2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

FMLTC-IDS: A Federated Meta-Learning and Adaptive Time Clustering-based IoT Intrusion Detection System

Jingxian Zhou[1], Zhou Liu[1], and Qiang Zhu[2]

[1] School of Safety Science and Engineering, Civil Aviation University of China, Tianjin, China
2541706816@qq.com
[2] Daxing Air Traffic Control Center, Civil Aviation North China Air Traffic Management Bureau, Beijing, China

Abstract. With the rapid proliferation of IoT devices, network security threats have intensified. Federated Learning (FL) has been applied to anomaly-based Network Intrusion Detection Systems (NIDS) to identify malicious traffic and mitigate risks. However, traditional FL struggles to handle Non-IID data, and while Personalized FL (PFL) improves adaptability, it remains insufficient in addressing the dynamic nature of time-series data. To address these issues, this paper proposed an IoT Intrusion Detection System based on Federated Meta-Learning and Adaptive Temporal Clustering (FMLTC-IDS). The method combines Model-Agnostic Meta-Learning (MAML) to optimize the initialization of the global model, enhancing personalized adaptation. It also introduces adaptive batch adjustment and gradient-weighted sampling strategies to improve local training efficiency. Additionally, Principal Component Analysis (PCA) is used for dimensionality reduction, and a time-series-based dynamic weighted DBA-K-Means clustering method is employed to optimize model clustering quality, enhancing the system's ability to handle spatiotemporal non-IID data. Experimental results show that FMLTC-IDS achieves excellent performance on CICIDS2017, BoT-IoT, and real IoT traffic datasets, outperforming existing methods (e.g., Fed-ANIDS, SSFL) by more accurately adapting to data heterogeneity, improving Accuracy, Recall, and F1-score, and accelerating model convergence. Furthermore, ablation experiments validate the effectiveness of dynamic batch adjustment, PCA dimensionality reduction, and time-series clustering strategies, demonstrating significant advantages in enhancing personalized detection capabilities and overall detection accuracy for FMLTC-IDS.

Keywords: IoT Security, Personalized Intrusion Detection, Federated Learning, spatial-temporal non-IID problem.

1 Introduction

The Internet of Things (IoT) is composed of information sensing devices, communication technologies, and intelligent computing systems, enabling deep integration between the physical and digital worlds through sensors, radio-frequency identification (RFID), and smart terminals. By facilitating extensive interconnectivity, IoT enables real-time communication and collaborative computing between devices, driving digital transformation in various domains such as smart homes and Industry 4.0 [1,2]. Within the IoT ecosystem, heterogeneous devices generate multimodal data streams (e.g., environmental parameters, device states, and user behaviors), which are integrated by edge computing nodes to form valuable data assets. However, the traditional centralized data processing paradigm faces significant security challenges during data transmission: data aggregation points may become attack targets, privacy breaches are exacerbated by ambiguous data-sharing boundaries, and unclear data ownership complicates regulatory compliance.

To address these challenges, Federated Learning (FL) has emerged as a core technology in privacy-preserving computing. FL is an advanced and secure distributed machine learning technique that enables multiple clients to train models locally without uploading raw data to a central server [3]. Under the FL framework, clients share only model parameter updates, while a central server aggregates these updates to optimize the global model. However, in real-world IoT systems, client data is often non-independently and identically distributed (non-IID). This Non-IID nature primarily stems from multiple factors, including differences in device deployment environments, user behavior patterns, and data collection times. For example, security checkpoint cameras in airports monitor both passengers and luggage inspections, causing significant traffic fluctuations, while boarding gate cameras primarily track passenger queues, resulting in more stable data.

Since the introduction of federated learning, addressing the non-IID data distribution problem has remained one of the core challenges and a fundamental characteristic of real-world FL scenarios. As non-IID issues arise due to heterogeneous client data, one potential solution is to personalize models for each client. Meta-learning is a powerful approach to achieving personalization, as it optimizes the global model to better adapt to diverse client data distributions, enabling fast adaptation across heterogeneous environments [4].

Moreover, IoT networks exhibit non-IID characteristics not only in the spatial domain (i.e., data across different clients) but also in the temporal domain. In the spatial domain, variations in network environments, device types, and traffic patterns across clients lead to significant distribution shifts. In the temporal domain, even for the same device, network dynamics (e.g., signal interference, bandwidth contention) and TCP retransmission mechanisms cause fluctuations in traffic patterns over time.

Furthermore, IoT intrusion detection systems (IDSs) often have stringent latency requirements. For example, in smart grids, a detection delay exceeding 10ms can lead to severe consequences, whereas maintaining a response time within 5ms significantly enhances security [5]. However, traditional Clustered Federated Learning (CFL) approaches struggle to meet the real-time demands of IoT environments due to their high computational and communication overhead. Specifically, CFL requires transmitting full high-dimensional model parameters (e.g., PointNet++), making a single clustering operation computationally expensive and inefficient for low-latency IoT applications.

To address the above issues, we propose an innovative Federated Meta-Learning and Adaptive Temporal Clustering framework (FMLTC-IDS). This approach integrates Federated Learning (FL), Meta-Learning, and Unsupervised Learning techniques to effectively handle cross-device and cross-temporal Non-IID data while preserving data privacy.

