An Efficient Federated Learning System for Network Intrusion Detection

Jianbin Li, Xin Tong, Jinwei Liu, *Member, IEEE*, and Long Cheng, *Senior Member, IEEE*

***Abstract*—Network intrusion detection is used to detect unauthorized activities on a digital network, with which the cybersecurity teams of organizations can then kick-start prevention protocols to protect the security of their networks and data. In real-life scenarios, due to the lack of high-quality attack instance data, building an in-depth network intrusion detection system (NIDS) is always challenging for a single enterprise, in terms of handling complex network security threats. To remedy the problem, this article proposes an efficient intrusion detection system called dynamic weighted aggregation federated learning (DAFL) based on federated learning. Specifically, DAFL has used the full advantages of federated learning for data privacy preservation. Moreover, compared to a conventional federated-learning based intrusion detection system, our scheme has implemented dynamic filtering and weighting strategies for local models. In this way, DAFL can perform better in detecting network intrusions with less communication overhead. We give the detailed designs of DAFL, and our experimental results demonstrate that DAFL can achieve excellent detection performance with a low network communication overhead, with data privacy preserved.**

***Index Terms*—Data privacy, deep learning, dynamic weighted aggregation, federated learning, network intrusion detection.**

I. Introduction

NOWADAYS, the external openness of networks makes cyber sercurity threats quite common in our daily life. Specifically, criminals can use loopholes frequently in a network to carry out illegal activities on a system, like modifying system permissions and tampering with system data [1]. These activities may lead to personal financial loss, privacy leakage, system paralysis, and severe threats to national interests. In such scenarios, establishing a network intrusion detection system (NIDS) that can effectively detect attacks to maintain a secure network environment becomes more and more desirable.

In recent years, building detection models for NIDS using machine learning or deep learning algorithms has become a mainstream [2]. Although all the relevant studies have shown that they can achieve significant performance improvement over conventional approaches [3], most of their models are established based on sufficient attack instance data. However, in real scenarios, the network system of an organization or enterprise only generates less high-quality attack sample data for training. Based on these data, the deep intrusion detection model has limited detection ability [4]. An effective way to address the problem is to collect data samples from different enterprises to a central server and train a deep learning model on that basis. Regardless, the fact is that the organizations are generally unwilling to share their network traffic data with others, because their data usually contain sensitive information and involves internal privacy [5].

Federated learning (FL) is a distributed framework proposed by the Google research team in 2016, which can achieve data expansion and privacy protection at the same time [6]. It treats companies, factories, or edge devices with data as clients participating in federated learning, and each client's data is private. Clients build the same deep local models and use their private datasets for training. Establish a global depth model with the same structure as the local model on the cloud center server. They transfer the global and local models through continuous communication between the central server and multiple clients participating in the training. Finally, a global depth model with excellent performance is jointly established to achieve specific learning tasks [7].

Different from centralized learning (CL) [8], FL transmits the deep model in the communication process rather than the local network traffic data of each client. Therefore, FL can indirectly expand the data and avoid the leakage of the original data. In addition, in the process of federated learning, some malicious or poorly performing local models will affect the global model, causing fluctuations in the accuracy of the global model, making quickly convergence is difficult [9]. Considering that, this article introduces an intrusion detection system called DAFL, i.e., *D*ynamic weighted *A*ggregation *F*ederated *L*earning, which can weight aggregate local models dynamically. Specifically, dynamic aggregation means that in the model aggregation stage, DAFL filters local models with poor performance through a threshold, calculates the aggregation weights of the local models based on the detection performance, and final weighted aggregates them dynamically. This scheme allows the training to focus more on local models with better performance and reduces the communication overhead between the client and the server.

In general, the main contributions of our work are summarized as follows:

Manuscript received 7 February 2022; revised 23 August 2022 and 2 December 2022; accepted 9 January 2023. Date of publication 30 January 2023; date of current version 8 June 2023. This work was supported by the Ministry of Science and Technology Key R&D Program under Grant 2020YFB1005804. *(Corresponding author: Long Cheng.)*

Jianbin Li, Xin Tong, and Long Cheng are with the School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China (e-mail: lijb87@ncepu.edu.cn; tongxin@ncepu.edu.cn; lcheng@ncepu.edu.cn).

Jinwei Liu is with the Department of Computer and Information Sciences, Florida A&M University, Tallahassee, FL 32307 USA (e-mail: jinwei.liu@famu.edu).

Digital Object Identifier 10.1109/JSYST.2023.3236995

TABLE I
LIST OF STATE-OF-THE-ART TECHNIQUES FOR FEDERATED LEARNING

Ref.	Method	Description	Desired goals		
			Clients-selection	Dynamic-Weighting	Less-overhead
[10]	Fedavg	Using the weighted average method for the sample size of each participant.	-	-	-
[11]	CMFL	Calculate the symbolic similarity ratio for local and global updates to select clients, weighting the sample size.	✓	-	✓
[12]	ASTW_FedAVG	Calculate and assign weights for local models trained and updated at different times.	-	✓	✓
[13]	Fedadp	Calculate the global and local similarities using cosine similarity and quantify to obtain the local weights.	-	✓	✓
[14]	MT-DNN-FL	Design a multi-task network. Each client performs multiple tasks simultaneously, weighted using the Fedavg algorithm.	-	-	-
[15]	DeepFed	Weighted using Fedavg algorithm. design a secure communication protocol to ensure the security of parameter transmission.	-	-	-
[16]	Segmented-FL	Design multiple global models, group similar local models into one global model to reduce the interaction with the central server.	✓	-	-
[17]	FL-LSTM	Use LSTM as a deep model, the Fedavg algorithm for weighting.	-	-	-
[18]	FedACNN	Calculate the distances of the local and global models using Euclidean and weight them based on the distances.	-	✓	✓
		Our scheme	✓	✓	✓

1) We propose an efficient federated-learning based NIDS DAFL. Considering the limited attack instances of a single organization, our system can effectively expand the sample instances and efficiently construct a high-performance deep detection model to deal with complex network environments.
2) We introduce a client filtering and local model weighting strategy in DAFL, which can reduce the impact of poorly performing local models on the global model during the training process. In the meantime, the approach can also reduce communication overhead and improve the global model performance on intrusion detection.
3) We present the detailed system design and key implementation of DAFL. Moreover, we have implemented a prototype of our system and made the source code publicly.[1]
4) We conduct an experimental evaluation of DAFL and compare its performance with existing approaches. Experimental results show that our proposed scheme can achieve better detection performance with lower communication overhead.

