

# Federated Intrusion Detection System with Cost-sensitive Learning for Internet of Things

Qiuzhen Lin, *Member, IEEE*, Wang Liu, Shaifeng Zheng, Junkai Ji, Ka-Chun Wong  
Jianqiang Li, and Carlos A. Coello Coello, *Fellow, IEEE*,

**Abstract**—Network Intrusion Detection System (NIDS) has become more important as a large number of diverse devices connect to the Internet of Things (IoT). Generally, training an effective NIDS requires a large amount of high-quality and centralized attack data. However, in real-world scenarios, it is difficult to centralize the distributed data for training NIDS in the IoT due to the privacy concerns and data format heterogeneity. To solve this problem, a novel NIDS combining federated learning and cost-sensitive learning is proposed, named FIDS-CL. Specifically, federated learning with dynamic weights aggregation tackles the problem of non-clusterable data, where multiple clients collaboratively enhance the overall performance while dynamically aggregating weights to maximize the retention of high-performing client models. Moreover, cost-sensitive learning is employed to alleviate the problem of class imbalance in NIDS by dynamically adjusting the gradient descent weights of different classes in the loss function, thereby emphasizing the importance of minority classes. Therefore, our method can effectively handle data imbalance while safeguarding data privacy of clients, which is more effective to detect network intrusions. The experiments conducted across various scenarios validate the superior detection capabilities and computational efficiency of FIDS-CL when compared to other state-of-the-art NIDSs.

**Index Terms**—network intrusion detection, deep learning, federated learning, cost-sensitive learning.

## I. INTRODUCTION

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62376163; in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076; in part by the Shenzhen Science and Technology Program under Grant JCYJ20220531101411027; and in part by the Shenzhen Natural Science Foundation (the Stable Support Plan Program) under Grant 20231122104038002. (*Corresponding author: Jianqiang Li.*)

Q.Z. Lin, L. Wang, and S.F. Zheng are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China (email of Q.Z. Lin: qiuzhlin@szu.edu.cn).

J.K. Ji and J.Q. Li are with the National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen, China.

K.C. Wong is with Department of Computer Science, City University of Hong Kong, Hong Kong, China.

C.A. Coello Coello is affiliated with the Department of Computer Science, CINVESTAV-IPN (Evolutionary Computation Group), México, D.F.07300, MÉXICO. He is also (as part of a sabbatical leave) a member of the Faculty of Excellence at the School of Engineering and Sciences, Tecnológico Monterrey, Monterrey, N.L., Mexico.

IN today's increasingly interconnected world, the prevalence of cybersecurity threats poses significant challenges, particularly within the context of the Internet of Things (IoT) [4]. Vulnerabilities in underpowered devices, such as sensors and cameras, are frequently exploited by cybercriminals to carry out illicit activities [5], [14], [16]. These actions can result in severe consequences, as cybercriminals can gradually infiltrate some core electronic devices (e.g., smartphones) through compromised underpowered devices, posing significant threats to personal privacy and finances. Since Network Intrusion Detection System (NIDS) can monitor network traffic in real time within networks and systems, potential malicious behaviors can be identified by recognizing suspicious signs and anomalies. After that, NIDS either proactively alerts security personnel or automatically executes immediate and effective responses. As a result, there is an urgent need for a highly effective NIDS that can accurately detect attacks and ensure a secure IoT environment. In general, NIDSs can be divided into signature-based and anomaly-based systems. Recent research efforts have primarily focused on anomaly-based detection due to its capability to identify previously unknown threats [3].

In earlier years, various machine learning methods, such as Random Forests (RF) [53], Support Vector Machines (SVM) [1], and K-Nearest Neighbors (KNN) [55], were widely adopted to design NIDSs. Although these methods could automatically identify unknown threats, they became inadequate for NIDSs due to the increasing complexity of network environments and the growing volume of network traffic data [46]. In recent years, due to the powerful pattern recognition abilities of Deep Learning (DL) methods in handling large-scale data, a large number of DL methods have been used to design NIDSs. Some representative methods include supervised learning models like Convolutional Neural Network (CNN) [17] and Recurrent Neural Network (RNN) [47], as well as unsupervised learning models like AutoEncoder (AE) [32] and Deep Belief Network (DBN) [6], all of which have significantly enhanced the detection capabilities of NIDSs. However, successfully training these methods requires access to a large volume of high-quality centralized training data, whereas in the IoT

environment, the training data and computing resources are inherently distributed. Moreover, privacy concerns, data format heterogeneity, and device heterogeneity (e.g., mobile phones and sensors) present significant barriers to aggregating distributed data for training NIDSs.

To address the challenges posed by data and resource distributions in IoT scenarios, Federated Learning (FL) has emerged as a promising paradigm for enabling collaborative model training across distributed clients while preserving data privacy [28]. It effectively promotes the training outcomes of all parties involved by utilizing local data and computing resources without compromising data confidentiality. In the FL paradigm, clients independently train local models and exchange only essential updates, ensuring that private data remains secure [39]. This distributed framework is well-suited to IoT scenarios, as it not only protects device privacy but also makes full use of distributed resources, which solves the problem of excessive consumption of computing and storage resources by centralized learning. The performance of FedAvg [31] has been improved in [13], [25] to detect network intrusions. However, these FL-based NIDSs fail to address the class imbalance issue in network traffic during training. Class imbalance in network traffic refers to the phenomenon where the number of normal (benign) traffic samples vastly exceeds that of abnormal (malicious) samples during model training. Specifically, in most real-world datasets, normal traffic can account for over 80% of the data, while various types of attacks or anomalies are severely underrepresented [48]. This pronounced skew leads to several adverse effects: DL models tend to be biased towards the majority class, resulting in high overall *Accuracy* but poor *Recall* and *Precision* for minority (attack) classes. Consequently, rare but critical security threats may go undetected, undermining the reliability and effectiveness of FL-based NIDSs in real-world deployment. The inability to accurately identify minority class instances not only reduces the practical value of these systems but also leaves networks vulnerable to sophisticated or emerging attacks.

To address the aforementioned challenges, a federated NIDS with cost-sensitive learning named FIDS-CL is proposed, as cost-sensitive learning [15] can well address the issue of class imbalance in FL by assigning different costs to classification errors, which can significantly improve the detection performance of NIDS by emphasizing the importance of learning from underrepresented attack categories [12]. FIDS-CL ensures data privacy while improving classification performance. Two key components (a loss weight module and a dynamic aggregation weight module) are designed in FIDS-CL, which can effectively improve the robustness and effectiveness of FL in IoT scenarios characterized by imbalanced data

and stringent privacy constraints. The main contributions are summarized as follows.

1) A loss weight module based on cost-sensitive learning is proposed, which calculates the loss weights based on the inverse relationship of the *F1-score* and is then used to adjust the loss function weights of the client model. It effectively adjusts the sensitivity of different categories and improves the detection effect of the model, especially for underrepresented categories.

2) A dynamic aggregation weight module is proposed, which employs weighted *F1-score* to calculate aggregation weights and then guides model aggregation. In this way, the superior detection performance of some clients for minority classes is preserved in the model aggregation process, thereby improving the overall intrusion detection performance.

3) Experimental results demonstrate that FIDS-CL significantly improves detection performance across multi-client scenarios. Specifically, on the UNSW-NB15 dataset [36], FIDS-CL achieves average improvements of 5.06% in recall and 2.48% in *F1-score* when compared to the baseline FL methods.

The remainder of the paper is organized as follows: Section II provides an overview of background knowledge on NIDSs, FL, FL-based NIDSs and cost-sensitive learning, along with a discussion of related work and the motivation for this study. Section III details the proposed methodology. The environment and parameter configurations are described in Section IV, followed by the discussions of the experimental results. Finally, the conclusions and potential directions for future research are outlined in Section V.

## II. RELATED WORK AND MOTIVATION

This section begins with an introduction of the related work regarding NIDSs and the background of FL. Subsequently, recent advancements in FL-based NIDSs are examined, focusing on the limitations inherent in these approaches. At last, an overview of cost-sensitive learning is presented, emphasizing its importance and necessity within the context of NIDS. The challenges associated with applying cost-sensitive learning to FL are also explored, including potential drawbacks and limitations of this integration.