Our key contributions are as follows:

- Global Generalization and Edge Personalization: By leveraging Model-Agnostic Meta-Learning (MAML) to optimize the global model initialization, FMLTC-IDS can rapidly adapt to different edge device data distributions, improving personalized detection accuracy. Additionally, we introduce adaptive batch adjustments and gradient-weighted sampling strategies to dynamically optimize local training, reducing communication overhead and accelerating model convergence.
- Spatiotemporal Non-IID Data Handling: We integrate Principal Component Analysis (PCA) to remove redundant information, enhancing clustering efficiency. Furthermore, we propose a time-series-based dynamic weighted DBA-K-Means clustering method to effectively group clients, optimizing personalized model aggregation and improving the adaptability and generalization of the global model.
- Experimental Validation and Performance Superiority: Extensive experiments on CICIDS2017 [6], BoT-IoT [7], and real IoT traffic datasets demonstrate that FMLTC-IDS consistently outperforms state-of-the-art methods (e.g., FedAvg [8], FedProx [9], Fed-ANIDS [10], and SSFL [11]) in highly heterogeneous environments in terms of accuracy, recall, and F1-score, while achieving faster convergence with fewer communication rounds. Ablation studies further validate the effectiveness of adaptive batch adjustments, PCA-based dimensionality reduction, and time-series clustering strategies.

2 Related Work

In recent years, Federated Learning (FL) has gained widespread attention in the field of Intrusion Detection Systems (IDS), as researchers aim to improve the generalization ability and robustness of detection models while ensuring data privacy. Mothukuri et al. [12] proposed a distributed FL architecture, utilizing GRU networks for time-series feature modeling in IoT networks and combining a random forest decision mechanism for multi-source information fusion to improve traffic classification accuracy. Tahir et al. [13] designed a decentralized FL framework with a hierarchical attention aggregation mechanism, achieving breakthrough results in false data injection detection tasks within the IEEE 39-node power system. Aouedi et al. [14] introduced a semi-supervised autoencoder (AE)-based FL scheme, which effectively extracts intrusion patterns and improves detection accuracy under low-labeled data conditions through distributed feature reconstruction. Chen et al. [15] proposed the FDAGMM model, optimizing DAGMM for detection capabilities in small sample environments by using deep autoencoders (DAE) and Gaussian Mixture Models (GMM), enhancing adaptability for anomaly traffic detection. Zhao et al. [16] introduced a FL and transfer learning-based network anomaly detection method to address the scarcity of training data in network anomaly detection. Their model achieved a 97.23% detection success rate for vulnerability attacks on the UNSW-NB15 dataset.

FL protects privacy and reduces communication costs through local training and parameter sharing. However, statistical heterogeneity limits the adaptability of global models. Personalized Federated Learning (PFL), combining global sharing with local optimization, allows IoT devices to adjust models to perform better in Non-IID environments. Lu et al. [17] proposed a self-labeling mechanism to address the labeling dependency problem in IoT intrusion detection for PFL, significantly reducing communication overhead and achieving traditional PFL detection performance without manual labeling. He et al. [18] introduced the FedLGS framework, which combines gradient masking and personalized model updates to solve the client drift problem in Non-IID data by dynamically allocating weights, ensuring efficient privacy protection. Deng et al. [19] presented the FedASA framework, which achieves high accuracy, low communication cost, and fairness in cross-location resource-constrained scenarios by adaptive architecture partitioning, modular aggregation, and mixed label optimization, providing efficient solutions for industrial IoT. Zhang et al. [20] proposed a model-layered and local perturbation dual mechanism, dividing the model into global and local layers, and introduced three types of gradient perturbations (zeroing gradients, random noise, and noise addition) to enhance defensive flexibility and improve the security and personalized adaptability of federated learning. This paper adopts meta-learning to implement personalized federated learning, enabling the model to quickly adapt to new tasks based on existing knowledge, enhancing learning ability and adaptability, while maintaining flexibility and accuracy in heterogeneous environments.

3 The Proposed Method

3.1 FMLTC-IDS Workflow

The proposed FMLTC-IDS framework consists of two main entities: edge nodes and the server, integrating federated learning (FL), meta-learning, and variational autoencoder (VAE) [21]to achieve efficient and personalized intrusion detection.

Edge Nodes: Responsible for local data training, gradient-adaptive sampling, and personalized optimization, and upload model parameters for global aggregation.

Server: Performs global model aggregation, feature extraction, time-series clustering, and dynamic adjustments, optimizing the personalized model update strategy.

The workflow of the FMLTC-IDS framework can be divided into four main stages:

(1) Local Dataset Creation: Collecting data using a layered simulation experiment platform and balancing low-traffic periods through ADWIN-based dynamic time window segmentation and SMOTE-TS, forming a series of temporal evolution subsets {D1, …, D10}.

(2) System Initialization: The central server trains the base model Mg using public datasets and optimizes it via the MAML framework, ensuring that the initial parameters $W0$have strong generalization capabilities.

(3) Feature Extraction and Dimensionality Reduction: The server collects the final layer parameters from each device to construct a parameter matrix, applies PCA

for dimensionality reduction to extract key features, and utilizes sliding window analysis to track feature evolution trends, optimizing subsequent clustering.

(4) Time-Series-Based Clustering: Using the features after PCA dimensionality reduction, a dynamically weighted DBA-optimized K-Means clustering method is employed to determine the optimal number of clusters using the elbow method. FedAvg is then used for model aggregation within each cluster.

During the FL iterations, IoT devices and the server continuously interact, with devices adjusting their models locally while the server aggregates updates to form a global model. By integrating diverse data from various clients, the global model enhances generalization. The server dynamically adjusts feature allocation and clustering, ensuring personalized optimization and ultimately generating a stable and accurate intrusion detection model. The FMLTC-IDS framework is shown in Fig. 1.

