Intrusion Detection for Secure Social Internet of Things Based on Collaborative Edge Computing: A Generative Adversarial Network-Based Approach

Laisen Nie, *Member, IEEE*, Yixuan Wu, Xiaojie Wang, Lei Guo, *Member, IEEE*, Guoyin Wang, *Senior Member, IEEE*, Xinbo Gao, *Senior Member, IEEE*, and Shengtao Li

***Abstract*— The Social Internet of Things (SIoT) now penetrates our daily lives. As a strategy to alleviate the escalation of resource congestion, collaborative edge computing (CEC) has become a new paradigm for solving the needs of the Internet of Things (IoT). CEC can provide computing, storage, and network connection resources for remote devices. Because the edge network is closer to the connected devices, it involves a large amount of users' privacy. This also makes edge networks face more and more security issues, such as Denial-of-Service (DoS) attacks, unauthorized access, packet sniffing, and man-in-the-middle attacks. To combat these issues and enhance the security of edge networks, we propose a deep learning-based intrusion detection algorithm. Based on the generative adversarial network (GAN), we designed a powerful intrusion detection method. Our intrusion detection method includes three phases. First, we use the feature selection module to process the collaborative edge network traffic. Second, a deep learning architecture based on GAN is designed for intrusion detection aiming at a single attack. Finally, we propose a new intrusion detection model by combining several intrusion detection models that aim at a single attack. Intrusion detection aiming at multiple attacks is realized through the designed GAN-based deep learning architecture. Besides, we provide a comprehensive evaluation to verify the effectiveness of the proposed method.**

***Index Terms*— Collaborative edge computing (CEC), generative adversarial network (GAN), intrusion detection, social internet of things (SIoT).**

Manuscript received November 30, 2020; revised February 14, 2021; accepted February 20, 2021. Date of publication March 22, 2021; date of current version January 31, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFE0206800; in part by the National Natural Science Foundation of China under Grant 61936001, Grant 61971084, Grant 62001073, Grant 61803238, and Grant 61701406; in part by the National Natural Science Foundation of Chongqing under Grant cstc2019jcyj-cxttX0002; and in part by the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University under Grant CX2020153. *(Corresponding author: Xiaojie Wang.)*

Laisen Nie and Yixuan Wu are with the School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: nielaisen@nwpu.edu.cn; wuyixuan1@mail.nwpu.edu.cn).

Xiaojie Wang and Lei Guo are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: xiaojie.kara.wang@ieee.org; guolei@cqupt.eddu.cn).

Guoyin Wang is with the Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: wanggy@cqupt.edu.cn).

Xinbo Gao is with the Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: gaoxb@cqupt.edu.cn).

Shengtao Li is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China (e-mail: saintaolee@sdnu.edu.cn).

Digital Object Identifier 10.1109/TCSS.2021.3063538

I. Introduction

With the continuous development of the Internet, a large number of devices are connected to the Internet and communicated with each other in real time. The Social Internet of Things (SIoT) [1], [2], combining users' social behaviors and physical Internet of Things (IoT) [3], can provide ubiquitous Internet access for users. Because of the widespread applications of SIoT, millions of sensors and devices continue to generate data and exchange important information [4]. In order to alleviate the problem of resource congestion, more and more service providers choose collaborative edge computing (CEC) [5], [6], which migrates data computation and storage to the network edge near the users [7]. Therefore, various nodes distributed on the network can offload computation away from the centralized data center, which can significantly reduce the waiting time of the message exchange. Meanwhile, CEC can collaboratively handle tasks, such as computation offloading through social networks and connections among users. CEC is a crucial technique for SIoT based on edge computing, which provides connections for users with low latency, high bandwidth, and high reliability [8], [9]. For instance, it can support high-quality communications for vehicles to implement unmanned driving [10], [11] and intelligent transportation system [12], [13]. However, CEC migrates the users' private information to network edge from data centers. In this case, it is much easier for attackers to steal the privacy of users by taking advantage of the vulnerability of edge nodes [8], [14]. The general environment of SIoTs based on collaborative edge networks is shown in Fig. 1. Sensors can capture and transmit network data via long term evolution (LTE), WiMAX, Wi-Fi, and satellites for SIoT services.

To ensure the security of CEC, it is crucial to use an intrusion detection mechanism in SIoTs. Intrusion detection is a good way to prevent jumped-up attacks and protect users' privacy. Currently, many works have used deep learning techniques to implement intrusion detection accuracy. For instance, the work in [15] uses edge network traffic as training data and trains convolutional neural networks to implement intrusion detection. It uses an auxiliary classifier generative adversarial network (AC-GAN) to expand abnormal data. The work in [16] uses the auto-encoder (AE) to reduce the data dimensionality and then realizes intrusion detection through AE-AlexNet based on deep learning. Using deep learning algorithms to train huge data sets in the CEC environment

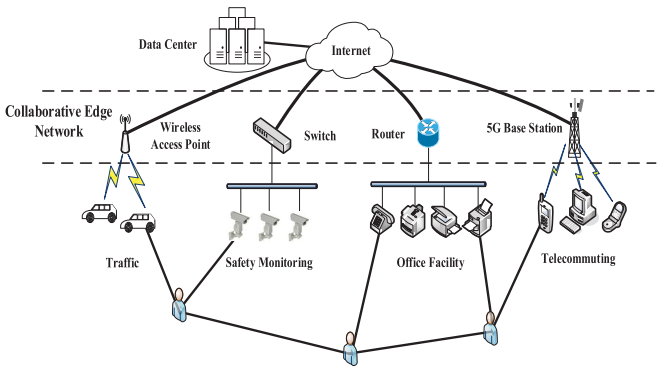


Fig. 1. Illustration of SIoTs based on CEC.

will make the overall process of computing more effective with low latency [17], [18].

Despite that intrusion detection technologies have been developing rapidly, existing intrusion detection algorithms for the security of CEC-based SIoT systems are still in their infancy. The main challenges of the current CEC-based SIoT system are summarized as follows [19]–[22].

1) In the edge network of SIoTs, a great number of network devices interact with each other, which makes the dimension of network data larger. Existing intrusion detection methods cannot achieve high accuracy when facing high-dimensional data.
2) In CEC-based SIoT, due to the complex network environment (e.g., interactions of human-to-human, human-to-thing, and thing-to-thing), our networks will be threatened by many types of attacks, such as brute force, Denial of Service (DoS), and Distributed Denial of Service (DDoS). Hence, it is difficult for existing intrusion detection methods to find various types of attacks from massive data.
3) Edge networks will often face a lot of new attacks. Existing intrusion detection technologies cannot accurately detect these attacks.

The unique and ubiquitous characteristics of CEC arise the main challenges for intrusion detection [23]. Motivated by that, we propose a generative adversarial network (GAN)-based mechanism to implement intrusion detection [24]. GAN consists of two networks, i.e., a discriminator and a generator. The task of the discriminator is to distinguish whether the data is extracted from the data set or generated by the generator. Besides, the task of the generator is to generate realistic data that the discriminator cannot distinguish whether the data are real. Over time, under careful supervision, two opponents compete with each other to successfully improve each other. The final result is that the well-trained generator can generate realistic results, and the well-trained discriminator can effectively distinguish between true and false. In deep learning, a large number of samples are needed to train the deep architecture. If the number of samples is too small, it often leads to insufficient network training. However, GAN can solve the problem of insufficient samples. The generator in GAN can theoretically generate data, which can greatly expand the diversity of samples.