### A. Network Intrusion Detection Systems

NIDS is an essential security tool that monitors and analyzes activities within computer networks or systems with the primary aim of detecting malicious activities, unauthorized access, or violations of security policies [8]. In the context of today's complex network environments, the risk of network attacks and data breaches has significantly increased. NIDS facilitates automated

security monitoring, allowing for the early identification of potential threats and issuing alerts to prevent damage from attacks. Consequently, NIDS plays a vital role in safeguarding sensitive data, which ensures the continuity of services and meets the compliance requirements. Generally, NIDSs can be mainly categorized into signature-based and anomaly-based systems according to their detection methods. Signature-based NIDSs [30] detect intrusions by identifying the characteristics or patterns of known attacks, achieving high detection efficiency for known threats; however, they often struggle to effectively respond to new or variant attacks. In contrast, anomaly-based NIDSs establish a baseline of normal activities for anomaly detection, offering advantages in identifying novel and unknown attacks. Some studies of NIDSs are summarized in Table I.

In anomaly-based NIDS, two types of methods (machine learning [33] and DL) are often used. Some machine learning approaches, such as Decision Trees (DT) [2], SVM [1], and KNN [55], are often adopted to model normal traffic behavior patterns by analyzing large-scale network data. A DT-based intelligent intrusion detection model was presented that efficiently detects cyber-attacks and reduces computational complexity [2]. In [55], an intelligent intrusion detection model integrating KNN within the edge intelligence framework was proposed to detect Denial of Service (DoS) attacks. Although these methods adapt well to changing network environments and provide a balanced trade-off between false positives and false negatives, the quality of the training dataset and the accuracy of feature selection considerably impact their performance. In addition, some DL approaches like CNN and RNN have been widely applied in NIDSs. For instance, in [17], one-dimensional vector data was converted into two-dimensional image data to generate both single-channel and three-channel datasets. Then, CNN was trained for intrusion detection on the transformed data. A DL model combining CNN and RNN was designed for intrusion detection, achieving a high *Accuracy* on the dataset [47].

Moreover, a new research paradigm focuses on designing NIDSs using unsupervised learning methods. In [6], a DBN was used as a feature extractor with a Multi-Layer Perceptron (MLP) as the classification head. The DBN was pre-trained in an unsupervised fashion, resulting in favorable outcomes for intrusion detection. In [32], multiple AEs were employed to form a single-head multi-tail architecture, which was also trained in an unsupervised manner, yielding excellent results on the dataset.

### B. Federated Learning

Federated learning is a paradigm of distributed learning that serves as an exemplary model for designing

distributed NIDSs. It is an emerging machine learning paradigm that aims at protecting data privacy while enabling collaborative model training [52]. By training multiple models in a decentralized manner across multiple parties (such as edge devices or local servers), FL circumvents centralized data management and thereby reduces the risk of data leakage. This characteristic renders it particularly advantageous in some applications having sensitive data. The concept of FL was initially proposed in FedAvg [31], where each client trains a model on its private data and periodically communicates with a server to update the global model. Throughout this process, data never leaves the local environment, offering data privacy protection. However, the limited effectiveness of FedAvg is demonstrated by preliminary experiments [44]. FedApt [50] improves upon FedAvg by addressing the heterogeneity issue among clients through personalized model adjustments, enhancing adaptability and personalized performance while maintaining data privacy. MOON [26] innovatively applies contrastive learning to FL by aligning the outputs of the local model closer to those of the global model while distancing them from the outputs of the previous local model. In FedNTD [24], a forgetting problem in FL was pointed out and addressed by modifying the loss function to balance the retention of existing knowledge with the acquisition of new knowledge.

### C. Intrusion Detection Based on Federated Learning

For distributed NIDSs, FL provides an effective solution to address some challenges related to data scarcity and privacy protection. To tackle these issues, a multi-task neural network was proposed, which can perform network anomaly detection, VPN traffic identification and traffic classification tasks simultaneously [56]. In [34], a FL-based anomaly detection method for IoT networks was reported. This approach employs a decentralized Gated Recurrent Unit (GRU) model to safeguard local data privacy while enhancing global model accuracy through an integration mechanism for aggregating updates. In [25], an actor filtering was utilized to enhance the global model by adjusting aggregate weights based on *Accuracy* of the client model. In [13], a gradient similarity was computed for each client's model update to determine aggregate weights. Moreover, an evaluation network was used to retain the best-performing global model, enabling rollback if its performance degrades after the model's update. Furthermore, FL was also studied to further improve the detection accuracy of NIDS in various IoT environments [41], [43], [54].

### D. Cost Sensitive Learning

Cost-sensitive learning is a pragmatic approach to mitigate the impact of training on imbalanced datasets. It is

TABLE I: Categories of Network Intrusion Detection Systems

Method	Algorithm	Description
ML	DT [2]	Establish an intelligent NIDS model for prioritizing security feature rankings through DT
	SVM [11]	A novel intrusion detection approach combines SVM and naive bayes feature embedding
	KNN [55]	An enhanced intrusion detection model is designed based on an improved KNN
DL	CNN [17]	Convert one-dimensional data into two-dimensional image data, and then use a CNN for intrusion detection
	CNN-RNN [47]	Design an intrusion detection model using a combination of CNN and RNN
	DBN [6]	A new NIDS combines DBN with a MLP classification head
	AE [32]	Form a real-time and unsupervised NIDS using an ensemble of multiple AEs
FL	DAFL [25]	Filter clients and adjust aggregation weights based on <i>Accuracy</i>
	Multi-NIDS [56]	A FL-based NIDS is capable of performing multiple tasks simultaneously
	EEFED [13]	A FL-based NIDS determines aggregation weights by calculating the gradient similarity of client models

a crucial method in machine learning that highlights the different costs associated with various types of errors. In practical applications, merely pursuing prediction accuracy is often insufficient, as different erroneous decisions can lead to significantly varied consequences and costs. For instance, in the medical field, misdiagnosing a sick person as healthy (false negative) can delay treatment and pose severe risks, whereas labeling a healthy person as sick (false positive) incurs a comparatively lower cost.

In neural networks, cost-sensitive learning addresses the issue of error cost differentiation by adjusting the loss function. In [9], the data sampling problem is framed as a random coverage problem, with error cost determined by counting the number of valid samples. Focal Loss [27] enhances cross-entropy loss [29] by incorporating an adjusting factor and a balancing factor, enabling the model to prioritize the minority class and mitigate the effects of class imbalance. In Ratio Loss [49], a gradient monitor on the server estimates the sample distribution by analyzing the model update direction, subsequently determining the loss weights of different classes. This approach can alleviate class imbalance and safeguard data privacy. Other studies have concentrated on leveraging FL to enhance the detection accuracy of NIDS across diverse IoT environments [10], [40].

#### E. Motivation

Although FL has been used in network intrusion detection, which can address difficulties with distributed data aggregation and privacy concerns in IoT, its imbalance classification problem for NIDS is still not well solved. As introduced in the above subsection that cost-sensitive learning can be used to mitigate class imbalance, the introduction of cost-sensitive learning into FL for intrusion detection is still not sufficiently explored in IoT environments. In our opinion, cost-sensitive learning is particularly suitable for NIDS due to the inherently imbalanced nature of network traffic data in real-world

scenarios, where normal packets significantly outnumber abnormal ones. Traditionally, this method adjusts class weights in the loss function to focus on underrepresented or poorly learned classes, thereby enhancing the model's performance across all classes. However, in FL, the direct application of existing cost-sensitive methods is infeasible due to privacy-preserving principles, which prevents the server from accessing the distribution of private datasets across clients. To address this problem, FIDS-CL is proposed in this paper, where the class weights in the loss function are dynamically adjusted based on the model's performance for each class. This design ensures adaptability to data imbalances while strictly preserving data privacy.

