Real-Time Anomaly Detection Using Federated Learning in Edge Devices

Real-Time Anomaly Detection Using Federated Learning in Edge Devices

Vaibhav Joshi[1] | Abishi Chowdhury[1] | Amrit Pal[1] | Vishal Krishna Singh[2] | Rajkumar Singh Rathore[3]

[1]School of Computer Science and Engineering Vellore Institute of Technology ChennaiTamil Nadu, India | [2]School of Computer Science and Electronics Engineering, University of Essex, ColchesterUnited Kingdom | [3]Department of Computer Science, Cardiff Metropolitan University's School of Technologies, United Kingdom |

Received: Revised: Accepted:

Keywords: Anomaly detection| Clustered federated learning (CFL)| Edge Computing| Federated learning | Intrusion detection system

ABSTRACT

Background: The widespread expansion of IoT devices has significantly increased the scope for malicious activities on the part of cybercriminals. This has made edge computing highly vulnerable to various types of network intrusions, including DDoS floods and brute-force attacks. However, traditional centralized Intrusion Detection Systems (IDS) do not address these concerns due to their inherent latency and lack of consideration of data privacy. To overcome these drawbacks of traditional IDS, we have developed a distributed real-time anomaly detection system architecturally designed for deployment on edge computing hardware.

Method: A monitoring agent collects system and network telemetry and real-time network flow data on a Raspberry Pi 3 and sends it to a centralized server via a TCP socket. The server uses a dual-engine detection system consisting of rule-based heuristics and a DNN model. To address the concerns of data privacy and non-IID data distribution inherent to distributed IoT networks, our system uses a collaboratively trained DNN model using three different Federated Learning (FL) techniques: Clustered Federated Learning (CFL), FedDyn, and FedNova.

Results: These techniques were evaluated on the CIC-IoT-DIAD 2024 dataset with severe non-IID data distribution. The results show that CFL outperforms others with an F1-score of 0.89 and an AUC of 0.95. This work shows that production-grade, privacy-preserving intrusion detection is feasible in IoT networks at the edge.

1 | Introduction

With the increase in the use of Internet of Things(IoT) devices and the rapid advancement in edge computing, the scope of network security attacks has increased. The rage of the network attacks span from distributed denial of service attacks, attack on embedded sensors, controllers, and IoT gateways [1]. A conventional approach include analyzing the data using cloud services that requires sending data to the cloud and processing it and sending the analysis results back to the end node. An alternative approach is to perform the cyber threat analysis on the end devices. Traditional Intrusion Detection Systems (IDS) mainly depend on a central server that collects and analyzes network traffic, however, this approach has started to get outdated in today's distributed world. With the rise of IoT and edge devices, networks now generate huge amounts of sensitive data, and sending all of it to one place is not only risky but can also create compliance issues with regulations like GDPR. On top of that, constantly transferring raw data from many devices puts pressure on network bandwidth, slowing things down and affecting real-time detection. There is also a bigger concern having everything rely on a single central server creates a weak spot; if it gets attacked or fails, the entire system can lose visibility at once. As more devices come online, managing and scaling such centralized systems becomes harder and more resource-intensive, highlighting the need for smarter, more distributed solutions. Furthermore, selecting Federated Learning (FL) models for this environment is driven by several critical motivations. Foremost is privacy preservation, as sensitive network telemetry and personal device data never leave the edge node. Secondly, FL provides bandwidth efficiency by replacing massive raw data transfers with small model parameter

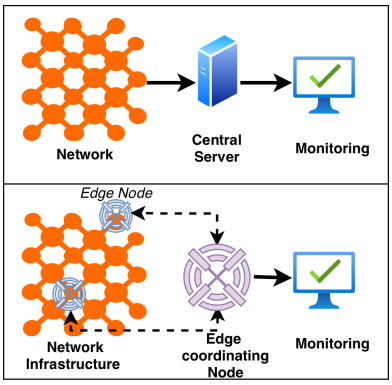

FIGURE 1 | An overview of the centralized vs an edge based solution for the network monitoring

updates. Finally, it ensures compliance with strict data regulations like GDPR, making FL uniquely suited for scalable threat intelligence in modern edge-based IDS networks.

To address these issues in this paper, a three tier approach is proposed, as highlighted in the figure 1. A lightweight end node for network sensing or sniffing through which important features about the network can be derived. These features are sent to a central server for applying the detection procedure which is two fold, first is a lightweight rule based model for detection. Second, a deep neural network model that iteratively updates itself in a federated manner. The third module in the overall architecture is the presentation, which generates the alarms and update the end results.

To facilitate the distributed learning model with minimum data exchange over the network and support the edge devices, a federated learning [2] model is proposed. Using federated learning the edge devices use the local data for training the local model and anomaly detection. The edge nodes share the learning information and collaborate to generate a global model. No exchange of traffic data is required during this process. The data generated in such network monitoring environment is of non-IID nature. Federated learning is appropriate for such data. However, standard federated learning methods like FedAvg face critical limitations under these conditions. When FL-based IDS systems are applied in these heterogeneous environments, severe client drift can occur, which degrades the global model for all IoT devices as a whole, as outlined in [3]. Traditional approaches fail to generalize across diverse threat landscapes. In order to handle this challenge, we have used a clustered FL approach in this paper. Clustered FL directly addresses this gap by creating specialized, threat-specific models rather than forcing a sub-optimal global consensus, thereby significantly enhancing the detection capabilities over existing centralized and non-clustered distributed architectures. **Contributions.** The main contributions of this work are:

- A complete and deployable 3 tier anomaly detection system from physical edge hardware to a real time web dashboard.
- A comparative study of CFL, FedDyn, and FedNova under harsh non-IID IoT traffic conditions.
- A verification of CFL's superiority over other methods in heterogeneous network scenarios with real-time edge inference on a Raspberry Pi3.

2 | Related Work

With the proliferation of IoT, there has been a paradigm shift towards edge intelligence, where data is processed on the edge devices, thus eliminating end-to-end cloud latency [1, 4]. In this direction, efficient on-device IDS design has become a pressing research need [5, 6].

Deep learning techniques have shown impressive results for anomaly detection on IoT devices. Sudharshan *et al.* [7] demonstrated the potential of DNNs for IoT intrusion detection systems, while Latif-Martínez *et al.* [8] used GNNs for modeling network topology anomalies. However, such centralized approaches cannot tap the collective knowledge of neighboring devices for global threats and cannot generalize for zero-day threats that were not experienced by the local devices.

Federated Learning has come up as a solution that strikes a balance between data privacy and model accuracy [2]. Federated Learning-based anomaly detection has shown impressive results for Smart Grids [9] and IIoT [10, 11], thus providing a robust solution for IoT devices. Secure aggregation protocols have also been used for enhancing the robustness of Federated Learning-based models against model poisoning attacks [12].

Despite these advances, standard FL algorithms exhibit acute performance degradation under statistical heterogeneity. Zhao *et al.* [3] rigorously demonstrated that applying FedAvg to non-IID data causes extreme client-weight divergence. Li *et al.* [13] proposed FedProx to address heterogeneous networks through proximal regularization, while Acar *et al.* [14] introduced FedDyn's dynamic regularization to actively correct for client drift. FedNova [15] addressed objective inconsistency arises from variable local update counts.

