while macro averages expose the behaviour on the minority classes; reporting only one of the two would give a misleading picture of an intrusion detection system. The values reported in the previous version of this manuscript were weighted averages.

Threat model and privacy scope. The system assumes an honest-but-curious server and honest-but-curious edge aggregators: every participant follows the protocol but may inspect whatever it receives. Under this model, keeping raw traffic records on the device removes the need to transmit them and therefore removes the corresponding exposure. No formal privacy guarantee is claimed. In particular, the system implements neither differential privacy [32] nor secure aggregation [33], and it is well established that model updates alone can leak information about the underlying training data

through gradient inversion and membership inference attacks [34, 35]. The privacy claims made in this paper are therefore limited to data minimization, meaning that raw records never leave the device. Integrating secure aggregation and a differentially private optimizer, and measuring the resulting utility cost, is left to future work.

$$Precision = \frac{TP}{TP+FP} \quad (9)$$

$$Recall = \frac{TP}{TP+FN} \quad (10)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (11)$$

$$F1 \text{ score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (12)$$

4. RESULTS AND DISCUSSION

This section reports the results of four models. All four use the identical network architecture of Section 3.5, the identical data partition of Section 3.2, and the identical held-out test split, so that the comparison isolates the training scheme:

- Centralized model: a single model trained at the server on the pooled training split.
- FedAvg (flat): standard federated averaging in which every client communicates directly with the central server.
- Hierarchical FedAvg: the global model of the proposed HFed-IDS, in which clients communicate with edge aggregators and only the edge aggregators communicate with the server. No fine-tuning is applied.
- HFed-IDS with few-shot fine-tuning: the hierarchical global model after client-level fine-tuning on a support set of $K = 20$ labelled samples per class.

4.1 Experimental setup

Table 2 lists every parameter needed to reproduce the experiments. The number of local epochs was selected on the validation split; the remaining values were fixed a priori. Local compute is matched across the federated baselines: flat FedAvg performs 6 local epochs per cloud round, while the hierarchical scheme performs 2 edge rounds of 3 local epochs each, so a client executes the same 180 local epochs in both cases, and the centralized baseline is trained for the same 180 epochs over the pooled data.

Table 2. Experimental configuration of the proposed hierarchical federated intrusion detection system (HFed-IDS)

Parameter	Value
Dataset	UNSW-NB15, 82,332 records, 10 classes, 42 features
Split (stratified, disjoint)	Train 57,632 / validation 12,350 / test 12,350 (70/15/15)
Preprocessing	Label encoding, mean imputation, standardization; all fitted on the training split only
Clients (K)	10
Edge aggregators (E)	2 (5 clients each)
Client partition	Dirichlet label skew, $\alpha = 0.5$ (non-IID)
Global (cloud) rounds	30
Edge rounds per cloud round	2
Local epochs per edge round	3
Local epochs, flat FedAvg	6 per round
Centralized training epochs	180
Model	MLP 42-30-16-10, ReLU, dropout 0.2, 1,956 parameters
Optimizer	Adam, learning rate 0.001, weight decay 0.0
Batch size	256

Loss	Cross-entropy (unweighted)
Few-shot support size (K-shot)	1, 5, 10, 20, 50
Fine-tuning	Last 2 linear layers, 500 full-batch steps, learning rate 0.01
Random seeds	0, 1, 2, 3, 4 (5 repetitions)
Reported statistic	Mean $\pm$ standard deviation over the seeds

4.2 Convergence of the federated global model

Figure 8 shows the federated convergence curves, that is, the evolution of global model accuracy on the held-out test split across the communication rounds, for both the flat and the hierarchical schemes; the shaded band is one standard deviation over the five seeds. Accuracy improves quickly during the early rounds and then flattens. For the hierarchical scheme, the global accuracy rises from 64.3% at round 1 to 76.5% at round 5 and 78.5% at round 10, and reaches $82.68 \pm 1.46\%$ at round 30. The gain over the last ten rounds is 1.7 percentage points, which indicates that the model has essentially converged under non-IID client data. The two curves are within one standard deviation of each other throughout, confirming that the hierarchical topology does not cost accuracy relative to flat FedAvg; its benefit lies in communication, quantified in Section 4.7.

