Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Intrusion Detection for Wireless Edge Networks Based on Federated Learning

Zhuo Chen, Na Lv, Pengfei Liu, Yu Fang, Kun Chen, Wu Pan
Information and Navigation College, Air Force Engineering University, Xi'an 710077, China

Corresponding author: Na Lv (e-mail: lvnn2007@163.com).

This work was funded by the National Natural Science Foundation of China youth program, grant number 61703427.

ABSTRACT Edge computing provides off-load computing and application services close to end-users, greatly reducing cloud pressure and communication overhead. However, wireless edge networks still face the risk of network attacks. To ensure the security of wireless edge networks, we present Federated Learning-based Attention Gated Recurrent Unit (FedAGRU), an intrusion detection algorithm for wireless edge networks. FedAGRU differs from current centralized learning methods by updating universal learning models rather than directly sharing raw data among edge devices and a central server. We also apply the attention mechanism to increase the weight of important devices, by avoiding the upload of unimportant updates to the server, FedAGRU can greatly reduce communication overhead while ensuring learning convergence. Our experimental results show that, compared with other centralized learning algorithms, FedAGRU improves detection accuracy by approximately 8%. In addition, FedAGRU's communication cost is 70% less than other federated learning algorithms, and it exhibits strong robustness against poisoning attacks.

INDEX TERMS wireless edge, intrusion detection, federated learning, gated recurrent unit

I. INTRODUCTION

Nowadays, with the sharp increasement of traffic and number of users in wireless communication networks, the crowded backbone networks have suffered heavy workloads. The high latency and loss caused by the traditional cloud computing model can no longer meet the requirements of massive data processing. In response to this challenge, WENs (Wireless Edge Networks) are proposed [1]. A WEN migrates part of the network functions and data processing originally located in a core network to the edge of the network closer to the end-user.

WENs face several challenges with respect to network security and performance [2]. The wireless channel characteristics of WENs make the connections unstable and vulnerable to malicious intrusion. Therefore, accurate intrusion detection algorithms are required to ensure data security and network performance. Most of the existing intrusion detection algorithms are centralized learning methods; thus, in the process of uploading massive data to the server, model performance is easily affected by unstable and low-speed transmission channels. The authors of [3] also pointed out that common machine learning algorithms usually follow the assumption that the data are IID (independent and identically distributed) as part of their derivation. However, in actual networks (especially edge networks), the device-generated data are typically non-IID (non-independent and uniformly distributed). Training on non-IID data with existing machine learning methods leads to low model accuracy and model convergence failure [4].

This paper proposes a federated learning intrusion detection model called FedAGRU that adopts to wireless edge networks. FedAGRU uses the computing resources of edge devices and local data sets for model training and uploads model parameters to a server for collaborative training. Compared with traditional centralized learning methods, FedAGRU does not require the transmission of original data to a central server. Consequently, the risk of data leakage is reduced while ensuring model accuracy. The FedAGRU algorithm uses the attention mechanism to improve its overall convergence speed and communication efficiency. We use three real network data sets to evaluate our method on both IID and non-IID data samples. FedAGRU achieves better detection performance than the other tested centralized models and distributed models while greatly reducing communication costs. Moreover, FedAGRU has strong robustness, which can provide effective protection from poisoning attacks.

The remaining sections of the paper are organized as follows: Section II reviews the research literature on federated learning and intrusion detection, Section III describes the intrusion detection framework based on federated learning, Section IV describes the experimental results, and Section VII concludes the paper.

II. RELATED WORK

A. FEDERATED LEARNING

Due to data privacy issues, network bandwidth limitations, equipment availability, etc., it is unrealistic to collect all the data from the edge devices of a data center and conduct centralized training. Federated learning is emerging as a solution to these problems [5]. The author of [6] proved that federated learning also helps improve the accuracy of local participants' models

because the limited data owned by any participant can easily fall into a local optimum. Furthermore, the models trained by other participants can effectively help participants discard local optima to obtain a more accurate model.

In the application scenario of distributed intrusion detection, the data is independently generated by each device; thus, the local data set of any specific device cannot represent the overall distribution and approximate non-IID data. In [7], the Federated Averaging (FedAVG) algorithm was proposed, which combines local SGD (stochastic gradient descent) on each client with a server that performs model averaging. The idea of model averaging is similar to dropout training, i.e., an average model of the shared parameters of different clients. In the federated learning scenario, the main constraint is communication cost. Compared with synchronous stochastic gradient descent, the number of communication rounds required by the FedAVG algorithm is 10-100 fold reduced.

The convergence of the federated learning algorithm is always a problem. Based on FedAVG, the authors of [8] proposed a federated optimization framework for heterogeneous networks called FedProx. FedProx solves the statistical heterogeneity problem of different devices by setting a correction term to make the local model on a device closer to the global model. Theoretical analyses and experiments have proven that the model converges quickly, which helps federated learning participants save time and resources. In the LoAdaBoost-FedAVG algorithm [9], participants train a model with local data and compare the cross-entropy loss with the median loss from the previous round of training. If the current cross-entropy loss is high, the model is retrained prior to global model aggregation to improve learning efficiency. Simulation results have shown that LoAdaBoost-FedAvg converges quickly.

The authors of [10] proposed that data poisoning is one of the most relevant and emerging security threats in data-driven machine learning algorithms, especially with untrusted data, but few general defense mechanisms against poisoning attacks in the federated learning framework have been proposed.

B. INTRUSION DETECTION

In recent years, deep learning has been increasingly applied to wireless network intrusion detection. Yang et al. [11] proposed a wireless network intrusion detection method based on an improved convolutional neural network (ICNN), and the KDD CUP99 data set was used to verify the algorithm's effectiveness. In [12], the author proposed an intrusion detection model called LA-GRU based on a novel imbalanced learning method and gated recurrent unit (GRU) neural network. LA-GRU not only obtained excellent overall detection performance with a low false alarm rate, but also effectively solved the learning problem of imbalanced traffic distribution. Intrusion detection models based on machine learning require sufficient data to establish normal behavior as a baseline for abnormal behavior. However, it is infeasible to centralize all training and testing data for the following reasons: (1) resource constraints, (2) data security and privacy issues, and (3) high transmission delay.

