



Research Article

Intelligent Intrusion Detection Based on Federated Learning for Edge-Assisted Internet of Things

Dapeng Man ¹, Fanyi Zeng,¹ Wu Yang,^{1,2} Miao Yu ³, Jiguang Lv,¹ and Yijing Wang⁴

¹Information Security Research Center, Harbin Engineering University, Harbin 150001, China

²Peng Cheng Laboratory, Guangdong 518055, China

³Beijing Institute of Network Data, Beijing 100031, China

⁴Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100864, China

Correspondence should be addressed to Miao Yu; yumiao8706@163.com

Received 9 June 2021; Revised 27 July 2021; Accepted 9 August 2021; Published 5 October 2021

Academic Editor: Qi Jiang

Copyright © 2021 Dapeng Man et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As an innovative strategy, edge computing has been considered a viable option to address the limitations of cloud computing in supporting the Internet-of-Things applications. However, due to the instability of the network and the increase of the attack surfaces, the security in edge-assisted IoT needs to be better guaranteed. In this paper, we propose an intelligent intrusion detection mechanism, FedACNN, which completes the intrusion detection task by assisting the deep learning model CNN through the federated learning mechanism. In order to alleviate the communication delay limit of federal learning, we innovatively integrate the attention mechanism, and the FedACNN can achieve ideal accuracy with a 50% reduction of communication rounds.

1. Introduction

By connecting the Internet with physical objects (including people and equipment) and transmitting information between objects [1], the Internet of Things (IoT) enables the integration of the real world and the data world, thus making our life more smart and fashionable [2]. Current popular applications of IoT include smart metering, smart cities, smart hospitals, smart agriculture, and smart transportation [2].

IoT and social networking applications are growing too fast, resulting in exponential growth of data generated at the edge of the network [3]. Previously, traditional centralized cloud computing architectures could provide centralized remote services by bringing all data together on a central server [4]. However, due to issues such as network bandwidth limitations and data privacy [5], it is impractical and often unnecessary to still send all data to the remote cloud in the same way as before [3] because this process incurs significant data transmission costs and does not meet the real needs of some low-latency services and applications [4].

As a result, edge computing has emerged as an innovative strategy [5]. Edge computing migrates some network functions and data processing to the edge of the network closer to the end user, where these tasks were previously performed in the core network [6], resulting in the so-called edge-assisted IoT. Edge computing has been identified as a viable option to address the limitations of cloud computing in supporting IoT applications [7, 8]. Compared to cloud computing, edge computing helps deliver efficient network communication services to users with lower latency, more flexible access, and protection of data privacy [7, 9].

While the edge-assisted IoT architecture offers unique features and enhanced quality of service, there are still many challenges to network stability and security, and damage from malicious attacks will undermine the benefits of edge computing [7]. In edge-assisted IoT, the threat factors mainly include information data in edge networks, edge nodes, cloud servers, and other system devices. These threats come from a variety of sources, including malware, hacking, exploited system vulnerabilities, unauthorized access, and

other human factors [10]. Once the intrusion is not detected in time, it will cause incalculable damage to the applications and devices in the IoT, especially the threat to personal safety [4]. The research on edge network security is still in the development stage [4], and most of the previous intrusion detection solutions are recommended to be deployed in the core backbone network [11]; however, this does not meet the real security needs, and due to the turnover of the network structure, the applicability of the intrusion detection technology has raised higher requirements.

The rest of this paper is organized as follows. Section 2 presents an overview of intrusion detection for IoT and an overview of federated learning. In Section 3, we present our proposed intrusion detection method. In Section 4, we present our experimental setup, the utilized performance, our experimental results, and the comparison of our work with other competing approaches. Finally, in Section 5, we present our conclusion and future work.

2. Related Work

2.1. Intrusion Detection for IoT. Intrusion detection is a strong active defense mechanism, and there are many intrusion detection methods [12]. Depending on the data source, intrusion detection techniques can generally be divided into two categories: network-based intrusion detection and host-based intrusion detection [13]. The network-based intrusion detection system is one of the solutions for the early detection of network attacks [14]. According to the different detection mechanisms, intrusion detection technology can be divided into two categories: misuse-based intrusion detection and anomaly-based intrusion detection. Misuse-based intrusion detection uses a set of predefined rules and patterns to detect attacks. Anomaly-based intrusion detection uses a precollected dataset with normal or malicious behaviour labels and uses this dataset to train and test a detection model; any network flow that deviates from the model threshold is flagged as an anomalous entry and reported as an attacker. Due to the excellent classification performance of machine learning, researchers have widely used machine learning methods in anomaly-based intrusion detection, such as the Bayesian model, support vector machine, genetic algorithm, and other machine learning models [13]. However, today's network data present larger, complex, and multidimensional features. When facing high-dimensional data features, traditional machine learning methods need to manually extract a large number of features. The process is complex and the computation is large, which cannot meet the accuracy and real-time requirements of intrusion detection [15].

Deep learning, as an important branch of machine learning, has attracted more and more attention since its theory was proposed. The deep learning model has good advantages in dealing with complex data, and it can automatically extract better representation features from large-scale data. For image classification, feedforward neural networks, especially convolutional neural networks (CNN), have achieved well-known advanced results. For language modelling tasks, recurrent neural networks (RNN), especially LSTM, have also achieved remarkable results [16].

Neural network models such as those described above, with good self-learning capabilities, high-speed optimization, and efficient parallel distributed processing, are also very suitable for handling complex data in network traffic [15].

Currently, the commonly used models in DL-based IoT intrusion detection are DBN, CNN, RNN, GAN, etc., [17, 18]. However, the network structure and training process of most depth models are usually complex, and the model parameters are more, which increases the difficulty and energy consumption of training. CNN has the property of weight sharing, which can effectively avoid the problem of more parameters in complex network structure and reduce the dimension of data by convolution layer and pooling layer, which can effectively reduce the network complexity and speed up the intrusion detection rate. Yang and Wang [19] proposed a wireless network intrusion detection method based on an improved convolutional neural network (ICNN). Simulation results show that the proposed model has acceptable performance on public datasets compared with traditional methods. Considering the limited resources of IoT edge equipment, Stahl et al. [20] proposed a solution to perform CNN model detection tasks in a distributed manner using multiple collaborative edge devices. The core mechanism of their method is to partition the CNN layer where the dominant weight is, distribute the weight data and calculation load evenly on all available devices, and then minimize the task execution time.