### III. SYSTEM DESIGN

In this section, the proposed FIDS-CL is introduced, which dynamically adjusts the aggregation and loss function weights for each local model based on their respective detection performance. Furthermore, to enhance the adaptability of FL algorithms to long-tail datasets, our algorithm prioritizes improving local training to better capture minority classes. During aggregation, some models demonstrating superior performance in minority classes are integrated more effectively, resulting in improved detection outcomes for these classes. Some key notations used in this paper are summarized in Table II.

#### A. Framework of FIDS-CL

The overall framework of FIDS-CL is depicted in Fig.1. First, local clients are trained and tested using their private datasets. Then, the clients upload their parameters and performance metrics to the central server. Upon receiving this information, the central server calculates the loss weights  $w$  and aggregation weights  $\gamma$  using the loss weight module (*LWM*) and the dynamic aggregation weight module (*DAGM*). Finally, the global model is updated and distributed to all clients, along with  $w$ .

TABLE II: List of key notations

Symbol	Description
$N$	Total number of clients
$C$	Total number of classes
$R$	Number of communication round
$D_n$	Local datasets for client $l_n$
$len(D_n)$	The size of $D_n$
$LD_n$	Test dataset for client $l_n$
$\theta_t^g$	Global model parameters in round $t$
$\theta_t^{l_n}$	Local model parameters for clients $l_n$ in round $t$
$w_c$	The weights of class $c$ in loss function
$fs_c^{l_n}$	The $FI$ -score of class $c$ in client $l_n$
$\gamma_t^{l_n}$	The aggregation weights of client $l_n$ in round $t$
$\phi_t^{l_n}$	The weighted $FI$ -score of client $l_n$ in round $t$

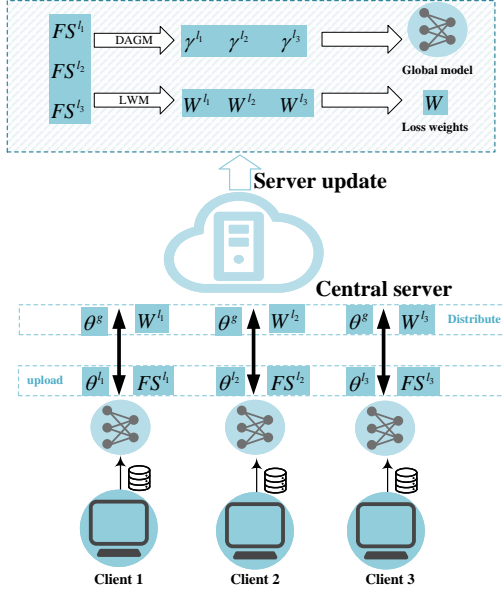


Fig. 1: Framework of the proposed FIDS-CL

The pseudocode for FIDS-CL is presented in Algorithm 1. In the initialization stage, several procedures are performed by the central server, including setting global model parameters and distributing both the global model and relevant experimental parameters to all clients, as presented in lines 1 and 2 of Algorithm 1. Specifically, the central server sets the communication round to  $t = 1$  and initializes the global model parameters  $\theta_0^g$ . Next, the class weights  $w^{l_n}$  is set uniformly across all classes. This initialization scheme reflects the model's initially equal exposure to all classes at the start of training, which aligns with standard gradient descent optimization practices in machine learning. Then, the server distributes  $B, O, E, f, \rho, w^{l_n}$  and  $\theta_0^g$  to all clients. In each communication round, the algorithm executes both the local training process and the global aggregation process, as detailed in lines 4–11 and 12–17, respectively. In the stage of local training process, each client trains and

evaluates its local model using its private dataset while adjusting the loss function weights based on  $w$ , which is computed by the *LWM*. The trained model parameters and corresponding performance metrics are then sent to the central server for further processing. In the stage of global aggregation process, as outlined in lines 14 and 15,  $w$  and  $\gamma$  are computed by the central server using *LWM* and *DAGM*, respectively. At last, in lines 16–17, the global model is updated, and the relevant information is distributed to all clients. In the following subsections, the details of local training process and global aggregation process are introduced, respectively.

---

**Algorithm 1: FIDS-CL Implementation Details**


---

**Input :** The number of selected indices  $a$ ; Batch size  $B$ ; Loss function  $f$ ; Local epoch  $E$ ; Learning rate  $\rho$ ;  $R, D_n, LD_n, C, w$ ;

**Output:** Updated global model parameters  $\theta^g$ ;

- 1 Initialize the communication round  $t = 1$  and global model parameters  $\theta_0^g$ ;
  - 2 The central server distributes  $B, O, E, f, \rho, w$  and  $\theta_0^g$  to all clients;
  - 3 **for**  $t \leq R$
  - 4   **Local Training Process;**
  - 5   **for each client**  $l_n$
  - 6      $\theta_t^{l_n} \leftarrow \text{CIntUpd}(l_n, \theta_{t-1}^g, w^{l_n})$ ;
  - 7      $fs^{l_n} \leftarrow \theta_t^{l_n}(LD_n)$  using Eqs.(8)-(10);
  - 8     **if**  $t = 1$  **then**
  - 9       Upload  $\theta_t^{l_n}, fs^{l_n}$  and  $len(D_n)$  to the server;
  - 10    **else**
  - 11      Upload  $\theta_t^{l_n}$  and  $fs^{l_n}$  to the server;
  - 12   **Global Aggregation Process;**
  - 13   **for server**
  - 14      $w \leftarrow \text{CalLweight}(\theta, fs)$  using Eq. (2);
  - 15      $\gamma \leftarrow \text{CalDagweight}(fs, D_{\text{lengths}}, a)$  in Algorithm 2;
  - 16      $\theta^g = \sum_{i=1}^{l_N} \gamma^{l_i} \theta^{l_i}$ ;
  - 17     Distribute the updated  $\theta_t^g$  and  $w$  to all clients;
  - 18 **Function**  $\text{CIntUpd}(l_n, \theta_{t-1}^g, w^{l_n})$ :
  - 19    $\theta_{t-1}^{l_n} \leftarrow \theta_{t-1}^g$ ;
  - 20   set weights for loss function using  $w^{l_n}$ ;
  - 21   **for local epoch**  $e$  **from** 1 **to**  $E$
  - 22     **for each batch**  $B$  **in**  $D_n$
  - 23        $\theta_t^{l_n} \leftarrow \theta_{t-1}^{l_n} - \rho \nabla f(\theta^{l_n}; B; w^{l_n})$ ;
  - 24   **return**  $\theta^{l_n}$ ;
-

### B. Local Training Process

The main objective of this process is to update local model  $\theta^{l_n}$  and calculate  $fs^{l_n}$ , and then upload them to central server. In the  $t$ -th iteration of local model training, client  $l_n$  initially downloads the global model  $\theta_{t-1}^g$  from the server as its local model  $\theta_{t-1}^{l_n}$ , and also retrieves  $w^{l_n}$  that is used to adjust the sensitivity of the model to each class during training.  $w^{l_n}$  is incorporated into the local model's loss function, influencing the model's learning process by emphasizing certain classes more than others. Client  $l_n$  then performs gradient descent on  $\theta_{t-1}^{l_n}$  using its private dataset  $D_n$ , resulting in an updated model  $\theta_t^{l_n}$ , as shown in line 6. The update is performed according to Eq. (1).

$$\theta_t^{l_n} \leftarrow \theta_{t-1}^{l_n} - \rho \nabla f(\theta^{l_n}; B; w^{l_n}) \quad (1)$$

This formula represents the process of gradient descent for the local model on the training dataset, where  $\rho$  denotes the learning rate, and  $\nabla f()$  is the gradient of the loss function.  $B$  is the batch size during training, and  $w^{l_n}$  refers to the weights of the loss function, controlling the model's gradient descent magnitude for different classes.

Subsequently,  $\theta_t^{l_n}$  is evaluated on the test set  $LD_n$  to determine the *F1-score* for each class:  $fs^{l_n} = \{fs_1^{l_n}, \dots, fs_C^{l_n}\}$ , as presented in line 7. Finally, client  $l_n$  uploads  $\theta_t^{l_n}$ ,  $fs^{l_n}$ , and the size of the dataset  $len(D_n)$  to the central server.