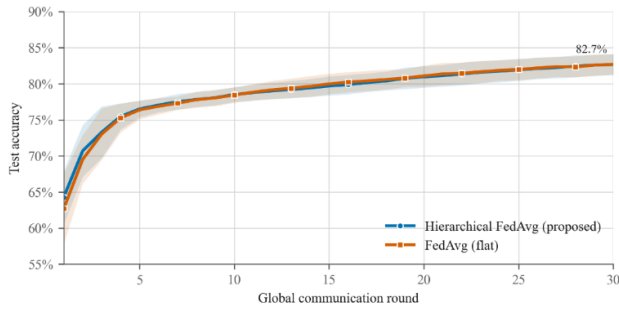


Figure 8. Convergence of the global model on the held-out test split for flat federated averaging (FedAvg) and for the proposed hierarchical scheme

Note: Lines are means, and bands are one standard deviation over five seeds.

4.3 Sensitivity to the size of the support set

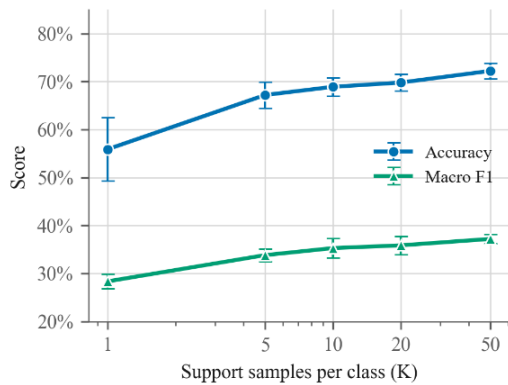


Figure 9. Effect of the support-set size K on accuracy and macro F1 after few-shot fine-tuning

Note: Error bars show one standard deviation over five seeds.

Figure 9 shows the sensitivity of the fine-tuning stage to the size of the support set. Each client fine-tunes the converged global model on K labelled samples per class and is evaluated on its own shard of the held-out test split; the reported values are means over the ten clients and the five seeds, with error bars showing one standard deviation across seeds. Both

accuracy and macro F1 increase monotonically with K, from $55.9 \pm 6.6\%$ accuracy and $28.4 \pm 1.5\%$ macro F1 at $K = 1$ to $72.2 \pm 1.6\%$ and $37.2 \pm 1.0\%$ at $K = 50$. Most of the gain is obtained by $K = 10$, and the variance at $K = 1$ is by far the largest, which shows that a single sample per class is not sufficient to estimate a stable decision boundary. The absolute accuracy of the fine-tuned models is below that of the unmodified global model; the reason is analyzed in Section 4.5.

4.4 Comparison with the centralized model and the federated baselines

Figure 10 compares the centralized model with the hierarchical federated model. The centralized model reaches $85.73 \pm 0.19\%$ accuracy. The federated model starts lower, at 64.3% in the first round, then converges to $82.68 \pm 1.46\%$, leaving a gap of 3.05 percentage points. Both models use the same architecture, the same training split, and the same held-out test split, so the gap is attributable to decentralization and to the non-IID client partition alone. Centralized training benefits from full visibility of the pooled dataset, but it is often impractical in real IoT deployments because of privacy regulation, bandwidth limits, and data ownership constraints. Table 3 reports the full set of metrics for all four models.

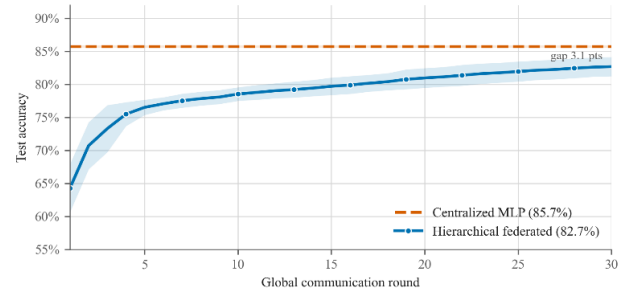


Figure 10. Test accuracy of the centralized model (dashed reference line with a one standard deviation band) against the hierarchical federated model across communication rounds

Table 3 compares the four models on the held-out test split using both weighted and macro averages, as defined in Section 3.8.