Based on GAN, we designed a powerful intrusion detection method. Our intrusion detection method includes three phases. First, the collection of network data sets is mainly to collect various types of information generated via network nodes. After that, we carry out feature extraction from the original data set. Second, we designed an intrusion detection algorithm aiming at a single attack based on GAN. Finally, we optimized the intrusion detection models aiming at a single attack so that they can detect various attacks at the same time.

The main contributions of our work are as follows.

1) We designed a novel intrusion detection method based on GAN. The GAN-based method can extract low-dimensional features from original network flows for feature learning.
2) Existing intrusion detection does not have high accuracy for detecting multiple attacks. We design a GAN-based algorithm that can be used to capture different types of attacks with high accuracy.
3) We design a multiple intrusion detection algorithms based on GAN. The designed algorithm can detect a variety of new types of attacks by training and learning existing attack types.

The following sections of this article are organized as follows. In Section II, we briefly introduce the existing intrusion detection methods. In Section III, we propose the approach put forward in this article. Besides, in Section IV, we use the CIC-DDoS2019 and CSE-CIC-IDS2018 data sets to evaluate the proposed method. Finally, we summarize our research and future works in Section V.

II. Related Work

Currently, many works apply machine learning techniques to address network security issues. For instance, Yang *et al.* [25], [26] proposed a deep post-decision state and prioritized experience replay schemes that can improve the communication systems. However, the application of intrusion detection technology is more extensive.

A. Traditional Intrusion Detection Algorithms

As a novel network security technique, intrusion detection is a crucial defense solution for our networks behind the firewall. Intrusion detection systems collect network traffic, security logs, and other data sets at first. Through analyzing these measures, intrusion detection systems can detect whether the network is intruded or not.

The concept of intrusion detection was first proposed by Anderson in [27]. Denning first proposed the intrusion detection expert system [28]. Furthermore, Conforti *et al.* [29] proposed a network-based intrusion detection system. Moustafa *et al.* [30] used machine learning methods for intrusion detection for the first time. After that, a great number of works using machine learning for intrusion detection had appeared, which can achieve very good results. Ambusaidi *et al.* [31] used the deep belief network for intrusion detection, and they verified the feasibility of deep learning in this field.

B. Deep Learning-Based Intrusion Detection

With the rapid development of deep learning, a set of neural network architectures has been proposed to solve the intrusion detection problem. For instance, Garg *et al.* [32]

proposed the Improved-Gray Wolf Optimization algorithm (Im-GWO) and the Improved-Convolution Neural Network (Im-CNN). Due to the complexity of cloud data and the huge amount of data, by combining Im-GWO and Im-CNN, they proposed a hybrid data processing model for intrusion detection. In this model, the Im-GWO algorithm was used to extract data features, and a certain tradeoff was made between the exploration of unknown features and the usage of known features. Then, the Im-CNN algorithm used the extracted features to implement intrusion detection. Besides, Saharkhizan *et al.* [33] proposed a method that integrated an ensemble of long short-term memory (LSTM) models to construct the detector to implement enhanced robustness. Specifically, they integrated a set of LSTM models. After that, they merged the decision tree method and the LSTM model to achieve intrusion detection. Unlike the previous methods, Tian *et al.* [17] took a different approach by using a distributed deep learning system to detect Web attacks. Multiple deep learning models were trained at the same time, and they were deployed in different servers. By using multiple parallel systems, the stability of this system can be enhanced.

At the same time, in terms of feature extraction using deep learning, Shone *et al.* [34] proposed an intrusion detection method that used the S-NDAE method for feature extraction and passed the extracted features to the random forest for classification. The algorithm has two phases, i.e., encoder and decoder. To express high-dimensional data in low dimensions, the decoder process reexpresses low-dimensional data in high dimensions. The nonsymmetric deep AE that they proposed has a certain effect on intrusion detection. Meanwhile, through the meta-learning framework, Xu *et al.* [35] proposed a new deep learning method based on feature extraction. The method that they proposed was to realize intrusion detection by distinguishing normal network traffic and abnormal network traffic. They designed a deep neural network called feature extraction and comparison network (FC-Net), which consists of a feature extraction network and a comparison network. FC-Net can realize network traffic intrusion detection by extracting network features.

As mentioned above, although many techniques are proposed for intrusion detection, there are few intrusion detection techniques aiming at CEC-based SIoTs. Using deep learning algorithms to train huge data sets in collaborative edge networks can greatly improve the efficiency of the computing process. Therefore, to solve the security problems of collaborative edge networks, this article takes advantage of GAN to realize intrusion detection. For CEC-based SIoTs, real-time intrusion detection is necessary. Thereby, GAN is used in this article to deal with the problem of insufficient data volume, which can improve the complexity of intrusion detection algorithm.

III. Our Methodology

Our intrusion detection method includes three phases. In detail, as shown in Fig. 2, we first use the feature selection model to preprocess the collaborative edge network data, and we select the required features for intrusion detection. Second, we designed an intrusion detection algorithm aiming at a single attack based on GAN. After that, by combining several intrusion detection models aiming at a single attack, we design an intrusion detection algorithm aiming at multiple attacks based on GAN.

A. Feature Selection

The collection of network data is mainly to record various types of information generated by network nodes. Then, the collected initial information is encapsulated in a standard format after analysis. After that, we extract features from the original data sets. In our method, we use the CICFlowMeter tool to process the data set [36]. CICFlowMeter is a network traffic generator written by Java, and it provides greater flexibility in selecting features to be calculated. CICFlowMeter can extract features of original data, such as quantity, the number of bytes, and packet length [20]. The output of the CICFlowMeter consists of more than 80 network traffic features, such as destination port, protocol, flow duration, the total number of packets in the forward direction, the number of packets per second of traffic flows, and the average size of the packet. A feature should be a unique characteristic of a data packet, through which the attack data can be found without affecting the normal network traffic.

In view of the increasing amount of data to be processed, preprocessing feature identification can be used before classification. The purpose of feature selection is to reduce the number of training and ensure the accuracy of training by extracting important features. Therefore, in this article, we use an ensemble-based multi-filter feature selection (EMFFS) method [37], which combines information gain (IG) to select important features. The feature selection method is a preprocessing phase toward selecting important features from a data set, and it is independent of the classification algorithm. The feature analysis method is based on the internal statistical test of the original training data set, and it takes the feature analysis scheme as the main standard for feature analysis by sorting. Then, the method selects features based on a specific threshold. As shown in Fig. 2, after receiving new flow, our method first preprocesses the flow, and it performs feature extraction. Then, we select specific features. Finally, the preprocessed features are input into the intrusion detection module.