### C. Global Aggregation Process

During the global aggregation phase, the server receives not only the model parameters submitted by the clients but also their evaluation metrics, i.e., the *F1-score* [7] obtained from the test set. The server uses the received model parameters and evaluation metrics to compute the weights  $w$  and  $\gamma$  using *LWM* and *DAGM* for the loss functions and model aggregation. Then, the server aggregates all client models into a single global model  $\theta^g$  based on  $\gamma$ . Finally, the server distributes  $\theta^g$  and  $w$  to all participating clients. Next, the details of *LWM* and *DAGM* are introduced, respectively.

1) *Loss Weight Module*: This module is designed for straightforward implementation. Its formula module is shown in Eq. (2), and the corresponding pseudocode is presented in line 14 of Algorithm 1.

$$w[i, j] = \frac{1}{\sum_{k=1}^C \frac{e^{\beta fs[i, j]}}{e^{\beta fs[i, k]}}} \quad (2)$$

Here,  $i$  refers to the client's ID,  $j$  indicates the class,  $\beta$  is the temperature coefficient that regulates the sensitivity of the *F1-score* to the weights, and  $C$  is the total number of classes. Finally,  $w$  is used in the local training process to adjust the loss function weights.

---

### Algorithm 2: CalDAGWeight

---

**Input** :  $fs, D_{\text{lengths}}, N$ ;  
**Output**:  $\gamma$ ;

- 1 **for** each class  $c$  in classes
- 2     // calculate average *F1-score* of class  $c$
- 3      $\bar{fs}_c \leftarrow \text{Average}(fs[:, c])$ ;
- 4     // store  $\bar{fs}$  into a new array  $\bar{FS}$
- 5      $\bar{FS}[c] \leftarrow \bar{fs}_c$ ;
- 6 // select the indices of the  $a$  smallest values from  $\bar{FS}$
- 7  $x_{\text{indices}} \leftarrow \text{Select\_the\_Lowest\_Indices}(\bar{FS}, a)$ ;
- 8 **for** each client  $l_n$  in clients
- 9      $\bar{fs}^{l_n} \leftarrow \text{Average}(fs[l_n, :])$ ;
- 10     $\dot{fs}^{l_n} \leftarrow \text{Average}(fs[l_n, x_{\text{indices}}])$ ;
- 11    // calculate weighted *F1-score*
- 12     $\phi^{l_n} \leftarrow (1 - \alpha)e^{\bar{fs}^{l_n}} + \alpha e^{\dot{fs}^{l_n}}$ ;
- 13    // calculate dynamic aggregation weights
- 14     $\gamma^{l_n} \leftarrow \frac{len(D[n]) \cdot \phi^{l_n}}{\sum_{i=1}^N len(D[i])}$ ;
- 15 **return**  $\gamma$ ;

---

2) *Dynamic Aggregation Weight Module*: During the global aggregation process, the server receives both local model parameters  $\theta^{l_n}$  and the corresponding *F1-score*  $fs^{l_n}$  from each client  $l_n$ . Upon receiving this information, the central server first computes the weighted *F1-score*  $\phi = \{\phi^{l_1}, \dots, \phi^{l_N}\}$  and subsequently uses  $\phi$  to calculate the aggregation weights  $\gamma = \{\gamma^{l_1}, \dots, \gamma^{l_N}\}$ . Following this, the server aggregates the local model parameters  $\theta = \{\theta^{l_1}, \dots, \theta^{l_N}\}$  using these aggregation weights  $\gamma$  to generate a global model  $\theta^g$ , which is then disseminated back to each client. The above process is covered in lines 15-17 of Algorithm 1. The detailed pseudocode of this module is shown in Algorithm 2.

A detailed and in-depth introduction to the operation of *DAGM* is provided as follows. Specifically, the server computes the average *F1-score*  $\bar{fs}_c$  for each class across all clients using Eq. (3), as shown in lines 2-5 of Algorithm 2. Subsequently, it identifies the indices of the classes with the lowest average *F1-score* in  $\bar{fs} = \{\bar{fs}_1, \dots, \bar{fs}_C\}$  as presented in line 7. Then, it recalculates the average *F1-score*  $\dot{fs}^{l_n}$  for these selected classes and computes the weighted *F1-score*  $\phi^{l_n}$  for each client using Eq. (4) as shown in lines 9-12.

$$\bar{fs}_c = \frac{\sum_{i=1}^N fs_c^i}{N} \quad (3)$$

$$\dot{fs}^{l_n} = \frac{\sum_{i=1}^C fs_i^{l_n}}{C}, \quad \phi^{l_n} = \frac{(1 - \alpha)e^{\bar{fs}^{l_n}} + \alpha e^{\dot{fs}^{l_n}}}{\sum_{i=1}^N \phi^i} \quad (4)$$

This score contributes to the aggregation weights. Next, the aggregation weights for each client are calculated using Eq. (5) as introduced in line 14, and the global model is computed using Eq. (6), which is then distributed to every client.

$$\gamma^{l_n} = \frac{\text{len}(D_N)\phi^{l_n}}{\sum_{i=1}^N \text{len}(D_i)} \quad (5)$$

$$\theta^g = \sum_{i=1}^{l_N} \gamma^{l_i} \theta^{l_i} \quad (6)$$

The aggregation weights  $\gamma^{l_n}$  are computed according to Eq. (5), where  $\phi^{l_n}$  represents the weighted *F1-score* of client  $l_n$ , and  $\text{len}(D_n)$  refers to the size of the local training dataset for client  $l_n$ . Subsequently, the global model parameters  $\theta^g$  are aggregated by applying the weights  $\gamma$ , as formalized in Eq. (6).

#### IV. EXPERIMENTS

In this section, the feasibility of the proposed method is evaluated through comprehensive experiments conducted on three selected datasets. First, a detailed description of the datasets and the experimental environment is provided. Next, the metrics employed for model evaluation are outlined. Finally, the effectiveness of the proposed method is analyzed based on the experimental results, along with additional parameter analysis and ablation studies.

##### A. Description of Datasets

A number of network intrusion detection datasets are available online. For this study, three widely recognized datasets are selected, including CICIDS2017 [45], UNSW-NB15 [36], and Bot-IoT [23].

The CICIDS2017 dataset, developed by the Canadian Institute for Cybersecurity (CIC), serves as a standard to test the performance of NIDSs. It simulates real-world network traffic and attack behaviors. The dataset has 78 network traffic attributes extracted from each network session, including basic features (e.g., flow duration, source IP, destination IP, and transport protocol), content features (e.g., average bytes per packet and flags), time features (e.g., time intervals between packets), and statistical information from the network and transport layers, among others. The CICIDS2017 dataset encompasses one type of normal traffic and 14 different types of attack traffic.

The UNSW-NB15 dataset, created by the Cyber Range Lab at the University of New South Wales (UNSW) in 2015, offers network traffic data that is more contemporary and aligns better with the current network environment compared to the datasets like KDD99 and NSL-KDD [37]. It contains 49 network traffic features, which are divided into three categories: basic features,

content features, and time features [38]. It records one type of normal traffic and nine types of attack traffic.

The Bot-IoT dataset, generated by the Cyber Range Lab at UNSW in 2018, simulates the communication patterns of modern IoT devices within real network environments and incorporate various types of attack traffic [21], [22]. Its purpose is to provide a platform for testing and evaluating NIDSs in the IoT environment [19]. It includes 16 features and five types of attacks, but does not have normal traffic types [20].

##### B. Dataset Analysis and Pre-processing

The distributions of attack categories in the above three datasets are plotted in Fig. 2. As shown in Fig. 2, all three datasets exhibit significant class imbalance, often referred to as long-tail distributions. A small number of classes, referred to as the head, constitute approximately 80% of the dataset, while the majority of classes, known as the long tail, account for a minor proportion. This characteristic indicates that *Accuracy* may not accurately reflect the model's classification performance on such datasets, as a model only needs to classify the head classes while the *Accuracy* may remain high even if the tail classes are not identified due to their low representation.