The remainder of this article is structured as follows. In Section II, we review the latest research on network intrusion detection and federated learning. In Section III, we introduce the designed deep model, federated learning intrusion detection system, and dynamic aggregation federated learning intrusion detection system. We report the results of our evaluation in Section IV. Finally, Section V concludes this article.

II. RELATED WORKS

In this section, we review some advanced research on network intrusion detection and federated learning. Specifically, some latest works have been presented in Table I.

A. Network Intrusion Detection

In recent years, many researchers in network security have designed different detection algorithms and proposed their solutions for network intrusion detection. For example, based on traditional machine learning algorithms, Teng et al. [19] proposed an improved genetic algorithm to optimize intrusion detection models based on support vector machines. It also designed a fitness function based on classification accuracy, false alarm rate, and data feature dimensions. Ren et al. [20] proposed a weighted naive Bayes intrusion detection model based on particle swarms, combining rough set theory, and improved particle swarm algorithm to improve detection capabilities. Liu et al. [21] used binary logistic regression statistical tools to detect attacks. This method obtained key data from the routing layer and analyzed sensor behavior.

Based on deep learning algorithms, Nathan et al. [22] designed a deep learning classification model combined with the NDAES and RF classification algorithms to perform network intrusion detection and achieved high accuracy at the base dataset. Dong et al. [23] proposed a network intrusion detection model based on multivariate correlation analysis-long short-time memory network. This model used the MCA algorithm for feature extraction and LSTM for intrusion classification. Liao et al. [24] used generated adversarial network to solve the imbalance of attack samples in intrusion detection. This method modified the binary classification model of the GAN and realized the multiclassification of attacks to improve the detection accuracy. Liu et al. [25] developed a NIDS based on adaptive composition oversampling technology and LightGBM. This system solved the problem of unbalanced network intrusion data and reduced the time complexity of the system.

B. Federated Learning

FL is a distributed machine learning technique that enables multiple participants to learn their data features without sharing data, ultimately building a common, powerful machine learning model. Thus, it can provide an effective solution to the problems of data privacy, data security, and data scarcity. In recent years, FL has been widely used in many industries such as telecommunications, the Internet of Things and national defence.

[1]https://github.com/tongxin-tx/DAFLNIDS

In contrast to traditional distributed learning, the local data faced by federated learning is nonindependent and identically distributed (non-IID). These local data often vary significantly in size and distribution, which may lead to the generation of partially under performing local models during the training process. Usually, these models affect global model convergence and performance during the aggregation phase. For this issue, many scholars have carried out some advanced research. McMahan et al. [10] proposed the Fedavg algorithm, which weighted the local model according to each client's sample size, reducing the communication overhead and improving the training efficiency. Wang et al. [11] proposed an improvement idea and designed the CMFL scheme. This scheme calculates the proportion of model parameters with the same sign between the global and local models and sets the threshold for client screening. Chen et al. [12] proposed a time-weighted aggregation strategy ASTW_FedAVG from the perspective of participation in training time. It effectively integrates information from previously trained local models and improves convergence and accuracy. Wu et al. [13] also improved Fedavg and proposed the Fedadp algorithm. This algorithm calculates the angle between the local and global gradient updates. Then, it quantifies the angle through the Gompertz function to obtain the aggregate weight of each local model. This method can significantly reduce the number of communication rounds but increases the computational load of the server and the client.

C. Intrusion Detection Based on Federated Learning

The network intrusion detection model based on federated learning can learn the characteristics of multiparty intrusion data, prevent the model trained by the participants from reaching the local optimum, and improve the detection ability of the model. In addition, it also protects data privacy, expands limited intrusion data resources, and improves the generalization ability of intrusion models.

At present, there are some research on the application of federated learning in intrusion detection. For example, Zhao et al. [14] proposed a multitask federated learning model MT-DNN-FL to achieve VPN traffic recognition, abnormal traffic detection, and traffic classification tasks. Li et al. [15] described a federated deep learning model based on industrial cyber-physical systems to detect attacks. They also designed secure communication protocols to protect the safety transfer of model parameters. In the same year, Sun et al. [16] proposed an adaptive intrusion detection system based on piecewise federated learning. This system could train multiple similar networks under the same global model and allow multiple participants to share multiple global models. Zhao et al. [17] also proposed an intelligent intrusion detection aided long short-term memory model based on federated learning in the same year. Compared with the traditional model, this model had higher precision, and better consistency. Man et al. [18] designed FedACNN and applied it to network intrusion detection. This method performs weighted aggregation of local models by calculating the euclidean distance between the global and local models in the previous round.

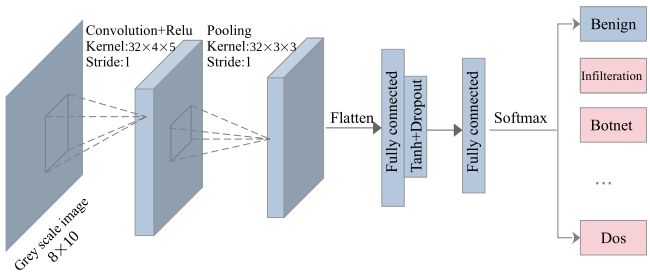

Fig. 1. Architecture of designed CNN model.

Compared to Fedavg, the number of communication rounds is reduced by 50%.

Different from the above work, our research considers the detection capabilities of each local model and its impact on the global model. The designed client filtering enables the federated learning model to select high-quality clients continuously. The designed dynamic weighted aggregation enables the global model to constantly adjust the weight of the local model following their detection results. As a result, the global model can achieve higher detection performance at a faster speed and shorter communication rounds and reduce the communication overhead between clients and the central server.

III. System Design

In this section, we mainly introduces the main structures and algorithms of the three system models we designed.

A. Convolutional Neural Network

Convolutional neural network (CNN) is a feedforward neural network with a deep structure. It has vital characterization functions that can accurately extract features [26]. Therefore, we use CNN as a local deep learning model for intrusion detection to extract traffic data features rapidly and identify attacks accurately. The overall architecture is shown in Fig. 1. The CNN designed in this article consists of a convolutional layer, a pooling layer, and two fully connected layers. After data preprocessing, we convert the one-dimensional network traffic data into two-dimensional grayscale images as the input of CNN.