Recently, researchers have been exploring whether collaborative intrusion detection systems [21] can be a mainstream option for detecting attacks in large and complex networks, such as the IoT [22]. By detecting real worm datasets with accuracy and overhead, Sharma et al. [11] evaluated the applicability and performance of centralized IDS and purely distributed IDS architectures. Through experiments, they observed that when a large number of edge nodes have attacks, centralized IDS has higher network requirements than distributed IDS and does not have more advantages in detection performance. It can be seen that Stahl et al. [20] did provide a reasonable solution for better completing the detection task on resource-constrained edge devices, but they ignored the protection of data privacy and the threat of malicious nodes.

2.2. Federated Learning. In recent years, the emergence of federal learning (FL) has enabled the deep learning model to effectively train, ensure the security and privacy of data, and effectively solve the problem of data island. The original design purpose of FL is to conduct efficient learning among multiple participants or computing nodes, on the premise of ensuring information security in the process of data exchange. FL can be used as the enabling technology of edge network because it can realize the collaborative training of ML and DL models and also be used for the optimization of edge network [23]. In FL, client devices (edge devices) use local data to train the ML model and then send updated model parameters (rather than raw data) to the cloud server for aggregation.

According to [24], federal learning can be divided into three categories: (1) horizontal federated learning, which applies to scenes with different samples but the same characteristics in multiple datasets, (2) vertical federated learning, which applies to scenarios where multiple datasets have the same sample but different feature spaces, and (3) federated transfer learning applies to scenarios where multiple datasets have different samples and different feature spaces. The federal learning architecture in our subsequent proposed approach belongs to horizontal federal learning.

Since the server needs to interact with the client in the edge network, the wireless connection between them is unstable and unreliable, reducing the number of rounds of communication between the server and the client which is necessary to provide security. The delay in communication is the performance bottleneck of the entire learning framework [25, 26]. For this reason, our goal is to ensure the good performance of the detection mechanism while reducing the cumulative number of communication rounds.

In addition, common machine learning (including deep learning) methods usually follow the assumption that data are independent and identically distributed (IID). However, in real networks, especially under the current edge-assisted IoT architecture, different IoT devices belong to a certain user, enterprise, or application scenario, and thus, data distribution often varies greatly, and due to factors such as user groups and geographical associations, these data often have certain correlations, so the data on edge devices are likely to be non-IID, and this situation deserves strong attention, especially when federated learning is introduced and used to collaborate with tens of thousands of devices and their private data for model training. Using only existing machine learning methods to handle non-IID data for training will lead to low model accuracy and poor model convergence [6]. Therefore, our study focuses on the treatment when the data are non-IID.

In this paper, we propose an intelligent intrusion detection mechanism, FedACNN, based on federated learning-(FL-) aided convolutional neural network (CNN) for edge-assisted IoT. FedACNN completes detection tasks using the CNN model in the framework of federated learning. Our FedACNN uses local datasets and computing resources of edge devices for model training and uploads model parameters to a central server for collaborative training. Compared with traditional centralized learning approaches, FedACNN does not require the transfer of raw data to a central server, ensuring model accuracy while reducing the risk of data leakage. At the same time, we incorporate an attention mechanism, which allows fewer communication rounds and lower detection latency while ensuring the performance of the detection model.

3. Proposed Methods

In this section, we first detail the composition of the learning model CNN and then introduce the common FL algorithms. Finally, we innovatively incorporate attention mechanisms into the FL model to constitute an intrusion detection mechanism based on FL-aided CNN.

3.1. CNN for Intrusion Detection. CNN is a kind of artificial neural network (ANN), which is inspired by the study of visual cortex cells. Its important feature is to reduce the number of parameters in the network by local connection and shared weight, to obtain some degree of affine invariance.

Using the neural network model for detection tasks, the structure of the network has a great influence on the detection results. However, most edge devices are limited by memory and computing resources [27], and they cannot store and execute complex CNN models. Therefore, when we set the hierarchical structure of the CNN model, we mainly start by adjusting parameters and optimizing the structure. While ensuring accuracy, the CNN model structure is relatively simple, as shown in Figure 1.

The CNN model structure is mainly composed of the convolution layer, the pooling layer, and the full connection layer. The first layer is the data input layer. In the training process, the data distribution will change, which will bring difficulties to the learning of the next network. So, after the convolution layer, we use batch normalization to force the data back to the normal distribution with mean 0 and variance 1; on the one hand, it makes the data distribution consistent, and on the other hand, it avoids gradient disappearance. We use the ReLU function as a nonlinear activation function to replace the Sigmoid or tanh function commonly used in traditional neural networks, which can effectively accelerate the speed of network convergence and training. The eighth layer is the pooling layer (down-sampling layer), which mainly conducts the downsampling of the input. The pooling operation reduces the output size of the convolution layer, thus reducing the calculation cost and avoiding overfitting. The commonly used pooling methods include the mean-pooling and the max-pooling, and the max-pooling method is selected in this paper. The ninth layer to the eleventh layer is the fully connected layers; the number of neurons in each layer has been marked in Figure 1. The last layer is the output layer of the network, which is mainly used for classification prediction. We use Softmax as the decision function, which generates a fractional vector for each class and returns the maximum index as the predicted class.