The dataset preprocessing ensures compatibility with the DL model input format. For the CICIDS2017 dataset, preprocessing includes the following steps: 1) Discarding constant or near-constant columns (i.e., columns with very low variance); 2) Removing columns with high correlation; 3) Eliminating columns with over 60% missing values; 4) Deleting records (rows) with missing values; 5) Applying label encoding to non-numeric features; and 6) Performing min-max scaling, as detailed in Eq. (7).

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (7)$$

where  $x$  represents the original value,  $x_{\min}$  is the minimum value in the column,  $x_{\max}$  is the maximum value in the column, and  $x_{\text{scaled}}$  is the normalized value obtained after scaling. The UNSW-NB15 dataset is characterized by a significant proportion of its features containing missing values [35], [42]. To address this problem, Step 4 in the standard approach is modified by substituting missing values in each record with the mode of the respective column, while all other steps remain consistent with the process used for the CICIDS2017 dataset. The Bot-IoT dataset, which had already undergone preprocessing and partitioning, is utilized directly in this study [18]. Each dataset is divided into training and testing sets, with 80% allocated for training and 20% reserved for testing.

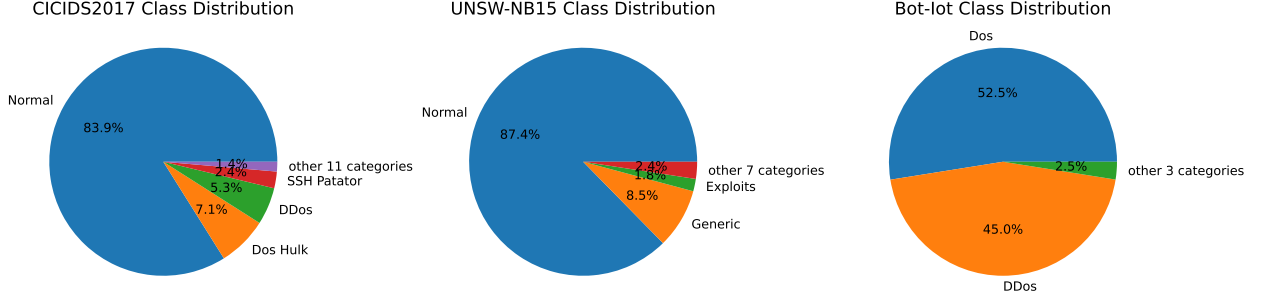


Fig. 2: Distributions of attack categories in the three datasets.

### C. Experimental Settings

The majority of the experiments are conducted on a server equipped with an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.10 GHz processor and 256 GB of RAM, running the Ubuntu 20.04 operating system. Model training and testing are performed on a Nvidia GeForce RTX 4090 graphics card with 24 GB of memory. The parameter setting of the experiments is presented in Table III. The source code of this paper is publicly available at <https://github.com/AzelaicAcid/FIDS-CL>.

TABLE III: Parameter setting of the proposed method

Parameter	Description	Value
$N$	Total number of clients	3,5,7
$R$	Number of communication rounds	50
$E$	Number of local epochs	1
$B$	Batch size	128
$f$	Loss function	Cross Entropy
$O$	Optimization Algorithm	Adam
$\beta$	Temperature of $F1$ -score	1.0
$a$	The number of selected attack types	3
$\alpha$	The importance of selected types	0.8
$\rho$	Learning rate	0.005

### D. Model Evaluation Metrics

1) *Detection Performance*: In this study, the classification evaluation metrics of *Precision*, *Recall*, and *F1-score* are employed. The rationale for excluding the commonly used *Accuracy* will be discussed later, as *Accuracy* is not a reliable measure of classification performance in the context of extremely imbalanced data. The definitions of these three evaluation metrics are provided in Eq. (8), Eq. (9), Eq. (10) and Eq. (11):

$$Precision = \frac{TP}{TP + FP} \quad (8)$$

$$Recall = \frac{TP}{TP + FN} \quad (9)$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (10)$$

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

2) *Time Cost*: FL differs from traditional centralized learning due to a distinctive communication time cost, which involves the duration required for transferring data between the server and clients during each communication round. To evaluate computational efficiency, the average time consumption per round for each algorithm is measured using the formula: *Time Per Round (TPR)* = *Model Training Time* + *Model Evaluation Time*. This metric offers a fair assessment of time expenditure, enabling a more holistic evaluation of each algorithm's performance by incorporating both detection and time efficiency. The experiments were executed on an RTX 4090 graphics card with 24 GB of memory, utilizing the UNSW-NB15 dataset across 20 communication rounds to determine *TPR* for the algorithms.

### E. Experimental Results

In this section, the comparison experiments on three datasets are first introduced. Then, the time consumption experiment is analyzed. Finally, hyperparameter analysis and ablation studies are presented, along with a metric analysis experiment. The algorithms compared in this study include five FL-based algorithms (DAFL [25], FedPAC [51], FedAvg [31], MOON [26], and FedNTD [24]) and Local. Here, Local denotes a scenario where clients are trained and evaluated on their own datasets without communication, serving as a baseline for experimental results. Tables IV, V, and VI present the performance of the compared algorithms in terms of *Precision*, *Recall*, and *F1-score* across three different scenarios and datasets. All metrics in Tables IV, V, and VI represent their average values from all client models in the best round and the best results in these tables are marked in **bold**.

1) *Performance Comparison*: Examining the experimental results over all three datasets, FL-based algorithms generally outperformed the Local algorithm, except for MOON [26] and FedNTD [24] on the Bot-IoT dataset. FedAvg [31] showed relatively stable performance in multi-client setups; however, its *Precision*

and *F1-score* experienced slight declines with an increasing number of clients. This indicates limitations in managing unevenly distributed data. Remarkably, FedNTD [24] maintained its *F1-score* effectively, even with more clients, underscoring its superior ability to preserve model quality. Across the datasets, FIDS-CL consistently exhibited outstanding performance, irrespective of client variations. Despite not always achieving the highest *Precision*, its *F1-score* remained exemplary, indicating its ability to balance *Precision* and *Recall* effectively.

**CICIDS2017.** Table IV illustrates the comparative results on the CICIDS2017 dataset. The Local algorithm, which does not employ a federated strategy, consistently performs the worst, irrespective of the number of clients. In the scenario with 3 clients, FIDS-CL outperforms others in all three metrics (*Precision*, *Recall*, and *F1-score*), achieving their values of 0.8852, 0.8458, and 0.8651, respectively. Moreover, our *Recall* and *F1-score* respectively reached 0.8660 and 0.8635 in the scenario with 5 clients, and 0.8462 and 0.8559 in the scenario with 7 clients, significantly surpassing other algorithms. The *Precision* in these scenarios also ranked at an intermediate level, exceeding more than half of the compared algorithms.

**UNSW-NB15.** Table V shows the comprehensive experimental results of FIDS-CL and other compared algorithms on the UNSW-NB15 dataset. In the scenarios with 3 and 7 clients, MOON [26] achieved the highest *Precision* of 0.7562 and 0.7310, respectively. Meanwhile, FedNTD [24] recorded the highest *Precision* in the 5-client scenario with a value of 0.7222. Despite these *Precision* achievements, FIDS-CL excelled in *Recall* across all three scenarios, with scores of 0.5920, 0.5853, and 0.5746, outperforming all other algorithms. Furthermore, FIDS-CL also led in *F1-score* measurements in every scenario, with scores of 0.6532, 0.6246, and 0.6194, respectively.

**Bot-IoT.** As observed from Table VI, each algorithm has notably enhanced performance on the Bot-IoT dataset compared to the UNSW-NB15 dataset, as the Bot-IoT dataset only includes abnormal traffic. It is noteworthy that both MOON [26] and FedNTD [24] exhibited inferior performance, as their *F1-score* were unable to surpass those of the Local algorithm across all scenarios. This suggests a counterproductive impact when incorrect FL strategies are employed. In contrast, FIDS-CL demonstrated superior performance, achieving the best results across almost all metrics. It only slightly underperformed DAFL [25] with the *Precision* score of 0.9956 in the 5-client scenario, yet excelled across other metrics in all scenarios.