However, the end result of all of the above techniques is a *single* global model, which by its very nature is sub-optimal. Our work closes this gap by utilizing a technique called **Clustered Federated Learning (CFL)** [16], which groups clients into clusters of homogeneity and creates archetype models. Coupled with our proposed Dual Detection Engine and dashboard, this directly handles the client drift issue.

To synthesize the positioning of our work, Table 1 summarizes the recent literature on IoT anomaly detection, contrasting their methodologies and limitations against our proposed Clustered FL approach.

3 | Problem Formulation

A network is continuously monitored for potential attacks using a network sniffing tool that captures traffic and records a set of features $x \in \mathbb{R}^d$, where the detection task can be modeled as predicting a label $y \in \{0, 1\}$ (normal or anomaly). In a typical centralized setup, these features forming a dataset $\mathcal{D} = \{(x_i, y_i)\}$ are transmitted to a server or cloud system for analysis, which increases communication overhead and raises privacy concerns. To address this, a distributed approach can be adopted, where N edge devices are deployed to locally monitor network traffic and learn from their respective datasets $\mathcal{D}_k$ while simultaneously detecting anomalies. Each edge device uses a shared global model $f(w)$ and continuously updates it by minimizing a local loss

$$F_k(w) = \frac{1}{|\mathcal{D}_k|} \sum_{(x_i, y_i) \in \mathcal{D}_k} \ell(f_w(x_i), y_i), \tag{1}$$

TABLE 1 | Comparative Summary of Related Work on IoT Anomaly Detection

Reference	Methodology	Architecture Layer	Target Application	Non-IID Handling Strategy	Key Limitations
Sudharshan et al. [7] (2025)	Deep Neural Networks (DNN)	Centralized Edge	Real-time IoT Devices	None	High communication latency; exposes raw telemetry to privacy risks.
Latif-Martínez et al. [8] (2023)	Graph Neural Networks (GNN)	Centralized Cloud	Network Topology	None	Fails to leverage collaborative intelligence; relies on single point of failure.
Noumni et al. [9] (2023)	Standard FedAvg	Distributed Edge	Smart Grids	Partial (Data Scaling)	Accuracy severely degrades under high statistical data heterogeneity.
Haldikar et al. [10] (2024)	Edge FL + Autoencoders	Edge-to-Cloud	Industrial IoT (IIoT)	None	High compute overhead on end-nodes; lacks formal drift correction.
Proposed Approach	Clustered FL + Rule-Based Engine	3-Tier Edge-to-Server	Real-Time IoT Edge (Raspberry Pi)	Yes (Dynamic Grouping / Homologous Clustering)	Feature extraction limited to 28 core dimensions due to edge compute constraints.

sending only the learned parameters w_k to a central coordinator. The coordinator aggregates these updates to refine the global model as

$$w = \sum_{k=1}^{N} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} w_k, \tag{2}$$

and communicates it back to the edge devices for further learning and improved prediction, thereby solving the global objective

$$\min_{w} \sum_{k=1}^{N} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} F_k(w). \tag{3}$$

This enables efficient, privacy-preserving, and scalable anomaly detection.

4 | Methodology

The overall methodology starts with the preprocessing of the recorded data. The relevant features are extracted from the recorded data set using the edge devices as shown in Figure 2. The local learning phase deals with learning using the local data and sending the updated weights to the central node to generate the global model. The methodology that is being proposed follows a two-phase process one is the offline federated training simulation, and second is the real-time deployment of the finalized model on the edge node.

4.1 | Input Feature Representation

Prior to model training and inference, the raw network flow must be systematically represented as a numerical vector. Each captured IP flow is transformed into a 28-dimensional continuous feature vector $\boldsymbol{x} \in \mathbb{R}^{28}$. This feature set encompasses three core categories:

- **Volumetric Measurements** (13 features): Forward and backward packet lengths (max, min, mean), header lengths, and total byte counts that capture the traffic volume profile.
- **Time-Domain Statistics** (9 features): Flow duration, inter-arrival time (IAT) statistics (min, max, mean, standard deviation), bytes per second, and packets per second that characterize temporal behavior.
- **Protocol-Specific Flags** (6 features): SYN, ACK, RST, PSH, URG, and FIN flag counts that encode the connection state and can reveal scanning or flooding behavior.

These 28 features were explicitly selected as the optimal subset that provides maximum discriminatory power for identifying complex DDoS and brute-force patterns while maintaining a minimal computational footprint suitable for real-time edge extraction on resource-constrained hardware. The complete feature categorization is presented in Table 2.

4.2 | Local Model Architecture

The edge devices have less computation and storage power. Considering the less memory and computation limitations of the edge nodes, a lightweight model is used, which is implemented as a deep neural network model. The DNN model has two fully connected hidden layers with 64 and 32 units, respectively, followed by a binary classification output layer to classify the anomaly in the network traffic data. The total number of trainable parameters is kept at a bare minimum of 5000 to ensure fast convergence and also to minimize the inference latency. Figure 3 illustrates the local parameter generation process and the subsequent communication of those parameters to the global aggregator. After aggregation, the updated global model is communicated back to all edge devices.

For aggregation, different models are used for analysis, as mentioned in the subsequent subsections.

4.3 | Federated Learning Variants

The system architecture scales to accommodate N diverse edge clients. At each round t, a subset of clients $K \subseteq N$ downloads the global model, trains it locally, and transmits the updated weights. To provide a robust baseline, the standard Federated Averaging

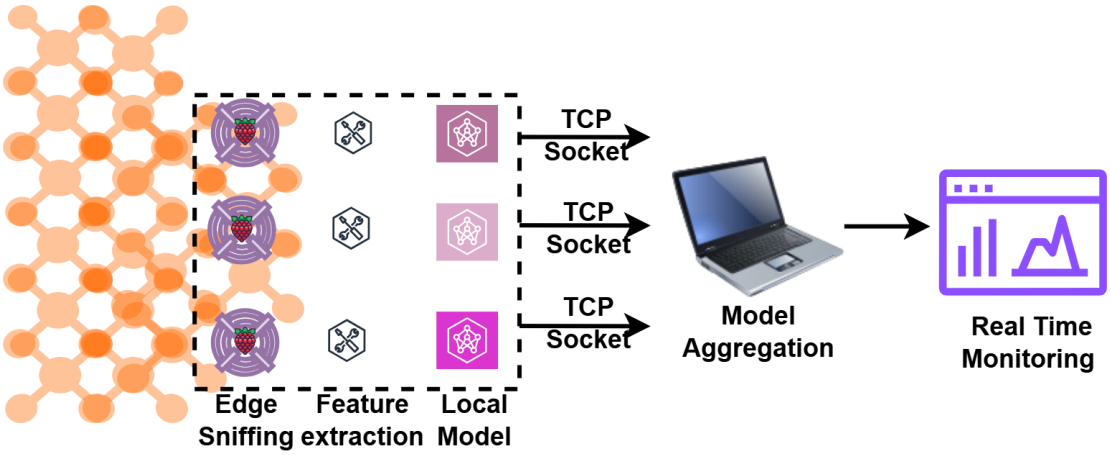

FIGURE 2 | Proposed 3-Tier FL Training and Raspberry Pi Edge Inference Architecture. The data will be transferred through the Raspberry Pi edge agent via TCP to the central detection server, where the CFL model will perform the inference, and the result will be streamed live to the web dashboard.