3.2. Federated Learning. The FL structure consists of one server and several clients. In this paper, the term “server” is the remote cloud server, and the term “client” is the edge network entities such as edge nodes and edge devices. The basic idea of FL is “the data does not move, the model moves” [27]. As shown in Figure 2, specifically, the server provides a global shared model and the client downloads the model and uses local datasets for training, while updating model parameters. In each communication between the server and the client, the server will get the current model parameters are distributed to each client (also can be described as the client download server model parameters), after the client training, and then, the updated model parameters are uploaded to the server; the server will aggregate client model parameters through some method, as the

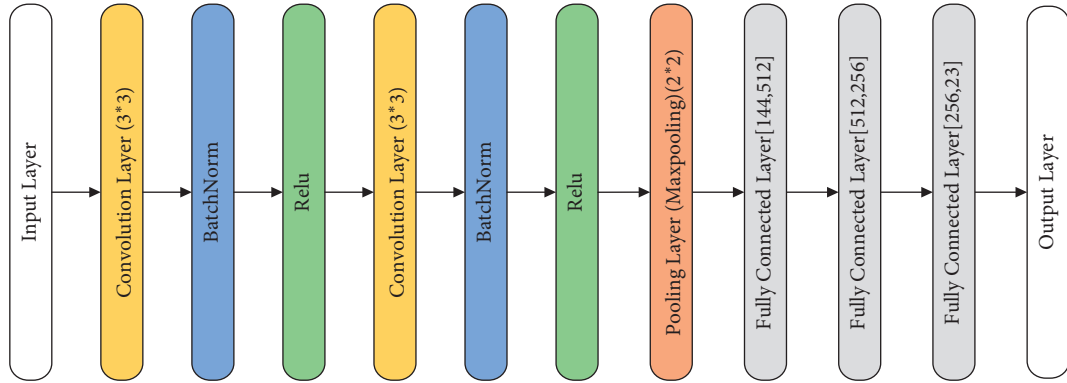


FIGURE 1: Structure of the proposed CNN model.

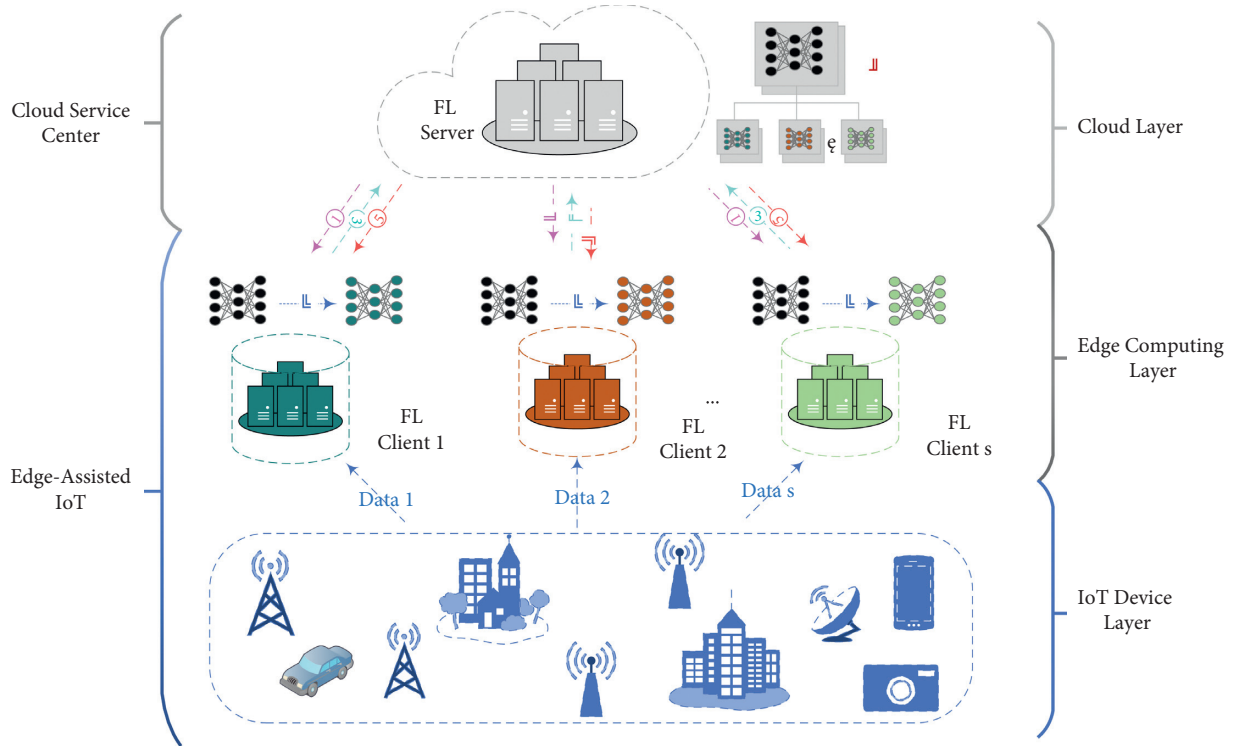


FIGURE 2: Intelligent intrusion detection based on FL-aided CNN for edge-assisted IoT. ① Model initialization; ② local model training and update; ③ upload the updated local model parameters; ④ server aggregates and updates parameters of the global model; ⑤ download the updated global model parameters. Repeat ②–⑤, until model convergence.

updated global model parameters, of this cycle. The server has a variety of aggregation methods for the model parameters uploaded by clients. In [16], the authors proposed a federal averaging algorithm, FedAVG. This algorithm combines the local stochastic gradient descent (SGD) of each client and the execution of model averaging on the server side. In FL scenarios, communication costs are very large constraints, compared with the synchronous random gradient descent algorithm; their FedAVG can greatly reduce the number of communication rounds.

The computation of FedAVG is mainly controlled by three key parameters: C , the fraction of clients, B , local batch size for client update, and E , the number of local epochs. In particular, when $B = \infty$ (i.e., the complete local dataset used

for the client update) and $E = 1$, then FedAVG is equivalent to FedSGD [16]. FedACNN refers to FedAVG as a baseline model framework. See Algorithm 1, for details of the FedAVG algorithm steps.