Moreover, according to the traditional cloud-centric intrusion detection scheme, high latency makes real-time detection difficult, and network security measures come too late. To make full use of the characteristics and advantages of edge computing, using the computing resources at the edge of a network for collaborative intrusion detection has become a research focus. In [13], a cooperative distributed intrusion detection system was proposed where each node in the network looks for abnormalities in local network data and a distributed coordination mechanism is used to detect network attacks. However, the efficiency of these collaborative detection schemes is not high, and the false alarm rate is high [14]. The authors of [15] proposed a novel distributed GAN (generative adversarial network) architecture that provides an effective intrusion detection system by adapting a mechanism in which every IoT (Internet of Things) device monitors the neighbor IoT devices, the proposed distributed intrusion detection system does not require any sharing of datasets among the IoT devices, compared with the standalone GAN-based intrusion detection system, the distributed GAN-based intrusion detection system has a higher accuracy rate.

The authors of [16] applied federated learning technology to intrusion detection, and their training and testing accuracy on the CICIDS2017 data set was approximately 97%, which signifies improved efficiency and confidentiality of training. The authors of [17] proposed an intrusion detection method based on the long and short-term memory framework of federated learning with higher classification accuracy than traditional methods. The authors of [18] found that an edge network intrusion detection system based on federated learning was vulnerable to poisoned samples and had incorrectly classified malicious traffic as benign.

III. INTRUSION DETECTION MODEL

In this section, we first propose an intrusion detection model by a combined model with GRU (gated recurrent unit) and SVM (Support Vector Machine), then describe common federated learning algorithms, and finally propose the improved federated learning algorithm called FedAGRU. FedAGRU features an aggregation mechanism in which different clients have different weights. The server merges the weighted model parameters to speed up model convergence, which effectively prevents the uploading of parameters harmful to the overall model and reduces the communication overhead.

A. GRU-SVM

Federated learning algorithms are typically used in models based on SGD (Stochastic Gradient Descent), mainly DNNs (deep neural networks) [19]. However, DNNs cannot simulate changes in time series. Network traffic data is essentially time series data, and RNNs (recurrent neural networks) are typically used to process sequence data[20]. However, due to shortcomings such as gradient disappearance and gradient explosion, traditional RNNs have limitations for long-term prediction [21]. In contrast, GRU has a relatively simple structure, fewer parameters and a faster training speed. Therefore, GRU model as shown in Fig. 1 is chosen as our training model for federated learning:

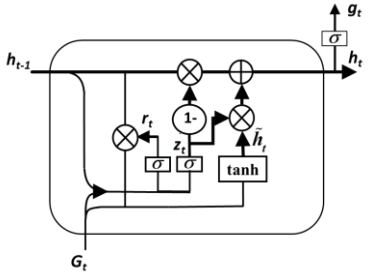


FIGURE 1. **GRU Model**

$$r_t = \sigma\left(w_r \cdot \left[h_{t-1}, X_t\right]\right) \quad (1)$$

$$z_t = \sigma\left(w_z \cdot \left[h_{t-1}, X_t\right]\right) \quad (2)$$

$$\tilde{h}_t = \tanh\left(w_h \cdot \left[r_t * h_{t-1}, X_t\right]\right) \quad (3)$$

$$h_t = \left(1 - z_t\right) * h_{t-1} + z_t * \tilde{h}_t \quad (4)$$

$$g_t = \sigma\left(w_o \cdot h_t\right) \quad (5)$$

where X_t is the input vector, σ is a sigmoid function, h is the output vector and $\tilde{h}$ is the candidate output. The subscript t represents the current time step, $t-1$ is the last time step, and w represents undetermined parameters. In Eqs. (1) and (2), r is denoted as a reset gate that determines how to combine new input information with previous memory, z is denoted as an update gate that defines how much of the previous information needs to be saved to the current time step, and g determines the final output in (5).

Deep learning classification models frequently use the Softmax activation function as their last neural network layer to obtain classification results and minimize cross-entropy loss. The Softmax function outputs the probability of a category. The authors of [22] proposed that the classification performance of a detection model would improve if a linear SVM (support vector machine) replaced Softmax in the final output layer of the GRU model. Thus, we use an SVM as the last layer of our model for classification, and use Hinge Loss as the objective function:

$$L = \min \frac{1}{2} \| w \|_2^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y_i\left(w x_i + b_i\right)\right)^2 \quad (6)$$

where y_i is the true label, x_k is the output of the GRU model and input to the SVM, w is the training weight, and C is a penalty parameter representing error tolerance. An excessive penalty coefficient leads to overfitting and poor model generalization ability; a small penalty coefficient leads to underfitting and increases the number of classification errors. The decision function generates a score vector for each class and returns the maximum value as the predicted class:

$$\text{label} = \text{argmax}(\text{sign}(w x_i + b_i)) \quad (7)$$

B. FEDERATED LEARNING FRAMEWORK

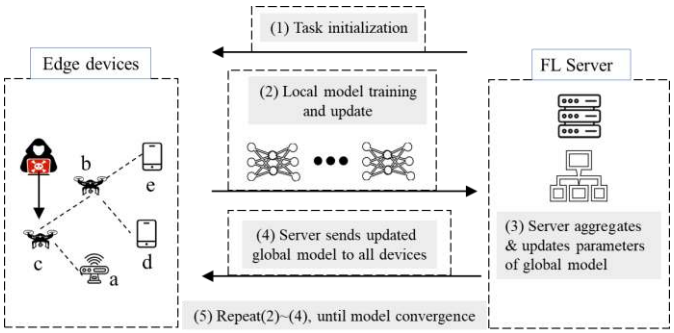


FIGURE 2. **Intrusion detection based on federated learning framework**

There are two main entities in a FL (federated learning) system, namely, the data owner (participant) and model owner (FL server). As shown in Fig. 2, edge devices use private data sets to train local intrusion detection models and only send local model parameters to the FL server (usually located at the cloud center); then, all the collected local models are aggregated to generate a global model. In this paper, the term "client" encompasses network entities such as devices, nodes, and sensors.