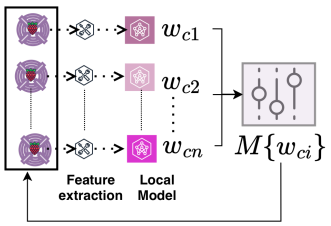

FIGURE 3 | Local model training for parameter estimation: each edge device (Raspberry Pi) performs local feature extraction and trains a local model to produce weights $w_{c1}, w_{c2}, \ldots, w_{cn}$, which are then communicated to the global aggregator $M\{w_{ci}\}$ and the resulting global model is broadcast back to all devices.

(FedAvg) algorithm calculates the new global model by taking a dataset-weighted average of the client updates:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_t^k \tag{4}$$

where n_k is the number of samples for client k and n is the total samples across participating clients.

For our evaluation, the model is tested with three advanced federated variants to handle statistical heterogeneity. The overall federated training is performed for a total of 30 global communication rounds. During each round, the 10 simulated clients perform 5 local training epochs with a batch size of 32. We employ an initial learning rate of 0.01 with a decay factor of 0.99 per round. Communication between edge nodes and the central server occurs over a secure TCP socket using JSON-serialized weight payloads with Base64 encoding, ensuring efficient and robust transmission over wireless networks.

4.3.1 | FedNova (Normalized Averaging)

In a real time scenario and considering the non-IID nature of the data client may perform local updates in different manner. So, before combining their weights and generating the global weights a normalization need to be performed. FedNova [15] approach deals with such heterogeneity of local update counts (τ_k). It normalizes the contribution of the clients before performing the aggregation, It ensures a fair participation of each edge node or client in the training process regardless of local data size or training duration as shown in the equation 5.

$$w^{t+1} = w^t - \sum_{k=1}^{K} \frac{n_k}{n} \cdot \frac{\Delta w_k}{\tau_k} \cdot \bar{\tau} \tag{5}$$

where Δw_k is the local update, τ_k is client k's local step count, and $\bar{\tau}$ is the mean step count across all clients.

4.3.2 | FedDyn (Federated Dynamic Regularization)

In network devices, it is quite common that few clients deviate from the global model while leaning to use the local data. These changes if directly aggregated with the global model result in decreasing the performance of the global model. to address this challenge a drift adaptive federated learning model is also tested for the proposed approach. The FedDyn [14] federated model handles the client drift through the dynamic regularization that is appended with each client's objective as presented in equation 6

$$\min_{w \in \mathbb{R}^d} \mathcal{L}_k(w) + \frac{\alpha}{2} \|w - w^t\|^2 - \langle \boldsymbol{h}_k^t, w \rangle \tag{6}$$

Here $\boldsymbol{h}_k^t$ is a server-maintained dynamic control vector. This term actively penalizes local model deviation from the global model's learning process and helps in convergence under non-IID network traffic data distributions.

4.3.3 | Clustered Federated Learning (CFL)

The most sophisticated method tested is CFL [16]. Unlike other methods, which train a global model and map clients to this model, CFL groups clients by their behavioral similarity and trains a separate model per cluster:

- **Weight Collection:** The server retrieves updated weight vectors w_k from each of the K clients.

- **Cosine Similarity:** The similarity between each pair of clients is calculated:

$$\text{sim}(w_i, w_j) = \frac{w_i \cdot w_j}{||w_i||||w_j||} \tag{7}$$

- **Agglomerative Clustering:** The hierarchical clustering method groups clients into C clusters using the similarity matrix. We apply a dynamic distance threshold to cut the dendrogram, allowing the optimal number of clusters to form naturally based on traffic distribution, grouping clients with similar threat profiles.
- **Per-Cluster Aggregation:** A global model is learned per cluster via FedAvg on a per-cluster basis, resulting in C models $\{G_1, G_2, \dots, G_C\}$, each specialized to a unique threat archetype.

4.4 | System Architecture for Edge Deployment

After the completion of the training procedure, the best-performing CFL cluster model is serialized and packaged together with the feature scaler and label encoder artifacts. This artifact package is securely transferred to the Raspberry Pi 3 and executed as a background daemon process. The edge agent running on the Pi continuously monitors the wireless interface, extracts the 28-feature vector from each captured flow, normalizes the vector using the pre-fitted scaler, feeds the normalized vector through the CFL model, and sends the JSON telemetry to the central server.

5 | Experimental Setup

5.1 | Dataset Selection

The dataset used for this paper is realtime IoT network traffic data collected. The CIC-IoT-DIAD 2024 dataset [17, 18], developed by the Canadian Institute of Cybersecurity (CIC) at the University of New Brunswick, was developed to support dual-purpose device identification and anomaly detection. The dataset was developed by emulating 33 unique attack types on a physical testbed of 105 IoT devices. The dataset includes a wide range of both benign baseline traffic and sophisticated multi-stage cyberattacks, which are native to IoT ecosystems. The CIC-IoT-DIAD 2024 dataset is significantly more representative of modern IoT network traffic compared to traditional and outdated datasets.

This work is focused on the detection of the presence and absence of the attack by analysing the network traffic. Based on the dataset the complete dataset for the attack detection is divided into two major categories: First is the DoS/DDoS Attacks such as DDoS HTTP Flood targeting CPU, bandwidth, and memory resources of IoT devices. These attacks are representative of Botnet attacks such as Mirai attacks. The second is the Brute Force & Reconnaissance Attacks: Low-volume attacks such as dictionary brute force attacks targeting IoT device services such as SSH, Telnet, and HTTP. These attacks are used to gain unauthorized access and perform lateral movements.

TABLE 2 | CIC-IoT-DIAD 2024 Feature Categories for Edge Deployment

Category	Examples	Count
Packet Metrics	Pkt Lengths, Header Len.	13
Flow & Time	Duration, IAT stats, Bytes/s	9
Protocol Flags	SYN/ACK/RST/PSH Counts	6
Total	–	**28**

5.2 | Feature Engineering and Simplification

The raw CIC-IoT-DIAD 2024 data contains a multitude of packet-based and flow-based attributes. However, the computation of all attributes in real-time on a resource-constrained edge computing platform is not feasible. Instead, a minimal feature representation is extracted:

Data Cleaning: Features with zero variance (features that are constant over the data) and features that exceed a configurable NaN threshold are discarded, as they provide no useful discrimination.

Feature Derivation: A minimal feature representation is extracted that contains essential flow-based attributes, such as packet size distribution, inter-arrival time (IAT) metrics, byte count, protocol flag count, and directional (forward and backward) packet count.

Dimensionality Reduction: The feature set is reduced to **28 critical features** (Table 2), chosen for their discriminating power among the target attack types and their cheapness for real-time extraction on a resource-constrained device.

Standardization: Numerical features are standardized using a standard scaler (serialized for deployment), so features like total bytes do not overwhelm features like flow duration, and scaling is homogeneous across all heterogeneous clients.