Convolutional Layer. The convolution kernel slide along horizontal and vertical axes in the spatial dimension of the feature map, then successively calculate the output of each position convolution [see (1)]. The grayscale image features are extracted through such a convolution operation. To fully extract features, we set the stride to 1. Because the ReLu function will make the output of some neurons be 0, which can reduce the dependence between parameters and avoid overfitting, we have used ReLu as the activation function in the convolutional layer, as presented in (2).

$$x_j^l = f_{\text{activation}}\left(\sum_{i \in M_j} x_j^{l-1} \times w_{ij}^l + b_j^l\right) \tag{1}$$

$$\mathrm{ReLU}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2}$$

where x_j^l represents the output of the lth layer, M_j represents the jth feature map of the input feature, x_j^{l-1} represents the feature map of the $(l-1)$th layer, w_{ij}^l represents the convolution kernel of the lth layer, b_j^l represents the bias of the lth layer, and $f_{\text{activation}}$ is the activation function.

Pooling layer. This article uses max-pooling to reduce the feature dimension and the feature vectors of the convolution layer output, aiming to prevent model overfitting.

Fully connected layer. In the first fully connected layer, the activation function tanh [see (3)] nonlinearly processes the data to enhance the model fitting and characterization ability. Dropout randomly discards some neurons to prevent overfitting. In the second fully connected layer, the softmax function maps the calculation results to the $0-1$ interval for intrusion classification prediction, finally outputs the predicted classification result [see (4)].

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3}$$

$$\mathrm{softmax}(x) = \frac{e^x}{\sum_{t=1}^{T} e^{x_t}} \tag{4}$$

where T represents the total number of categories.

Model training. The model training carries out locally. Since the CNN model performs the multiclassification task to detect T categories of attacks, we use the multiclassification cross-entropy function as the loss function [see (5)]. The optimizer used in this article is Nesterov-accelerated Adaptive Moment Estimation (Nadam). It is a gradient descent algorithm proposed by Timothy Dozat's improved Adam algorithm. Based on Adam, Nesterov momentum is added, which can more effectively accelerate convergence and suppress oscillations [27]. Back-propagation learning is performed during the entire model training process through the Nadam optimizer and multiclassification cross-entropy loss function. Then, the parameters of neural network layer are updated and optimized until the loss function converges.

$$f_{\text{loss}} = -\sum_{t=1}^{T} y_t \log(\hat{y}_t) \tag{5}$$

where y_t represents the true label corresponding to the tth category, $\hat{y^t}$ represents the predicted label corresponding to the tth category.

B. NIDS Based on Federated Learning

FL-NIDS is a NIDS based on CNN and the federated learning algorithm Fedavg [28]. The local depth model uses the CNN designed above (see the previous section for details). The entire framework is shown in Fig. 2, and the algorithm description is shown in Algorithm 1. This system mainly includes three entities: central server, clients, and communication network. The specific description is as follows:

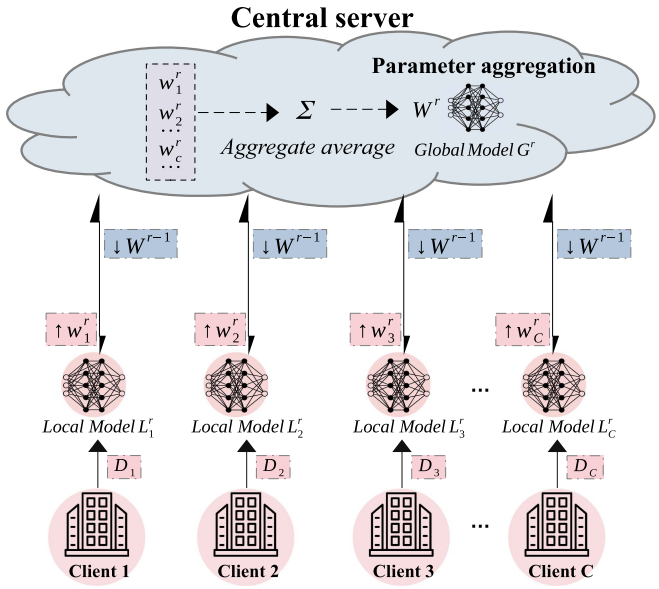

Fig. 2. Architecture of designed FL-NIDS.

Central server. The primary function of the central server is to aggregate locally learned intrusion model parameters and establish a global model. It has usually deployed in the cloud. In order to obtain a better and more comprehensive model, the central server demands interaction with clients in multiple rounds.

Client. Each client represents the corporate agent participating in the joint model training. It is responsible for building a deep local model and training the local model with the network traffic data it collects. By regularly interacting with the central server, they help update the parameters of the global intrusion detection model.

Communication network. It is mainly accountable for the parameters transferring between the central server and clients.

In the entire system model, the basic workflow of a round of federated learning is described as follows:

1) *Initialization*: The central server builds a global intrusion detection model G^0 and initializes the model parameters W^0. Clients download G^0 from the central server and create the local intrusion detection model L^0. It is worth noting that downloading G^0 and creating L^0 are only performed in the first round of communication.
2) *Local model training and upload*: Client c initiates the $ClientUpdate$ procedure and updates the local model L_c^{r-1} with the G^{r-1}'s model parameters W^{r-1}. Train L_c^{r-1} through local data D_c to get L_c^r. Upload the local model parameters w_c^r to the central server.
3) *Local models aggregation*: The central server aggregates the uploaded parameters (e.g., w_1^r, w_2^r,..., w_c^r,...) to obtain the model parameters W^r of global model G^r in the $r-th$ round. Then, it sends W^r to each client participating in federated learning

$$W^r = \sum_{c=1}^{C} \frac{N_c}{N} \times w_c^r \tag{6}$$

Algorithm 1: The Implementation Details of Fedavg Algorithm.

Input: Number of communication rounds R, the batch size B, number of clients C, number of epochs E, the proportion of the number of clients K, client sample resource D_c, the learning rate ρ, the loss function f.

Output: Deep learning model.