We assume that the local data set stored by client k is D_k of size $|D_k|$, the main steps of intrusion detection based on the federated learning framework are as follows:

Step 1 (task initialization): The server determines the training task of intrusion detection and specifies the global model and hyperparameters of the training process. The server then broadcasts the initialized global model parameters W_G^t to selected participants.

Step 2 (local model training and update): Based on the global model w_G^t, each client uses local private data to update local model parameters w_k^{t+1}. The goal of client k at iteration t is to find the optimal parameter that minimizes the loss function: $w_k^{t+1} = \arg\min L\left(w_k^t, D_k\right)$, so as to effectively detect network intrusions.

Step 3 (global model aggregation and update): All K clients collaboratively train a global prediction model. In the FedAVG algorithm, the server averages the client model parameters to obtain a new global model: $w_G^{t+1} = w_G^t + \sum_{k=1}^{K} \frac{n_k}{N} w_k^{t+1}$, where $N = \sum_{k=1}^{K} D_k$, and sends the updated global model parameters back to the data owner. Algorithm 1 gives a pseudo-code description of the FedAVG algorithm.

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; η is the leaning rate;

procedure GLOBALOPTIMIZATION:
  initialize w_o on public dataset
  for each round t = 1,2,... **do**
    m←max($C \cdot K$,1)
    S_t←(random set of m clients)
  for each client k$\in S_t$ in parallel **do**
    w_k^{t+1} = ClientUpdate(k, w_k^t)
    $w_G^{t+1} = w_G^t + \sum_{k=1}^{K} \frac{n_k}{N} w_k^{t+1}$

procedure ClientUpdate(k, w): //Run on selected client k
  B_k←(split local client data into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in B_k$ **do**
      $w = w - \eta \cdot L\left(w; b\right)$
  return w to server

The calculation volume of FedAVG is controlled by three key parameters: C, the proportion of clients performing calculations in each round; B, the local mini-batch size used for client updates; E, the number of training rounds for each client on local data. When $B = \infty$ (complete local data set), and E=1 corresponds to FedSGD (FederatedSGD) [7], FedSGD is the fastest update algorithm in FL. It only requires each local client to calculate the average gradient once before uploading it to the server, so it requires minimal computing resources. This explains why most studies use FedSGD as a baseline.

C. IMPROVED FEDERATED LEARNING ALGORITHM

Federated learning for edge networks requires interaction with multiple edge nodes. On the one hand, because the wireless connection between the client and server is typically slow or unreliable, the number of communication rounds between the client and server is minimized. On the other hand, the data set of any single client is smaller than the total data set, and the edge client has a relatively fast processor. Thus, the computational cost is often negligible compared to the communication cost, and the delay in communication is the performance bottleneck of the entire learning framework [23]. For this reason, our goal is to minimize the accumulated number of communication rounds while ensuring the convergence of the learning algorithm:

$$\begin{aligned} &\text{minimize} \sum_{t=1}^{T} C_t \\ &s.t. \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \left| f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^*\right) \right| = 0 \end{aligned} \tag{8}$$

where C_t is the number of clients uploading local updates to the server at iteration t.

The authors of [4] verified that several client-specific local optimizations are not closely related to the aggregation of collaborative training on the server. CMFL [24] proposed that, to avoid the transmission of irrelevant local updates, each client should be aware of the collaborative optimization trend in global aggregation. At each learning iteration, the client should compare local and global updates to determine if they are relevant. Assuming that $\boldsymbol{W} = \{w_1, w_2, ..., w_N\}$ is the local model parameter update, and $\overline{\boldsymbol{W}}$ is the global model parameter, CMFL calculates the percentage of the same symbolic parameter as the correlation measure:

$$e(\boldsymbol{W}, \overline{\boldsymbol{W}}) = \frac{1}{N} \sum_{i=1}^{N} I\left(\operatorname{sign}\left(\boldsymbol{W}_j\right) = \operatorname{sign}\left(\overline{\boldsymbol{W}_j}\right)\right) \tag{9}$$

where $I\left(\operatorname{sign}\left(\boldsymbol{W}_j\right) = \operatorname{sign}\left(\overline{\boldsymbol{W}_j}\right)\right)=1$ if $\boldsymbol{W}_j$ and $\overline{\boldsymbol{W}_j}$ are of the same sign, and 0 otherwise.

This paper proposes an improved federated learning method, FedAGRU, which uses the idea of the attention mechanism to calculate the importance of uploaded model parameters. The criterion of importance is based on improvement of the classification performance of the global model so that different clients are sorted according to their importance. When bandwidth resources are scarce, clients with high importance (the weight of upload parameters is high) can be preferentially allowed to upload over those with low importance, whose impact on the overall performance of the model is reduced. If a local model has an importance lower than a certain threshold and exceeds a set number of times, it is no longer calculated, and only the global model parameters are accepted for updating. This decreases the calculation cost and greatly reduces the communication cost while ensuring model convergence. The FedAGRU algorithm is described in Algorithm 2. It improves upon the FedAVG algorithm in several areas:

(1) Central server: As with the previous federated learning algorithm, we initialize and issue model parameters on the server, wait for client model parameters to be uploaded, and perform aggregation updates. Considering the wireless edge network environment, we propose an asynchronous update process for the server. After receiving a client update (multiple clients completing their local calculations at the same time results in multiple updates), the central server starts updating the model parameters w without waiting for other clients to complete their calculations.

Suppose that in round t, the server receives updates from s clients, the central server model is $\boldsymbol{w}(t)$, and $N' = |D_1| + |D_2| + ... + |D_s|$ is the total number of client data samples. We then calculate server updates by summarizing client updates as follows:

$$\boldsymbol{w}(t) = \sum_{k=1}^{s} \frac{|\mathcal{D}_k|}{N'} \boldsymbol{w}_k(t) \tag{10}$$

where $\boldsymbol{w}_k(t)$ is the local model parameter of client k at round t. In the update aggregation process, the attention mechanism is introduced to increase the weights of important update parameters.