3.3. Improving FedAVG for Intrusion Detection. Based on FedAVG, inspired by the idea of attention mechanism, we match different importance degrees for the edge nodes involved in collaborative detection. In other words, different clients have different weights. The standard of importance degree is based on the contribution of local model classification performance to the improvement of global model classification performance. The weight is high when the

```

procedure Server:
  initialize  $w_0$ 
  for each round  $t=1, 2, \dots$  do
     $m \leftarrow \max(C \cdot K, 1)$ 
     $S_t \leftarrow$  (random set of  $m$  Clients)
    for each Client  $k \in S_t$  in parallel do
       $w_k(t+1) \leftarrow$  ClientUpdate( $k, w_k(t)$ )
       $w_G(t+1) \leftarrow$  ClientUpdate( $k, w_k(t+1)$ )
procedure Client (k, w): //Run on Client k
   $B \leftarrow$  (split local Client data into batches of size  $B$ )
  for each local epoch  $i$  from 1 to  $E$  do
    for batch  $b \in B$  do
       $w \leftarrow w - \eta * \iota(w; b)$ 
    return  $w$  to Server

```

ALGORITHM 1: FedAVG. Given K clients (indexed by k), B is the local minibatch size, E is the number of local epochs, R is the number of global rounds, C is the fraction of clients, and η is the learning rate.

contribution is large, and vice versa. The server does not only aggregate the average model of clients but also aggregate the weighted model parameters, which can accelerate the convergence of the global model. At the same time, it reduces the adverse effects of model parameters that contribute less to global models. The FedACNN algorithm is described by Algorithm 2. The innovative contributions of this algorithm are mainly reflected as follows.

3.3.1. Server. Like the previous FedAVG algorithm, the server initializes and publishes the model parameters. Each client device downloads the initial model parameters of the server, uses the local dataset for training, and uploads the updated model parameters to the server. After each client device uploads the updated model parameters, the server aggregates the parameters. It is necessary to note that we only use the idea of FedAVG to process non-IID data and apply it to the FedACNN algorithm.

Assuming that the server communicates with the client total t round, in the first round of communication, the server receives updates from s clients, the model parameter after the average aggregation of the server is $W_G(1)$, and $N' = |D_1| + |D_2| + \dots + |D_s|$ is the total number of client data samples. We update the global model on the server side by

$$W_G(1) = \sum_{k=1}^s \frac{|D_k|}{N'} W_k(1), \quad (1)$$

where $W_k(1)$ is the local model parameter of client k in the first round of updating. In the later update aggregation process, an attention mechanism is introduced to aggregate weighted parameters.

3.3.2. Weighted Parameter Aggregation. Inspired by the attention mechanism and FedAGRU [25], FedACNN implements the attention mechanism on the server side, assigns different weights to different clients, and applies the weights to the model parameter aggregation process. The model parameter vector set updated by s clients is expressed

as $W = [W_1, W_2, \dots, W_s]$. Assuming that the model parameter vector W_k updated by the client k is n -dimensional; after the first round of communication ($t > 1$), the Euclidean distance $d(W_G(t), W_k(t))$ between the parameter matrices are calculated by

$$d(W_G(t), W_k(t)) = \sqrt{\sum_{i=1}^n (W_G(t)_i - W_k(t)_i)^2}. \quad (2)$$

Compare the magnitude of d value between the model parameters updated by each client and the model parameters after global aggregation by the server, which is used to measure the contribution of each client's local parameters to the global parametric model optimization. Due to the large dimensional difference between different parameters, the Sigmoid function (see (3)) is used for normalization. The normalized result is $a_k(t)$ (see (4)). Furthermore, using (5) to assign importance $h_k(t)$ to client k ,

$$f_{\text{Sigmoid}} = \frac{1}{1 + e^{-x}}, \quad (3)$$

$$a_k(t) = f_{\text{Sigmoid}}(d(W_G(t), W_k(t))), \quad (4)$$

$$h_k(t) = \frac{s * a_k(t)}{\sum_{k=1}^s a_k(t)}. \quad (5)$$

In the next round of global aggregation, the server will be based on the importance of each client $h_k(t)$ and aggregate model parameters for each the client using (6). This cycle will continue until optimal performance is achieved:

$$W_G(t) = \sum_{k=1}^s \frac{|D_k|}{N^*} h_k(t-1) * W_k(t), \quad (6)$$

where N^* is the total number of data samples for selected clients.

It is important to note that, except for the average aggregation of parameters by the server in the first round of communication, the server aggregates weighted parameters

```

procedure Server:
  initialize  $W_0$ 
  for each round  $t=1, 2, \dots$  do
     $m \leftarrow \max(C \cdot K, 1)$ 
     $S_t \leftarrow$  (random set of  $m$  Clients)
    for each Client  $k \in S_t$  in parallel do
       $W_k(t+1) \leftarrow$  ClientUpdate( $k, W_k(t)$ )
    if  $t=1$  then
       $W_G(1) = \sum_{k=1}^S |D_k|/N' W_k(1)$ 
      Calculate the Client  $K$  importance degree  $h_k(1)$  following equations (3)–(6)
    else
       $W_G(t) = \sum_{k=1}^S |D_k|/N' h_k(t-1) * W_k(t)$ 
      Calculate the Client importance degree  $h_k(t)$  following equations (3)–(6)
      Update parameter  $W_G(t+1), h_k(t+1)$ .
  procedure Client Update (k, w)://Run on Client k
   $B \leftarrow$  (Split local Client data into batches of size B)
  for each local epoch  $i$  from 1 to  $E$  do
    for batch  $b \in B$  do
       $w \leftarrow w - \eta * \nabla(w; b)$ 
    return  $w$  to Server

```

ALGORITHM 2: FedACNN. Given K clients (indexed by k), B is the local minibatch size, E is the number of local epochs, R is the number of global rounds, C is the fraction of clients, and η is the learning rate.

in each subsequent round of communication. According to the contribution of the updated parameters of the client in the previous round to the global model optimization, the importance of each client in each round will be dynamically allocated.

4. Experiments and Results

We use the NSL-KDD dataset to evaluate the model. Our experiments can dynamically select the number of clients, and each client uses local data to train its model and upload updated model parameters to the server for aggregation.