B. Intrusion Detection Based on GAN Aiming at Single Attack

We use GAN to implement intrusion detection by using network traffic features of SIoTs. The main idea of GAN comes from the Nash equilibrium. In a game model with two parties, it is composed of a generator and a discriminator. The generator will capture the potential distribution of real data samples, and then, it generates novel data samples. Meanwhile, the discriminator is two classifiers that distinguish the input data from the generated data. In order to win the competition, two-game players need to continuously optimize and improve their own generation and discrimination abilities. This learning optimization process is to find a Nash equilibrium between

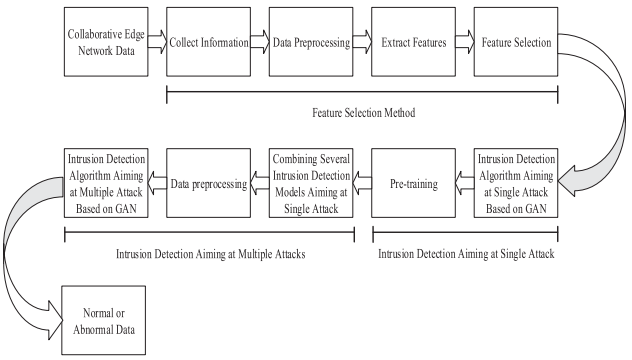


Fig. 2. Illustration of our method.

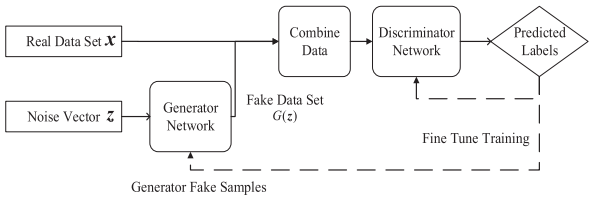


Fig. 3. GAN.

TABLE I
NOTATIONS

Symbol	Meaning
E	Expectation
D	Differential functions represented by a multilayer perceptron used for discriminator aiming at single attack
G	Differential functions represented by a multilayer perceptron used for generator aiming at single attack
$p_{data}(\boldsymbol{x})$	Input data
$p_{\boldsymbol{z}}(\boldsymbol{z})$	Noise variables
$\boldsymbol{x}$	Input vector
$\boldsymbol{z}$	Noise vector
$V(G, D)$	Value function
l	Represent the number of samples of $\boldsymbol{z}$
L	Represent the number of samples of and $p_{\boldsymbol{z}}(\boldsymbol{z})$
j	Represent the number of features
r	Represents the number of features of input data $\boldsymbol{x}$
$\boldsymbol{x_z}$	Fake data generated by a generator aiming at single attack
$\boldsymbol{x_D}$	$\boldsymbol{x_D}$ is composed of $\boldsymbol{x}$ and $\boldsymbol{x_z}$
$p_{data}(\boldsymbol{x})'$	Input data set only contains an attack
p	Represent the number of samples of $\boldsymbol{x}$
P	Represent the number of samples of $p_{data}(\boldsymbol{x})'$
$\boldsymbol{y_D}$	The discriminator aiming at single attack generates data
F	Accuracy function

two players. GAN is a two-player zero-sum game. The sum of the interests of both players is a constant [24]. Some of the notations used in this section are summarized in Table I.

GAN uses adversarial methods to generate data, and the basic illustration is shown in Fig. 3. Generating an adversarial network is playing a fighting game. The learning process is to constantly find other opponents to fight against and accumulate experience in the confrontation in order to improve your skills. GAN can be described as follows:

$$\min_G \max_D V(G, D) = E_{\boldsymbol{x}\in p_{\text{data}}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))] + E_{z\in p_z(z)}[\log(1 - D(G(z))] \quad (1)$$

where E denotes the expectation. D and G are both differential functions represented by a multilayer perceptron. D and G represent a discriminator and a generator, respectively. $p_{\text{data}}(\boldsymbol{x})$ and $p_z(z)$ represent input data and noise variables, respectively. Accordingly, $\boldsymbol{x}$ and $\boldsymbol{z}$ are input and noise vectors, respectively. Meanwhile, $V(G, D)$ represents the value function.

According to the loss functions of discriminant and generative models, the back-propagation (BP) algorithm can be used to update the parameters of GAN. The parameters of GAN can be calculated effectively through the gradient BP algorithm. Therefore, we use the method based on GAN to handle the problem of intrusion detection.

As mentioned above, we use the CICFlowMeter tool to process the original data set, and then, we preprocess the data to get the training data $p_{\text{data}}(\boldsymbol{x})$ we need. Then, we use this data set to train the proposed model.

The generator G selects the example z from random noise $p_z(z)$. Hence, $G(z)$ denotes the fake data generated by the generator. G is the differential functions represented by a multilayer perceptron.

Meanwhile, the discriminator D recognizes the example $\boldsymbol{x}$ from the input $p_{\text{data}}(\boldsymbol{x})$, and then

$$E_{\boldsymbol{x}\in p_{\text{data}}(\boldsymbol{x})} \log(F(D(\boldsymbol{x}))). \quad (2)$$

$F(D(\boldsymbol{x}))$ is the output result of the discriminator model, which is a real value in the range of $0 - 1$. It is used to judge the probability of data accuracy. Maximizing 2 means that D can predict normal values accurately, i.e., $F(D(\boldsymbol{x})) = 1$ when $\boldsymbol{x} \in p_{\text{data}}(\boldsymbol{x})$.

Then, we verify the data generated by the generator, which is

$$E_{z\in p_z(z)} \log(1 - F(D(G(z)))). \quad (3)$$

Maximizing 3 means that $F(D(G(z))) \approx 0$, and so G cannot generate excellent fraud data. The purpose of the generator is to generate data that can fool the discriminator.

During network training, the objective function of the discriminator can be defined as

$$\max_D E_{z\in p_z(z)}[\log(1 - D(G(z))] + E_{\boldsymbol{x}\in p_{\text{data}}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))]. \quad (4)$$

It can be seen that the purpose of the objective function is to find the discriminator function D that can maximize the sum of the following two expressions. Hence, we can define the value function as

$$V(G, D) = E_{\boldsymbol{x}\in p_{\text{data}}(\boldsymbol{x})}[\log(F(D(\boldsymbol{x})))] + E_{z\in p_z(z)}[\log(1 - F(D(G(z))))]. \quad (5)$$

When G remains unchanged, then $D_G^* = \arg\max_D V(G, D)$ can be expressed as the best discriminator. Correspondingly, when D is constant, to obtain the optimal generator G, G should satisfy $G_D^* = \arg\min_G V(G, D_G^*)$. After a certain amount of training, we can get the desired optimal discriminator D_G^*, which satisfies $D_G^* = \arg\max_D V(G_D^*, D)$.

The deep feedforward network is so-called multilayer perceptron, which is a classical deep-learning method. The destination of a multilayer perceptron is to approximate a certain function f^*. For instance, classifier $\boldsymbol{y} = f^*(\boldsymbol{x})$ maps $\boldsymbol{x}$,

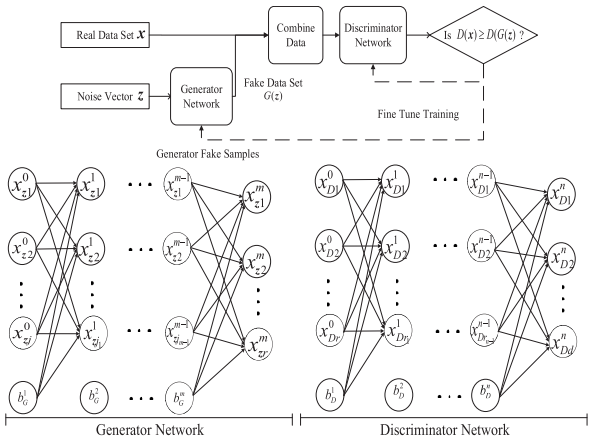


Fig. 4. Intrusion detection structure diagram aiming at single attack.

which is the input data to a certain category $\boldsymbol{y}$. The multilayer perceptron defines a mapping $\boldsymbol{y} = f^*(\boldsymbol{x}; \boldsymbol{\theta})$ and learns the value of the parameter $\boldsymbol{\theta}$ so that it can get the best function approximation.