(2) Parameter importance calculation: Attention mechanisms have a significant effect on feature representation [25]. Initially applied to image recognition, the attention mechanism can select the most relevant information in a large amount of information. Inspired by the attention mechanism and the CMFL algorithm, the importance of uploaded model parameters should be differentiated to increase the weights of parameters uploaded by the client that are beneficial to the model and optimize the performance of the global model. The FedAGRU algorithm uses the attention mechanism on a central server to assign different weights to the model parameters of different clients. The set of model parameter vectors output by k clients is expressed as $\boldsymbol{W} = [w_1, w_2, ..., w_k]$, and the attention mechanism generates a weighted output representation:

$$\boldsymbol{M} = \tanh(\boldsymbol{W}) \tag{11}$$

$$\boldsymbol{A} = \operatorname{softmax}\left(\boldsymbol{W}_m{}^T \boldsymbol{M}\right) \tag{12}$$

$$\boldsymbol{W}^* = \boldsymbol{H} \cdot \boldsymbol{A}^T \tag{13}$$

where $\boldsymbol{W}_m$ is a parameter matrix to be learned that is realized by the linear layer, and weight matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_k]$ represents the importance of the parameters of different clients. In addition, we combine weight normalization to reduce the calculation cost. The larger the weight, the more important the client is to overall intrusion detection performance. If the final obtained parameter weight of the client is less than the set threshold v, it does not participate in the parameter aggregation of the server. Assuming that the obtained model parameter set with parameter weights greater than threshold v is $\boldsymbol{W} = [w_1, w_2, ..., w_s]$, the final converged model parameter is:

$$\boldsymbol{W}_G = \sum_{i=1}^{s} \frac{|D_i|}{N^*} a_i w_i \tag{14}$$

where N^* is the total number of data samples of the final selected clients, $\boldsymbol{W}_G$ and weight matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_k]$ are broadcast to the selected participants.

(3) Local optimization on the client. Client-side optimization is obtained by using client data to perform local model training iterations. If the client importance is greater than threshold v, a local model update is calculated and uploaded. If the client importance is less than threshold v for multiple consecutive time steps, and the number of times exceeds a set tolerance threshold, the client is marked as "False", and the client no longer uses the local data set to calculate and upload parameters, but only

accepts global broadcasted model parameters for updating. This not only reduces communication costs, but also prevents irrelevant updates from depleting computing resources and affecting detection performance. This local optimization method also effectively prevents poisoning attacks.

Assuming $\boldsymbol{X}=\{x_1,x_2,\ldots,x_n\}$ is an input time series of the GRU-SVM model, and $\boldsymbol{Y}=\{y_1,y_2,\ldots,y_n\}$ is the true label set, the FedAGRU pseudocode is described in Algorithm 2.

Algorithm 2: FedAGRU. 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; η is the leaning rate; v is the importance threshold; T is the tolerance threshold

procedure GLOBALOPTIMIZATION:
According to X, Y and Eq.(1)-(7), initialize the cloud GRU-SVM model $f(\boldsymbol{W},\boldsymbol{X},\boldsymbol{Y})$, $\boldsymbol{W}_G^t(w_r,w_z,w_h,w_o)$, and $\boldsymbol{H}^t$;
for each round t = 1,2,... **do**
m←max(C·K ,1)
S_t←(random set of m clients)
for each client k∈S_t in parallel **do**
W_k^{t+1} = ClientUpdate(k, W_G^t)
Calculate the client importance $\boldsymbol{A}=[a_1,a_2,...,a_k]$ following Eq.(11)-(14)
obtain the global aggregation update
$$\boldsymbol{W}_G=\sum_{i=1}^{s}\frac{|D_i|}{N^*}a_i\boldsymbol{w}_i \quad \text{where} \quad a_k>v$$
Update parameter W_G^{t+1}, $\boldsymbol{H}^{t+1}$
procedure ClientUpdate(k, w): //Run on selected client k
B_k←(split local client data into batches of size B)
if $a_k<v$ **then** $c_k=c_k+1$
else c_k=0
if c_k>T **then**
client is marked as False, early stop
else
for each local epoch i from 1 to E **do**
for batch $\{x_i,y_i\}\in B_k$ **do**
$$w=w-\eta\cdot L(w;b)$$
return w to server

IV. EXPERIMENTS AND RESULTS

We combine the KDD CUP 99 data set [26], the CICIDS2017 data set provided by the Canadian Institute of Network Security [27] and the WSN-DS wireless network data set [28] to evaluate the model. Since each data set has different modes and characteristics such as network topology, traffic changes, attack methods, etc., using multiple data sets comprehensively verifies the adaptability of the model. Each data set is divided into 60%, 20%, and 20% subsets for training, verification, and testing, respectively. The training set is divided into 10 clients on average, and each client learns its own model and uploads model parameters to the server for aggregation. Similar to the data setting method in [7], for the IID setting, each client is assigned the uniform distribution of all labels. For non-IID settings, labeled data are divided into 20 partitions, and each client randomly allocates data from 2 partitions of the two categories, which allows us to study the stability of each algorithm on highly non-IID data.

To simulate the real federated learning scenario as much as possible, the model and data set are not located on a single high-performance machine [16]. In the experiment, the server is a desktop computer equipped with Core i9-9820x and GTX 1080 Ti, and the client is a laptop equipped with a Core I5-8300H CPU and a Raspberry Pi 3 equipped with an ARM Cortex A7 CPU. We use the PySyft federated learning framework [29] to build the experimental environment.

TABLE I
SUMMARY OF DATASET

Dataset	Types of network attacks
WSN-DS	Blackhole, Grayhole, Flooding, Scheduling
KDD CUP99	DoS, R2L, U2R, Probing
CICIDS2017	PortScan, Botnet, DoS/DDoS, Web Attack, Infiltration, Brute Force

A. MODEL ARCHITECTURE

In deep learning algorithms, the proper selection of hyperparameters, such as the number of hidden layers and hidden units in each hidden layer, greatly affects the performance of an algorithm. We studied the classification performance of FedAGRU under different hyperparameter configurations to determine the federated learning parameters of the FedAGRU algorithm: C(fraction of clients)=0.5, v(importance threshold)=0.05, T(tolerance threshold)=5. We also used the grid search method to find the best architecture of the GRU-SVM model in the FedAGRU algorithm. The final parameters are shown in Table II.