The server used in this experiment is the Windows10 operating system and the processor is Intel(R) Core (TM) i5-10210U CPU@2.11 GHz. We use Python’s deep learning library Pytorch to program FedACNN.

4.1. Dataset Preprocessing. ML algorithms rely on large amounts of data to train models to provide better results. Data is usually stored in storage container devices such as files and databases, which cannot be used directly for training [28]. Before passing the data to the learning model for training, we must preprocess the data.

A key component of an effective intrusion detection system is a good dataset that reflects real-world reality. To verify the effectiveness of the approach proposed in the paper, we conduct experiments using a public intrusion detection dataset-NSL-KDD. NSL-KDD dataset is a commonly used dataset for intrusion detection. The NSL-KDD dataset has advantages over the original KDD CUP 99 dataset, such as it does not include redundant records in the training set, so the classifier is not biased towards more frequent records, etc. The dataset contains normal networks’ behaviour data, as well as four major categories of abnormal

attack data, which are denial-of-service (DoS) attack, user to root (U2R), remote to local attack (R2L), and probing attack (Probe), and their specific descriptions are shown in Table 1. Our work focuses on classifying and detecting four major categories of anomalous attacks. Each sample in the NSL-KDD dataset contains 41 feature attributes and one class identification, and the class identification is used to indicate whether the connection record is normal or a specific type of attack.

To better use the NSL-KDD data as input data for the CNN model, we performed a preprocessing operation on the original dataset. The preprocessing procedure includes numerical, normalization, and visualization.

4.1.1. Numerical. The original NSL-KDD dataset has four character-based feature attributes, namely, protocol, service, flag, and label. We have used the label encoder function to numerically encode each of the four character-based values.

4.1.2. Normalization. After the numerical processing is performed, there is a large difference in magnitude between the values. This situation tends to cause problems such as slower convergence of the network and saturation of neuron outputs; therefore, normalization of the original data is required. We used the Min-Max normalization method (see (7)) to normalize the data to the interval [0, 1]:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad (7)$$

where x^* is the normalized data, x is the original data, x_{\min} is the minimum data value in the current attribute, and x_{\max} is the maximum data value in the current attribute.

TABLE 1: Specific description of each type of attack.

Category	Description	Specific attacks included in the training set	Specific attacks included in the test set
DoS	Sending a large number of packets to the server to make it busy, the attacker tries to prevent legitimate users from accessing the server	back, land, Neptune, pod, smurf, teardrop	apache2, mailbomb, processtable
U2R	The attacker accesses the system through a normal user account and then attempts to gain root access to the system using certain vulnerabilities	buffer-overflow, perl, load module, rootkit	htptunnel, ps, sqlattack, xterm
R2L	The attacker can remotely log into the computer and then use the computer's account and weak password to enter the computer to operate	ftp-write, imap, multihop, phf, guess-passwd, warezclient, spy, warezmaster	sendmail, worm, named, xlock, snmpgetattack, snmpguess, xsnoop
Probe	The attacker purposefully collects information about a computer network to bypass its security controls	ipsweep, Satan, nmap, portsweep	Mscan, saint

4.1.3. *Visualization.* After preprocessing the data, we obtained a numerical dataset with values between $[0, 1]$, and we converted the numerical type data into an $8 * 8$ two-dimensional matrix, with the empty space to be filled with zeros.

In the following experiments, Accuracy, Precision, Recall, and $F1_score$ are selected as the evaluation indexes for evaluating the performance of various methods. The specific description of these evaluation indexes is shown in Table 2. TP and TN correctly classify positive/negative samples; in fact, FP indicates a false alarm that misidentifies a negative sample as a positive sample by mistake negative samples into positive samples and FN means that a positive sample is mistaken for a negative sample, indicating a missing alarm [29].

4.2. *Applicability of FL Model.* We first use Accuracy to evaluate the performance of the centralized learning (CL) model and the federated learning model on NSL-KDD. For a centralized model, we use our proposed local CNN model for experiments. Centralized algorithm uploads data to the server for centralized training. FedAVG [29] is selected as one of the comparison methods of federal learning. In order to better illustrate the advantages of the proposed algorithm, the training models of the comparison federal learning methods are CNN. Reasonable selection of hyperparameters will greatly affect the performance of the algorithm [30]. Under different hyperparameter configurations, we studied the classification performance of the centralized learning model CNN (CL-CNN) and the FL model and determined the reasonable parameters of the algorithms (the specific parameter configuration is shown in the table). The simulation results (see Table 3) show that, under the condition of 40 rounds of iteration, CL-CNN has the highest accuracy of 99.65%; this is because the CL-CNN model has more complete datasets, so its accuracy can be used as the theoretical upper limit of the accuracy of the federal learning model. Compared with FedAVG, FedACNN has higher accuracy of 99.12%. Although it is slightly lower than CL-CNN, it is within an acceptable gap. Experiments show that collaborative training of federal learning can achieve ideal accuracy while protecting data

privacy. In other words, FedACNN sacrifices a bit of accuracy to protect data privacy.

4.3. *Classification Performance Evaluation.* After verifying the applicability of the federal learning model, we use Accuracy to measure the overall classification performance of FedACNN without limiting the number of communication rounds, and use Recall, Precision, and $F1_score$ to measure the specific classification performance of FedACNN. First, we use FedAVG to complete the comparative experiment. The hyperparameters of the FL model are set as $B = 128$, $E = 5$, $K = 10$, $C = 1$, and $\eta = 1e - 2$. Through the experimental results shown in Table 4, we observe that since there are fewer samples of R2L and U2R types of attacks, the indicators of these two types of attacks are lower than those of other types, followed by Probe. Since there are too few samples of the U2R class, FedAVG cannot accurately classify such attacks, while FedACNN has certain classification ability for such attacks. FedACNN also has obvious advantages in detecting other types of attacks, and its overall classification accuracy can reach 99.76%. Figure 3 shows the detection performance of FedACNN for various attacks.