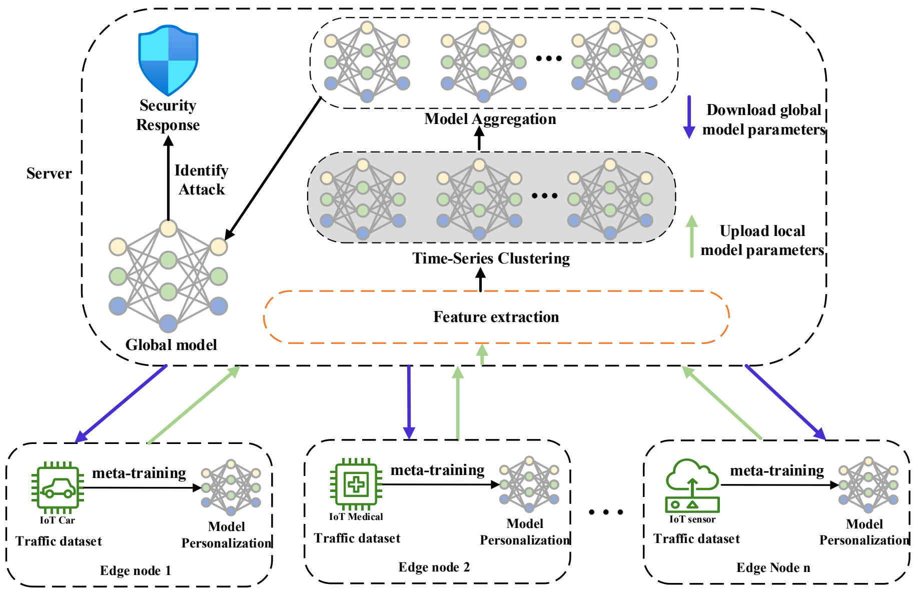

Fig. 1. Overall Flowchart of FMLTC-IDS

3.2 Personalized Federated Meta-Learning in Edge Nodes

We draw inspiration from MAML and train a base model Mg on the central server using a public dataset, optimizing the initial parameters W_0 to provide a generalized initialization. This enables the global model to quickly adapt to local tasks on different edge devices.

In federated meta-learning, fixed batch sizes can strain resource-limited devices or underutilize those with little data. To overcome this, we propose a gradient-based adaptive sampling strategy that adjusts batch size dynamically based on each device's computational capacity and data volume, improving training efficiency and personalization.

We dynamically adjust the batch size using device-specific factors such as computational capacity (GPU/CPU specifications, memory) and data volume. The formula is as follows:

$$B_i = \max\left(B_{\min}, \min\left(B_{\max}, \frac{|D_i|}{c} \times R_i\right)\right) \tag{1}$$

Where: B_i is the batch size used by device i. $B_{\min}$ and $B_{\max}$ are the predefined minimum and maximum batch sizes (e.g., 16 and 128). $|D_i|$ represents the local dataset size of device i. c is a control factor to prevent excessively large batch sizes. R_i denotes the computational resource score of the device (e.g., GPU computing power) with a range of $0<R_i\leq 10$

This mechanism ensures that devices with higher computational capacity and larger datasets can utilize larger batch sizes for training, while resource-constrained or data-limited devices adopt smaller batch sizes. This improves overall training efficiency and reduces computational burden.

To further optimize the training process, we introduce Gradient-Based Weighted Sampling, which prioritizes samples that contribute more significantly to model updates. The specific steps are as follows:

Step1: For model parameters W and loss function L, compute the gradient for each sample x:

$$g(x) = \|\nabla_W L(W,x)\|_2 \tag{2}$$

Where $g(x)$ represents the L2 norm of the gradient for sample x, measuring its impact on model updates.

Step2: Compute the gradient norm for all samples and sort them in descending order, prioritizing samples with larger gradients:

$$\{x_1,x_2,...,x_n\}, \quad g(x_1) \geq g(x_2) \geq ... \geq g(x_n) \tag{3}$$

Normalize the gradient norms of all samples to ensure their sum equals 1:

$$p_i = \frac{g_i}{\sum_{j=1}^{n} g_j} \tag{4}$$

Where p_i represents the sampling probability of sample i; samples with larger gradients have a higher probability of being selected.

Step3: Based on the computed batch size, perform weighted random sampling using a multinomial distribution:

$$S \sim Multinomial(B_i,p) \tag{5}$$

Where S is the selected subset of data samples, B_i is the adaptive batch size, and $p(x)$ represents the sampling probability of each sample.

Through this approach, we ensure that the batch size adapts to the device's computational capacity while prioritizing samples that contribute more significantly to the loss function, thereby accelerating model convergence.

To further enhance convergence efficiency and stability, we employ the Adam optimizer for local gradient updates and incorporate a learning rate decay strategy. As training progresses, the learning rate gradually decreases to prevent oscillations, allowing the model to converge more smoothly.

After local updates, devices upload their results to the server, where the global model is aggregated and optimized. Through multiple training rounds, each device fine-tunes the model based on its local data, forming a personalized model that better adapts to the local data distribution.

3.3 Feature Extraction Method

The server extracts key features by analyzing the model updates uploaded by each client to gain a more comprehensive understanding of the differences between clients and their data distribution characteristics. Specifically, the server focuses on the parameters of the final layer (i.e., the output layer) in client j's model. The weights in this layer directly reflect the relationship between the neurons and the output categories.

Assume that the connection weight between the i-th neuron and the r-th output unit in the final layer of client j's model is $W^{j}_{i,r}$.The server collects the final layer parameters from all clients and constructs the parameters for each client j into the following matrix:

$$W_j = \begin{bmatrix} w^j_{1,1} & w^j_{1,2} & \cdots & w^j_{1,R} \\ w^j_{2,1} & w^j_{2,2} & \cdots & w^j_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w^j_{M,1} & w^j_{M,2} & \cdots & w^j_{M,R} \end{bmatrix} \tag{6}$$

Where: M represents the number of neurons in the final layer, R represents the number of output classes, W_j records the weight parameters of the final layer for client j.

By constructing such matrices, the server can systematically gather features from each client, laying the groundwork for subsequent feature processing and dimensionality reduction. However, since W_j has high dimensionality, using it directly may lead to significant computational overhead and be susceptible to noise. Therefore, on the server side, we perform Principal Component Analysis (PCA) on the collected client parameter matrices to extract more representative features.