TABLE II
STRUCTURE OF GRU-SVM

Hyper-parameters	value
Hidden Layers	1
Hidden Units	128
Dropout Rate	0.8
SVM C	0.5
Learning Rate	1e-4

B. Comparison of different methods

We use Accuracy, FAR, and F1score to evaluate the performance of the centralized model and federated learning model on the CICIDS2017 data set.

For the centralized model, we select several popular methods in intrusion detection, GRU-SVM, GRU-softmax, ICNN and VAE, for comparison. These methods upload data to a server for centralized training. FedAVG is selected as a comparison method for federated learning. To better elucidate the advantages of the proposed algorithm, the training model of all the compared federated learning methods is GRU-SVM, and the specific hyperparameters are identical to those in Table II. The simulation results in Table III show that FedAGRU and the local model GRU-SVM have the highest detection accuracy and F1score as well as the lowest false alarm rate. In addition, their classification performance is 3%~8% higher than ICNN. This is because FedAGRU inherits the intrusion detection advantages of GRU-SVM, and the test accuracy of FedAGRU is approximately 1% higher than FedAVG. The experiment proves that the distributed learning of federated learning can achieve the training effect of the centralized model. We also found that the

distributed learning method has a performance advantage on non-IID data over centralized learning.

TABLE III
PERFORMANCE OF DIFFERENT METHODS

Method	IID			non-IID		
	Accuracy (↑)	FAR (↓)	F1score (↑)	Accuracy (↑)	FAR (↓)	F1score (↑)
GRU-SVM(local)	99.84%	0.42%	98.25%	98.84%	1.32%	97.65%
GRU-softmax(local)	98.94%	1.12%	97.96%	96.84%	3.15%	95.46%
ICNN(local)	95.84%	5.16%	94.55%	90.74%	9.89%	90.55%
VAE(local)	97.84%	3.28%	96.23%	95.84%	4.54%	94.23%
FedAVG(GRU-SVM)	99.84%	1.82%	97.44%	97.84%	2.08%	97.02%
FedAGRU	**99.28%**	**0.73%**	**98.12%**	**98.82%**	**1.43%**	**97.12%**

C. IMPACT OF FEDERATED PARAMETERS

The IID and non-IID samples of the KDD CUP99 data set are used to analyze the influence of the federated parameters B (batch size) and E (number of local epochs) on communication rounds. Table IV compares the communication rounds required for FedSGD and FedAGRU when the F1 score of the model reaches 98% under different B and E. We found that increasing the number of local epochs (E) and setting a smaller batch size (B) reduces the communication cost. The FedAGRU algorithm in this paper has the smallest communication round with small batch size (B) and local epochs (E)=20. Compared with the FedSGD baseline algorithm, FedAGRU's communication rounds are reduced 21 fold on IID sample data and 29 fold on non-IID sample data. Thus, the FedAGRU algorithm has higher communication efficiency than FedSGD.

TABLE IV
COMMUNICATION ROUND COMPARISON

Method	E	B	IID	non-IID
FedSGD	1	∞	238	560
FedAGRU	5	small	63(3.7×)	73(7.7×)
FedAGRU	10	small	26(9.2×)	36(15.6×)
FedAGRU	**20**	**small**	**11(21.6×)**	**19(29.4×)**
FedAGRU	5	large	78(3.1×)	89(6.3×)
FedAGRU	10	large	54(4.4×)	74(7.6×)
FedAGRU	20	large	39(6.1×)	57(9.8×)

D. MODEL PERFORMANCE & COMMUNICATION EFFICIENCY

We evaluate the learning speed of the federated learning algorithm through communication rounds, and compare the classification accuracy performance of the three federated learning algorithms, FedAGRU, CMFL and FedAVG, using the GRU-SVM model under different numbers of communication rounds. On the basis of FedAVG, the CMFL algorithm uses the correlation between the global and local model to limit the updating of upload parameters. The correlation threshold of the CMFL algorithm is {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, and we found it performs best at 0.7. In Fig. 3~5, we describe the relationship between the classification accuracy of the algorithms in each data set and the communication round. FedAGRU achieves the best performance and fastest learning speed under the same communication round. In other words, our method requires fewer communication rounds to achieve model convergence. On the IID and non-IID samples of the WSN-DS data set, the standard method FedAVG reaches a test accuracy of 90% by 21 and 32 communication rounds respectively, while FedAGRU requires only 9 and 22 communication rounds, respectively. Thus, our method reduces the number of communication rounds on the IID and non-IID samples by 57% and 31%, respectively. Similarly, on the CICIDS 2017 data set, FedAGRU achieves a communication round reduction of 69% and 41% on the IID and non-IID samples, respectively, compared to the standard method at 95% test accuracy. On the WSN-DS data set, FedAGRU achieves a 70% and 38% communication round reduction on the IID and non-IID samples, respectively, compared to the standard method at 95% test accuracy. Compared with the FedAVG algorithm, the CMFL algorithm improves the communication efficiency, but it is slightly weaker than the FedAGRU algorithm.