We compare the results of the NSL-KDD dataset after applying traditional machine learning for training and testing with FedACNN and also compare the classification performance of typical deep learning models along with several current innovative mechanisms. The traditional machine learning models used for comparison include Random Forest (RF), SVM, and Naive Bayes [31], typical deep learning models include LeNet-5, DBN, and RNN [19], and innovative mechanisms include IoTDefender [24] and IBWNIDM [19]. Combined with the analysis of the experimental results in [31], it is clear that the RF classifier has the best classification performance compared to SVM and Naive Bayes, and its detection accuracy can reach 99.1% for normal traffic, 98.7% for DoS, and 97.5%, 96.8%, and 97.6% for U2R, R2L, and Probe, respectively. Comparing the above results with the detection results produced by our model, FedACNN has higher detection accuracy for normal traffic and various types of attacks than the three types of traditional machine learning models mentioned above (see Table 5).

TABLE 2: Method evaluation metrics.

Metrics	Narrative description	Equation to describe
Accuracy	The percentage of correct classification records in total records	$TP + TN / TP + TN + FP + FN$
Precision	The percentage of the number of prediction pairs of this category to all the prediction number of this category	$TP / TP + FP$
Recall	The percentage of the number of prediction pairs of this category to all the number of this category	$TP / TP + FN$
F1_score	A measure of classification problems is a harmonic average of precision and recall	$2 * Precision * Recall / Precision + Recall$

TABLE 3: Comparison of overall classification accuracy between the CL model and the FL model.

Method ($B = 128, E = 1, K = 10, C = 1, \eta = 1e-2$)	Accuracy (%) ($R = 10$)	Accuracy (%) ($R = 20$)	Accuracy (%) ($R = 40$)
CL-CNN	98.97	99.40	99.65
FedAVG (CNN)	98.13	98.58	98.90
FedACNN (proposed)	98.73	99.02	99.12

TABLE 4: Comparison of specific classification performance of different methods.

	FedACNN			FedAVG (CNN)		
	Recall	Precision	F1_score	Recall	Precision	F1_score
DoS	99.92	99.93	99.93	99.79	99.77	99.78
U2R	45.59	90.00	60.52	23.07	63.91	33.90
R2L	85.54	90.15	87.78	69.25	80.00	74.24
Probe	88.79	93.86	91.26	75.84	84.08	79.75
Normal	99.81	99.43	99.62	99.49	98.99	99.23
All	83.92	94.67	88.97	73.49	85.35	78.98

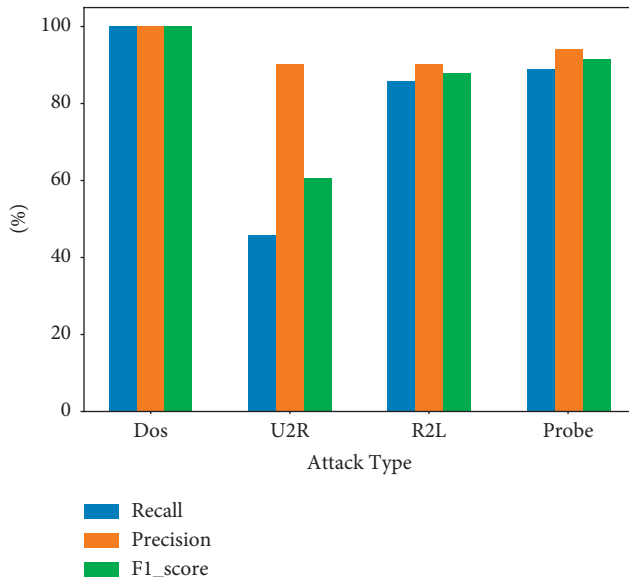


FIGURE 3: The detection performance of FedACNN for various attacks.

As shown in [19], IBWNIDM is a detection mechanism based on the improved CNN model, and its overall detection accuracy reaches 95.36%, which is already higher than the typical deep learning models, LeNet-5, DBN, and RNN.

Comparing the above results with the detection performance of our designed FedACNN, the overall detection accuracy of FedACNN is 4% higher more than. The authors of [24] claim that their IoTDefender is the first framework to apply federated transfer learning to 5G IoT IDSs, but the detection accuracy of IoTDefender for the NSL-KDD dataset is only 81.99%, which is much worse than that of our FedACNN (see Table 6). Through the above comparative analysis, we can learn that FedACNN has better detection performance.

4.4. Communications' Efficiency Assessment. Through the relationship between the number of communication rounds and the accuracy, we evaluate the learning speed of the proposed algorithm and compare the classification accuracy performance of FedACNN and FedAVG (CNN) under different communication rounds. Figures 4 and 5 show the relationship between the number of communication rounds and accuracy. Compared with FedACNN and FedAVG, FedACNN has better performance under the same communication round. In other words, FedACNN needs fewer communication rounds to achieve model convergence. FedACNN can achieve the accuracy of FedAVG in 40 rounds of communication when communicating for 20 rounds, and the number of communication rounds is reduced by 50%. This is due to the addition of the attention mechanism in FedACNN, so the server aggregation of client model parameters is no longer an average aggregation, but a

TABLE 5: Comparison of specific classification performance with traditional machine learning methods.

Classification algorithm	Accuracy for different classes of attacks				
	Normal	Dos	U2R	R2L	Probe
Random forest	99.1	98.7	97.5	96.8	97.6
SVM	98.1	97.8	93.7	91.8	90.7
Naive Bayes	70.3	72.7	70.7	69.8	70.9
FedACNN (proposed)	99.8	99.9	99.1	99.0	99.2

TABLE 6: Comparison of specific classification performance with traditional machine learning methods.