GAN is composed of a generator and a discriminator. The generator and discriminator are composed of several hidden layers, respectively. The output of each layer is used as the input of the subsequent layer. Fig. 4 shows the architecture of intrusion detection aiming at an attack based on GAN for SIoTs. Here, the superscript number indicates the identity of the hidden layer, and the subscript number indicates the size of the layer.

In the generator phase, the generator G selects the example $\boldsymbol{z}$ from random noise $p_z(z)$, where $\boldsymbol{z}$ and $p_z(z)$ are $j \times l$ and $j \times L$ matrices. Meanwhile, l and L represent the number of samples of $\boldsymbol{z}$ and $p_z(z)$, respectively. Both $\boldsymbol{z}$ and $p_z(z)$ have j features. Then, the generator G generates fake data $\boldsymbol{x}_z$, where $\boldsymbol{x}_z = G(z)$. It is obvious that $\boldsymbol{x}_z$ is a $r \times l$ matrix, and r represents the number of features of the input data $\boldsymbol{x}$. In the generator, the kth layer of the generator can train itself by using the result of the $(k-1)$th layer. Hence, the generator can be expressed as

$$\boldsymbol{x}_z^k = \mathrm{g}^k(\boldsymbol{w}_G^k \boldsymbol{x}_z^{k-1} + \boldsymbol{b}_G^k), k = 1, \ldots, m \tag{6}$$

where $\boldsymbol{x}_z^0$ represents the input data $\boldsymbol{z}$. $\boldsymbol{x}_z^k$ denotes the output of the kth layer, and $\boldsymbol{x}_z^m$ represents the output data $\boldsymbol{x}_z$. $\boldsymbol{x}_z$ is the fraud data generated by the generator. Meanwhile, m is the number of layers. We use $\boldsymbol{\theta}_G^k = \{\boldsymbol{w}_G^k, \boldsymbol{b}_G^k\}$ to represent the parameters corresponding to the kth layer of the generator. The notation g^k represents the operation of the rectified linear unit (ReLU) function or the tanh function, and the last layer g^m represents the tanh function.

For the generator model, we choose mean square error (MSE) as the loss function. In our method, MSE for anomaly based intrusion detection in CEC-based SIoTs can be defined as

$$\mathrm{MSE}(\boldsymbol{q}, \boldsymbol{q}') = \frac{\sum_{k=1}^{p} (\boldsymbol{q}_k - \boldsymbol{q}'_k)^2}{p} \tag{7}$$

where $\boldsymbol{q}'_k$ is the correct answer of the kth data in a batch, and $\boldsymbol{q}$ is the predicted value given by the neural network. MSE is a function to find the average error of a batch. p represents the number of samples. We use BP and adaptive moment estimation (ADAM) optimizer to determine the minimum loss value of the generator network so that the corresponding learning parameters can be derived and optimized model.

In the discriminator phase, the input of the discriminator D is the combined data $\boldsymbol{x}_D$, where $\boldsymbol{x}_D$ is composed of $\boldsymbol{x}$ and $\boldsymbol{x}_z$. $\boldsymbol{x}_D$ is a $r \times (l + p)$ matrix, and $\boldsymbol{x}$ is a $r \times p$ matrix. Here, $\boldsymbol{x}$ is a sampling from the input data set $p_{\mathrm{data}}(\boldsymbol{x})'$, where the input data $p_{\mathrm{data}}(\boldsymbol{x})'$ are preprocessed. The input data set only contains an attack. $p_{\mathrm{data}}(\boldsymbol{x})'$ is a $r \times P$ matrix, where p and P represent the number of samples of $\boldsymbol{x}$ and $p_{\mathrm{data}}(\boldsymbol{x})'$, respectively. Then, the discriminator generates data $\boldsymbol{y}_D$, where $\boldsymbol{y}_D = D(\boldsymbol{x}_D)$ and $\boldsymbol{y}_D$ is a $2 \times (l + p)$ matrix. Here, $\boldsymbol{y}_z = D(\boldsymbol{x}_z)$ and $\boldsymbol{y} = D(\boldsymbol{x})$. $\boldsymbol{y}_D$ represents the abnormal situation of the overall combined data, and $\boldsymbol{y}$ is the abnormal situation of the real input data. Meanwhile, $\boldsymbol{y}_z$ represents the abnormal situation of the generated data.

In the discriminator, the kth layer of the discriminator can train itself by using the result of the $(k-1)$th layer. Hence, the discriminator can be expressed as

$$\boldsymbol{x}_D^k = f^k(\boldsymbol{w}_D^k \boldsymbol{x}_D^{k-1} + \boldsymbol{b}_D^k), k = 1, \ldots, n \tag{8}$$

where $\boldsymbol{x}_D^0$ represents the input data $\boldsymbol{x}_D$, and $\boldsymbol{x}_D^k$ denotes the output of the kth layer. $\boldsymbol{x}_D^n$ represents the output data $\boldsymbol{y}_D$, where n is the number of the discriminator layers. We use $\boldsymbol{\theta}_D^k = \{\boldsymbol{w}_D^k, \boldsymbol{b}_D^k\}$ to represent the parameters corresponding to the kth layer discriminator. The notation f^k represents the operation of the tanh function, and the last layer f^n represents the softmax function.

In the discriminator phase, we use the cross-entropy loss function. In the case of two classifications, the cross-entropy loss function can optimize the model very well. By minimizing the loss function, the model can reach a state of convergence and reduce the error of the model's predicted value. The cross-entropy loss function can be described as follows:

$$L(\boldsymbol{q}, \boldsymbol{q}') = \frac{1}{p} \sum_{k}^{p} -[\boldsymbol{q}'_k \log(\boldsymbol{q}_k) + (1 - \boldsymbol{q}'_k) \log(1 - \boldsymbol{q}_k)] \tag{9}$$

where $\boldsymbol{q}'$ is the correct answer, and $\boldsymbol{q}$ is the predicted value given by the neural network. p represents the number of samples. The cross-entropy loss function is used as a loss function to determine the minimum loss value of the discriminator network. Similarly, we also use BP and ADAM to update the discriminator.