2) *Time Consumption:* In the time consumption experiments, identical hardware configurations were maintained across all algorithms to ensure experimental fair-

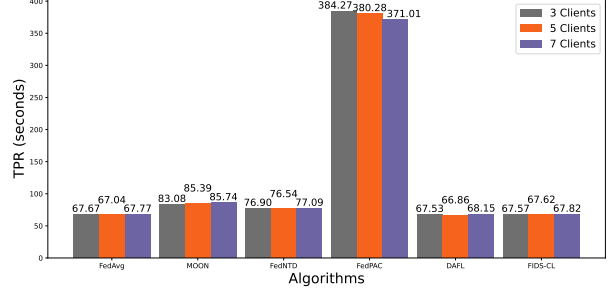


Fig. 3: The *TPR* of various algorithms under different numbers of clients.

ness. The *TPR* of each FL-based algorithm is depicted in Fig. 3. Notably, the FedPAC [51] algorithm yielded a *TPR* approximately four to five times higher than the others. This significant outcome is primarily due to the unique approach of FedPAC [51], in which each model consists of an encoder and a classification head that are trained alternately and independently. Furthermore, when determining the final global classification head, each client reviews its local dataset to compute specific statistics, which is then used to establish the global classification head in the server. These additional local and global computational processes significantly increase the *TPR*.

Following FedPAC [51], the MOON [26] algorithm exhibits an increased *TPR* due to its incorporation of a contrastive learning process, resulting in higher computation time. Similarly, FedNTD [24] demonstrates an elevated *TPR* because it uses a distillation method. The *TPRs* of DAFL [25] and FIDS-CL are similar to that of FedAvg [31], underscoring the efficacy of FIDS-CL. Thus, it is reasonable to conclude that FIDS-CL enhances detection performance without substantially increasing computational burden.

3) *Communication Complexity:* In FL, the network load for parameter exchanges (uploads and downloads) between the server and clients is a critical performance metric. The communication cost of FL is directly determined by the size of this network load. Optimizing the upload and download overhead can significantly reduce communication latency, thereby enhancing FL efficiency. Furthermore, such optimization enables the system to support a larger client population while simultaneously minimizing computational and bandwidth resource consumption.

In a 5-client FL scenario, this study evaluates the per-communication-round network load (upload/download) of multiple algorithms, where all clients employ identical DL models as their local models. The experimental results are presented in Table VII. Notably, FedNTD [24], MOON [26], and FedAvg [31] exhibit identical

TABLE IV: The results of comparative experiments on the CICIDS2017 dataset

method	3 clients			5 clients			7 clients		
	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>
Local	0.8689	0.8271	0.8475	0.8581	0.8084	0.8325	0.8406	0.8004	0.8200
FedAvg [31]	0.8762	0.8354	0.8553	0.8684	0.8331	0.8504	0.8444	0.8226	0.8333
MOON [26]	0.8831	0.8316	0.8565	0.8654	0.8372	0.8511	0.8419	0.8316	0.8367
FedNTD [24]	0.8708	0.8270	0.8483	<b>0.8697</b>	0.8396	0.8544	0.8540	0.8368	0.8453
FedPAC [51]	0.8724	0.8288	0.8500	0.8666	0.8388	0.8525	0.8421	0.8189	0.8303
DAFL [25]	0.8762	0.8354	0.8553	0.8669	0.8311	0.8486	<b>0.8718</b>	0.8181	0.8441
FIDS-CL	<b>0.8852</b>	<b>0.8458</b>	<b>0.8651</b>	0.8610	<b>0.8660</b>	<b>0.8635</b>	0.8658	<b>0.8462</b>	<b>0.8559</b>

TABLE V: The results of comparative experiments on the UNSW-NB15 dataset

method	3 clients			5 clients			7 clients		
	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>
Local	0.6820	0.5230	0.5920	0.6526	0.5144	0.5752	0.6242	0.5068	0.5593
FedAvg [31]	0.7434	0.5458	0.6292	0.6850	0.5198	0.5910	0.6892	0.5172	0.5909
MOON [26]	<b>0.7562</b>	0.5422	0.6315	0.7144	0.5322	0.6099	<b>0.7310</b>	0.5261	0.6119
FedNTD [24]	0.7532	0.5402	0.6292	<b>0.7222</b>	0.5345	0.6143	0.7237	0.5267	0.6096
FedPAC [51]	0.7354	0.5814	0.6493	0.7071	0.5513	0.6196	0.6770	0.5430	0.6026
DAFL [25]	0.7240	0.5462	0.6226	0.6992	0.5280	0.6016	0.6986	0.5216	0.5972
FIDS-CL	0.7284	<b>0.5920</b>	<b>0.6532</b>	0.6696	<b>0.5853</b>	<b>0.6246</b>	0.6718	<b>0.5746</b>	<b>0.6194</b>

TABLE VI: The results of comparative experiments on the Bot-IoT dataset

method	3 clients			5 clients			7 clients		
	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>
Local	0.9878	0.9740	0.9809	0.9892	0.9696	0.9793	0.9814	0.9544	0.9677
FedAvg [31]	0.9963	0.9727	0.9844	0.9910	0.9664	0.9785	0.9925	0.9650	0.9786
MOON [26]	0.9750	0.9764	0.9757	0.9488	0.9684	0.9585	0.9564	0.9618	0.9591
FedNTD [24]	0.9611	0.9665	0.9638	0.9572	0.9608	0.9590	0.9584	0.9670	0.9626
FedPAC [51]	0.9926	0.9768	0.9846	0.9828	0.9768	0.9798	0.9825	0.9645	0.9734
DAFL [25]	0.9961	0.9749	0.9854	<b>0.9956</b>	0.9650	0.9800	0.9871	0.9616	0.9742
FIDS-CL	<b>0.9983</b>	<b>0.9830</b>	<b>0.9906</b>	0.9943	<b>0.9784</b>	<b>0.9863</b>	<b>0.9884</b>	<b>0.9788</b>	<b>0.9836</b>

network loads, as their communication is strictly limited to the exchange of model parameters ( $\theta^{l_n}$ ) between the server and clients. In contrast, FedPAC [51] introduces additional overhead: clients upload local prototypes and download global prototypes, with the total network load comprising both prototype transfers and model parameter transmissions. DAFL [25] demonstrates dynamic client-server participation due to its server-side client filtering mechanism based on an *Accuracy* threshold. Consequently, its network load fluctuates across rounds and is reported as an averaged value. For FIDS-CL, beyond standard model parameter exchanges, clients transmit the current round's *F1-score* and receive loss weights ( $w$ ). However, these auxiliary data are negligible in size (order of bytes), contributing minimally to the

overall communication overhead. In summary, FIDS-CL significantly improves the model's detection capability without adding virtually any communication overhead.

TABLE VII: Communication complexity

method	theory	practice
FedAvg [31]	$\sum_{n=1}^N \theta^{l_n} \times 2$	200.68MB
MOON [26]	$\sum_{n=1}^N \theta^{l_n} \times 2$	200.68MB
FedNTD [24]	$\sum_{n=1}^N \theta^{l_n} \times 2$	200.68MB
FedPAC [51]	$\sum_{n=1}^N (\theta^{l_n} + C \times d) \times 2$	200.98MB
DAFL [25]	$\sum_{n=1}^{\lfloor N \times 0.7 \rfloor} \theta^{l_n} \times 2$	140.48MB
FIDS-CL	$\sum_{n=1}^N 2 \times \theta^{l_n} + w + F$	200.68MB

4) *Hyperparameters Analysis*: In this section, the impact of the hyperparameter  $r$  on the performance of FIDS-CL is investigated, where  $r$  refers to the number of communication rounds in which both the *LWM* and *DAGM* are simultaneously activated. Fig. 4 shows the variation in the average *F1-score* across all clients when both *LWM* and *DAGM* are activated at communication round  $r$ . In Fig. 4, the activation rounds of *LWM* always align with those of *DAGM*, gradually increasing with a step size of 5, as depicted by the blue line. The red line represents the detection performance of FIDS-CL when both proposed modules are consistently disabled. The experimental data analysis reveals the following findings: first, regardless of the specific values of  $r$ , as long as both modules are active, the performance of FIDS-CL surpasses the scenario where the modules are inactive. Second, FIDS-CL achieves the best performance, attaining an *F1-score* of 0.8705, when  $r = 25$ . Third, activating *LWM* and *DAGM* early in training to assist the process does not show significant benefits when  $0 \leq r < 25$ . This may be due to the fact that, during the early stages of training, the client models have not yet converged, and their detection capabilities are still unstable. As a result, enabling *LWM* at this stage cannot effectively adjust the classification error costs. Similarly, since *DAGM* computes aggregation weights based on the model's detection performance, prematurely enabling *DAGM* results in imprecise aggregation weights. Finally, when  $r > 25$ , the detection performance of FIDS-CL remains at a moderate level. This may be because *LWM* and *DAGM* did not have enough rounds to fully exert their effects, resulting in only a slight improvement in the performance of FIDS-CL.