The centralized model performs best on aggregate accuracy, as expected from unrestricted access to the pooled training data. The hierarchical federated model loses 3.05 percentage points of accuracy relative to it, and is statistically indistinguishable from flat FedAvg ($82.70 \pm 1.42\%$ versus $82.68 \pm 1.46\%$), which is the expected outcome given the equivalence noted in Section 3.3. Few-shot fine-tuning reduces accuracy to $72.44 \pm 2.35\%$ but raises macro recall from 38.16% to 49.28% and macro F1 from 38.35% to 41.19%. This is a genuine trade-off rather than an improvement on every axis: the class-balanced support set moves the decision boundary away from the majority classes, which costs accuracy on the dominant Normal and Generic traffic and buys recall on the minority attack classes. Which operating point is preferable depends on the deployment, and Section 4.9 discusses this further.

Table 3. Performance of the centralized model, flat federated averaging (FedAvg), the hierarchical global model, and the hierarchical model after few-shot fine-tuning, on the held-out test split (mean $\pm$ standard deviation over five seeds, in %)

Model	Accuracy	Weighted Precision	Weighted Recall	Weighted F1	Macro Precision	Macro Recall	Macro F1
Centralized (upper bound)	85.73 $\pm$ 0.19	83.96 $\pm$ 0.24	85.73 $\pm$ 0.19	83.91 $\pm$ 0.41	45.45 $\pm$ 0.50	42.22 $\pm$ 0.51	42.47 $\pm$ 0.68
FedAvg (flat)	82.70 $\pm$ 1.42	81.09 $\pm$ 1.33	82.70 $\pm$ 1.42	80.13 $\pm$ 1.87	42.85 $\pm$ 1.81	38.37 $\pm$ 1.99	38.27 $\pm$ 2.06
Hierarchical FedAvg (HFed-IDS)	82.68 $\pm$ 1.46	81.05 $\pm$ 1.24	82.68 $\pm$ 1.46	80.13 $\pm$ 2.14	43.21 $\pm$ 1.66	38.16 $\pm$ 2.20	38.35 $\pm$ 2.50
HFed-IDS + few-shot fine-tuning (K = 20)	72.44 $\pm$ 2.35	82.28 $\pm$ 0.68	72.44 $\pm$ 2.35	76.02 $\pm$ 1.82	41.23 $\pm$ 1.01	49.28 $\pm$ 1.62	41.19 $\pm$ 1.51

Table 4. Per-class precision (P), recall (R), and F1 score on the held-out test split for the hierarchical federated global model and for the same model after few-shot fine-tuning (K = 20), averaged over five seeds, in %

Attack Category	Test Support	Global P	Global R	Global F1	Fine-Tuned P	Fine-Tuned R	Fine-Tuned F1
Analysis	102	0.0	0.0	0.0	9.3	34.1	14.6
Backdoor	87	0.0	0.0	0.0	10.9	34.6	16.3
DoS	613	29.2	28.9	23.9	31.7	44.5	36.8
Exploits	1,670	65.3	74.6	67.9	74.6	46.9	57.5
Fuzzers	909	74.1	37.5	44.4	38.8	55.2	45.2
Generic	2,831	99.6	96.2	97.9	98.4	96.3	97.3
Normal	5,550	87.3	98.8	92.7	95.8	77.6	85.7
Reconnaissance	525	76.7	45.6	56.7	46.1	51.3	47.3
Shellcode	57	0.0	0.0	0.0	6.0	38.9	10.1
Worms	6	0.0	0.0	0.0	0.7	13.3	1.3
Macro average	12,350	43.2	38.2	38.4	41.2	49.3	41.2
Weighted average	12,350	81.1	82.7	80.1	82.3	72.4	76.0