Before entering the judgment phase, we need to process the data. To judge the abnormal situation directly, the processing conditions are as follows:

$$Y(\boldsymbol{y}_D) = \begin{cases} y_{1k} = 0, y_{2k} = 1 & if \ y_{1k} < y_{2k} \\ y_{1k} = 1, y_{2k} = 0 & if \ y_{1k} \geq y_{2k} \end{cases} \tag{10}$$

where y_{1k} and y_{2k} are the discrimination results of the kth data, respectively. In the judgment phase, we define the accuracy function F, which can reflect the accuracy of the discriminator and the effect of the generator. The accuracy function F can

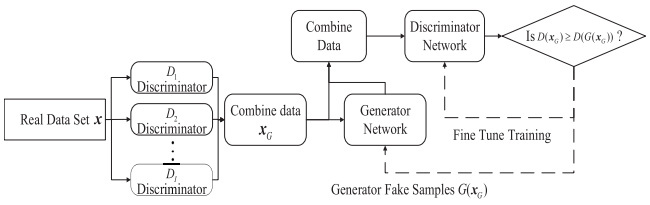


Fig. 5. Intrusion detection for various attacks.

be shown by

$$F = \frac{TP + TN}{TP + FN + FP + TN}. \tag{11}$$

The four parameters are described as follows.

1) TP—The forecast is a positive example, and the actual is a positive example.
2) FP—The forecast is a positive example, and the actual is a negative example.
3) TN—The forecast is a negative example, and the actual is a negative example.
4) FN—The forecast is a negative example, and the actual is a positive example.

Finally, we train the network by judging the size of $F(Y(\boldsymbol{y}), Y_y)$ and $F(Y(\boldsymbol{y}_z), Y_z)$, where Y_y and Y_z are the actual label of the real data set and the actual label of the generated data, respectively. If $F(Y(\boldsymbol{y}), Y_y)$ is greater than $F(Y(\boldsymbol{y}_z), Y_z)$, we train the generator; otherwise, we train the discriminator. The details of our intrusion detectiom method for an attack are shown in Algorithm 1.

C. Intrusion Detection Aiming at Multiple Attacks

Our intrusion detection method aiming at multiple attacks is a combination of several GAN model discriminators. We train I discriminators in advance, and the discriminators $D_1, D_2, \ldots, D_I$ come from the intrusion detection models aiming at an attack. The types of attacks trained by these models are different. The specific model of intrusion detection aiming at multiple attacks is shown in Fig. 5. We adjust the number of discriminators from intrusion detection aiming at an attack according to the number of attack types. The destination of network is to obtain the optimal discriminator $D_G^* = \arg\max_D V(G_D^*, D)$.

In the generator phase of intrusion detection aiming at multiple attacks, we select the input sample $\boldsymbol{x}$ from the input data set $p_{\text{data}}(\boldsymbol{x})$, where the input data $p_{\text{data}}(\boldsymbol{x})$ are preprocessed, and it contains various attacks types. $p_{\text{data}}(\boldsymbol{x})$ also is a $r \times P'$ matrix. After that, the input $\boldsymbol{x}$ will first be processed by discriminators $D_1, D_2, \ldots, D_I$. Therefore, for the input $\boldsymbol{x}$, it will pass through I discriminators to form the feature quantity $\boldsymbol{x}_i$, where $i = 1, 2, \ldots, I$. Then, we combine I feature quantities into a new feature matrix $\boldsymbol{x}_G$, where $\boldsymbol{x}_G$ is a $R \times p$ matrix, and $R = 2I$. Thus, $\boldsymbol{x}_i$ and $\boldsymbol{x}_G$ can be denoted by

$$\begin{cases} \boldsymbol{x}_i = D_i(\boldsymbol{x}), & i = 1, 2, 3, \ldots, I \\ \boldsymbol{x}_G = [\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_I] \end{cases}. \tag{12}$$

The generated feature matrix $\boldsymbol{x}_G$ will pass the generator G' to generate the fake data $\boldsymbol{x'}_z$ that can deceive the discriminator,

Algorithm 1 Intrusion Detection Network Training Aiming at Single Attack

Require: input data $p_{\text{data}}(\boldsymbol{x})'$; number of iterations T; real data label Y_y; corresponding label of generate data Y_z
Ensure: generator G; discriminator D

$p_z(\boldsymbol{z})$ is randomly generated L numbers from 1 to 0
$t \leftarrow 1$
while $t \leq T$ **do**
  Samples minibatch of l noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(l)}\}$ from noise prior $p_z(\boldsymbol{z})$
  $\boldsymbol{x}_z = G(\boldsymbol{z})$
  Samples minibatch of p examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(p)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})'$
  $\boldsymbol{x}_D = [\boldsymbol{x}, \boldsymbol{x}_z]$
  $\boldsymbol{y}_D = D(\boldsymbol{x}_D)$
  $Y(\boldsymbol{y}_D) = \begin{cases} y_{1k} = 0, y_{2k} = 1 & if\ y_{1k} < y_{2k} \\ y_{1k} = 1, y_{2k} = 0 & if\ y_{1k} \geq y_{2k} \end{cases}$
  $\boldsymbol{y}_D = [\boldsymbol{y}, \boldsymbol{y}_z]$
  $F = \frac{TP+TN}{TP+FN+FP+TN}$
  if $F(Y(\boldsymbol{y}), Y_y) > F(Y(\boldsymbol{y}_z), Y_z)$ **then**
    Generator update: The mean squared error is used as a loss function to determine the minimum loss value of the discriminator network. And the network uses BP and Adam to optimize.
  else $\{F(Y(\boldsymbol{y}), Y_y) \leq F(Y(\boldsymbol{y}_z), Y_z)\}$
    Discriminator update: The cross entropy loss function is used as a loss function to determine the minimum loss value of the discriminator network. And the network uses BP and Adam to optimize.
  end if
end while
return discriminator D

where $\boldsymbol{x'}_z = G'(\boldsymbol{x}_G)$. Thereby, the generator can be expressed as

$$\boldsymbol{x'}^k_z = \mathrm{g}'^k(\boldsymbol{w'}^k_G \boldsymbol{x'}^{k-1}_z + \boldsymbol{b'}^k_G), k = 1, \ldots, m \tag{13}$$

where $\boldsymbol{x'}^0_z$ represents the input data $\boldsymbol{x}_G$, and $\boldsymbol{x'}^k_z$ denotes the output of the kth layer. Meanwhile, $\boldsymbol{x'}^m_z$ represents the output data $\boldsymbol{x'}_z$, where m is the number of layers. We use $\boldsymbol{\theta'}^k_G = \{\boldsymbol{w'}^k_G, \boldsymbol{b'}^k_G\}$ to represent the parameters corresponding to the kth layer generator. The notation g'^k represents the operation of the ReLU function or the tanh function, and the last layer g'^m represents the tanh function. For the generator, the MSE is used as a loss function to determine the minimum loss value of the discriminator network.

In the discriminator, the input of the intrusion detection discriminator D' is the combined data $\boldsymbol{x'}_D$, where $\boldsymbol{x'}_D$ consists of data $\boldsymbol{x}_G$ and $\boldsymbol{x'}_z$. Then, the discriminator generates the data $\boldsymbol{y'}_D$, where $\boldsymbol{y'}_D = D'(\boldsymbol{x'}_D)$ and $\boldsymbol{y'}_D$ is a $2 \times 2p$ matrix. Here, $\boldsymbol{y'}_z = D'(\boldsymbol{x'}_z)$ and $\boldsymbol{y'} = D'(\boldsymbol{x}_G)$. $\boldsymbol{y'}_D$ represents the abnormal situation of the overall combined data, and $\boldsymbol{y'}$ represents the abnormal situation of the real input data. Besides, $\boldsymbol{y'}_z$ represents the abnormal situation of the generated data.