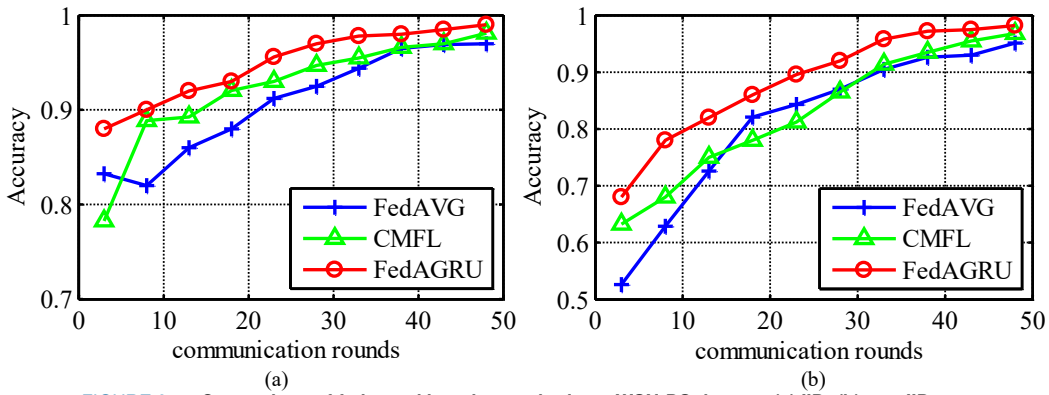


FIGURE 3. **Comparison of federated learning methods on WSN-DS dataset: (a) IID; (b) non-IID.**

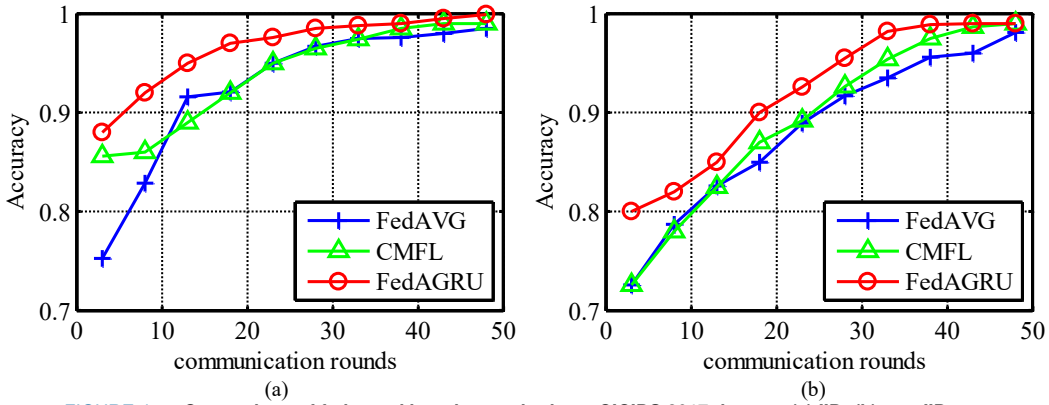


FIGURE 4. **Comparison of federated learning methods on CICIDS 2017 dataset: (a) IID; (b) non-IID.**

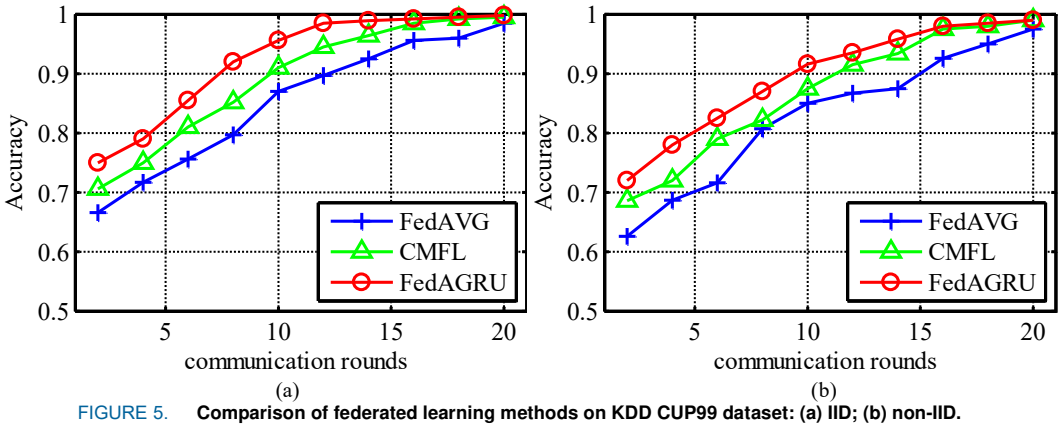


FIGURE 5. **Comparison of federated learning methods on KDD CUP99 dataset: (a) IID; (b) non-IID.**

E. PERFORMANCE COMPARISON OF FEDERATED LEARNING MODELS UNDER DIFFERENT CLIENT NUMBERS

In this experiment, we explored the influence of the number of clients on the federated learning models. Their classification accuracy is shown in Fig. 6 on the non-IID sample of the KDD CUP99 data set under identical simulation environments. An increase in the number of clients was found to adversely affect the performance of FedAVG and CMFL.

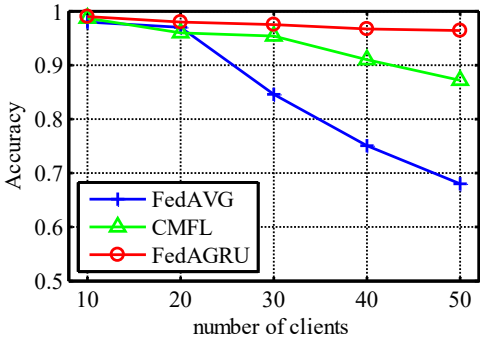


FIGURE 6. **Comparison of federated learning methods under different numbers of clients**

This degradation in performance is because the added clients bring more updated parameters, making it more difficult for the server to perform simultaneous model parameter aggregation. In addition, the model parameters of some clients may also adversely affect the global model. In addition, this overhead can affect the upload of important client parameters, reducing the accuracy of the global model. Meanwhile, FedAGRU determines client importance, which potentially reduces the number of participating clients. In this manner, updates that benefit model performance are aggregated, ensuring model training reliability.

F. POISONING ATTACKS

In a poisoning attack, an attacker injects poisoned samples into a training data set to maximize the error of the learning classifier [30]. Poisoning attacks are typically used to test model robustness [31-32]. The most common poisoning strategy for federated learning is label-flipping, under the assumption that an attacker can manipulate the labels of client training samples. The training sets of the WSN-DS and KDD CUP99 data sets are randomly and evenly divided into 20 small labeled data sets and assigned to 20 clients. Then, 3 of the data sets are randomly selected to simulate poisoning attacks, One-fifth of the labels of the scheduling class in the training samples are selected and modified to the normal class. The characteristics of the data remain unchanged, causing incorrect model classification. The left side of Fig. 7 and Fig. 8 shows model classification results after training with normal samples, and the right side shows those after training with poisoned samples.