Classification algorithm	LeNet-5	DBN	RNN	IBWNIDM	IoTDefender	FedACNN
Accuracy (%)	86.54	92.45	93.08	95.36	81.99	99.76

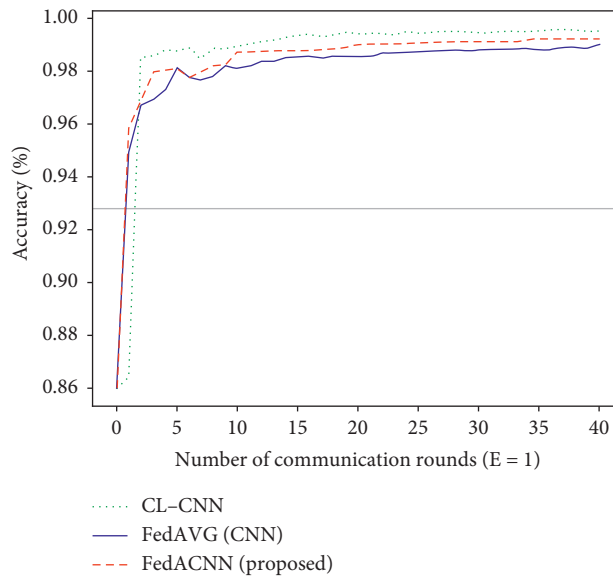


FIGURE 4: Relationship between number of communication rounds and accuracy ($B = 128, E = 1, K = 10, C = 1,$ and $\eta = 1e - 2$).

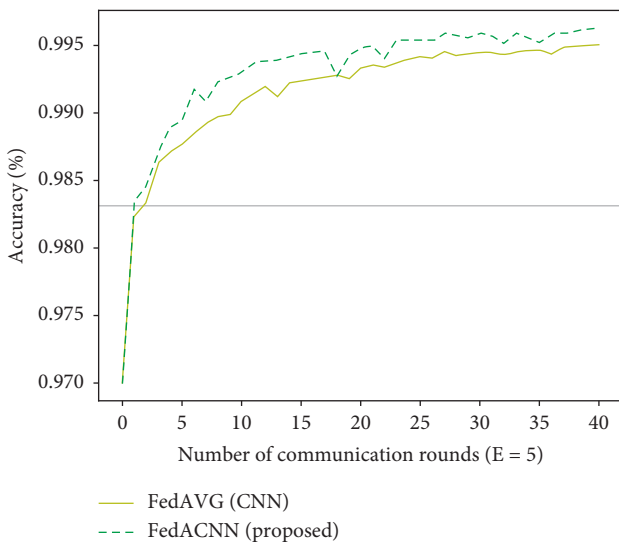


FIGURE 5: Relationship between number of communication rounds and accuracy ($B = 128, E = 5, K = 10, C = 1,$ and $\eta = 1e - 2$).

weighted aggregation according to different importance, allowing parameters that contribute more to the overall model to play a greater role, thus speeding up convergence and reducing the number of communication rounds.

4.5. Comparison of Classification Performance between the Local Model and Global Collaboration Model. We use FedACNN to complete this part of the experiment. On the one hand, each client model uses the local dataset for training, and the local iterations are 40, 120, and 200 rounds, namely, $E = 40, 120,$ and 200 . On the other hand, the FL mechanism is used for collaborative training, the parameters $E = 1,$ and R is 40, 120, and 200, respectively. In each iteration number (communication rounds), we use Accuracy and Precision as evaluation indexes to evaluate the classification performance of these two mechanisms. The results in Table 7 show that due to the incomplete characteristics of the dataset, each client model shows overfitting. In the same number of iterations, the overall classification performance

TABLE 7: Comparison of classification performance between the local model and the global collaboration model.

Model $B=128, K=10, C=1, \eta=1e-2$	Local model: $E=40$ FL model: $E=1, R=40$		Local model: $E=120$ FL model: $E=1, R=120$		Local model: $E=200$ FL model: $E=1, R=200$	
	Accuracy (%)	Precision (%)	Accuracy (%)	Precision (%)	Accuracy (%)	Precision (%)
Client model01	99.07	92.38	99.22	93.04	99.54	93.99
Client model02	98.97	91.95	99.09	91.90	99.53	92.30
Client model03	99.06	91.31	99.27	92.01	99.57	94.53
Client model04	99.08	91.67	99.15	91.58	99.53	91.12
Client model05	99.06	90.11	99.21	92.92	99.56	93.78
Client model06	98.96	91.64	99.23	92.19	99.56	91.03
Client model07	98.98	91.50	99.18	93.01	99.58	94.17
Client model08	98.84	90.89	99.11	90.70	99.36	88.59
Client model09	99.06	90.86	99.22	91.52	99.57	91.62
Client model10	99.03	91.42	99.19	92.20	99.57	91.27
FedACNN	99.12	92.74	99.45	93.06	99.63	94.56

of each client under the FL mechanism is much better than that only using local data to train in the local. This is because when the client only uses local data for model training, the dataset is small and the number of samples is limited. FedACNN, through the federal learning mechanism where the Clients update parameters and the server aggregates them and sends them to the clients, makes each client no longer only use small dataset for training, but use the current global optimal parameters for training, which makes the model detection performance trained by each client better.

5. Conclusions

In this paper, we propose an intelligent intrusion detection mechanism for edge-assisted IoT, FedACNN, which is based on FL-aided CNN. Under the premise of protecting data privacy, the FedACNN can complete the intrusion detection task with relatively ideal performance, and the overall classification accuracy of FedACNN for attack data can reach 99.76%. Since we innovatively integrate the attention mechanism, FedACNN can obtain higher detection accuracy with less communication overhead. By comparing the detection results on the USL-CUP dataset, FedACNN has better accuracy performance than three traditional machine learning models, three typical deep learning models, and two innovative mechanisms. Compared with FedAVG, the number of communication rounds is reduced by 50%.

We believe and expect that FedACNN can make some contributions to the research on security protection for edge-assisted IoT. In the future, we will conduct intrusion detection research on encrypted traffic data of edge-assisted IoT and look forward to making greater contributions to protecting edge-assisted IoT.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant nos. 61771153, 61831007, and 61971154) and the Fundamental Research Funds of the Central Universities (Grant no. 3072020CF0601).