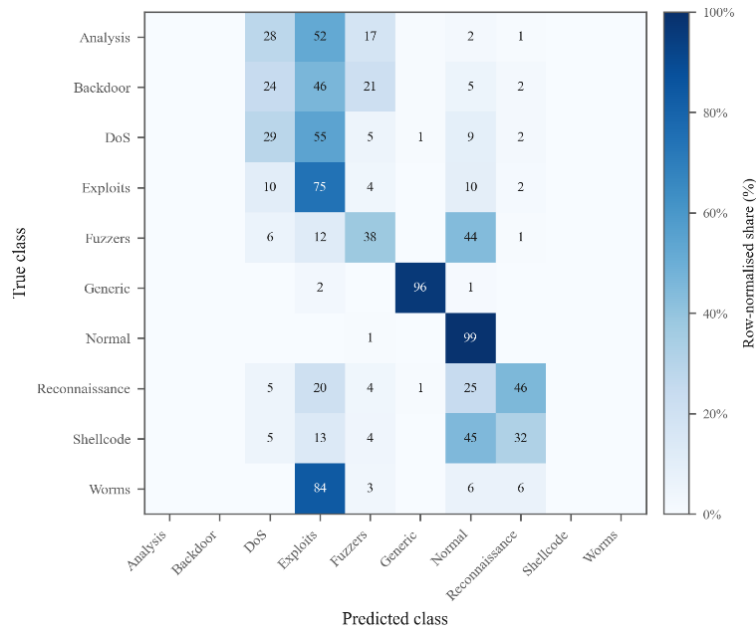


Figure 11. Row-normalized confusion matrix of the hierarchical federated global model on the held-out test split, averaged over five seeds

Note: Values are percentages of the true class.

The centralized model performs best on aggregate accuracy, as expected from unrestricted access to the pooled training data. The hierarchical federated model loses 3.05 percentage points of accuracy relative to it, and is statistically indistinguishable from flat FedAvg (82.70 $\pm$ 1.42% versus 82.68 $\pm$ 1.46%), which is the expected outcome given the equivalence noted in Section 3.3. Few-shot fine-tuning reduces accuracy to 72.44 $\pm$ 2.35% but raises macro recall from 38.16% to 49.28% and macro F1 from 38.35% to 41.19%. This is a genuine trade-off rather than an improvement on

every axis: the class-balanced support set moves the decision boundary away from the majority classes, which costs accuracy on the dominant Normal and Generic traffic and buys recall on the minority attack classes. Which operating point is preferable depends on the deployment, and Section 4.9 discusses this further.

4.5 Per-class performance and confusion matrix

Aggregate accuracy hides the behaviour that matters most

for an intrusion detection system, so Table 4 reports precision, recall, and F1 for every class, and Figure 11 shows the corresponding row-normalized confusion matrix of the hierarchical global model. The picture is clear and unfavourable for the rare categories: Analysis, Backdoor, Shellcode, and Worms are never predicted by the global model, so their recall is exactly zero, which is why the macro F1 of 38.35% is so much lower than the weighted F1 of 80.13%. The classes that the model does detect reliably are the frequent ones, Normal and Generic, together with Exploits. The confusion matrix also shows where the errors concentrate: Analysis, Backdoor, and DoS are absorbed almost entirely into Exploits, and Shellcode is split between Normal and Reconnaissance, which is consistent with the well-documented overlap of these categories in UNSW-NB15 [26].

The same table shows the effect of few-shot fine-tuning. After fine-tuning with $K = 20$, the previously undetected classes acquire non-zero recall; for example, Analysis 34.1%, Backdoor 34.6%, Shellcode 38.9%, while recall on the dominant Normal class falls from 98.8% to 77.6%. This is the

mechanism behind the accuracy-versus-macro-recall trade-off reported in Table 3.

4.6 Adaptation to attack classes unseen during training

The experiments above vary the size of the support set for classes that are already present during federated training, and therefore do not by themselves demonstrate adaptation to a new attack. To test that claim directly, a leave-one-attack-out protocol is used: one attack category is removed from the local data of every client, the federated system is trained to convergence without it, and the resulting global model is then fine-tuned on a support set that contains K samples of the withheld class alongside K samples of each known class. The support samples are drawn from the training split only, and evaluation is performed on the untouched test split. Four categories spanning a range of frequencies were withheld in turn, and the procedure was repeated over 3 seeds. Results are given in Table 5 and Figure 12.

Table 5. Leave-one-attack-out results. Recall on the withheld attack class before and after few-shot fine-tuning, together with overall accuracy and macro F1 score (mean $\pm$ standard deviation over 3 seeds, in %)

Withheld Class	Recall, no Adaptation	Recall, K = 5	Recall, K = 10	Recall, K = 20	Recall, K = 50	Accuracy, no Adaptation	Accuracy, K = 50	Macro F1, no Adaptation	Macro F1, K = 50
DoS	0.0	22.1 $\pm$ 11.7	33.6 $\pm$ 5.2	33.2 $\pm$ 14.4	46.9 $\pm$ 3.5	82.6	72.5	36.2	41.3
Reconnaissance	0.0	39.9 $\pm$ 10.1	29.7 $\pm$ 13.2	18.1 $\pm$ 6.4	33.1 $\pm$ 2.1	81.1	72.2	34.1	41.4
Fuzzers	0.0	24.8 $\pm$ 13.9	41.0 $\pm$ 7.3	47.9 $\pm$ 5.6	57.0 $\pm$ 5.1	80.6	67.8	31.5	39.5
Shellcode	0.0	38.0 $\pm$ 26.7	55.0 $\pm$ 3.6	50.3 $\pm$ 9.8	72.5 $\pm$ 8.8	82.3	71.1	39.1	40.9