For Server:
Initialize W^0;
for $r \leq R$ **do**
  $m \leftarrow max(K \times C, 1)$;
  $S^r \leftarrow (setofmclients)$;
  for *each client* $c \in S^r$ **do**
    $w_c^r \leftarrow ClientUpdate(c, W^{r-1})$;
    Calculate W^r following Eq. 6;
  end
end
For Client:
ClientUpdate(c, W)**:**
$B_c \rightarrow$ (Split the local client data D_c into batches of size B) ;
for *each local epoch* e *from* 1 *to* E **do**
  for *batch* $b \in B_c$ **do**
    $w = w - \rho \nabla f(w; b)$;
  end
end
return w to the central server.

where r is the round index, c is the client index, C ($C \in N_+$) is the total number of clients, N ($N \in N_+$) is the total number of client samples participating in the training, N_c ($N_c \in N_+$) is the sample number of the client c.

The system model finally obtains a global model with an excellent detection effect through R ($R \in N_+$) rounds of federated learning.

C. NIDS Based on Dynamic Weighted Aggregation Federated Learning

During federated learning, the performance of the global model will be interfered with by some poor detection performance or malicious models. As a result, it cannot converge quickly and achieve better detection results. Therefore, to reduce the communication time and the adverse effects of some local models on the global model, we improve the Fedavg algorithm and the FL-NIDS. Specifically, according to the detection accuracy of the local model in each round of training, we introduce local model filtering and dynamic weighted aggregation, leading to DAFL-NIDS, the overall system architecture of which is shown in Fig. 3. There, in each round of communication, we will perform client selection, dynamic weight calculation and refresh operations. This kind of setting allows the model to reduce the influence of poorly performing local models in time. In detail, Algorithm 2 presents the complete workflow of the system, which is described as follows:

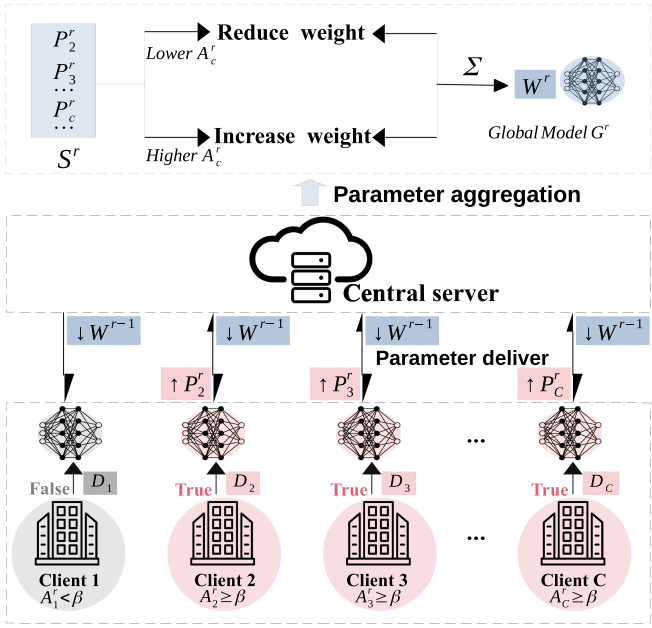

Fig. 3. Architecture of the proposed DAFL-NIDS.

1) *Initialization*: The central server sends W^0 (the initial parameters of the global model), ρ (learning rate), B (batch size), f (loss function), O (optimizer), R (the total number of communication rounds), β (detection accuracy threshold), and E (epoch size) to all clients participating in the training. Then, clients set their communication status to $True$, allowing them to upload parameters to the central server.
2) *Local model update and training*: After clients receive the global model parameters W^{r-1} of the previous round, they start the $ClientUpdate$ procedure. Specifically, clients replace the local model parameters w_c^{r-1} update to W^{r-1}. Then, they use their local data resource D_c to train the local CNN model. After one epoch training, clients get the parameters w_c^r of this round.
3) *Filter and upload local model parameters*: At this stage, we mainly filter out some local models that are useless or even disadvantageous to the global model through the threshold β. Specifically, in each round of training, the client first calculates the detection accuracy A_c^r of the local intrusion detection model. If $A_c^r < \beta$ exists in the local model, set their client communication status to False. These clients suspend communication with the central server and do not upload model parameters. While $A_c^r \geq \beta$, the parameter set $P_c^r = \{N_c, w_c^r, A_c^r\}$ is packaged and uploaded to the central server. Then, the clients end the $ClientUpdate$ process. About β, we reference the floating range of the detection accuracy rate on the test set after multiple local models are trained with minority rounds, and the average range is selected as the β value. If β is too low, the model filtering effect is not obvious, and if β is too high, good local models will be filtered out. Since the training is performed locally during the β value setting

Algorithm 2: The Proposed Dynamic Weighted Aggregated Federated Learning Algorithm.

Input: R, β, C, D_c, B, f, E, ρ, O.
Output: Deep learning model.
Initial:
(1).Communication round $r = 1$, the global model parameters W^0, clients status are $True$;
(2).The central server distributes B, O, E, β, f, ρ, and W^0 to all clients;
For Server:
for $r \leq R$ **do**
  for $c \leq C$ **do**
    Set the client status to $True$;
    $P_r^c = ClientUpdate(c, W^{r-1})$;
    if *c status is not* $False$ **then**
      Add P_r^c to array S^r;
  end
  for $c \in S^r$ **do**
    Calculate the client sample contribution ratio following Eq. 7;
    Calculate the aggregation weight ratio following Eq. 8;
  end
  Calculate global model parameters following Eq. 9.
end
For Client:
ClientUpdate(c, W)**:**
Update client parameters: w$\leftarrow W$;
$B_c \rightarrow$ (Split the local client data D_c into batches of size B);
for *each local epoch e from* 1 *to* E **do**
  for *batch* $b \in B_c$ **do**
    $w = w - \rho \nabla f(w; b)$;
  end
end
Calculate local model accuracy following Eq. 10;
if $A \geq \beta$ **then**
  return $P = (w, A, N)$ to the central server;
else
  Set the client status to $False$.

process, the central server and the client only transmit β, and the communication overhead is negligible.