To reduce data redundancy and improve computational efficiency, the server first computes the covariance matrix C:

$$C = \frac{1}{N}\sum_{j=1}^{N}\left(W_j - \bar{W}\right)\left(W_j - \bar{W}\right)^T \tag{7}$$

Where $\bar{W}$ represents the mean of the parameters across all clients. Next, the covariance matrix C is subjected to eigenvalue decomposition, and the principal components that together explain 95% of the cumulative variance are selected to form the dimensionality reduction matrix P:

$$W_j' = P^T W_j \tag{8}$$

Here, W_j' represents the dimension-reduced client feature vector, providing a more compact and efficient client feature representation that enables the server to analyze different client characteristics more effectively.

The dimension-reduced feature vectors not only preserve key information but also significantly reduce storage and computation costs, enhancing overall system efficiency and performance. By eliminating noise and redundancy, these features provide more reliable inputs for client clustering, classification, and personalized model generation, making federated learning more adaptive and robust.

3.4 Time-Series Based Clustering Method

In the time-series clustering task, we employ a dynamically weighted DBA (DTW Barycenter Averaging) optimized K-Means algorithm to enhance the processing capability for sequential data. Combined with PCA-reduced feature vectors—viewed as time-series data extracted from model parameters—this method produces smoother and more representative cluster centroids, improving clustering quality and enhancing model personalization.

K-Means Clustering Algorithm Based on Dynamically Weighted DBA:

Step1: Initializing Cluster Centers: On the server side, during the training of the MAML initial model using the public dataset, we simultaneously extract the encoder output feature vectors as meta-cluster centers. Leveraging prior knowledge of feature distributions obtained through meta-learning, we initialize the cluster centers of DBA-K-Means to better align with the real data's temporal evolution patterns.

Step2: Computing DTW Distance: We introduce gradient importance weights at each time step, prioritizing the alignment of time periods that contribute more significantly to model updates.

$$\mathrm{DTW}_{\mathrm{grad}}(Q_i, \mu_j) = \sum_{(p,q) \in W} (Q_i(p) - \mu_j(q))^2 \cdot \sqrt{g(Q_i(p))} \tag{9}$$

Where,$g(Q_i(p))$ represents the L2 norm of the gradient of the sample at time point p, which is calculated during local training on the client and then uploaded.

Step3: Dynamic Feature Fusion: On the server side, we concatenate the PCA-reduced feature vector h_j with the gradient statistics uploaded by the client (e.g., gradient mean and variance) to form an enhanced feature representation:

$$h_j^{\mathrm{enhanced}} = [\, h_j;\ \mathrm{mean}(g_j);\ \mathrm{var}(g_j)\,] \tag{10}$$

The enhanced features not only capture the temporal patterns of the time-series data but also incorporate dynamic information from the model updates, thereby improving the clustering's discriminative power.

Step4: Dynamic Weighted DBA to Compute New Cluster Centroids: To obtain more stable and representative cluster centroids, we use the dynamically weighted DBA method to update the center sequence of each cluster. For each centroid time step t∈[1,T$_{\max}$], we calculate the client contribution weight:

$$\gamma_i(t) = \begin{cases} \exp\left(-\underbrace{\frac{(g_i(t)-\overline{g})^2}{2\sigma_g^2}}_{\text{Gradient Stability Term}}\right) \times \underbrace{\frac{B_i}{B_{\max}}}_{\text{Batch Quantization Term}}, & \text{if confrontation exists} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Where, the gradient term: An exponential decay function with a normal distribution is used. If $g_i(t)$ deviates from the global mean g, the weight is reduced. Batch term: Linear normalization is applied to ensure that the weights of larger batch clients dominate, exceeding 50%. Then, perform a weighted average update of the centroid.

$$\mu^{(k+1)}(t) = \frac{\sum_{i=1}^{N} \gamma_i(t) x_i(\pi_i^{-1}(t))}{\sum_{i=1}^{N} \gamma_i(t) + \epsilon} \tag{12}$$

Where, $\epsilon=10^{-8}$ is used to prevent division by zero. Numerator: The sum of the weighted observations of all clients at time step t. Denominator: The sum of all weights, achieving normalization.

Step5: Iterate until convergence:

Repeat steps 2-4 and compute the centroid change.

$$\Delta\mu = \frac{1}{T_{\max}} \sum_{t=1}^{T_{\max}} |\mu^{(k+1)}(t) - \mu^{(k)}(t)| \tag{13}$$

The algorithm terminates when $\Delta\mu<10^{-4}$ or the maximum number of iterations (usually 50) is reached.

To determine the optimal number of clusters K, we use the Elbow Method. First, calculate the Sum of Squared Errors (SSE) for different values of K.

$$\text{SSE}(K) = \sum_{k=1}^{K} \sum_{i \in C_k} \text{DTW}_{grad}(Q_i, \mu_k) \tag{14}$$

The K-SSE curve tracks the sum of squared errors (SSE) as cluster numbers (K) rise. Initially, SSE drops sharply with higher K, but at the "elbow point," the decline rate slows, signaling optimal K. This balances precision and computational cost, preventing overfitting. Clusters beyond this yield marginal gains. Servers then employ FedAvg within each cluster, aggregating local models to harmonize parameters and enhance the global model's generalization across varied data patterns, optimizing federated learning efficiency and cluster utility.

4 Experiment

4.1 Experimental Setup and Datasets

Current research on IoT intrusion detection predominantly relies on limited datasets, which exhibit significant limitations. On one hand, attack samples are confined to simplistic scenarios with static patterns, failing to reflect the dynamic evolution of real-world network traffic. On the other hand, the absence of complex background traffic (e.g., concurrent normal operations and attack scenarios) leads to insufficient model generalization capabilities.