Table 6. Cumulative communication cost of a complete training run of 30 rounds, separated by link type

Scheme	Device to edge (MB)	Edge to Cloud, Backbone (MB)	Total (MB)	Backbone Traffic Relative to Centralized Upload
Centralized (raw data upload)	—	9.23	9.23	baseline
FedAvg (flat)	—	4.48	4.48	−51.5%
Hierarchical FedAvg (HFed-IDS)	8.95	0.90	9.85	−90.3%

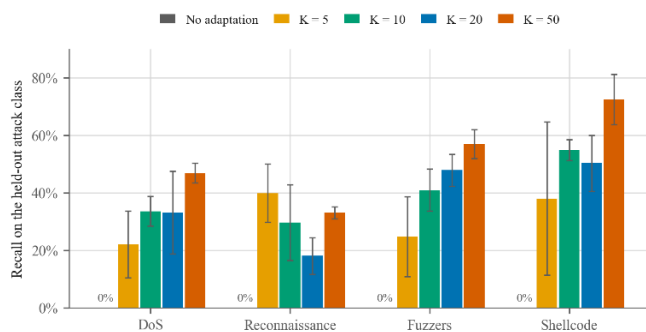


Figure 12. Recall on the withheld attack class in the leave-one-attack-out experiment, before adaptation and after few-shot fine-tuning with K support samples per class

Note: Error bars show one standard deviation over three seeds.

Before adaptation, the withheld class is never predicted: its recall is exactly 0% in every case, which confirms that a federated model cannot detect an attack category it has never

observed. Fine-tuning on a handful of labelled examples recovers a substantial part of that capability. With 50 samples per class, recall on the withheld category reaches $72.5 \pm 8.8\%$ for Shellcode, and 46.9% for DoS, 33.1% for Reconnaissance, and 57.0% for Fuzzers. Even five samples per class lift recall from zero to between 22.1% and 39.9% . The cost is again a fall in overall accuracy, from about 81.7% before adaptation to about 70.9% afterwards, while macro F1 improves. Two further observations should temper the claim. First, the recovered recall is far from the recall achievable when the class is present throughout training. Second, the seed-to-seed variance at small K is large, so a single run would not support a reliable conclusion; this is precisely why repetitions are reported.

4.7 Cumulative communication cost

The previous version of this manuscript quoted a traffic reduction derived from a single model transfer. Table 6 instead accounts for the whole training run. The model payload is

1,956 parameters, which is 7.64 KB as 32-bit floats. Over 30 rounds, flat FedAvg moves $2 \times 10 \times 30$ payloads across the backbone, which is 4.48 MB, while the hierarchical scheme moves only $2 \times 2 \times 30$ payloads across the backbone, which is 0.90 MB. Uploading the raw training split instead would cost 9.23 MB. Backbone traffic is therefore reduced by 90.3% relative to centralized data collection and by 80.0% relative to flat FedAvg. One consequence must be stated honestly, because it is a direct result of the hierarchy rather than a side effect: the device-to-edge links carry 8.95 MB, since each client exchanges its model 2 times per cloud round, so total traffic summed over all links (9.85 MB) is higher than for either alternative. The scheme is therefore beneficial where the backbone or wide-area link is the constrained and costly resource and the device-to-edge links are local and inexpensive, which is the usual situation in edge-enabled IoT deployments, and it is not beneficial where the device-to-edge link is itself the bottleneck. Figure 13 illustrates this split.