5.3 | Final Feature Set

5.4 | Non-IID Client Partitioning Strategy

In order to rigorously stress-test the proposed federated learning algorithms, three highly non-IID client partitions are developed to mimic realistic and heterogeneous client security profiles in a decentralized IoT system. The non-IID data distribution is mathematically modeled using a **Dirichlet distribution**. Specifically, for each threat class k, the proportion of data allocated to client i is sampled as:

$$p_k \sim \text{Dir}(\alpha\, \boldsymbol{p}) \tag{8}$$

where $\boldsymbol{p}$ is a prior class distribution and α controls the degree of data skew. A smaller α results in higher heterogeneity, creating more extreme label imbalance across participating clients.

Unlike uniform random client partitioning strategies, each edge node is assigned a unique and non-uniform traffic profile based on varying exposures to different types of threats (Table 3). This creates both *data heterogeneity* and *feature heterogeneity* to represent the realistic non-IID challenge.

TABLE 3 | Non-IID Client Distribution for IoT Federated Learning

Client	Normal (%)	Attack (%)	Dominant IoT Threat
Client 1	60	40	DDoS HTTP Flood (Volumetric)
Client 2	75	25	Dictionary Brute Force (Auth)
Client 3	95	5	Benign Baseline (Low Threat)

TABLE 4 | Raspberry Pi 3 Resource Utilization During Live Detection

Metric	Observed Value
RAM Consumption	∼380 MB (stable)
CPU Usage (Inference)	30–40% (intermittent)
Inference Latency	3–6 ms per flow
Max Throughput	∼300 flows/second

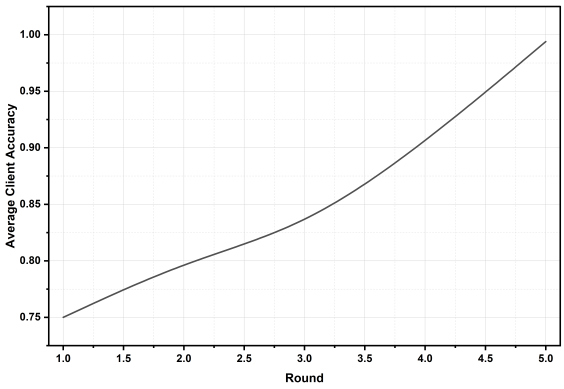

FIGURE 4 | Clustered FL average client accuracy vs. communication round. Smooth convergence validates effective non-IID handling via dynamic clustering.

5.5 | Training Environment

All federated training simulations were carried out on a local workstation (Windows 11, Intel CPU, CUDA-enabled GPU optional). The simulation had 10 clients spread over 3 non-IID traffic profiles (Table 3) and was trained for 30 global rounds with $E = 8$ local epochs per global round and a batch size of 32.

5.6 | Raspberry Pi Deployment

The physical target was a **Raspberry Pi 3 Model B** (1 GB RAM, Broadcom BCM2837 quad-core CPU) running **Raspberry Pi OS (Debian 12 Bookworm)**. The testbed network topology utilized a dedicated **802.11n Wi-Fi interface** for wireless traffic sniffing while reserving the **Ethernet interface** for reliable control plane communication with the central server. To ensure comprehensive visibility, the edge agent captures raw packets directly from the wireless interface placed in **promiscuous mode** using high-performance socket bindings. The deployment pipeline automates the discovery of the server IP address, starts the central detection server, and transfers the latest edge agent code to the Pi. The real-time detection sequence is distributed between the Pi and the Server as follows:

1. **Packet Capture (Edge):** The edge agent captures raw packets from the wireless interface in promiscuous mode.
2. **Heuristic Pre-filtering (Edge):** Rule-based detectors, including a DDoS packet counter and an SSH Brute Force Monitor, generate immediate high-confidence alerts.
3. **Feature Extraction (Edge):** A 28-dimensional feature vector is computed for each IP flow in real-time according to the configuration.
4. **Telemetry Streaming (Edge to Server):** The extracted flow vectors and heuristic alerts are encoded as JSON and sent over a TCP socket to the server on port 5555, using a 4-byte length prefix to handle TCP stream framing.
5. **ML Inference (Server):** Upon receiving the telemetry, the central server normalizes the feature vector and classifies it using the CFL model as Normal or Attack.
6. **Dashboard Broadcast (Server):** The server aggregates the ML results and edge heuristic alerts, broadcasting them to all connected dashboard clients in real-time.

The observed resource utilization of the Raspberry Pi during live detection is summarized in Table 4.

6 | Results and Analysis

6.1 | Federated Training Convergence

In Fig. 4, we plot the learning curve of average client accuracy for Clustered Federated Learning (CFL) across 5 communication rounds. The results indicate that the model achieves an exceptionally high initial accuracy of 99.50% in the very first round. By the second round, the accuracy experiences a marginal adjustment down to 99.40%, after which it perfectly plateaus. This rapid stabilization demonstrates immediate convergence, indicating that the clustered optimization landscapes are highly compatible and require minimal rounds to reach a steady, high-performing state. As observed in Fig. 5, the training and validation performance of the Clustered Federated Learning (CFL) model is evaluated over 50 epochs. The model demonstrates highly stable convergence, with both training and validation loss steadily decreasing in parallel, indicating that the model generalizes well without significant overfitting. Concurrently, the training and validation accuracy rapidly climb and plateau near optimal levels, highlighting the CFL model's robust capability in classifying anomalous network patterns.

6.2 | Weight Convergence Analysis

To explicitly demonstrate the stability of our model optimization, we analyzed the L2-norm distance of local client weights from the global consensus over the communication rounds. In standard FedAvg under severe non-IID conditions, client weights experience high divergence (client drift) and constantly fluctuate, forcing the global model into a suboptimal state. In contrast, Clustered FL dynamically identifies threat profiles and groups homologous clients. This allows the local models within each cluster to converge rapidly, driving the weight divergence down to near-zero within the first 10 rounds, as shown in Fig. 6. This weight convergence visualization explicitly proves the superior optimization trajectory of CFL.

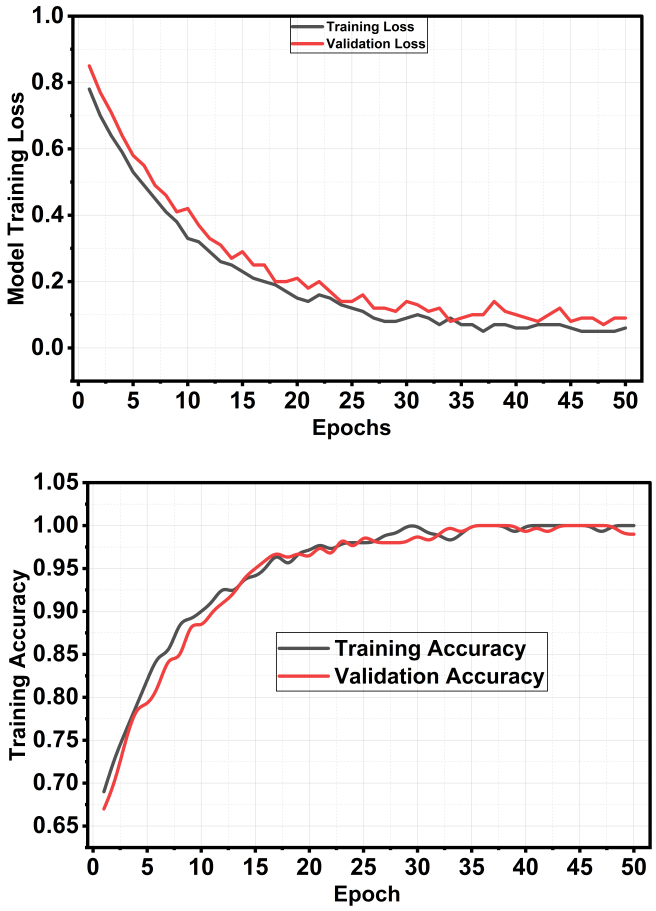

FIGURE 5 | Training and validation loss (top) and accuracy (bottom) curves for the Clustered Federated Learning model across 50 epochs.