For evaluation, we first train an initial model on the server using a public dataset (CICIDS2017) to capture cross-domain attack features and universal traffic patterns, establishing foundational detection capabilities. This model is then deployed to local clients and fine-tuned with our collected real-world dataset to adapt to time-sensitive network environments and traffic heterogeneity.

To construct a high-quality temporally evolving dataset, we developed a hierarchical simulation experimental platform (Fig 2), comprising firewalls, smart routers, traffic acquisition systems, and IoT terminal devices (e.g., smart cameras, sensors, and smart plugs).

Our preprocessing pipeline includes missing value imputation, outlier detection, PCA dimensionality reduction, and standardization to ensure data accuracy and consistency. Additionally, we employ the ADWIN algorithm for dynamic time window segmentation (1-hour windows with 30-minute sliding steps) and integrate SMOTE-TS oversampling to balance low-traffic periods, partitioning the dataset into temporally evolving subsets {D1,…,D10}. During federated learning, the model is incrementally updated in the order from D1 to D10, effectively capturing the dynamic evolution and personalized characteristics of IoT traffic.

We implement our method using the Flower federated learning framework, which offers scalable server and client support for flexible architectures. Experiments are conducted on Ubuntu 20.04.3 LTS with an NVIDIA GeForce RTX 3090 GPU.

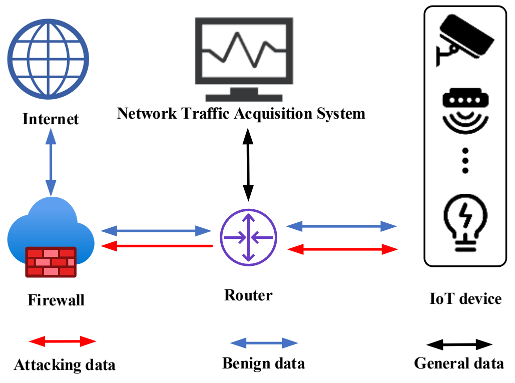

Fig. 2. Layered Simulation Experiment Platform Architecture

4.2 Personalized Performance of FMLTC-IDS

We evaluated the personalized detection capability of FMLTC-IDS on the CICIDS2017, BoT-IoT, and real IoT traffic datasets, comparing it with baseline methods including FedAvg, FedProx, Fed-ANIDS, and SSFL. To comprehensively assess model performance, we used Accuracy, Recall, and F1-Score as evaluation metrics.

The real IoT traffic dataset follows the same construction method as the dataset in Section 4.1 (without being divided into subsets) and contains 10,000 instances of normal and attack traffic. All experimental datasets were configured with α values of [0.5, 1.0, 10] to control data heterogeneity, where smaller α values indicate highly imbalanced distributions, while larger α values approximate an IID setting. The test nodes followed the same distribution parameters to ensure evaluation consistency.

The experimental results (Fig 3) show that as the α value increases, the accuracy of each model improves, and the convergence speed accelerates. Notably, FMLTC-IDS not only converges faster but also achieves higher accuracy, particularly performing well on the real IoT traffic dataset. For example, when α=0.5, FMLTC-IDS converges in just 5 communication rounds, and after 10 rounds, the accuracy reaches 74.7%, outperforming FedAvg, FedProx, Fed-ANIDS, and SSFL by 4.6%, 4.5%, 2.9%, and 2.6%, respectively.

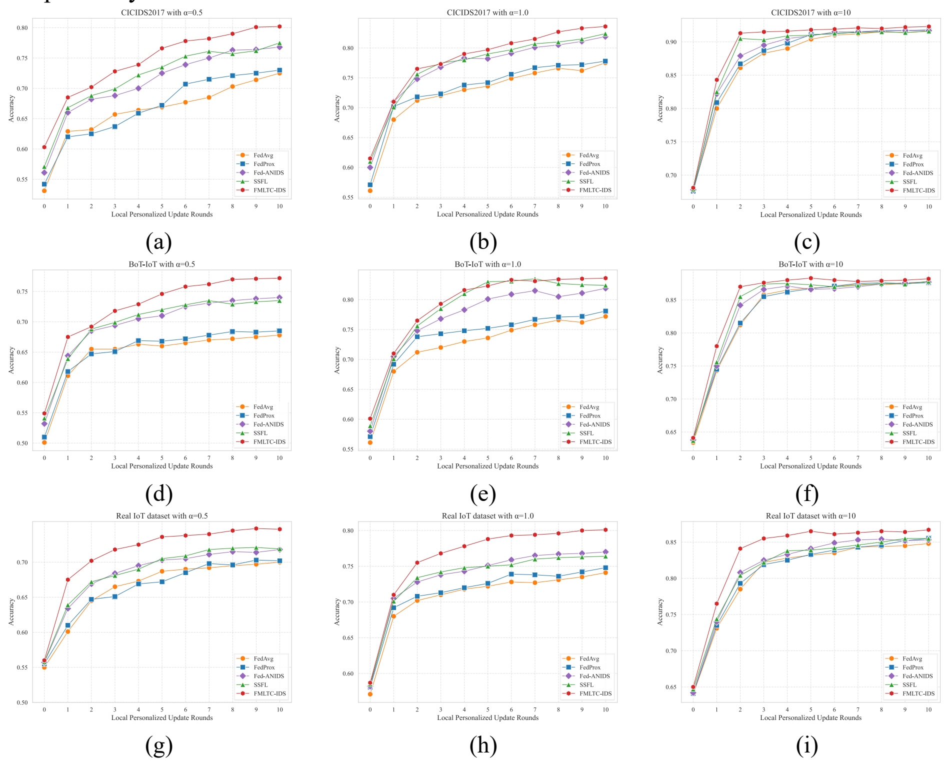

Fig. 3. Accuracy comparison of different models on the CICIDS2017, BoT-IoT, and Real IoT datasets with α values of 0.5, 1.0 and 10.