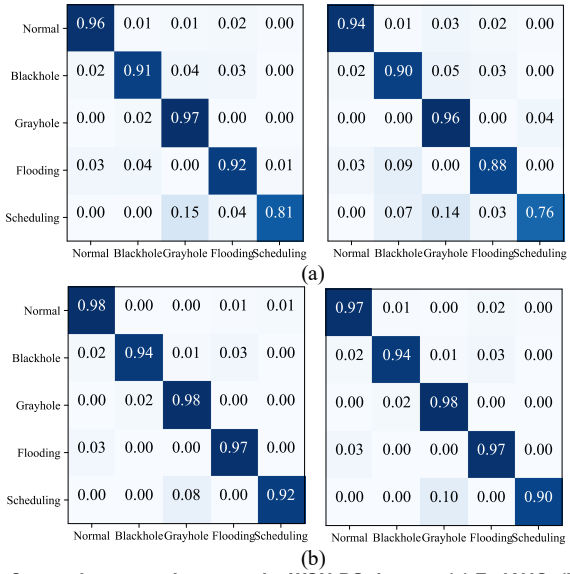


FIGURE 7. **Comparison experiment on the WSN-DS dataset: (a) FedAVG; (b) FedAGRU.**

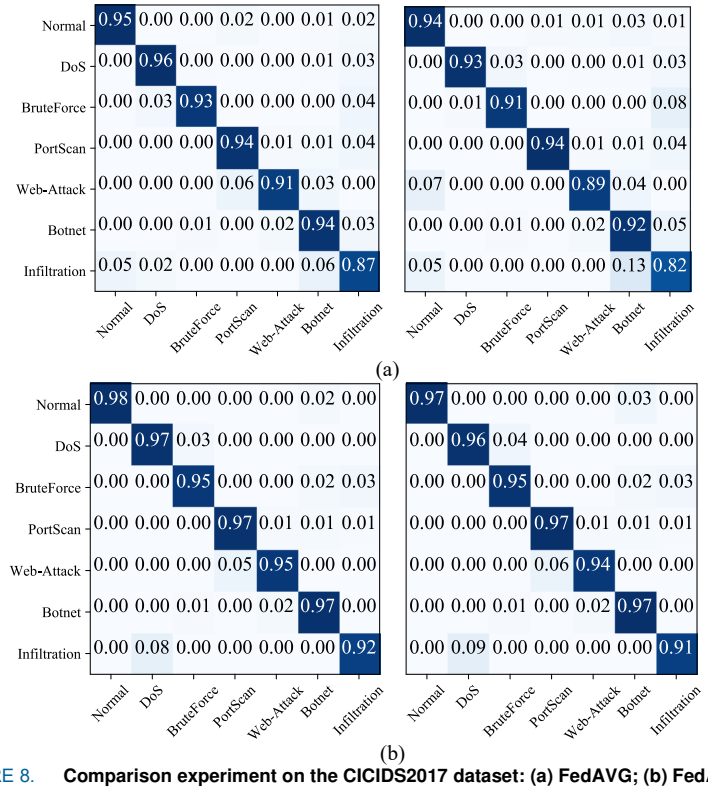


FIGURE 8. **Comparison experiment on the CICIDS2017 dataset: (a) FedAVG; (b) FedAGRU.**

The FedAVG algorithm is severely affected by the poisoning attack. As a result, in the WSN-DS data set, the recognition rates of Scheduling, Flooding, Blackhole, Grayhole and Normal Traffic decrease by 5%, 4%, 1%, 1%, and 2%, respectively. In the CICIDS2017 dataset, the recognition rate of Infiltration decreases by 5%, that of DoS decreases by 3%, and that of Brute Force, Botnet, and Web Attack decrease by 2%. The poisoning attack affects the recognition rate of all labels. However, FedAGRU calculates client importance and suppresses the impact of poisoned samples on global classification performance. The recognition rate of Scheduling in the WSN-DS data set decreases by 2%, that of normal traffic decreases by 1%, and the other labels are unaffected. The r recognition rate of each category in the CICIDS2017 data set decreases within 1%. Experiments have proved that FedAGRU has strong robustness and can provide effective protection against poisoning attacks.

VI. CONCLUSION

This paper proposes an improved federated learning algorithm called FedAGRU for intrusion detection in wireless edge networks. FedAGRU effectively prevents the upload of parameters that do not benefit the overall model and reduces communication overhead. In a comparison with other methods, FedAGRU had better accuracy, robustness and efficiency.

(1) The FedAGRU algorithm obtained higher detection accuracy with less communication overhead. Comparing the detection performance of the centralized model and the distributed model, we found that FedAGRU and GRU-SVM centralized learning have a higher detection accuracy and F1 score, a lower false alarm rate, and 3%~8% improved classification performance compared with ICNN. The distributed learning of federated learning also achieved the

training effect of the centralized model experimentally. Compared with FedAVG, the learning accuracy of the FedAGRU algorithm improved by approximately 1%, and the number of communication rounds was reduced by up to 70%.

(2) We explored the influence of federated learning parameters and the number of clients on the federated learning model. By setting the parameters properly, the communication cost could be further reduced, and the performance of FedAGRU remained stable even with a large number of clients.

(3) We simulated poisoning attacks, a new attack method against machine learning algorithm models. The FedAVG algorithm was severely affected by poisoning attacks, resulting in a decline in the recognition rate of each label. In contrast, the FedAGRU algorithm effectively inhibited the influence of poisoned samples and exhibited strong robustness.

In the future, we plan to study the safe aggregation of federated learning model parameters to further improve the reliability of our detection model.

REFERENCES

[1] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, "Proactive retention-aware caching with multi-path routing for wireless edge networks," *IEEE Journal on Selected Areas in Communications,* vol. 36, no. 6, pp. 1286-1299, 2018.

[2] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine,* vol. 54, no. 9, pp. 22-28, 2016.