4) *Dynamic weighted aggregation of model parameters*: The central server receives the parameters sent by the client whose status is $True$ and composes P_c^r into an array S^r. For the clients in S^r, we perform the following weighted aggregation operation: first, the central server calculates the average detection accuracy rate $\hat{A}^r$. Next, it calculates the contribution ratio μ_c^r according to the sample size [see (7)]. Then, weight λ_c^r is calculated according to the detection accuracy of each client [see (8)]. Next, we combine the contribution ratio, aggregation weight, and model parameters to calculate different aggregation parameters $\hat{w_i^r}$, $\hat{w_j^r}$. After merging them, the global model parameters W^r for round r [see (9)] are calculated. Finally, the central server sends W^r to all clients. After receiving W^r, all clients change their communication status to $True$.

$$\mu_c^r = \frac{N_c^r}{\sum_{c \in S^r} N_c^r} \tag{7}$$

$$\lambda_c^r = \frac{e^{A_c^r}}{\sum_{c \in S^r} e^{A_c^r}} \tag{8}$$

$$W^r = \sum_{c \in S^r} \frac{\mu_c^r \times \lambda_c^r}{\sum_{c \in S^r} \mu_c^r \times \lambda_c^r} \times w_c^r \tag{9}$$

5) *Stop learning*: After training R times between steps 2, 3, and 4, we can get the final deep model G^R.

TABLE II
EXPERIMENTAL PARAMETERS

Parameter	Description	Value
C	Initial total number of clients	$3, 5, 7$
R	Communication round	20
β	Detection accuracy threshold	0.75
B	Batch size	512
f	Loss function	Cross Entropy
O	Optimization Algorithm	NAdam
E	Number of epochs	1
ρ	Learning rate	0.002
s_c^r	Uploaded parameters file size	6.23MB

IV. EXPERIMENTAL EVALUATION

This section conducts experiments on the designed scheme to evaluate its feasibility. First, we give the experiment settings, including the data resource descriptions and environment settings. Second, we describe the indicators used to evaluate the model system. Finally, the superiority and effectiveness of the proposed method are verified through the comparison and analysis of different experimental results.

A. Experimental Settings

1) Experimental Environment: In the experiment, the programming language we use is Python, and the TensorFlow high-level interface Keras is used to complete the design of the entire model. The specific experimental model parameters are shown in Table II.

2) Dataset: Similar to other works (e.g., [29], [30], [31]), we have chosen the CSE-CIC-IDS2018 as the test dataset in our experiments. Specifically, the dataset was developed by the communications security establishment (CSE) and the Canadian Institute for Cybersecurity Research for the purpose of intrusion detection research. Compared to some old school datasets (e.g., KDD99), IDS2018 includes some latest common attacks. In detail, it records network traffic data for ten days from February 14 to March 2 in the year 2018 with a series of attacks generated by actual traffic characteristics, including distributed denial of service (DDoS), denial of service (Dos), brute force, XSS, SQL injection, botnet, infiltration, and many other classic attack types.

IDS2018 is preprocessed to contain 79 features, mainly describing the protocol packet size distribution, length of the data stream, time, number of bytes, the data stream payload size, etc. In this article, our experiments focus on detecting DDos, Dos, infiltration, botnet, SQL injection, and brute force attacks. Moreover, we split this dataset into training and test sets at random. And further, randomly divide the train set again as multiple client sample data for experiments.

B. Model Evaluation

1) Detection Performance: In this work, we have used the commonly used indicators $Accuracy$, $Precision$, $Recall$, and $F1-score$ for the evaluation of the proposed system, which are defined as (10)–(14) respectively.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{10}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{11}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{12}$$

$$F1-\text{Score} = \frac{2\times \text{Precision}\times \text{Recall}}{\text{Precision}+\text{Recall}} \tag{13}$$

2) Communication Overhead: In federated learning, constant communication between the server and the remote client is often required to exchange model update information. A large number of clients can easily cause a vast bandwidth burden on the communication network. Reducing communication overhead has become a significant problem to be solved in the further development and application of federated learning [11]. This article uses the number of communications, communications time, and the total number of bits S_{bit} of transmitted data [see (14)] as the leading indicators for evaluating the communication overhead of the system model. Because the proposed system model does not improve the global model parameters delivery stage, this article only calculates the total number of bits in the stage of uploading local model parameters.

$$S_{\text{bit}} = \sum_{r=1}^{R}\sum_{c=1}^{C} s_c^r \tag{14}$$

where s_c^r represents the size of the parameters file uploaded by the cth client in the rth round.

C. Experimental Results

In this article, we conduct experiments on the centralized scheme. After that, we experiment with the designed FL-NIDS based on the traditional Fedavg algorithm under different scenarios (i.e., $C=3$, $C=5$, $C=7$). Finally, the DAFL scheme and some advanced federated learning schemes are compared, including the some latest works, like Wu's [13] Wang's [11], and Man's [18].

In the federated learning experiment, we set up some clients whose data show the non-IID compared with other clients (e.g., containing only a single attack category) to simulate different data in different clients. We randomly divide the data into a training set and a test set, which account for 70% and 30%, respectively. For the purpose of training, we further randomly split the training set into C parts, in which there are C_2 parts with the non-IID data. To better validate the effectiveness of our algorithm and avoid the excessive effect of non-IID data on the global model during the training, we set $C_2 < \frac{1}{2}\cdot C$. Specifically, we set $C=3$, $C_2=1$; $C=5$, $C_2=2$; $C=7$, $C_2=3$ in our experiments.

TABLE III
CL-NIDS DETECTION CAPABILITY VALUE

	Accuracy	Precision	Recall	F1-score
Benign	0.9457	0.9100	0.9816	0.9444
Infilteration	0.9476	0.6273	0.1745	0.2731
Botnet	0.9997	0.9952	0.9992	0.9972
DoS	0.9993	0.9912	0.9995	0.9953
SQL Injection	0.9988	0.0165	0.7143	0.0323
Brute Force	0.9992	0.9837	0.9944	0.9890
DDos	0.9985	0.9953	0.9997	0.9975

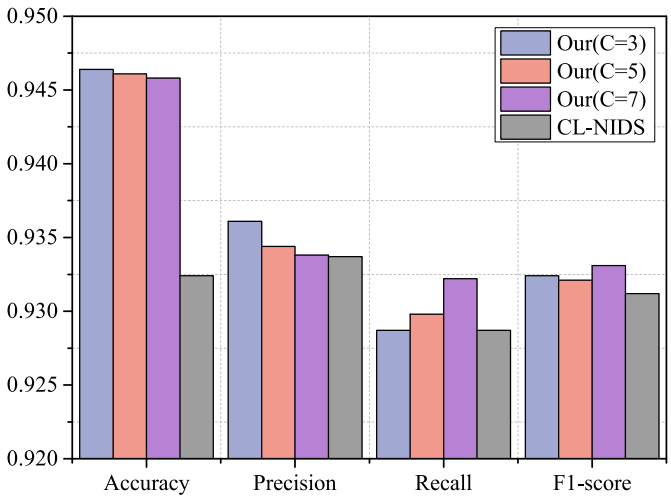

Fig. 4. Performance comparison of our scheme and centralized scheme.