FMLTC-IDS achieves higher accuracy in high-heterogeneity environments due to two main optimizations: First, the initial parameter optimization based on MAML enhances the generalization ability of the global model, and combined with gradient-weighted sampling, it prioritizes training on data that contributes more to the model. Second, we optimized the dynamic weighted DBA and K-Means clustering method, which integrates time-series features and gradient information to optimize the clustering centers, allowing the model to better adapt to the traffic patterns of each edge node.

When approaching IID (α=10), the performance gap among methods narrows, but FMLTC-IDS still maintains a significant advantage in convergence speed, demonstrating stronger personalized adaptation capabilities. Additionally, before personalized training (i.e., at round 0), the model already achieves an accuracy of 50%-60%, thanks to the introduction of public data during the initial training phase, which allows the model to have high detection capability even before personalization.

Tables 1 and 2 show the experimental results on the BoT-IoT and Real IoT datasets. FMLTC-IDS consistently outperforms in terms of Recall and F1-score, achieving the best overall performance. On the Real IoT dataset, when α=0.5, FMLTC-IDS achieves Accuracy, Recall, and F1-score of 74.7%, 72.5%, and 73.5%, respectively; when α=10, these metrics improve to 86.7%, 85.5%, and 86.1%. Although FMLTC-IDS's Accuracy increases by 2.9% compared to SSFL when α=0.5, the more stringent decision boundary reduces false positives (increasing Precision), which leads to a rise in missed detections and a slight decrease in Recall and F1-score.

The advantage of FMLTC-IDS in high-heterogeneity (low α) scenarios comes from its time-series clustering and gradient-weighted mechanisms, which allow it to better adapt to the traffic patterns of different edge devices. However, when α=10, the data distribution becomes more balanced, diminishing the personalized differences in time-series data, and other methods also achieve high performance, thus narrowing the performance gap.

Table 1. Experimental Results on the BoT-IoT Dataset.

Method	α=0.5			α=1.0			α=10		
	Acc	Rec	F1	Acc	Rec	F1	Acc	Rec	F1
FedAvg	0.678	0.652	0.664	0.771	0.742	0.755	0.878	0.863	0.870
FedProx	0.685	0.661	0.672	0.781	0.753	0.766	0.876	0.858	0.866
Fed-ANIDS	0.740	0.718	0.728	0.819	0.795	0.805	0.877	0.862	0.869
SSFL	0.735	0.712	0.722	0.824	0.802	0.812	0.878	0.864	0.871
FMLTC-IDS	**0.772**	**0.772**	**0.761**	**0.836**	**0.816**	**0.825**	**0.882**	**0.868**	**0.875**

Table 2. Experimental Results on the Real IoT Dataset.

Method	α=0.5			α=1.0			α=10		
	Acc	Rec	F1	Acc	Rec	F1	Acc	Rec	F1
FedAvg	0.700	0.675	0.687	0.741	0.713	0.726	0.848	0.832	0.840
FedProx	0.702	0.680	0.690	0.748	0.723	0.735	0.855	0.842	0.848
Fed-ANIDS	0.719	0.698	0.708	0.770	0.745	0.757	0.853	0.840	0.846
SSFL	0.718	**0.764**	**0.740**	0.764	0.735	0.749	0.855	0.843	0.849
FMLTC-IDS	**0.747**	0.725	0.735	**0.801**	**0.778**	**0.789**	**0.867**	**0.855**	**0.861**

4.3 Ablation Study

In this section, we evaluate the effectiveness of each component design strategy in FMLTC-IDS through ablation experiments, aiming to answer the following questions: RQ1: Does the dynamic batch adjustment strategy impact model training efficiency and convergence speed? RQ2: Is our dimensionality reduction strategy superior to other methods? RQ3: Does time-series clustering help in personalized model aggregation?

4.3.1 Answer to RQ1

To answer RQ1, we designed three comparison models to assess the impact of the dynamic batch adjustment strategy: Baseline1 (Fixed small batch, B=16): Simulates training on resource-constrained devices. Baseline2 (Fixed large batch, B=128): Simulates high-performance devices but lacks dynamic adjustment. Baseline3 (Random dynamic batch, $B_i \in [16,128]$): Randomly selects batch size in each training round.

In FMLTC-IDS, the batch size is dynamically adjusted (as shown in formula (1)) to adapt to the device's computing resources and data size, with the minimum batch size $B_{\min}$ =16 and maximum batch size $B_{\max}$ =128. We conducted experiments on the BoT-IoT dataset with heterogeneity parameter α=1, comparing the training time, convergence speed, detection performance (Accuracy, Recall, F1-score), and convergence time (T_C) of different methods.

The experimental results are shown in Table 3. Baseline1 (B=16) has the longest per-round training time (138.21s) due to the smaller batch size, which requires more parameter updates per round. However, its stable gradient estimation helps accelerate convergence. Baseline2 (B=128) uses a larger batch size, reducing the number of parameter updates and shortening training time, but it may introduce gradient estimation bias, affecting the model's generalization ability. Baseline3 (random dynamic batch) introduces instability by randomly selecting batch sizes, which may impact the consistency of the gradient direction, thereby affecting convergence speed and stability.

In contrast, FMLTC-IDS optimizes the training time per round to 94.36s (a 31.8% reduction compared to Baseline1) through adaptive batch adjustment, requiring only 6 rounds to converge, with total convergence time significantly lower than other methods. Additionally, FMLTC-IDS, combining dynamic batch adjustment with gradient-weighted sampling, achieves a Pareto-optimal balance between gradient quality and computational efficiency, improving training efficiency by 2-3 times and detection accuracy by 2-3%.