The discriminator can be expressed as

$$\boldsymbol{x'}^k_D = f'^k(\boldsymbol{w'}^k_D \boldsymbol{x'}^{k-1}_D + \boldsymbol{b'}^k_D), k = 1, \ldots, n \tag{14}$$

where ${x'}_D^0$ represents the input data ${x'}_D$, and ${x'}_D^k$ denotes the output of the kth layer. ${x''}_D^n$ represents the output data ${y'}_D$, where n is the number of the discriminator layers. We utilize ${\theta'}_D^k = \{{w'}_D^k, {b'}_D^k\}$ to represent the parameters corresponding to the kth layer discriminator. Similarly, f'^k represents the operation of the ReLU function or the tanh function, and the last layer f''^n represents the softmax function.

In the discriminator of intrusion detection for various attacks, we also use the cross-entropy loss function. Besides, BP and ADAM optimizers are used in our intrusion detection method for various attacks to determine the minimum loss value of the generator network so that the corresponding learning parameters can be derived and optimized model.

Similarly, in the judgment phase, we judge the abnormal situation by

$$Y'({y'}_D) = \begin{cases} {y'}_{1k} = 0, {y'}_{2k} = 1 & if\ {y'}_{1k} < {y'}_{2k} \\ {y'}_{1k} = 1, {y'}_{2k} = 0 & if\ {y'}_{1k} \geq {y'}_{2k} \end{cases} \tag{15}$$

where ${y'}_{1k}$ and ${y'}_{2k}$ are the discrimination results of the kth data, respectively. Furthermore, we use the same way to train the proposed architecture. The details of our intrusion detection method for multiple attacks are shown in Algorithm 2.

After training the intrusion detection model aiming at multiple attacks, we use the trained all discriminators to detect the network data. Namely, the network data are preprocessed by the discriminators $D_1, D_2, \ldots, D_I$ to build the combined data x_G, and then, the data x_G pass the discriminator D'. Finally, D' will output the data y' to judge whether the input data x are normal or abnormal.

Algorithm 2 Intrusion Detection Network Training Aiming at Multiple Attacks

Require: input data $p_{\text{data}}(x)$; number of iterations T; real data label $Y'_{y'}$; corresponding label of generate data Y'_z; discriminators $D_1, D_2, \ldots, D_{\text{I}}$

Ensure: generator G'; discriminator D'

$t \leftarrow 1$

while $t \leq T$ **do**

Samples minibatch of p examples $\{x^{(1)}, \ldots, x^{(p)}\}$ from data generating distribution $p_{\text{data}}(x)$

$$\begin{cases} x_i = D_i(x), & i = 1, 2, 3, \ldots, \text{I} \\ x_G = [x_1; x_2; \ldots; x_1] \end{cases}$$

${x'}_z = G'(x_G)$

${x'}_D = [x_G, {x'}_z]$

${y'}_D = D'({x'}_D)$

$$Y'({y'}_D) = \begin{cases} {y'}_{1k} = 0, {y'}_{2k} = 1 & if\ {y'}_{1k} < {y'}_{2k} \\ {y'}_{1k} = 1, {y'}_{2k} = 0 & if\ {y'}_{1k} \geq {y'}_{2k} \end{cases}$$

${y'}_D = [{y'}, {y'}_z]$

$F = \frac{TP+TN}{TP+FN+FP+TN}$

if $F(Y'({y'}), Y'_{y'}) > F(Y'({y'}_z), Y'_z)$ **then**

Generator update: The mean squared error is used as a loss function to determine the minimum loss value of the discriminator network. And the network uses BP and Adam to optimize.

else $\{F(Y'({y'}), Y'_{y'}) \leq F(Y'({y'}_z), Y'_z)\}$

Discriminator update: The cross entropy loss function is used as a loss function to determine the minimum loss value of the discriminator network. And the network uses BP and Adam to optimize.

end if

end while

return discriminator D'; generator G'

IV. Experiments

A. Data Set

In our simulation, we use CSE-CIC-IDS2018 and CIC-DDoS2019 [20], [38] to evaluate our method. CICFlowMeter is utilized to preprocess the CSE-CIC-IDS2018 data set and CIC-DDoS2019 data set, which consists of 83 features. Here, we select a certain amount of data to evaluate the intrusion detection mechanism for various attacks. The CSE-CIC-IDS2018 data set contains seven different attack scenarios, namely Botnet, DoS, DDoS, Brute-Force, Heartbleed, infiltration of the network from inside, and Web attacks. At the same time, these seven attack scenarios are included 14 different intrusion attacks. We take advantage of seven attacks as the training data set and 14 attacks as the testing data set. There are different attacks in the training data set, such as Botnet attack, DoS attacks-Hulk, DoS attacks-slow hypertext transfer protocol test (SlowHTTPTest), Brute Force-Web, Infilteration, file transfer protocol (FTP)-BruteForce, and DDOS attack-High Orbit Ion Canon (HOIC). The attacks in the testing data set consist of a Botnet attack, DoS attacks-Hulk, DoS attacks-SlowHTTPTest, Brute Force-Web, Brute Force-Cross Site Scripting (XSS), Structured Query Language (SQL) Injection, DDoS attacks-Low Orbit Ion Canon (LOIC)-Hyper Text Transfer Protocol (HTTP), Infilteration, DoS attacks-GoldenEye, DoS attacks-Slowloris, FTP-BruteForce, Secure Shell (SSH)-Bruteforce, DDOS attack-HOIC, and DDOS attack-LOIC-user datagram protocol (UDP).

At the same time, we take advantage of the data on the first day of the CIC-DDoS2019 data set as the training data set and the data on the second day as the testing data set. The training and testing data sets consist of seven and 12 anomalies, respectively. There are different DDoS attacks in the training data set, such as PortMap, network basic input/output system (NetBIOS), lightweight directory access protocol (LDAP), Microsoft Structured Query Language (MSSQL), UDP, UDP-Lag, and synchronize sequence numbers (SYN). The DDoS attacks in the testing data set consist of network time protocol (NTP), domain name system (DNS), LDAP, MSSQL, NetBIOS, simple network management protocol (SNMP), simple service discovery protocol (SSDP), UDP, UDP-Lag, WebDDoS, SYN, and trivial file transfer protocol (TFTP).