1) CL: This experiment uses the entire training set for training to simulate a centralized scenario. We perform 20 rounds of training and learning, followed by model performance tests using the test set, and the results are shown in Table III. It describes the detection capabilities of the scheme for various types of attacks. We can see the centralized system has satisfactory detection effects on normal traffic, botnets, Dos, brute force, and DDoS. Its accuracy, precision, recall, and F1 score can reach more than 95% and even 99%. The detection performance for SQL injection and infiltration attacks is unsatisfactory because the amount of data in the dataset is minimal. The overall detection performance of CL-NIDS is shown in Fig. 4, with accuracy, precision, recall, and F1 scores of 94.44%, 93.37%, 92.87%, and 93.12%. Although CL-NIDS has satisfactory detection performance overall, because its essentially simulated scenario is a combination of data samples from multiple clients, privacy leakage is risky.

2) FL: In this section, we conduct experiments on the designed FL-NIDS based on the experimental parameters given in

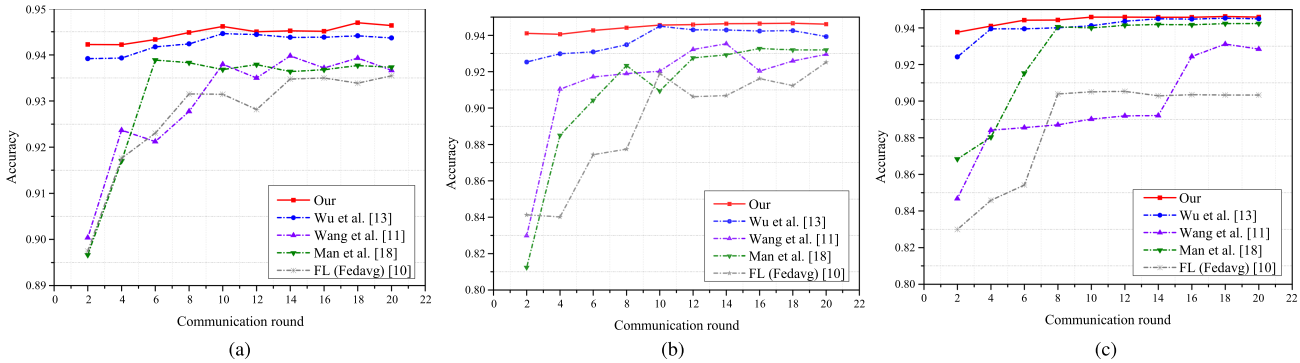

Fig. 5. Comparison of detection accuracy trends when C = 3, 5, 7 and R = 20. (a) C = 3, R = 20. (b) C = 5, R = 20. (c) C = 7, R = 20.

TABLE IV
DETECTION CAPABILITY VALUES OF NIDS UNDER DIFFERENT FEDERATED LEARNING ALGORITHMS AND DIFFERENT SCENARIOS

Attack	Indicators	Benign	Infilteration	Botnet	DoS	SQL Injection	Brute Force	DDos
Our	Accuracy	0.9462	0.9473	0.9999	0.9997	0.9999	0.9996	0.9994
	Precision	0.9074	0.6164	0.9999	0.9963	1.0000	0.9982	0.9996
	Recall	0.9862	0.1747	0.9986	0.9996	0.1429	0.9918	0.9986
	F1-score	0.9452	0.2722	0.9992	0.9980	0.2500	0.9950	0.9991
Wu et al. [13]	Accuracy	0.9436	0.9451	0.9995	0.9955	0.9994	0.9958	0.9996
	Precision	0.9096	0.5375	0.9933	0.9447	0.0402	0.9977	0.9989
	Recall	0.9772	0.1916	0.9979	0.9995	0.8571	0.8889	0.9996
	F1-score	0.9422	0.2825	0.9956	0.9714	0.0770	0.9402	0.9993
Wang et al. [11]	Accuracy	0.9439	0.9455	0.9998	0.9854	0.9999	0.9860	0.9984
	Precision	0.9093	0.5480	0.9980	0.9006	0.8443	1.0000	0.9973
	Recall	0.9782	0.1937	0.9981	0.9911	0.1429	0.6237	0.9997
	F1-score	0.9425	0.2862	0.9980	0.9118	0.2500	0.7675	0.9973
Man et al. [18]	Accuracy	0.9378	0.9407	0.9998	0.9943	0.9999	0.9996	0.9916
	Precision	0.9128	0.4508	0.9988	0.9943	0.0000	0.9975	0.9732
	Recall	0.9593	0.2333	0.9981	0.9307	0.0000	0.9925	0.9999
	F1-score	0.9354	0.3075	0.9985	0.9614	0.0000	0.9950	0.9864
FL(Fedavg) [10]	Accuracy	0.9324	0.9384	0.9998	0.9922	0.9998	0.9996	0.9879
	Precision	0.9166	0.4234	0.9985	0.9637	0.1220	0.9973	0.9620
	Recall	0.9420	0.2550	0.9982	0.9323	0.7143	0.9931	0.9999
	F1-score	0.9290	0.3180	0.9983	0.9477	0.2083	0.9952	0.9806

Table II. Considering that the size of CIC-IDS2018 is large, we have normalized the dataset so that CNN can learn data features in a more effective way. Furthermore, to fully validate the algorithm and avoid excessive impact from non-IID data on a global model, we set a relatively small number of clients with non-IID data in our experiments. Therefore, we can see from Fig. 5 that FL can achieve a high accuracy rate in the first round of training. However, in terms of the overall performance, we can see that the method has larger fluctuations, compared to other approaches. This indicates that the global model is actually affected by some local models during the aggregation phase, resulting in continuous fluctuations for the accuracy.