Table 3. Training Efficiency and Convergence Speed of Different Models.

Approach	Training Time(s)	Convergence Speed	Acc	Rec	F1	T_C (s)
Baseline1	138.21	12	0.814	0.798	0.803	1658.52
Baseline2	85.13	19	0.801	0.774	0.787	1617.47
Baseline3	102.62	14	0.807	0.781	0.794	1436.68
FMLTC-IDS	94.36	6	0.836	0.816	0.825	566.2

4.3.2 Answer to RQ2

To answer RQ2, we compared our method with UMAP and selected the first layer, the last layer, and all layers of the model to construct different feature extraction schemes. We tested the impact of different feature extraction methods on clustering runtime using the BoT-IoT dataset (heterogeneity parameter α=1). The experimental results are shown in Fig 4.

The results indicate that using all layers for clustering resulted in a computational time of 21.42s, significantly higher than the other methods, demonstrating that high-dimensional data increases computational overhead. In contrast, our method (last layer + PCA) only took 0.36s, reducing the computational time by 98.3% compared to the all-layer method. It was also faster by 0.09s compared to the first layer + PCA method (0.45s), validating the effectiveness of combining PCA with the last layer.

While UMAP, as a nonlinear dimensionality reduction method, can preserve richer local structural information, it has a slightly higher computational cost. For feature dimensionality reduction tasks in large-scale neural networks, PCA provides higher computational efficiency while maintaining model performance, significantly reducing the computational burden.

Our feature extraction method, which effectively combines PCA with the last layer of the neural network, greatly reduces clustering runtime. As the number of clients and the number of neural network layers increase, and data distribution becomes more complex, the time advantage of this method will become even more significant.

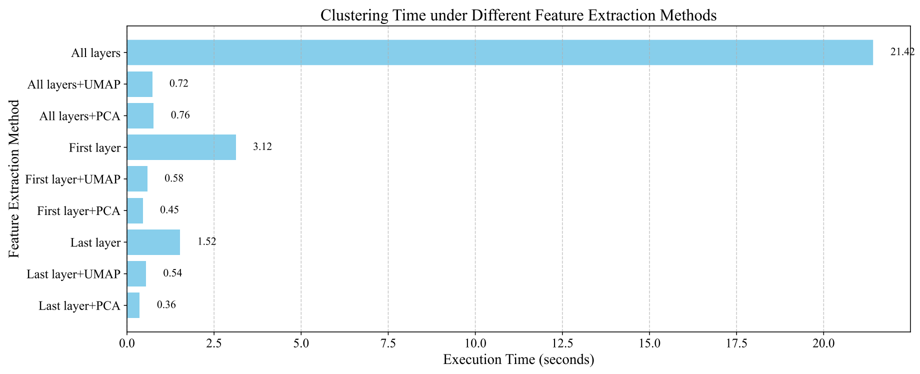

Fig. 4. Clustering Time under Different Feature Extraction Methods.

4.3.3 Answer to RQ3

To answer RQ3, we conduct experiments using the Real IoT dataset with a heterogeneity parameter of α = 1. The experiment included three groups: Experimental group (TS-Cluster): Time series-based clustering. Control group 1 (Static-Cluster): K-Means clustering based on static features. Control group 2 (FedAvg): No clustering, traditional FedAvg for global aggregation. To comprehensively assess the effectiveness of the time series clustering method, we measured classification performance using Accuracy, Recall, and F1-score, and evaluated personalization capability using ΔAcc = Acc_local -

Acc_global (i.e., the accuracy improvement of the local model compared to the global model), as shown in Table 4.

The experimental results show that the TS-Cluster group outperformed the control groups in all classification performance metrics. In terms of Accuracy, TS-Cluster was 6.3% higher than FedAvg and 2.7% higher than Static-Cluster. Recall improved by 6.3% (compared to FedAvg) and 2.7% (compared to Static-Cluster), and F1-score increased by 7.9% and 2.7%, respectively. These results demonstrate that time series clustering more effectively utilizes the time series features of clients, enhancing intrusion detection performance. Furthermore, the ΔAcc of TS-Cluster reached 4.1%, far exceeding that of FedAvg (0.8%) and Static-Cluster (2.3%), indicating that time series clustering better adapts to local data distributions and strengthens personalized detection capabilities.

Table 4. Evaluation of Personalization Capability of Different Models.

Approach	Accuracy	Recall	F1-score	ΔAcc
TS-Cluster	0.801	0.778	0.789	**+4.1**
Static-Cluster	0.774	0.751	0.762	+2.3
FedAvg	0.738	0.715	0.726	+0.8

We also evaluated clustering quality using Silhouette Score and intra-cluster gradient similarity. As shown in Fig 5, TS-Cluster had a Silhouette Score of 0.58, higher than Static-Cluster's 0.45, indicating that time series clustering forms tighter and more separable clusters. TS-Cluster's gradient similarity (0.84) also surpassed Static-Cluster's (0.73), suggesting that clients in the TS-Cluster share more similar update patterns, thereby enhancing personalized optimization.

In summary, the results indicate that TS-Cluster achieves the best performance in clustering quality and personalization capability, further validating the effectiveness of the time-series clustering strategy in optimizing personalized federated learning.