We select a certain amount of data from the data set for experimentation, and the specific conditions are shown in Table II. We set up low-flow attacks in the training data set, and we also set up a variety of new types of attacks and new low-flow attacks in the testing data set. Here, in order to verify the performance of the proposed GAN-based model, we conducted two data sets for evaluation. The first experiment is to evaluate the performance of the GAN-based model

TABLE II
SAMPLE DISTRIBUTION

Data set	Class	Attack Type	Abnormal Data	Normal Data
CSE-CIC-IDS 2018	Training	Infilteration	3500	2500
		DoS attacks-Hulk	3500	2500
		DoS attacks-SlowHTTPTest	3500	2500
		Brute Force-Web	150	100
		Botnet attack	3500	2500
		FTP-BruteForce	3500	2500
		DDoS attacks-HOIC	3500	2500
		Overall data	36250	
	Testing	Infilteration	40000	5000
		DoS attacks-Hulk	40000	5000
		DoS attacks-SlowHTTPTest	40000	5000
		Brute Force-Web	611	100
		Brute Force-XSS	230	50
		SQL Injection	87	20
		DDoS attacks-LOIC-HTTP	40000	5000
		Botnet attack	40000	5000
		DoS attacks-GoldenEye	40000	5000
		DoS attacks-Slowloris	10990	1000
		FTP-BruteForce	40000	5000
		SSH-Bruteforce	40000	5000
		DDoS attacks-HOIC	40000	5000
		DDoS attacks-LOIC-UDP	1730	250
		Overall data	420068	
CIC-DDoS 2019	Training	PortMap	3500	2500
		NetBIOS	3500	2500
		LDAP	3500	2500
		MSSQL	3500	2500
		UDP	3500	2500
		UDP-Lag	180	100
		SYN	3500	2500
		Overall data	36280	
	Testing	NTP	40000	5000
		DNS	40000	5000
		LDAP	40000	5000
		MSSQL	40000	5000
		NetBIOS	40000	5000
		SNMP	40000	5000
		SSDP	40000	5000
		UDP	40000	5000
		UDP-Lag	40000	5000
		WebDDoS	439	100
		SYN	40000	5000
		TFTP	40000	5000
		Overall data	495539	

for intrusion detection aiming at an attack. The second is implementing our method to detect various attacks.

In this article, we use five performance metrics to evaluate detection performance, and they are

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{16}$$

Accuracy means that our model predicts the correct sample as a percentage of all the samples involved

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{17}$$

Precision indicates the proportion of positive examples that are classified as positive examples in fact

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{18}$$

The recall rate is related to the original sample, which indicates that the predicted number of positive cases in the sample is correct

$$\text{FalseAlarm} = \frac{FP}{FP + TN}. \tag{19}$$

False alarm is the proportion of correct samples that are wrongly classified as wrong

$$\text{F\_Measures} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{20}$$

The F_Measures value is the harmonic average of precision and recall.

TABLE III
INTRUSION DETECTION PARAMETERS AIMING AT SINGLE ATTACK

Parameter	Value
Number of generator layers	7
Number of generator input layer units	1
Number of hidden layer units of generator	1280
Number of generator output layer units	10
Generator single layer activation function	Tanh
Generator even layer activation function	ReLU
Generator loss function	MSE
Generator gradient optimizer	Adam
Generator learning rate	0.0001
Number of discriminator layers	5
Number of discriminator input layer units	10
Number of units in the first hidden layer	28
Number of units in the second hidden layer	16
Number of units in the third hidden layer	8
Number of discriminator output layer units	2
Discriminator hidden layer activation function	Tanh
Discriminator output layer activation function	Softmax
Discriminator loss function	Cross Entropy Loss Function
Discriminator gradient optimizer	Adam
Discriminator learning rate	0.0001
Network training times per cycle	200
Overall training times	5000

The proposed GAN-based model is implemented using MATLAB running on a machine with the configuration of an Intel Core processor (3.2 GHz) with 8-GB RAM. The hyperparameter settings of the GAN-based intrusion detection model aiming at single and various attacks are shown by Table IV, respectively. In this work, we use an EMFFS method that combines the output of IG, gain ratio, and chi-squared to extract important features. In this article, to reduce the amount of calculation, based on the IG, we select ten features related to attacks. For instance, we select ten features related to the attack, namely, Flow inter arrival time (IAT) Mean, Subflow Bwd Byts, Flow Byts/s, Flow IAT Std, Init Win bytes backward, Subflow Bwd Pkts, Idle Std, Idle Mean, Fwd IAT Mean, and Flow Pkts/s. The selected features are shown in Table V.

B. Performance Evaluation

We use five performance metrics to evaluate the detection performance, namely, Accuracy, Precision, Recall, False Alarm, and F-Measures. Besides, we compare the performance of the proposed GAN-based intrusion detection framework with existing deep learning models, namely, vector convolutional deep learning (VCDL) [39], stacked nonsymmetric deep auto-encoders (S-NDAE) [34], self-taught learning (STL) [40], and stacked contractive auto encoder and support vector machine (SCAE+SVM) [19]. VCDL is the VCDL approach, which comprises two modules, i.e., the fully

TABLE IV
INTRUSION DETECTION PARAMETERS AIMING AT VARIOUS ATTACKS

Parameter	Value
Number of generator layers	16
Number of generator input layer units	14
Number of hidden layer units of generator	1400
Number of generator output layer units	14
Generator single layer activation function	ReLU
Generator even layer activation function	Tanh
Generator loss function	MSE
Generator gradient optimizer	Adam
Generator learning rate	0.0001
Number of discriminator layers	54
Number of discriminator input layer units	14
Number of hidden layer units of discriminator	1104
Number of discriminator output layer units	2
Discriminator single layer activation function	ReLU
Discriminator even layer activation function	Tanh
Discriminator output layer activation function	Softmax
Discriminator loss function	Cross Entropy Loss Function
Discriminator gradient optimizer	Adam
Discriminator learning rate	0.0001
Network training times per cycle	200
Overall training times	5000

TABLE V
SELECTED FEATURES

Feature Name	Description
Flow IAT Mean	Average time between two flows
Subflow Bwd Byts	The average number of bytes in a sub flow in the backward direction
Flow Byts/s	Flow byte rate
Flow IAT Std	Standard deviation time between two flows
Init Win bytes backward	Number of bytes sent in initial window in the backward direction
Subflow Bwd Pkts	The average number of packets in a sub flow in the backward direction
Idle Std	Standard deviation time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Flow Pkts/s	Flow packets rate

connected network and the vector convolutional network. The vector convolutional network is involved to extract the features, and the fully connected network learns these extracted features. S-NDAE comprises two modules, i.e., S-NDAE and random forest. This method uses S-NDAE for feature learning, and the random forest is used for classification. STL is a combination method consisting of unsupervised feature learning and classification models. SCAE+SVM is a combination method consisting of a stacked contractive AE and a support vector machine. This method uses the stacked contractive AE for feature extraction, and it classifies the trained data by using a support vector machine.