For the detection capabilities of the method, Table IV and Fig. 6 provide the detailed results. From there, we can observe that, compared to the centralized scheme (see Table III), the overall detection metric values for the FL scheme are 92.51%, 91.37%, 92.11%, and 91.74%, a reduction of about 2% (Take $C = 5$ as an example). Its detection performance for all types of attacks decreases by about 1%.

In this section, we conduct experiments on the DAFL scheme and the baseline studies [11], [13], [18] under the same environment and settings as FL. Our scheme implements filtering of client-side models and dynamic aggregation of local model parameters, as detailed in Section III for the framework and workflow. Then, we compare all the experimental results in Figs. 5, 6, and Table IV. Among them, Fig. 5 plots the trend of detection accuracy, which gradually increases for all schemes as the number of communication rounds increases. Compared with other federated learning solutions, our scheme has higher accuracy, smoother line trends, and can reach high detection levels quickly and consistently. In addition, since the DAFL scheme can filter unsatisfactory local models and highlight the good ones, it can also reduce the overhead.

Fig. 6 depicts the value results of the four metrics under the FL, DAFL, and baseline schemes. When $C = 3$, the detection indicators of our scheme are 94.64%, 93.61%, 92.87%, 93.24%; when $C = 5$, they are 94.61%, 93.44%, 92.98%, 93.21%; when $C = 7$, they are 94.58%, 93.38%, 93.22%, 93.3%. By

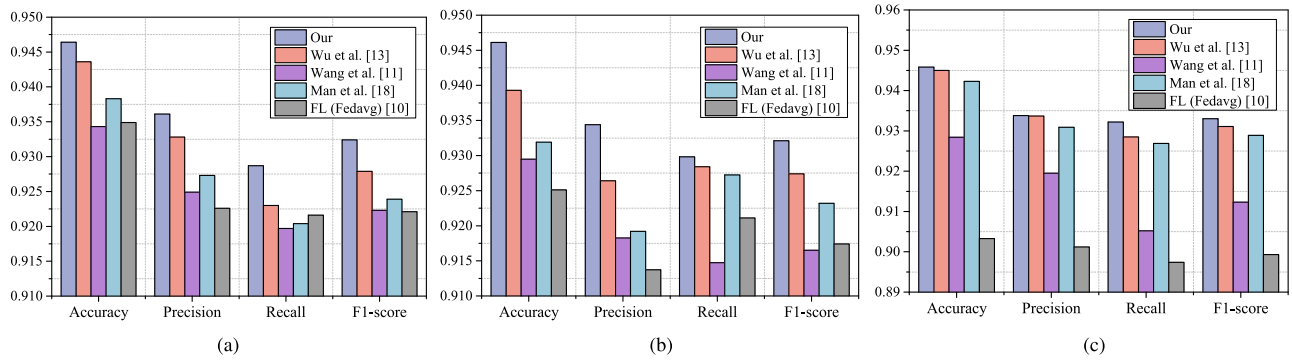

Fig. 6. Comparison of detection performance indicators when C=3, 5, 7 and R = 20. (a) C = 3, R = 20. (b) C = 5, R = 20. (c) C = 7, R = 20.

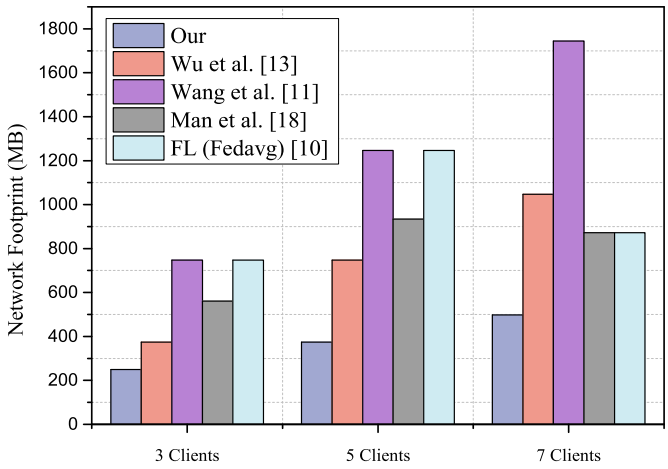

Fig. 7. Comparison of communication overhead.

comparison, we observe that these values are about 1% to 2% higher than other schemes.

At the same time, we also compare the centralized scheme (see Table IV), and from the results, we can see that the DAFL metrics are close to or even slightly higher than the CL scheme. Although all the schemes have poor detection performance for a few categories like infiltration and injection attacks, DAFL's detection metric values are better than other schemes for other attacks and normal traffic.

3) Communication Overhead: Assuming that the network environment is consistent during training learning, this section compares the communication overhead of DAFL, FL schemes, and other advanced research schemes. We evaluate it mainly regarding the number of communication rounds and the network footprint consumed. Regarding the communication rounds, during the 20 rounds of communication learning, the DAFL and Wu's schemes have the best stability, can reach a stable convergence state around the 10th round (see Fig. 5). Wang's and FL schemes have the worst stable convergence, requiring more than 20 rounds of communication iterations to reach a steady state. Man's scheme can achieve stable convergence in about 15 rounds, and its accuracy is lower than that of DAFL. If the detection effect reaches a steady state, the communication between client and server can be stopped. Therefore, in practice, DAFL has the least number of communication training rounds. We can save at least half of the communication rounds compared to other FL solutions. At the same time, DAFL selects the client, reducing some of the model parameters to be uploaded and reducing some of the communication overhead. To better illustrate the efficiency of DAFL, we measure the network footprint consumed by the five schemes during the learning process, as shown in Fig. 7. In the three cases of $C = 3$, $C = 5$, and $C = 7$, our scheme reduces the overhead by $33\% - 71\%$, in contrast to the other schemes.

V. Conclusion

In this work, we propose a federated learning system called DAFL, capable of dynamic weighted aggregation for network intrusion detection. Specifically, it can prevent uploading local parameters unfavorable to the global model and dynamically weigh different local models, preserving data privacy. We have given the detailed design of our system, and our experimental results show that DAFL can achieve faster convergence speed and higher accuracy, precision, and recall with lower communication overhead. In general, the detection performance of our scheme is close to that of the centralized method, with slight improvements compared to other federated learning schemes, and can achieve a communication reduction of 33%–71% in the federated learning process. In our future work, we plan to explore secure and efficient schemes for parameter transferring in DAFL to enhance the security of our system.