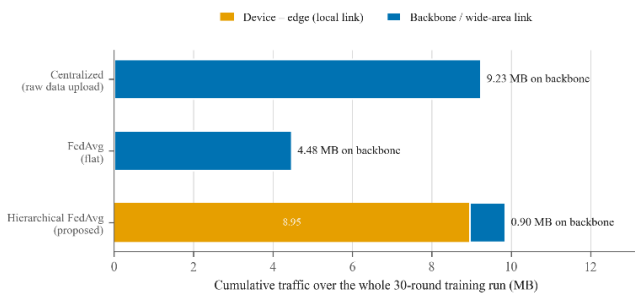


Figure 13. Cumulative traffic of a complete training run, separated into device-to-edge and backbone links

4.8 Robustness under the official dataset partitions

The results above use a stratified split of a single UNSW-NB15 partition, in which the training and test distributions match. Because the official UNSW-NB15 release provides two partitions that are deliberately not identically distributed, an additional check was run in which the federated system is trained on the 175,341-record partition and evaluated on the 82,332-record partition, with 2 seeds. Accuracy drops for both the centralized model ($73.2 \pm 0.8\%$) and the hierarchical federated model ($73.9 \pm 4.2\%$), and the seed-to-seed spread widens considerably. This confirms that the absolute numbers reported in Sections 4.2 to 4.7 are specific to the matched-distribution setting and should not be read as an estimate of performance under distribution shift. Results are summarized in Table 7.

Table 7. Robustness check across the official UNSW-NB15 partitions: training on the 175,341-record partition and evaluation on the 82,332-record partition (mean $\pm$ standard deviation over 2 seeds, in %)

Model	Accuracy	Weighted F1	Macro F1
Centralized	73.2 ± 0.8	73.8 ± 0.6	39.9 ± 0.1
Hierarchical FedAvg (HFed-IDS)	73.9 ± 4.2	72.3 ± 1.5	33.5 ± 1.0

4.9 Discussion and limitations

Taken together, the results support a narrower set of claims

than the previous version of this manuscript made. Hierarchical federated aggregation trains an intrusion detector to within 3.05 percentage points of a centralized model of identical capacity while moving 90.3% less traffic across the backbone, and few-shot fine-tuning gives the system the ability to cover attack categories that are rare or entirely absent from federated training. The claims that are not supported by these experiments, and which have accordingly been removed, are that the method performs meta-learning and that it provides a formal privacy guarantee.

Minority classes are the dominant limitation. Four of the ten categories are never predicted by the global model, and Worms, with only a handful of test records, remains essentially undetectable even after adaptation. The unweighted cross-entropy loss, the small model capacity, and the extreme class imbalance of UNSW-NB15 all contribute. Class-weighted losses, focal loss, or resampling at the client would be the natural next step, and none of them was used here.

Client heterogeneity affects the results in a measurable way. Under the Dirichlet partition with $\alpha = 0.5$, the client shard sizes in a single run ranged from about 1,300 to 14,262 records, and the standard deviation of federated accuracy across seeds (1.46 percentage points) is roughly seven times that of the centralized model (0.19), which quantifies the sensitivity of the federated result to how the data happen to be distributed.

Several practical aspects were not modelled. All clients participate in every round, so the effect of stragglers and partial participation is not measured. Edge-node failure is not simulated, although the hierarchy introduces exactly such a single point of failure for the clients attached to a failed aggregator; replicating edge state or reassigning clients would be required in a real deployment. Communication is accounted analytically from payload sizes and does not include protocol overhead, retransmission, or encryption expansion, and no measurement of energy consumption, on-device latency, or monetary cost of deployment was performed. Finally, UNSW-NB15 is a single, offline, and now somewhat dated benchmark whose partition into clients is synthetic; validation on live traffic from a real deployment would be required before operational claims could be made.

5. CONCLUSIONS AND FUTURE WORK

This paper studied multi-class intrusion detection for edge-enabled IoT using FL on the ten categories of the UNSW-NB15 dataset. The previous version of this text described the work as DDoS detection; that description was inaccurate because the dataset contains no separate DDoS class, and the terminology has been corrected throughout. The proposed HFed-IDS combines hierarchical edge aggregation with a localized few-shot fine-tuning stage. The hierarchical global model reaches $82.68 \pm 1.46\%$ accuracy and $80.13 \pm 2.14\%$ weighted F1 on a held-out test split, which is 3.05 percentage points below an identically configured centralized model, while cumulative backbone traffic over the 30 training rounds falls by 90.3% relative to uploading the raw data and by 80.0% relative to flat FedAvg, at the price of additional traffic on the local device-to-edge links. Few-shot fine-tuning raises macro recall from 38.16% to 49.28% and restores non-zero recall on attack categories that were entirely withheld from federated training, at a cost in overall accuracy. Because raw records never leave the device, the approach achieves data minimization, but it provides no formal privacy guarantee.