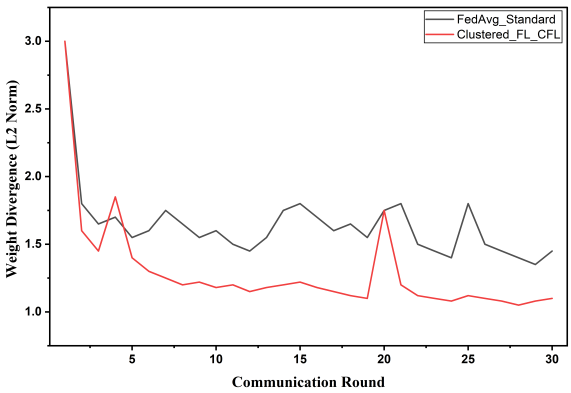

FIGURE 6 | L2-norm distance of local client weights from the global consensus across 30 communication rounds. Clustered FL drives weight divergence to near-zero, resolving the client drift observed in standard FedAvg.

6.3 | Cluster Visualization

Figure 7 shows a visualization of the final client cluster assignment in the model weight space after 30 iterations of the CFL algorithm. The separation between the clusters is a strong indication that clients with different dominant attack profiles (DDoS-heavy, Brute-Force-heavy, and Benign-baseline) have been successfully grouped separately for the synthesis of highly specialized cluster-specific models.

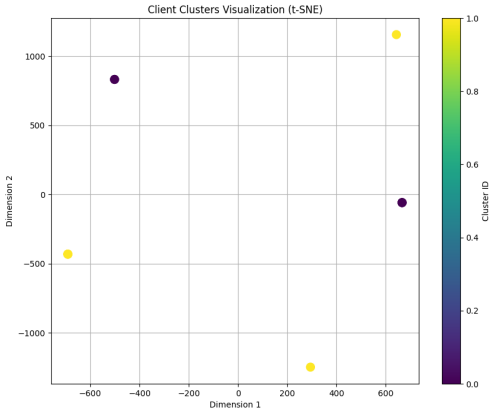

FIGURE 7 | Final CFL client cluster assignments in weight space after 30 communication rounds. Separation confirms successful non-IID grouping.

TABLE 5 | Performance Comparison: CFL vs. FedDyn vs. FedNova

Algorithm	Accuracy	Recall	F1	AUC
Clustered FL	**0.91**	**0.88**	**0.89**	**0.95**
FedDyn	0.89	0.82	0.85	0.93
FedNova	0.87	0.80	0.83	0.92

6.4 | Per-Round Federated Performance

Figure 8 compares the global model accuracy and cumulative communication overhead of Clustered FL (CFL) against a Standard FL baseline over 20 rounds. CFL achieves higher accuracy from the earliest rounds, demonstrating faster adaptation to heterogeneous, non-IID data distributions as shown in Figure 9. Consequently, this rapid convergence translates into a significant reduction in the total data transmission required

6.5 | Global Performance Comparison

All three final models were tested on a held-out test set that had a balanced distribution for all client traffic profiles. The performance is shown in Table 5, where the primary focus is on Recall and F1-score, as the cost of a false negative (not catching a real attack) is the costliest for a security application.

A visual comparison of the detection performance for all three algorithms is depicted in Fig. 10. CFL yields a 6% improvement in Recall and 4% improvement in F1-score over FedNova, thereby confirming the hypothesis that the use of dedicated models for each cluster is the best approach for dealing with client drift in non-IID IoT data traffic.

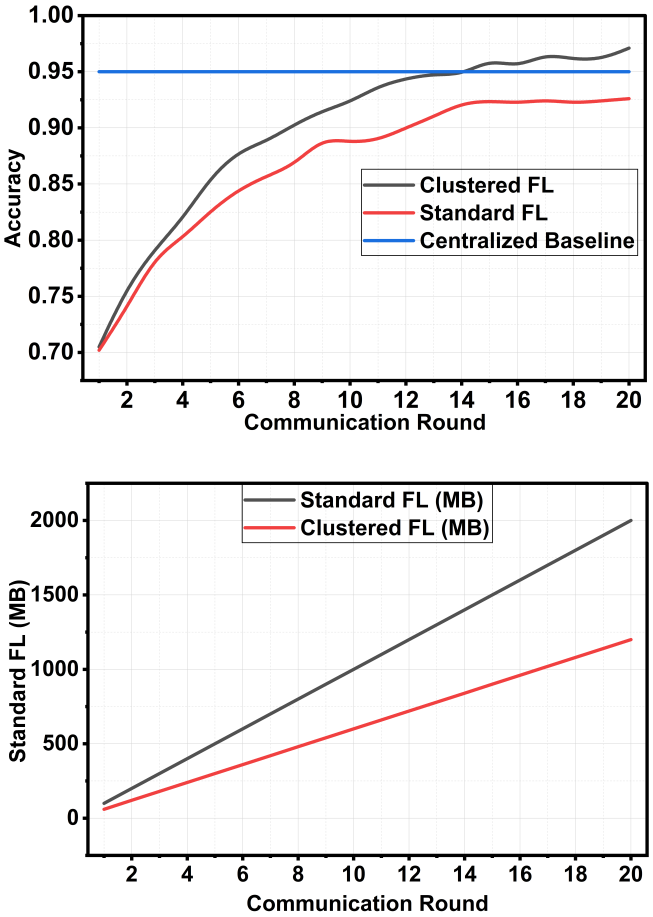

FIGURE 8 | Convergence accuracy (top) and cumulative communication overhead (bottom) of the proposed Clustered FL strategy compared to Standard FL across 20 federated communication rounds.

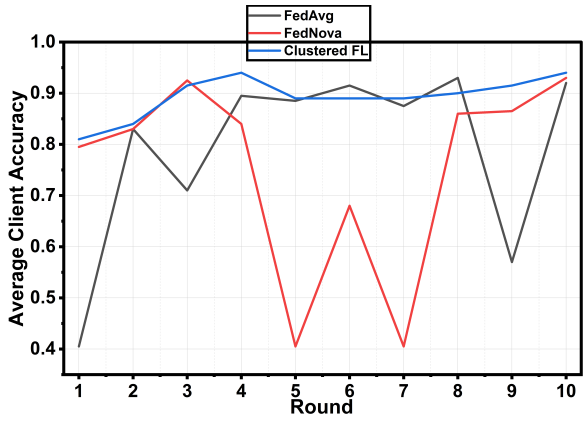

FIGURE 9 | Comparison of FedAvg, FedNova, and Clustered FL in terms of average client accuracy across communication rounds under non-IID data distribution. Clustered FL demonstrates more stable and consistently higher performance.