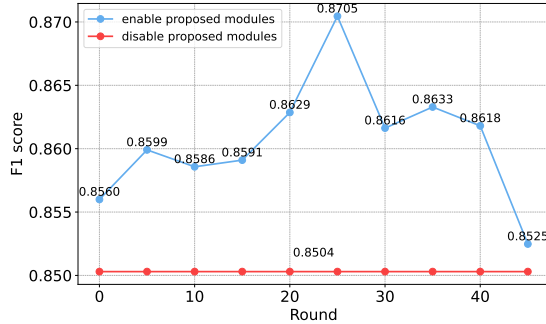


Fig. 4: Detection performance of FIDS-CL with varying activation rounds for *LWM* and *DAGM*.

5) *Ablation Study*: The ablation study examined in detail the effect of disabling a module on *F1-score*. Table VIII displays the outcomes. FedAvg refers to the case where both *LWM* and *DAGM* are disabled, while *w/o LWM* indicates the scenario with *LWM* disabled,

and *w/o DAGM* represents the case with *DAGM* disabled. The experimental results show that both modules significantly improve detection performance. Notably, enabling both *LWM* and *DAGM* simultaneously provides the greatest performance boost, especially on the UNSW-NB15 dataset, where the *F1-score* increases by around 3%.

TABLE VIII: *F1-score* of ablation study with FIDS-CL. The statistically best results are marked in **bold**

Dataset	FedAvg	<i>w/o LWM</i>	<i>w/o DAGM</i>	FIDS-CL
CICIDS2017	0.8504	0.8531	0.8590	<b>0.8635</b>
UNSW-NB15	0.5910	0.6093	0.6115	<b>0.6246</b>
Bot-IoT	0.9785	0.9812	0.9794	<b>0.9863</b>

6) *Metrics Analysis*: In this subsection, the rationale for selecting the *F1-score* as a central evaluation metric is discussed. While *Accuracy* is a commonly used and intuitive measure for assessing classification performance, it can be misleading when applied to highly imbalanced datasets, such as network traffic datasets. Table IX presents the detection performance of the Local algorithm after 50 training epochs on the CICIDS2017 dataset in a 5-client scenario, focusing on one specific client. As observed from this, it is evident that the model performs nearly perfectly on the majority of classes (e.g., Benign, DDoS, and DoS), which collectively account for over 90% of the CICIDS2017 dataset. As a result, the model's overall *Accuracy* is expected to be very high. However, the model's performance on minority classes, such as Infiltration and Web Attack XSS, is notably poor. This highlights the phenomenon that high *Accuracy* may result from a model's ability to classify a predominantly large class effectively.

Fig. 5 shows the variations of *Accuracy* and *F1-score* in 50 training epochs of the Local algorithm on the CICIDS2017 dataset in a 5-client scenario. Compared to the *F1-score*, the inflated *Accuracy* is depicted, which may lead to a misjudgment of the model's detection performance. Conversely, the *F1-score* is a comprehensive metric combining *Precision* and *Recall*. *Precision* reflects the model's conservatism, indicating its tendency to avoid false positives, while *Recall* indicates its aggressiveness, or its tendency to capture all positive instances. When both *Precision* and *Recall* are elevated, the *F1-score*, as defined in Eq. (10), will be accordingly increased, indicating that the model employs a well-balanced classification strategy, which can accurately identify and classify positive instances without errors. Therefore, in evaluating network traffic datasets, the *F1-score* is more suitable than *Accuracy*. Consequently, this paper endorses the *F1-score* as the principal evaluation metric and fundamental element of our algorithm.

TABLE IX: Detection performance of the local algorithm on the CICIDS2017 dataset.

Class	Precision	Recall	F1-score
Benign	0.9993	0.9991	0.9991
Bot	0.8393	0.6026	0.7015
DDoS	0.9990	0.9996	0.9990
DoS GoldenEye	0.9926	0.9830	0.9877
DoS Hulk	0.9985	0.9972	0.9978
DoS Slowhttptest	0.9761	0.9855	0.9807
DoS Slowloris	0.9906	0.9953	0.9929
FTP Patator	0.9955	1.0000	0.9977
Heartbleed	1.0000	1.0000	1.0000
Infiltration	0	0	0
PortScan	0.9807	0.9974	0.9889
SSH Patator	0.9683	0.9919	0.9799
Web Attack Brute Force	0.6951	0.9828	0.8142
Web Attack Sql Injection	0	0	0
Web Attack XSS	0.5000	0.0385	0.0715

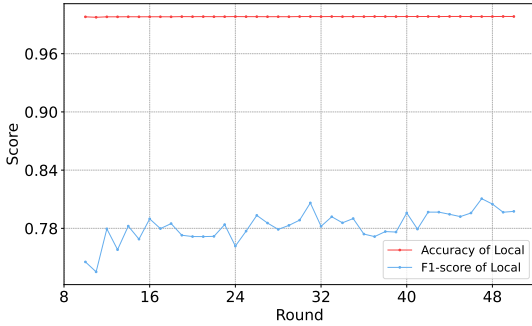


Fig. 5: Comparison of *Accuracy* and *F1-score* on the CICIDS2017 dataset.

## V. CONCLUSION

In this work, a novel NIDS (FIDS-CL) is proposed, which incorporates *LWM* and *DAGM* to innovatively combine federated learning and cost-sensitive learning, trying to tackle the challenges of distributed data and class imbalance when running intrusion detection in IoT environments. Specifically, *LWM* dynamically adjusts the sensitivity of the loss function for each local model, while *DAGM* assigns aggregation weights based on the *F1-score*. Their integration significantly enhances the model's detection performance, particularly with notable improvements in *Recall* and *F1-score*. The detection performance of FIDS-CL is outstanding, achieving the highest *F1-score* compared to other state-of-the-art NIDSs in three datasets. Furthermore, the deployment of these modules only introduces a few computational overhead, making them practical and well-suited for real-world applications.

Although FIDS-CL demonstrates superior detection performance compared to state-of-the-art approaches, several limitations remain. Most notably, the current framework does not account for the inherent hetero-

geneity of IoT environments, particularly in terms of hardware disparities across client devices. For instance, computational capabilities vary significantly among clients—while some can efficiently execute complex deep learning models for inference, others lack the necessary resources to support such computations. This hardware heterogeneity may impact the practical deployment and scalability of FIDS-CL in real-world IoT ecosystems. Future work will explore the usage of heterogeneous FL algorithms in NIDSs, where the clients possess varying hardware capabilities, resulting in some differences in the parameter scales of local models. Our goal is to improve the detection performance of NIDSs across diverse environments, thereby accommodating varied computational resources while enhancing the robustness and adaptability of FL in intrusion detection.