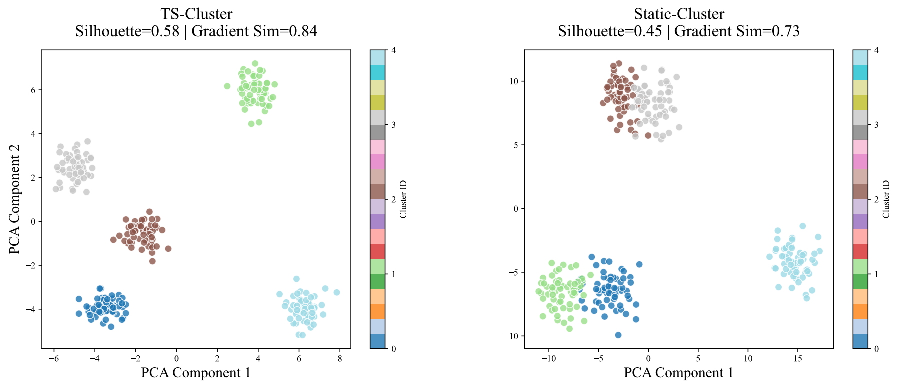

Fig. 5. Comparison of Silhouette Score and Intra-cluster Gradient Similarity between TS-Cluster and Static-Cluster.

5 Conclusion and Outlook

This study proposes a novel IoT intrusion detection framework (FMLTC-IDS), which combines federated meta-learning with adaptive time clustering techniques to enhance intrusion detection's personalization and generalization capabilities while ensuring data privacy. To optimize local training efficiency, we introduced adaptive batch adjustment and gradient-weighted sampling strategies, dynamically adjusting batch sizes and prioritizing samples that contribute significantly to model updates. Additionally, to address the spatial and temporal heterogeneity of IoT data, we proposed a method based on PCA dimensionality reduction and dynamic weighted DBA-K-Means clustering to improve the accuracy and adaptability of model aggregation. Experimental results demonstrate that FMLTC-IDS outperforms traditional methods in both detection accuracy and convergence speed in high-heterogeneity scenarios, offering an efficient solution for security defense in edge computing environments.

However, challenges remain in real-time detection and pattern capturing, especially when facing advanced persistent threats (APT) and other covert, dynamically evolving attack patterns. Future research will focus on enhancing temporal feature representation, optimizing real-time detection, and upgrading defense systems to improve APT detection capabilities, reduce detection latency, and enhance the system's robustness and privacy security against emerging attacks.

Acknowledgments. This research was supported by the fundamental research funds for the central universities (3122025094).

References

1. Song, Yanxing, et al. "Applications of the Internet of Things (IoT) in smart logistics: A comprehensive survey." IEEE Internet of Things Journal 8.6 (2020): 4250-4274.
2. Qin W, Chen S, Peng M. Recent advances in Industrial Internet: insights and challenges[J]. Digital Communications and Networks, 2020, 6(1): 1-13.
3. Rahman A, Hasan K, Kundu D, et al. On the ICN-IoT with federated learning integration of communication: Concepts, security-privacy issues, applications, and future perspectives[J]. Future Generation Computer Systems, 2023, 138: 61-88.
4. Liu, Xiaonan, et al. "Federated learning and meta learning: Approaches, applications, and directions." IEEE Communications Surveys & Tutorials 26.1 (2023): 571-618.
5. Elrawy M F, Awad A I, Hamed H F A. Intrusion detection systems for IoT-based smart environments: a survey[J]. Journal of Cloud Computing, 2018, 7(1): 1-20.
6. Sharafaldin I, Lashkari A H, Ghorbani A A. Toward generating a new intrusion detection dataset and intrusion traffic characterization[J]. ICISSp, 2018, 1(2018): 108-116.
7. Koroniotis N, Moustafa N, Sitnikova E, et al. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset[J]. Future Generation Computer Systems, 2019, 100: 779-796.

8. McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.
9. Li T, Sahu A K, Zaheer M, et al. Federated optimization in heterogeneous networks[J]. Proceedings of Machine learning and systems, 2020, 2: 429-450.
10. Idrissi M J, Alami H, El Mahdaouy A, et al. Fed-anids: Federated learning for anomaly-based network intrusion detection systems[J]. Expert Systems with Applications, 2023, 234: 121000.
11. Zhao, Ruijie, et al. "Semisupervised federated-learning-based intrusion detection method for Internet of Things." IEEE Internet of Things Journal 10.10 (2022): 8645-8657.
12. Mothukuri, Viraaji, et al. "Federated-learning-based anomaly detection for IoT security attacks." IEEE Internet of Things Journal 9.4 (2021): 2545-2554.
13. Tahir, Bushra, Alireza Jolfaei, and Muhammad Tariq. "Experience-driven attack design and federated-learning-based intrusion detection in industry 4.0." IEEE Transactions on Industrial Informatics 18.9 (2021): 6398-6405.
14. Aouedi, Ons, et al. "Federated semisupervised learning for attack detection in industrial internet of things." IEEE Transactions on Industrial Informatics 19.1 (2022): 286-295.
15. Chen Y, Zhang J, Yeo C K. Network anomaly detection using federated deep autoencoding gaussian mixture model[C]//International Conference on Machine Learning for Networking. Cham: Springer International Publishing, 2019: 1-14.
16. Zhao R, Yin Y, Shi Y, et al. Intelligent intrusion detection based on federated learning aided long short-term memory[J]. Physical Communication, 2020, 42: 101157.
17. Lu, Wenting, et al. "Stones from Other Hills: Intrusion Detection in Statistical Heterogeneous IoT by Self-labeled Personalized Federated Learning." IEEE Internet of Things Journal (2025).
18. He, Zaobo, et al. "Privacy-enhanced personalized federated learning with layer-wise gradient shielding on heterogeneous IoT data." IEEE Internet of Things Journal (2024).
19. Deng, Dongshang, et al. "Fedasa: A personalized federated learning with adaptive model aggregation for heterogeneous mobile edge computing." IEEE Transactions on Mobile Computing (2024).
20. Zhang, Guangsheng, et al. "PPFed: A privacy-preserving and personalized federated learning framework." IEEE Internet of Things Journal 11.11 (2024): 19380-19393.
21. Kingma D P, Welling M. Auto-encoding variational bayes[EB/OL].(2013-12-20)