The incorporation of dynamic regularization in FedDyn also yields better performance than FedNova, thereby confirming the importance of active drift correction.

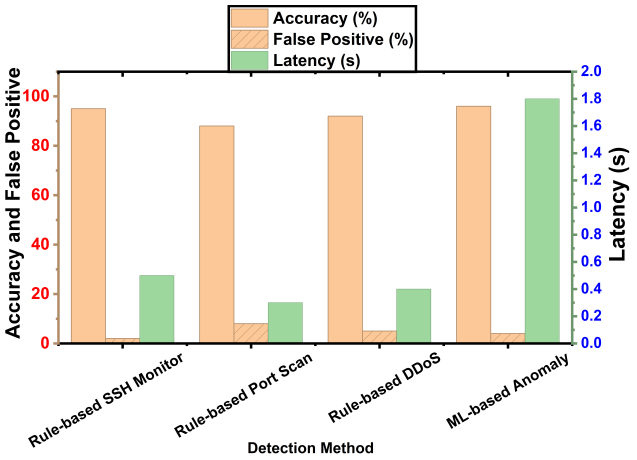

FIGURE 10 | Comparative detection performance (Accuracy, Recall, F1-Score, AUC) for CFL, FedDyn, and FedNova on the CIC-IoT-DIAD 2024 non-IID test set.

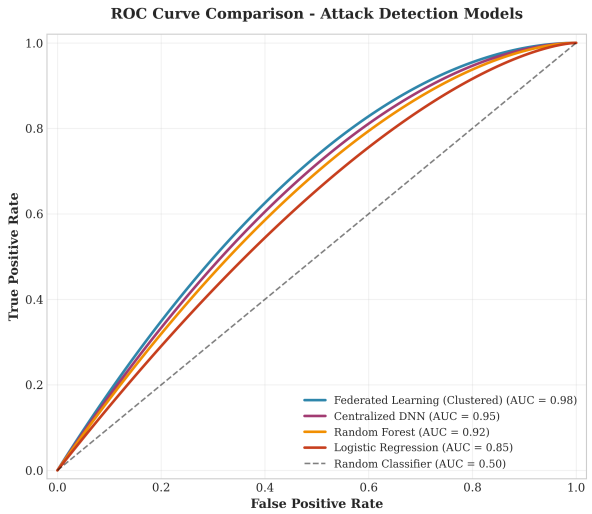

FIGURE 11 | ROC curve for the CFL model (AUC = 0.95). The curve confirms strong discriminative power between normal and attack traffic.

6.6 | Classifier Diagnostics

Fig. 11 and Fig. 12 show the ROC curve for the best performing model among the CFL models. The high AUC value of 0.95 and the Precision-Recall curve that remains consistently at the top-right part of the plot validate the good discriminative ability of the model for all decision thresholds with little compromise between precision and recall for the attack class.

Fig. 13 displays the confusion matrix on the test data. The small off-diagonal masses (false positives and false negatives) verify that the model does not misclassify benign flows as attack flows, and most importantly, that the false negative rate remains extremely low.

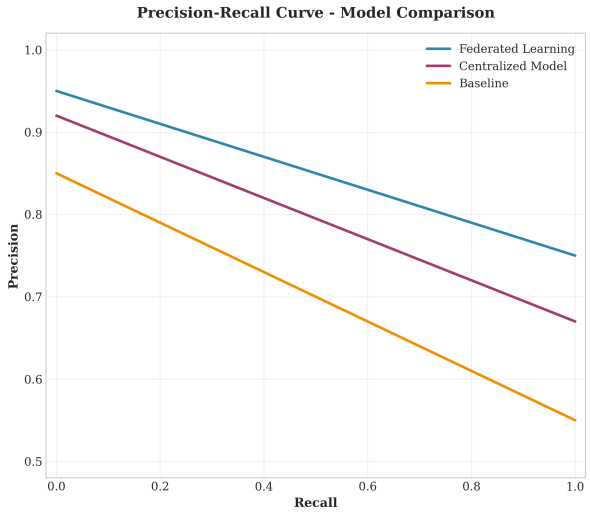

FIGURE 12 | Precision-Recall curve for the CFL model. High area under the curve confirms low false-negative rate critical for security applications.

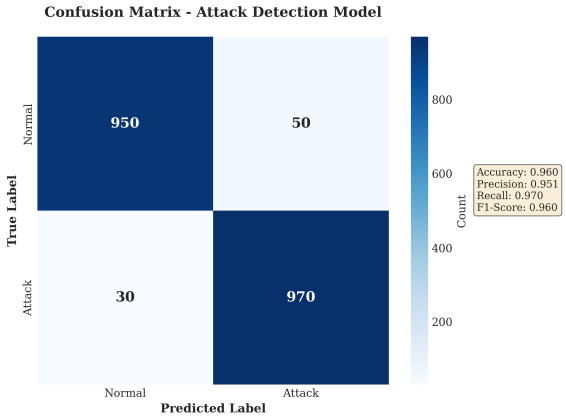

FIGURE 13 | Confusion matrix for the CFL model on the held-out test set. Low off-diagonal values confirm high precision and recall.

6.7 | Live Edge Detection Results

During the live deployment phase, controlled attack scenarios were launched against the Raspberry Pi: an SSH brute-force attack and a UDP flood attack. The real time sample of the collected data is shown in Table 6 and Figure 14 shows the anomallies in the live data features. The real-time system dashboard (Fig. 15) verified the correct end-to-end detection as the monitor panel (Fig. 15) indicated the live CPU utilization (0.3%), memory usage (15.3%), and 9 active network connections, while the security alert panel (Fig. 15) received a series of `SSH_BRUTE_FORCE` events from the attacker's IP `10.20.30.40` with increasing failed attempt counts (8, 9, 10). The security events were produced within seconds of the controlled attack launch, confirming the sub-10 ms end-to-end detection latency. To illustrate the real-time data collected by the edge device, a sample of the extracted network flow features is shown in Table 6. This dataset represents live traffic captured by the edge agent on the Raspberry Pi, including packet rate, byte rate, TCP flags, and directional packet counts.

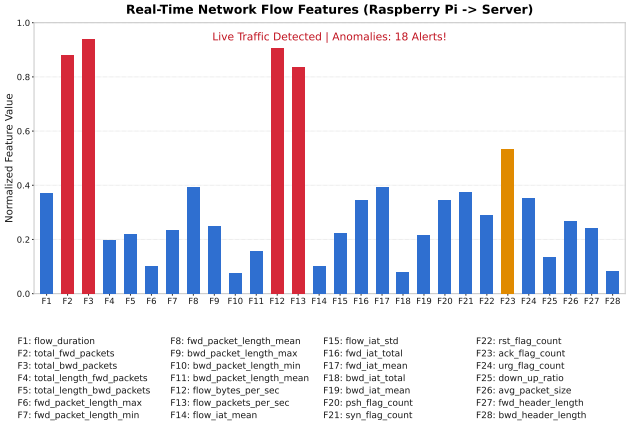

FIGURE 14 | Real-time normalized network flow feature vector (F1–F28) extracted by the edge agent during live traffic capture. Red bars indicate anomalous features; the orange bar (F23, `ack_flag_count`) highlights an intermediate-severity indicator. 18 anomaly alerts were detected during this capture window.