## REFERENCES

- [1] Laith Abualigah, Saba Hussein Ahmed, Mohammad H Almomani, Raed Abu Zitar, Anas Ratib Alsoud, Belal Abuhaija, Essam Said Hanandeh, Heming Jia, Daa Salama Abd Elminaam, and Mohamed Abd Elaziz. Modified aquila optimizer feature selection approach and support vector machine classifier for intrusion detection system. *Multimedia Tools and Applications*, pages 1–27, 2024.
- [2] Mohammad Al-Omari, Majdi Rawashdeh, Fadi Qutaishat, Mohammad Alshira'H, and Nedal Ababneh. An intelligent tree-based intrusion detection model for cyber security. *Journal of Network and Systems Management*, 29(2):20, 2021.
- [3] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z Emam. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189:105124, 2020.
- [4] Mohammed Almehdhar, Abdullatif Albaseer, Muhammad Asif Khan, Mohamed Abdallah, Hamid Menouar, Saif Al-Kuwari, and Ala Al-Fuqaha. Deep learning in the fast lane: A survey on advanced intrusion detection systems for intelligent vehicle networks. *IEEE Open Journal of Vehicular Technology*, 2024.
- [5] Fatima Alwahedi, Alyazia Aldhaheri, Mohamed Amine Ferrag, Ammar Battah, and Norbert Tihanyi. Machine learning techniques for iot security: Current research and future vision with generative ai and large language models. *Internet of Things and Cyber-Physical Systems*, 2024.
- [6] Othmane Belarbi, Aftab Khan, Pietro Carnelli, and Theodoros Spyridopoulos. An intrusion detection system based on deep belief networks. In *International Conference on Science of Cyber Security*, pages 377–392. Springer, 2022.
- [7] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21:1–13, 2020.
- [8] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2020.
- [9] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.
- [10] Othmane Friha, Mohamed Amine Ferrag, Lei Shu, Leandros Maglaras, Kim-Kwang Raymond Choo, and Mehdi Nafaa. Fedlids: Federated learning-based intrusion detection system for agricultural internet of things. *Journal of Parallel and Distributed Computing*, 165:17–31, 2022.
- [11] Jie Gu and Shan Lu. An effective intrusion detection approach using svm with naïve bayes feature embedding. *Computers & Security*, 103:102158, 2021.

- [12] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications*, 73:220–239, 2017.
- [13] Xianting Huang, Jing Liu, Yingxu Lai, Beifeng Mao, and Hongshuo Lyu. Eefed: Personalized federated learning of execution&evaluation dual network for cps intrusion detection. *IEEE Transactions on Information Forensics and Security*, 18:41–56, 2022.
- [14] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M. Hadi Amini. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24, 2022.
- [15] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587, 2017.
- [16] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.
- [17] Jiyeon Kim, Jiwon Kim, Hyunjung Kim, Minsun Shim, and Eunjung Choi. Cnn-based network intrusion detection against denial-of-service attacks. *Electronics*, 9(6):916, 2020.
- [18] Nickolaos Koroniotis. *Designing an effective network forensic framework for the investigation of botnets in the Internet of Things*. PhD thesis, UNSW Sydney, 2020.
- [19] Nickolaos Koroniotis and Nour Moustafa. Enhancing network forensics with particle swarm and deep learning: The particle deep framework. *arXiv preprint arXiv:2005.00722*, 2020.
- [20] Nickolaos Koroniotis, Nour Moustafa, Francesco Schiliro, Praveen Gauravaram, and Helge Janicke. A holistic review of cybersecurity and reliability perspectives in smart airports. *IEEE Access*, 8:209802–209834, 2020.
- [21] Nickolaos Koroniotis, Nour Moustafa, and Elena Sitnikova. A new network forensic framework based on deep learning for internet of things networks: A particle deep framework. *Future Generation Computer Systems*, 110:91–106, 2020.
- [22] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Jill Slay. Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques. In *Mobile Networks and Management: 9th International Conference, MONAMI 2017, Melbourne, Australia, December 13-15, 2017, Proceedings 9*, pages 30–44. Springer, 2018.
- [23] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796, 2019.
- [24] Gihun Lee, Minchan Jeong, Yongjin Shin, Sangmin Bae, and Se-Young Yun. Preservation of the global knowledge by not-true distillation in federated learning. *Advances in Neural Information Processing Systems*, 35:38461–38474, 2022.
- [25] Jianbin Li, Xin Tong, Jinwei Liu, and Long Cheng. An efficient federated learning system for network intrusion detection. *IEEE Systems Journal*, 17(2):2455–2464, 2023.
- [26] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.
- [27] T Lin. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.
- [28] Zili Lu, Heng Pan, Yueyue Dai, Xueming Si, and Yan Zhang. Federated learning with non-iid data: A survey. *IEEE Internet of Things Journal*, 11(11):19188–19209, 2024.
- [29] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International conference on Machine learning*, pages 23803–23828. PMLR, 2023.
- [30] Mohammad Masdari and Hemn Khezri. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Applied Soft Computing*, 92:106301, 2020.
- [31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [32] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018.
- [33] Preeti Mishra, Vijay Varadharajan, Uday Tupakula, and Emmanuel S Pilli. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE communications surveys & tutorials*, 21(1):686–728, 2018.
- [34] Virraji Mothukuri, Prachi Khare, Reza M. Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. Federated-learning-based anomaly detection for iot security attacks. *IEEE Internet of Things Journal*, 9(4):2545–2554, 2022.
- [35] Nour Moustafa, Gideon Creech, and Jill Slay. Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models. *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*, pages 127–156, 2017.
- [36] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.
- [37] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal: A Global Perspective*, 25(1-3):18–31, 2016.
- [38] Nour Moustafa, Jill Slay, and Gideon Creech. Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. *IEEE Transactions on Big Data*, 5(4):481–494, 2017.
- [39] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.
- [40] Babatunde Olanrewaju-George and Bernardi Pranggono. Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models. *Cyber Security and Applications*, 3:100068, 2025.
- [41] Souradip Roy, Juan Li, and Yan Bai. Federated learning-based intrusion detection system for iot environments with locally adapted model. In *2023 IEEE 10th International Conference on Cyber Security and Cloud Computing (CSCloud)/2023 IEEE 9th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 203–209. IEEE, 2023.
- [42] Mohanad Sarhan, Siamak Layeghy, Nour Moustafa, and Marius Portmann. Netflow datasets for machine learning-based network intrusion detection systems. In *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10*, pages 117–135. Springer, 2021.
- [43] Raed Abdel Sater and A Ben Hamza. A federated learning approach to anomaly detection in smart buildings. *ACM Transactions on Internet of Things*, 2(4):1–23, 2021.
- [44] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.
- [45] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.
- [46] Shivani Singh, Razia Sulthana, Tanvi Shewale, Vinay Chamola, Abderrahim Benslimane, and Biplob Sikdar. Machine-learning-assisted security and privacy provisioning for edge computing: A survey. *IEEE Internet of Things Journal*, 9(1):236–260, 2022.
- [47] Jay Sinha and M Manollas. Efficient deep cnn-bilstm model for network intrusion detection. In *Proceedings of the 2020 3rd*

- International Conference on Artificial Intelligence and Pattern Recognition*, pages 223–231, 2020.
- [48] Ankit Thakkar and Ritika Lohiya. A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167:636–645, 2020.
  - [49] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Addressing class imbalance in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10165–10173, 2021.
  - [50] Hongda Wu and Ping Wang. Fast-convergent federated learning with adaptive weighting. *IEEE Transactions on Cognitive Communications and Networking*, 7(4):1078–1088, 2021.
  - [51] Jian Xu, Xinyi Tong, and Shao-Lun Huang. Personalized federated learning with feature alignment and classifier collaboration. *arXiv preprint arXiv:2306.11867*, 2023.
  - [52] Liangqi Yuan, Ziran Wang, Lichao Sun, Philip S. Yu, and Christopher G. Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 11(21):34617–34638, 2024.
  - [53] Jiong Zhang, Mohammad Zulkernine, and Anwar Haque. Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5):649–659, 2008.
  - [54] Tuo Zhang, Chaoyang He, Tianhao Ma, Lei Gao, Mark Ma, and Salman Avestimehr. Federated learning for internet of things. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 413–419, 2021.
  - [55] Fanyi Zhao, Mingxuan Zhang, Shiji Zhou, and Qi Lou. Detection of network security traffic anomalies based on machine learning knn method. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, 1(1):209–218, 2024.
  - [56] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, and Shui Yu. Multi-task network anomaly detection using federated learning. In *Proceedings of the 10th international symposium on information and communication technology*, pages 273–279, 2019.