Future work follows directly from the limitations identified in Section 4.9. The imbalance problem should be addressed with class-weighted or focal losses and client-side resampling, since four categories are currently never predicted. Formal privacy should be added through secure aggregation [33] and a differentially private optimizer [32], with the resulting utility cost measured rather than assumed. The evaluation should be extended to partial client participation, straggling and failing edge nodes, and a measured rather than analytical communication budget. Finally, intelligent selection of the participating IoT devices according to their computational load or data quality, and an assessment of resilience against adversarial evasion, remain valuable directions for deployment in high-security environments.

REFERENCES

- [1] Abd-elaziem, A.H., Soliman, T.H. (2023). A multi-layer perceptron (MLP) neural networks for stellar classification: A review of methods and results. *International Journal of Advances in Applied Computational Intelligence*, 3(2): 29-37. <https://doi.org/10.54216/IJAACI.030203>
- [2] Abadi, M., Chu, A., Goodfellow, I., et al. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Vienna, Austria, pp. 308-318. <https://doi.org/10.1145/2976749.2978318>
- [3] Abreha, H.G., Hayajneh, M., Serhani, M.A. (2022). Federated learning in edge computing: A systematic survey. *Sensors*, 22(2): 450. <https://doi.org/10.3390/s22020450>
- [4] Adhikari, T., Khan, A.K., Kule, M. (2025). An analytical review of security issues in centralized and distributed SDN environments. *Information Security Journal: A Global Perspective*, 34(6): 713-746. <https://doi.org/10.1080/19393555.2025.2550762>
- [5] Ali, B., Gregory, M.A., Li, S. (2021). Multi-access edge computing architecture, data security and privacy: A review. *IEEE Access*, 9(1): 18706-18721. <https://doi.org/10.1109/ACCESS.2021.3053233>
- [6] Almesleh, Z., Gouissem, A., Hamila, R. (2024). Federated learning with Kalman filter for intrusion detection in IoT environment. In *2024 IEEE 8th Energy Conference (ENERGYCON)*. Doha, Qatar. <https://doi.org/10.1109/ENERGYCON58629.2024.10488796>
- [7] Althiyabi, T., Ahmad, I., Alassafi, M.O. (2024). Enhancing IoT security: A few-shot learning approach for intrusion detection. *Mathematics*, 12(7): 1055. <https://doi.org/10.3390/math12071055>
- [8] Balik, M.Y. (2024). Comparing federated stochastic gradient descent and federated averaging for predicting hospital length of stay. *arXiv preprint arXiv:2407.12741*. <https://doi.org/10.48550/arXiv.2407.12741>
- [9] Bonawitz, K., Ivanov, V., Kreuter, B., et al. (2017). Practical secure aggregation for privacy-preserving machine learning. In *CCS '17: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas, Texas, USA, pp. 1175-1191. <https://doi.org/10.1145/3133956.3133982>
- [10] Emmanuel, T., Maupong, T., Mpoeleng, D., et al. (2021). A survey on missing data in machine learning. *Journal of Big Data*, 8(1): 140. <https://doi.org/10.1186/s40537-021-00516-9>
- [11] Finn, C., Abbeel, P., Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 70: 1126-1135. <https://dl.acm.org/doi/10.5555/3305381.3305498>
- [12] Haq, R.U., Aman, F., Majeed, M.A., et al. (2025). Developing edge computing solutions for IoT devices to reduce latency and enhance real-time decision-making. *Spectrum of Engineering Sciences*, 3(4): 961-970.
- [13] Ismael, O. (2025). The Standard Deviation Score: A novel similarity metric for data analysis. *Journal of Big Data*, 12(1): 58. <https://doi.org/10.1186/s40537-025-01091-z>
- [14] Lazzarini, R., Tianfield, H., Charissis, V. (2023). Federated learning for IoT intrusion detection. *AI*, 4(3): 509-530. <https://doi.org/10.3390/ai4030028>
- [15] Liu, L., Zhang, J., Song, S.H., Letaief, K.B. (2020). Client-edge-cloud hierarchical federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. Dublin, Ireland. <https://doi.org/10.1109/ICC40277.2020.9148862>
- [16] Long, G., Xie, M., Shen, T., et al. (2023). Multi-center federated learning: Client clustering for better personalization. *World Wide Web*, 26(1): 481-500. <https://doi.org/10.1007/s11280-022-01046-x>
- [17] McMahan, B., Moore, E., Ramage, D., et al. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 54: 1273-1282.
- [18] Moustafa, N., Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*. Canberra, ACT, Australia. <https://doi.org/10.1109/MilCIS.2015.7348942>
- [19] Moustafa, N., Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 25(1-3): 18-31. <https://doi.org/10.1080/19393555.2015.1125974>
- [20] Munirathinam, S. (2020). Industry 4.0: Industrial internet of things (IIOT). In *Advances in Computers*, 117: 129-164. <https://doi.org/10.1016/bs.adcom.2019.10.010>
- [21] Nasr, M., Shokri, R., Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA, pp. 739-753. <https://doi.org/10.1109/SP.2019.00065>
- [22] Nguyen, T.D., Marchal, S., Miettinen, M., et al. (2019). DiIoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. Dallas, TX, USA, pp. 756-767. <https://doi.org/10.1109/ICDCS.2019.00080>
- [23] Pande, P., Babu, B.M., Bhargav, P., et al. (2025). Attention-driven hierarchical federated learning for privacy-preserving edge AI in heterogeneous IoT networks. *International Journal of Advanced Computer Science & Applications*, 16(5): 45.