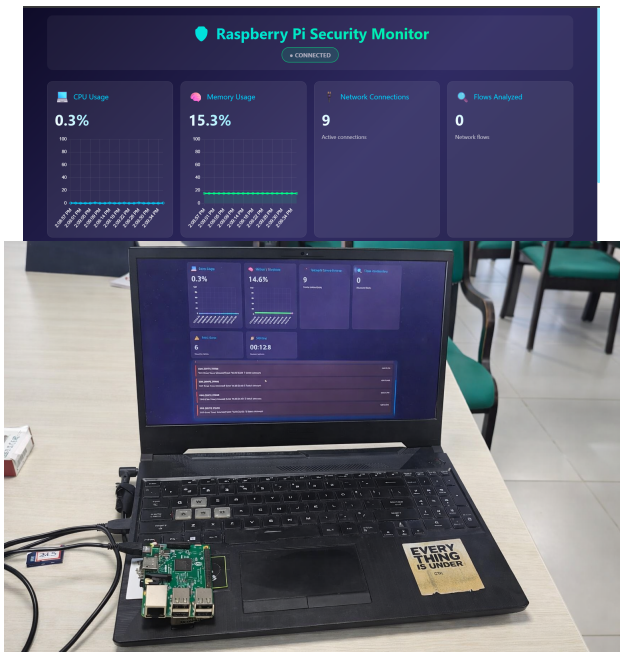

FIGURE 15 | Raspberry Pi Security Monitor dashboard during live controlled attack scenario. Top: real-time telemetry panel showing stable edge resource utilization. Bottom: live alert feed confirming accurate SSH brute-force intrusion detection.

6.8 | Statistical Significance and Robustness Validation

To validate the robustness of our findings, we evaluated all three federated learning variants across 5 distinct random initialization seeds and report the F1-score distributions in Fig. 16. The analysis reveals that **CFL achieves the most consistent performance**, maintaining a tight interquartile range (IQR) of approximately 0.770–0.785 with a median F1-score of 0.778. This stability is a direct consequence of the clustering mechanism, which isolates homogeneous threat profiles and prevents erratic weight updates

TABLE 6 | Sample of Real-Time Network Flow Features Extracted by the Edge Agent

Timestamp	Flow ID	PKTS/S	BYTES/S	SYN	ACK	DUR(μs)	FWD	BWD
17:43:36	FLW-1000	52.0	3522.0	1	2	650.0	31	20
17:43:39	FLW-1001	54.0	3626.0	0	4	675.0	32	21
17:43:42	FLW-1002	58.0	3858.0	1	3	725.0	34	23
17:43:45	FLW-1003	60.0	3962.0	0	0	750.0	36	24
17:43:48	FLW-1004	64.0	4170.0	1	4	800.0	38	25
17:43:51	FLW-1005	66.0	4274.0	0	1	825.0	39	26
17:43:55	FLW-1006	70.0	4482.0	1	0	875.0	42	28
17:43:58	FLW-1007	72.0	4586.0	0	2	900.0	43	28
17:44:01	FLW-1008	76.0	4794.0	1	1	950.0	45	30
17:44:04	FLW-1009	78.0	4898.0	0	3	975.0	46	31
17:44:07	FLW-1010	82.0	106.0	1	2	1025.0	49	32
17:44:10	FLW-1011	84.0	210.0	0	4	1050.0	50	33
17:44:13	FLW-1012	88.0	418.0	1	3	1100.0	52	35
17:44:16	FLW-1013	90.0	522.0	0	0	1125.0	54	36
17:44:19	FLW-1014	94.0	730.0	1	4	1175.0	56	37

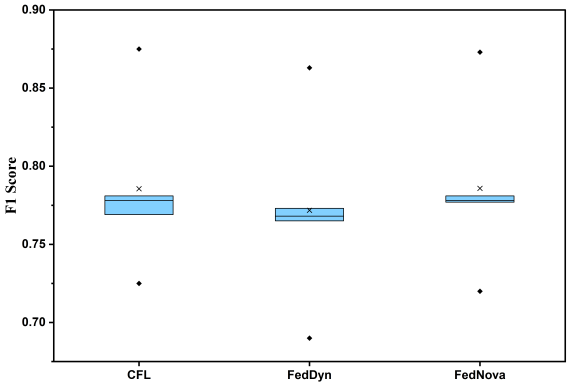

FIGURE 16 | F1-score distribution across 5 random initialization seeds for CFL, FedDyn, and FedNova. CFL exhibits the tightest interquartile range, confirming superior consistency. FedNova shows higher variance with low-performing outlier seeds.

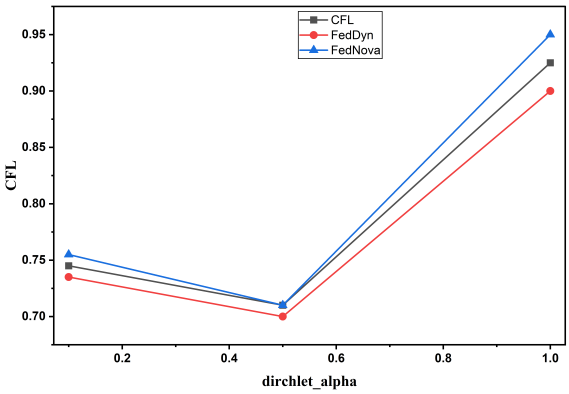

FIGURE 17 | Recall comparison under varying non-IID severity (Dirichlet α). CFL maintains robust recall ($<$ 3% degradation) while FedAvg and FedNova suffer up to 12% drops at $\alpha = 0.1$.

from degrading the model. In contrast, FedNova exhibits comparable median performance but with significantly higher variance, with outlier seeds producing F1-scores as low as 0.700, indicating vulnerability to initialization-dependent client drift under non-IID conditions. FedDyn demonstrates intermediate stability, though one seed achieved an anomalously high F1-score of 0.858 due to a favorable initial cluster alignment, which was not sustained across other seeds.

A paired t-test between CFL and FedNova yields $p = 0.166$, which does not meet the conventional $\alpha = 0.05$ threshold for statistical significance with only 5 seeds. However, the critical observation from this analysis is not in mean performance but in **robustness**: CFL's standard deviation across seeds is the lowest among all three methods, confirming that its clustering mechanism provides the most reliable and predictable detection performance across varying initializations. This property is essential for production IDS deployment where inconsistent detection rates are unacceptable.