References

- [1] S. S. Swarna and R. Ratna, "Investigation of machine learning techniques in intrusion detection system for IoT network," in *Proceeding of the 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 1164–1167, IEEE, Thoothukudi, India, December 2020.
- [2] S. Anand and A. Sharma, "Assessment of security threats on IoT based applications," *Materials Today: Proceedings*, 2020, In press.
- [3] D. Wu, J. Yan, H. Wang, and R. Wang, "Multiattack intrusion detection algorithm for edge-assisted internet of Things," in *Proceeding of the International Conference on Industrial Internet (ICII)*, pp. 210–218, IEEE, OL, USA, November 2019.
- [4] S. Wang, T. Tuor, T. Salonidis et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.
- [5] A. Alwarafy, K. A. A. Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4004–4022, 2021.
- [6] W. Y. B. Lim, C. Luong, T. Hoang, Y. Jiao, and C. Liang, "Federated learning in mobile edge networks: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.
- [7] F. Lin, Y. Zhou, X. An, I. You, and K. R. Choo, "Fair resource allocation in an intrusion-detection system for edge computing: ensuring the security of internet of Things devices," *IEEE Consumer Electronics Magazine*, vol. 7, no. 6, pp. 45–50, 2018.
- [8] G. W. Cassales, H. Senger, E. R. de Faria, and A. Bifet, "IDSA-IoT: an intrusion detection system Architecture for IoT networks," in *Proceeding of the Symposium on Computers and Communications (ISCC)*, pp. 1–7, IEEE, 2019.
- [9] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy

- for industry 4.0,” *SCIENCE CHINA: Information Sciences*, vol. 64, 2020.
- [10] Q. Cui, C. Huang, Z. Zhu et al., “Lightweight security mechanism based on heterogeneous construction for virtualized edge system: a markov decision process approach,” in *Proceedings of the International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1003–1009, Nanjing, Jiangsu, China, October 2020.
- [11] R. Sharma, C. A. Chan, and C. Leckie, “Evaluation of centralised vs distributed collaborative intrusion detection systems in multi-access edge computing,” in *Proceedings of the 2020 IFIP Networking Conference (Networking)*, pp. 343–351, Paris, France, June 2020.
- [12] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, “Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques,” *Mobile Networks and Management (MONAMI)*, vol. 235, pp. 30–44, 2017.
- [13] R. Zhao, Y. Yin, Y. Shi, and Z. Xue, “Intelligent intrusion detection based on federated learning aided long short-term memory,” *Physical Communication*, vol. 42, 2020.
- [14] S. Hosseini, “A new machine learning method consisting of GA-LR and ANN for attack detection,” *Wireless Networks*, vol. 26, no. 6, pp. 4149–4162, 2020.
- [15] A. P. D. C. Kelton, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. D. Albuquerque, “Internet of Things: a survey on machine learning-based intrusion detection approaches,” *Computer Networks*, vol. 151, pp. 147–157, 2019.
- [16] B. McMahan, E. Moore, D. Ramage et al., “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54, pp. 1273–1282, Ft. Lauderdale, FL, USA, April 2017.
- [17] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, “Understanding node capture attacks in user authentication schemes for wireless sensor networks,” *IEEE Transactions on Dependable and Secure Computing*, 2020.
- [18] X. Yuan, C. Li, and X. Li, “DeepDefense: identifying DDoS attack via deep learning,” in *Proceeding of the International Conference on Smart Computing (SMARTCOMP)*, pp. 1–8, IEEE, Hong Kong, China, May 2017.
- [19] H. Yang and F. Wang, “Wireless network intrusion detection based on improved convolutional neural network,” *IEEE Access*, vol. 7, Article ID 64366, 2019.
- [20] R. Stahl, A. Hoffman, D. M. Gritschneider et al., “DeeperThings: fully distributed CNN inference on resource-constrained edge devices,” *International Journal of Parallel Programming*, vol. 49, 2021.
- [21] J. Arshad, M. A. Azad, M. M. Abdeltaif, and K. Salah, “An intrusion detection framework for energy constrained IoT devices,” *Mechanical Systems and Signal Processing*, vol. 136, 2020.
- [22] X. An, X. Zhou, X. Lü, F. Lin, and L. Yang, “Sample selected extreme learning machine based intrusion detection in fog computing and MEC,” *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 7472095, 2018.
- [23] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, “Federated learning: a survey on enabling technologies, protocols, and applications,” *IEEE Access*, vol. 8, Article ID 140699, 2020.
- [24] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, “IoTDefender: a federated transfer learning intrusion detection framework for 5G IoT,” in *Proceeding of the 14th International Conference on Big Data Science and Engineering (BigDataSE)*, pp. 88–95, IEEE, Erode, India, April 2021.
- [25] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, “Intrusion detection for wireless edge networks based on federated learning,” *IEEE Access*, vol. 8, Article ID 217463, 2020.
- [26] N. A. A. A. Marri, B. S. Ciftler, and M. M. Abdallah, “Federated mimic learning for privacy preserving intrusion detection,” in *Proceeding of the International Black Sea Conference on Communications and Networking (Black-SeaCom)*, pp. 1–6, IEEE, Odessa, Ukraine, May 2020.
- [27] S. Qiu, D. Wang, G. Xu, and S. Kumari, “Practical and provably secure three-factor Authentication protocol based on extended chaotic-maps for mobile lightweight devices,” *IEEE Transactions on Dependable and Secure Computing*, 2020.
- [28] A. S. Ahanger, S. M. Khan, and F. Masoodi, “An effective intrusion detection system using supervised machine learning techniques,” in *Proceeding of the 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1639–1644, IEEE, Erode, India, April 2021.
- [29] P. Singh, J. Jayakumar, A. Pankaj, and R. Mitra, “Edge-detect: edge-centric network intrusion detection using deep neural network,” in *Proceeding of the 18th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, IEEE, Las Vegas, Nevada, January 2021.
- [30] D. Yuan, K. Ota, M. Dong et al., “Intrusion detection for smart home security based on data augmentation with edge computing,” in *Proceeding of the International Conference on Communications (ICC)*, pp. 1–6, IEEE, Dublin, Ireland, June 2020.
- [31] S. Revathi and A. Malathi, “A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection,” *International Journal of Engineering Research and Technology*, vol. 2, pp. 1848–1853, 2013.