References

[1] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surv. Tut.*, vol. 16, no. 1, pp. 303–336, Jan.–Mar. 2014.

[2] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, 2020, Art. no. 102419.

[3] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput.*, vol. 22, no. 1, pp. 949–961, 2019.

[4] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowl.-Based Syst.*, vol. 189, 2020, Art. no. 105124.

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.
[6] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Comput. Ind. Eng.*, vol. 149, 2020, Art. no. 106854.
[7] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, 2021, Art. no. 106775.
[8] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 603–618.
[9] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.
[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[11] W. Luping, W. Wei, and L. Bo, "CMFL: Mitigating communication overhead for federated learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 954–964.
[12] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.
[13] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, Dec. 2021.
[14] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proc. 10th Int. Symp. Inf. Commun. Technol.*, 2019, pp. 273–279.
[15] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021.
[16] Y. Sun, H. Esaki, and H. Ochiai, "Adaptive intrusion detection in the networking of large-scale lans with segmented federated learning," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 102–112, Dec. 2020.
[17] R. Zhao, Y. Yin, Y. Shi, and Z. Xue, "Intelligent intrusion detection based on federated learning aided long short-term memory," *Phys. Commun.*, vol. 42, 2020, Art. no. 101157.
[18] D. Man, F. Zeng, W. Yang, M. Yu, J. Lv, and Y. Wang, "Intelligent intrusion detection based on federated learning for edge-assisted Internet of Things," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, 2021.
[19] L. Teng et al., "A collaborative and adaptive intrusion detection based on SVMs and decision trees," in *Proc. IEEE Int. Conf. Data Mining Workshop*, 2014, pp. 898–905.
[20] X. Ren, W. Pao, and D. Zhou, "Intrusion detection model of weighted navie Bayes based on particle swarm optimization algorithm," *Comput. Eng. Appl.*, vol. 52, no. 7, pp. 122–126, 2016.
[21] Y. Liu and D. Pi, "A novel kernel SVM algorithm with game theory for network intrusion detection," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 8, pp. 4043–4060, 2017.
[22] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[23] R.-H. Dong, X.-Y. Li, Q.-Y. Zhang, and H. Yuan, "Network intrusion detection model based on multivariate correlation analysis–long short-time memory network," *IET Inf. Secur.*, vol. 14, no. 2, pp. 166–174, 2020.
[24] D. Liao, S. Huang, Y. Tan, and G. Bai, "Network intrusion detection method based on GAN model," in *Proc. Int. Conf. Comput. Commun. Netw. Secur.*, 2020, pp. 153–156.
[25] J. Liu, Y. Gao, and F. Hu, "A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM," *Comput. Secur.*, vol. 106, 2021, Art. no. 102289.
[26] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proc. Int. Conf. Eng. Technol.*, 2017, pp. 1–6.
[27] A. Tato and R. Nkambou, "Improving adam optimizer," 2018.
[28] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proc. 2nd Workshop Distrib. Infrastructures Deep Learn.*, 2018, pp. 1–8.
[29] M. Abdel-Basset, H. Hawash, R. K. Chakrabortty, and M. J. Ryan, "Semi-supervised spatiotemporal deep learning for intrusions detection in IoT networks," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12251–12265, Aug. 2021.
[30] R. Zuech, J. Hancock, and T. M. Khoshgoftaar, "Detecting web attacks in severely imbalanced network traffic data," in *Proc. IEEE 22nd Int. Conf. Inf. Reuse Integr. Data Sci.*, 2021, pp. 267–273.
[31] R. Zuech, J. Hancock, and T. M. Khoshgoftaar, "Feature popularity between different web attacks with supervised feature selection rankers," in *Proc. IEEE 20th Int. Conf. Mach. Learn. Appl.*, 2021, pp. 30–37.

Jianbin Li received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1992, and the M.S. degree in computer science from the Analysis and Forecast Center, State Seismological Bureau, Beijing, China, in 1995.

He was the Dean with the Institute of Information Security and Big Data, Central South University, Changsha, China. He is currently a Full Professor with the North China Electric Power University, Beijing, China. His research interests mainly include information security, Big Data, and deep learning.

Xin Tong received the graduate degree in computer science from the Henan University of Economics and Law, Zhengzhou, China, in 2020. She is currently working toward the master's degree in computer science with North China Electric Power University (NCEPU), Beijing, China.

Her research interests include cyber security, machine learning, and federated learning.

Jinwei Liu (Member, IEEE) received the M.S. degree in computer science from Clemson University, Clemson, SC, USA and the University of Science and Technology of China, Anhui, China, in 2012, and the Ph.D. degree in computer engineering from Clemson University in 2016.

He has worked with the University of Virginia and University of Central Florida. He is currently an Assistant Professor with the Department of Computer and Information Sciences, Florida A&M University, Tallahassee, FL, USA. His research interests include cloud computing, Big Data, machine learning and data mining, cybersecurity, wireless sensor networks, social networks, high-performance compting, and the Internet of Things.

Prof. Liu is a member of the ACM.

Long Cheng (Senior Member, IEEE) received the B.E. degree from the Harbin Institute of Technology, Harbin, China, in 2007, the M.Sc. degree from the University of Duisburg-Essen, Duisburg, Germany, in 2010, and the Ph.D. degree from the National University of Ireland, Maynooth, Ireland, in 2014, all in electronic engineering.

He is currently a Full Professor with the School of Control and Computer Engineering, North China Electric Power University, Beijing, China. He was an Assistant Professor with Dublin City University, and a Marie Curie Fellow with the University College Dublin. He also has worked with the organizations such as Huawei Technologies Germany, IBM Research Dublin, TU Dresden, and TU Eindhoven. He has authored or coauthored around 80 papers in journals and conferences like TPDS, TON, TC, TSC, TASE, TCAD, TCC, TBD, TITS, TVLSI, JPDC, IEEE NETWORK, IEEE SYSTEMS JOURNAL, HPCA, CIKM, ICPP, CCGrid and Euro-Par, etc. His research focuses on distributed systems, deep learning, cloud computing and process mining.

Prof Cheng is a Co-Chair of *Journal of Cloud Computing*.