- <https://doi.org/10.14569/IJACSA.2025.0160545>
- [24] Peng, H., Wu, C., Xiao, Y. (2025). FD-IDS: Federated learning with knowledge distillation for intrusion detection in non-IID IoT environments. *Sensors*, 25(14): 4309. <https://doi.org/10.3390/s25144309>
- [25] Reyes, J., Di Jorio, L., Low-Kam, C., Kersten-Oertel, M. (2025). Precision-weighted federated learning. *Computational Intelligence*, 41(6): e70150. <https://doi.org/10.1111/coin.70150>
- [26] Sathyanarayanan, S., Tantri, B.R. (2024). Confusion matrix-based performance evaluation metrics. *African Journal of Biomedical Research*, 27(4S): 4023-4031. <https://doi.org/10.53555/AJBR.v27i4S.4345>
- [27] Shafiq, M., Gu, Z., Cheikhrouhou, O., Alhakami, W., Hamam, H. (2022). The rise of “Internet of Things”: review and open research issues related to detection and prevention of IoT-based security attacks. *Wireless Communications and Mobile Computing*, 2022(1): 8669348. <https://doi.org/10.1155/2022/8669348>
- [28] Tan, A.Z., Yu, H., Cui, L., Yang, Q. (2023). Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12): 9587-9603. <https://doi.org/10.1109/TNNLS.2022.3160699>
- [29] Tripathi, S., Singh, L., Sermanraja, J. (2024). Complete data using exploratory data analysis and ML algorithms. In 2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS). Tashkent, Uzbekistan. <https://doi.org/10.1109/ICTACS62700.2024.10840870>
- [30] Vibhute, A.D., Khan, M., Patil, C.H., et al. (2024). Network anomaly detection and performance evaluation of Convolutional Neural Networks on UNSW-NB15 dataset. *Procedia Computer Science*, 235(1): 2227-2236. <https://doi.org/10.1016/j.procs.2024.04.211>
- [31] Wen, J., Zhang, Z., Lan, Y., et al. (2023). A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14(2): 513-535. <https://doi.org/10.1007/s13042-022-01647-y>
- [32] Wilson, A., Anwar, M.R. (2024). The future of adaptive machine learning algorithms in high-dimensional data processing. *International Transactions on Artificial Intelligence*, 3(1): 97-107. <https://doi.org/10.33050/italic.v3i1.656>
- [33] Zhang, R., Xu, Q., Yao, J., et al. (2023). Federated domain generalization with generalization adjustment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Vancouver, BC, Canada, pp. 3954-3963. <https://doi.org/10.1109/CVPR52729.2023.00385>
- [34] Zhao, L., Li, J., Li, Q., Li, F. (2021). A federated learning framework for detecting false data injection attacks in solar farms. *IEEE Transactions on Power Electronics*, 37(3): 2496-2501. <https://doi.org/10.1109/TPEL.2021.3114671>
- [35] Zhu, L., Liu, Z., Han, S. (2019). Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32.