[3] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582,* 2018.

[4] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems,* 2019.

[5] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials,* 2020.

[6] P. P. Liang, T. Liu, L. Ziyin, R. Salakhutdinov, and L.-P. Morency, "Think locally, act globally: Federated learning with local and global representations," *arXiv preprint arXiv:2001.01523,* 2020.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017: PMLR, pp. 1273-1282.

[8] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127,* 2018.

[9] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "Loadaboost: Loss-based adaboost federated machine learning on medical data," *arXiv preprint arXiv:1811.12629,* 2018.

[10] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning Attacks on Federated Learning-based IoT Intrusion Detection System," 2020.

[11] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," IEEE Access, vol. 7, pp. 64366-64374, 2019.

[12] B. Yan and G. Han, "LA-GRU: building combined intrusion detection model based on imbalanced learning and gated recurrent unit neural network," security and communication networks, vol. 2018, 2018.

[13] P. Albers, O. Camp, J.-M. Percher, B. Jouga, L. Me, and R. S. Puttini, "Security in Ad Hoc Networks: a General Intrusion Detection Architecture Enhancing Trust Based Approaches," in Wireless Information Systems, 2002, pp. 1-12.

[14] J. Sen, "An intrusion detection architecture for clustered wireless ad hoc networks," in 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks, 2010: IEEE, pp. 202-207.

[15] A. Ferdowsi and W. Saad, "Generative adversarial networks for distributed intrusion detection in the internet of things," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019: IEEE, pp. 1-6.

[16] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," Applied Sciences, vol. 8, no. 12, p. 2663, 2018.

[17] R. Zhao, Y. Yin, Y. Shi, and Z. Xue, "Intelligent intrusion detection based on federated learning aided long short-term memory," Physical Communication, p. 101157, 2020.

[18] Y. Zhao, J. Zhao, L. Jiang, R. Tan, and D. Niyato, "Mobile edge computing, blockchain and reputation-based crowdsourcing iot federated learning: A secure, decentralized and privacy-preserving system," arXiv preprint arXiv:1906.10893, 2019.

[19] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," IEEE Network, vol. 33, no. 5, pp. 156-165, 2019.

[20] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2018: Springer, pp. 149-158.

[21] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, "LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems," *arXiv preprint arXiv:1611.01726*, 2016.

[22] A. F. M. Agarap, "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," in Proceedings of the 2018 10th International Conference on Machine Learning and Computing, 2018, pp. 26-30.

[23] X. Lu, Y. Liao, P. Lio, and P. Hui, "Privacy-Preserving Asynchronous Federated Learning Mechanism for Edge Network Computing," IEEE Access, vol. 8, pp. 48970-48981, 2020.

[24] W. Luping, W. Wei, and L. Bo, "Cmfl: Mitigating communication overhead for federated learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019: IEEE, pp. 954-964.

[25] A. Vaswani et al., "Attention is all you need," in Advances in neural information processing systems, 2017, pp. 5998-6008.

[26] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, 2009: IEEE, pp. 1-6.

[27] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," International Journal of Engineering & Technology, vol. 7, no. 3.24, pp. 479-482, 2018.

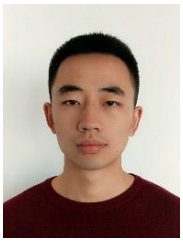
[28] I. Almomani, B. Al-Kasasbeh, and M. Al-Akhras, "WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks," Journal of Sensors, vol. 2016, p. 4731953, 2016/09/29 2016, doi: 10.1155/2016/4731953.

[29] T. Ryffel *et al.*, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017,* 2018.

[30] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389,* 2012.

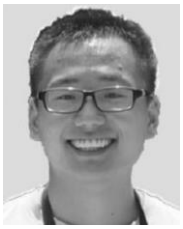
[31] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," IEEE Transactions on Communications, 2020.

[32] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in 29th {USENIX} Security Symposium ({USENIX} Security 20), 2020, pp. 1605-1622.

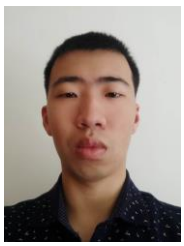


ZHUO CHEN received the B.S. degree in communication engineering from Yunnan University, Kun'ming, China, in 2018. He is currently pursuing the M.S. degree in Information and Navigation College, Air Force Engineering University, Xi'an, China. His research interests include communication network, machine learning and data mining.

NA LV received the B.S. degree in testing technology and instrumentation, the M.S. degree in control theory and applications, and the Ph.D. degree in armament science and technology from Northwestern Polytechnical University, Xi' an, China, in 1992, 1995, and 2010, respectively. She is currently a Full Professor with Air Force Engineering University, Xi' an. Her current research interests include aviation datalink system, military air communications and machine learning.

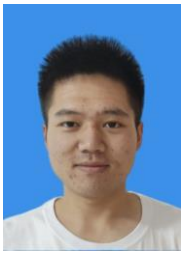


PENGFEI LIU received the B.S. degree from the Xi' an Jiaotong University, Xi' an, China, in 2012, and the M.S. degree from the National University of Defense Technology, Changsha, China, in 2014. He is currently pursuing the Ph.D. degree in the Air Force Engineering University. His research interests include airborne network, traffic classification, machine learning and aviation data link system.



YU FANG received the B.S. degree in Dalian University of Technology, Da' lian, China, in 2018. He is currently pursuing the M.S. degree in Information and Navigation College, Air Force Engineering University, Xi'an, China. His research interests include communication network, machine learning and data mining.

KUN CHEN received the B.S. degree in Information and Navigation College, Air Force Engineering University, Xi'an, China, in 2018. He is currently pursuing the M.S. degree in Information and Navigation College, Air Force Engineering University, Xi'an, China. His research interests include airborne tactical network, software-defined networking and deep learning.



WU PAN received the B.S. degree in Information and Navigation College, Air Force Engineering University, Xi'an, China, in 2018. He is currently pursuing the M.S. degree in Information and Navigation College, Air Force Engineering University, Xi'an, China. His research interests include wireless network and deep learning.