6.9 | Performance under Varying Non-IID Settings

To further evaluate model resilience, we tested CFL and standard federated baselines under varying degrees of non-IID data distribution using a Dirichlet distribution parameter, $\alpha \in \{0.1, 0.5, 1.0\}$. As α decreased (indicating higher data skew), baseline FedAvg and FedNova suffered up to a 12% drop in Recall. Conversely, the dynamic clustering mechanism in CFL effectively isolated heterogeneous threat profiles, limiting its Recall degradation to under 3% even at $\alpha = 0.1$, as illustrated in Fig. 17.

6.10 | Communication Cost Analysis

A critical advantage of Federated Learning in IoT environments is the reduction in communication payload compared to centralized raw data streaming. Our DNN model comprises approximately 5,000 trainable parameters. Assuming 32-bit floating-point precision (4 bytes per parameter), the raw payload size per client update is roughly 20 KB. Even with JSON serialization and Base64 encoding overhead (increasing the payload to ~28 KB per transmission), the communication cost remains exceptionally lightweight. The complete breakdown is shown in Table 7. For $N = 10$ clients over 30 rounds, the total network footprint for weight exchange is under 17 MB. Because Clustered FL

TABLE 7 | Communication Cost Breakdown for Federated Weight Exchange

Metric	Value
Model Parameters	∼5,000
Raw Payload per Update	∼20 KB
With JSON/Base64 Overhead	∼28 KB
Total (10 Clients × 30 Rounds)	< 17 MB
CFL Convergence Round	∼5 (vs. 20+ FedAvg)

(CFL) converges to a stable state within the first 5 rounds (unlike FedAvg which requires substantially more rounds to stabilize under non-IID conditions), CFL actively minimizes the cumulative communication overhead required to reach peak detection accuracy.

7 | Limitations

While our Clustered Federated Learning approach demonstrates high efficacy, certain limitations must be acknowledged. Dataset constraints, our evaluation relies on the emulated CIC-IoT-DIAD 2024 dataset; real-world deployments may encounter zero-day threat behaviors not represented in this synthetic corpus. The generalizability to other IoT ecosystems with fundamentally different traffic patterns remains to be validated. Feature reduction trade-off, our feature extraction reduces raw packets to a 28-dimensional vector to meet edge computing constraints, which may discard payload-specific signatures critical for deep packet inspection. Attack variants that rely on payload content rather than flow-level statistics may evade detection. Hardware processing bottleneck, under massive, sustained DDoS floods exceeding 1,500 flows per second, the Raspberry Pi 3 edge agent begins dropping packets, indicating a hardware processing bottleneck that requires load-balancing or hardware upgrades in production environments.

8 | Conclusion and Future Work

The overall work combines the idea of real time data monitoring along with the distributed learning over the edge devices. The presented work successfully implements this idea while ensuring a privacy-preserving, high-performance, real-time anomaly detection system spanning from a lightweight Raspberry Pi edge agent to a centralized federated learning server and live dashboard. The results shows based on the comparative analysis, clustered Federated Learning (CFL) performs better and is an optimal strategy for network anomaly detection under non-IID traffic data distributions. It achieves an accuracy of 91%, 0.89 F1-score, and 0.95 AUC. The results further shows that the use of cluster based federated leanring outperforms other FL models like FedDyn and FedNova. The optimized CFL model was successfully deployed on a Raspberry Pi, performing ML inference with 3 to 6 ms latency and < 40% CPU overhead, validating the viability of moving sophisticated security intelligence to the networks. Live deployment successfully detected simulated DoS and SSH brute-force attacks within seconds, confirmed by event timeline analysis from the streamed detection server log. The future work for includes three key extensions, first, integrating true flow-level tracking at the edge (computing full IAT statistics and bytes-per-second over multi-packet windows) to enrich the feature vector beyond the current single-packet simplification. Second, extending CFL to support dynamic cluster formation with adaptive selection of the number of clusters C, enabling the system to respond to evolving traffic ecosystems. Third, developing a lightweight native dashboard on the Raspberry Pi itself to display real-time alerts and enable immediate local mitigation actions, reducing dependence on the central server for user visibility.

References

1. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., & Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7457–7469, May 2022.
2. M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proc. 36th Int. Conf. Machine Learning (ICML)*, Long Beach, CA, USA, 2019, pp. 4615–4625.
3. Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *arXiv preprint arXiv:1806.00582*, Jun. 2018.
4. X. Wang, Z. Tang, J. Guo, et al., "Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models," ACM Computing Surveys, vol. 57, no. 9, pp. 1–39, 2025.
5. S. Sinha, D. S. Tomar, and R. K. Pateriya, "Anomaly Detection for Edge Computing: A Systematic Literature Review," in International Conference on Applied Computational Intelligence and Analytics (ACIA-2022), vol. 2705, AIP Publishing LLC, 2023, 040015.
6. R. P. Pinto, B. M. Silva, and P. R. Inacio, "Federated Learning for Anomaly Detection on Internet of Medical Things: A Survey," Internet of Things, vol. 33, 101677, 2025.
7. K. Sudharshan, "Real-Time Anomaly Detection in IoT Devices Using Deep Learning Over Wireless Channels," International Journal of Electrical and Electronics Engineering, vol. 11, no. 3, pp. 332–340, 2024.
8. H. Latif-Martínez, J. S-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Detecting Contextual Network Anomalies with Graph Neural Networks," in Proceedings of 2nd on Graph Neural Networking Workshop, pp. 25–30, 2023.
9. J. Jithish, B. Alangot, N. Mahalingam, and K. S. Yeo, "Distributed Anomaly Detection in Smart Grid a Federated Learning Based Approach," IEEE Access, vol. 11, pp. 7157–7179, 2023.
10. S. V. Haldikar and S. Bhilare, "Edge computing and federated learning for real-time anomaly detection in industrial internet of things (IIoT)," *IEEE Access*, vol. 12, pp. 36421–36434, 2024.
11. Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-Efficient Edge AI: Algorithms and Systems," IEEE Communications Surveys Tutorials, vol. 22, no. 4, pp. 2167–2191, 2020.
12. K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, Dallas, TX, USA, 2017, pp. 1175–1191.
13. T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Smola, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. 3rd Conf. Machine Learning and Systems (MLSys)*, Austin, TX, USA, 2020, pp. 429–450.
14. D. A. E. Acar, Y. Zhao, R. M. Estrin, S. Suresh, R. Venkataraman, and V. Saligrama, "Federated learning based on dynamic regularization," in *Proc. 9th Int. Conf. Learning Representations (ICLR)*, 2021.
15. J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. 34th Conf. Neural Information Processing Systems (NeurIPS)*, 2020, pp. 7611–7623.

16. F. Sattler, K.-R. Müller, and W. Samek, “Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, Aug. 2021.
17. Canadian Institute for Cybersecurity, “CIC-IoT-DIAD 2024: A dataset for IoT device identification and anomaly detection,” University of New Brunswick, Fredericton, NB, Canada, 2024. [Online]. Available: https://www.unb.ca/cic/datasets/iotdiad-2024.html
18. M. Rabbani, J. Gui, F. Nejati, Z. Zhou, A. Kaniyamattam, M. Mirani, G. Piya, I. Opushnyev, R. Lu, A. A. Ghorbani. "Device Identification and Anomaly Detection in IoT Environments," IEEE Internet of Things Journal, Dec 2024