The single attack intrusion detection situation based on GAN is shown in Table VI. For each type of attack, we select 6000 samples as training data set, in which 2500 samples are normal data, and the other 3500 samples are abnormal. For the CSE-CIC-IDS2018 data set, we set the Brute Force-Web attack as a low traffic attack. At the same time, we select 45 000 corresponding abnormal types of samples as the testing data set, where 40 000 samples are abnormal and the others

TABLE VI
INTRUSION DETECTION RESULT AIMING AT SINGLE ATTACK

Data Set	Attack Type	Accuracy	Precision	Recall	False Alarm	F_Measures
CSE-CIC-IDS 2018	Infilteration	0.9804	0.9986	0.9793	0.0106	0.9889
	DoS attacks-Hulk	0.9952	0.9971	0.9975	0.0234	0.9973
	DoS attacks SlowHTTPTest	0.9997	0.9997	0.9999	0.0024	0.9999
	Brute Force-Web	0.9999	0.9999	0.9999	0.0001	0.9999
	Botnet attack	0.9758	0.9994	0.9734	0.005	0.9862
	FTP-BruteForce	0.9988	0.9998	0.9989	0.0016	0.9993
	DDoS attacks-HOIC	0.9974	0.9971	0.9999	0.0232	0.9986
CIC-DDoS 2019	PortMap	0.9834	0.986	0.9954	0.113	0.9907
	NetBIOS	0.9976	0.9987	0.9985	0.01	0.9986
	LDAP	0.9987	0.9985	0.999	0.0038	0.9993
	MSSQL	0.9966	0.9976	0.9986	0.0194	0.9981
	UDP	0.9594	0.9941	0.9599	0.0454	0.9767
	UDP-Lag	0.9162	0.9139	0.9999	0.7532	0.9549
	SYN	0.9674	0.9901	0.9731	0.0782	0.9815

TABLE VII
INTRUSION DETECTION RESULT AIMING AT MULTIPLE ATTACKS BASED ON THE CSE-CIC-IDS2018 DATA SET

Method	Attack Type	Accuracy	Precision	Recall	False Alarm	F_Measures
GAN	Overall Data	**0.9532**	**0.9988**	**0.9485**	**0.0089**	**0.9730**
SCAE+SVM	Overall Data	**0.9504**	**0.9963**	**0.9478**	**0.0279**	**0.9715**
VCDL	Overall Data	**0.8821**	**0.9988**	**0.8685**	**0.0084**	**0.9291**
STL	Overall Data	**0.8695**	**0.9983**	**0.8547**	**0.0114**	**0.9209**
S-NDAE	Overall Data	**0.9399**	**0.9961**	**0.9362**	**0.0296**	**0.9652**

TABLE VIII
INTRUSION DETECTION RESULT AIMING AT MULTIPLE ATTACKS BASED ON THE CIC-DDOS2019 DATA SET

Method	Attack Type	Accuracy	Precision	Recall	False Alarm	F_Measures
GAN	Overall Data	**0.9853**	**0.9959**	**0.9876**	**0.0326**	**0.9917**
SCAE+SVM	Overall Data	**0.9371**	**0.9897**	**0.939**	**0.078**	**0.9637**
VCDL	Overall Data	**0.9685**	**0.9955**	**0.9689**	**0.0352**	**0.9821**
STL	Overall Data	**0.9503**	**0.9951**	**0.9487**	**0.0369**	**0.9714**
S-NDAE	Overall Data	**0.9027**	**0.9836**	**0.9056**	**0.1209**	**0.9429**

are normal. We select 611 Brute Force-Web attacks samples as the testing data set, where 100 are abnormal. For the CIC-DDoS2019 data set, we set the UDP-Lag attack as a low traffic attack. Similarly, we select 45 000 corresponding abnormal types of samples as the testing data set, where 40 000 samples are abnormal.

According to these simulation results, we can find that the proposed intrusion detection model aiming at a single attack can obtain excellent intrusion detection accuracy. Our method also has high accuracy for known low-flow attacks.

In intrusion detection aiming at multiple attacks, we select the data in Table II as the testing data set. In the

TABLE IX

INTRUSION DETECTION RESULT AIMING AT MULTIPLE ATTACKS BASED ON THE CSE-CIC-IDS2018 DATA SET

Method	Attack Type	Accuracy	Precision	Recall	False Alarm	F_Measures
GAN	Overall Data	**0.9532**	**0.9988**	**0.9485**	**0.0089**	**0.9730**
	Infilteration	0.9776	0.999	0.9758	0.0076	0.9873
	DoS attacks-Hulk	0.9933	0.999	0.9934	0.0076	0.9962
	DoS attacks-SlowHTTPTest	0.9992	0.9991	0.9999	0.0076	0.9995
	Brute Force-Web	0.8678	0.9999	0.8646	0.0001	0.9167
	Brute Force-XSS	0.6107	0.9999	0.5261	0.0001	0.6895
	SQL Injection	0.8131	0.9999	0.7701	0.0001	0.8701
	DDoS attacks-LOIC-HTTP	0.9992	0.9991	0.9999	0.0076	0.9995
	Botnet attack	0.9991	0.9991	0.9999	0.0076	0.9995
	DoS attacks-GoldenEye	0.6709	0.9985	0.6307	0.0076	0.7731
	DoS attacks-Slowloris	0.7463	0.9954	0.7266	0.0037	0.84
	FTP-BruteForce	0.9992	0.9991	0.9999	0.0076	0.9995
	SSH-Bruteforce	0.9992	0.9991	0.9999	0.0076	0.9995
	DDoS attacks-HOIC	0.9992	0.9991	0.9999	0.0076	0.9995
	DDoS attacks-LOIC-UDP	0.9828	0.9807	0.9999	0.136	0.9903
SCAE+SVM	Overall Data	**0.9504**	**0.9963**	**0.9478**	**0.0279**	**0.9715**
	Infilteration	0.9723	0.9969	0.9719	0.0238	0.9842
	DoS attacks-Hulk	0.9951	0.997	0.9975	0.0238	0.9973
	DoS attacks-SlowHTTPTest	0.9974	0.997	0.9999	0.0238	0.9985
	Brute Force-Web	0.879	0.9999	0.8592	0.0001	0.9243
	Brute Force-XSS	0.9999	0.9999	0.9999	0.0001	0.9999
	SQL Injection	0.8318	0.9999	0.7931	0.0001	0.8846
	DDoS attacks-LOIC-HTTP	0.9974	0.997	0.9999	0.0238	0.9985
	Botnet attack	0.9974	0.997	0.9999	0.0238	0.9985
	DoS attacks-GoldenEye	0.6526	0.9952	0.6121	0.0238	0.758
	DoS attacks-Slowloris	0.7675	0.986	0.7571	0.118	0.8565
	FTP-BruteForce	0.9974	0.997	0.9999	0.0238	0.9985
	SSH-Bruteforce	0.9974	0.997	0.9999	0.0238	0.9985
	DDoS attacks-HOIC	0.9974	0.997	0.9999	0.0238	0.9985
	DDoS attacks-LOIC-UDP	0.9449	0.9407	0.9999	0.436	0.9695
VCDL	Overall Data	**0.8821**	**0.9988**	**0.8685**	**0.0084**	**0.9291**
	Infilteration	0.9656	0.9991	0.9623	0.0072	0.9803
	DoS attacks-Hulk	0.9934	0.9991	0.9934	0.0072	0.9963
	DoS attacks-SlowHTTPTest	0.9992	0.9991	0.9999	0.0072	0.9996
	Brute Force-Web	0.872	0.9999	0.8511	0.0001	0.9195
	Brute Force-XSS	0.6071	0.9999	0.5217	0.0001	0.6857
	SQL Injection	0.8505	0.9999	0.8161	0.0001	0.8987
	DDoS attacks-LOIC-HTTP	0.9906	0.9991	0.9903	0.0072	0.9947
	Botnet attack	0.9992	0.9991	0.9999	0.0072	0.9995
	DoS attacks-GoldenEye	0.6213	0.9984	0.5749	0.0072	0.7297
	DoS attacks-Slowloris	0.7714	0.9958	0.7538	0.0035	0.858
	FTP-BruteForce	0.9992	0.9991	0.9999	0.0072	0.9996
	SSH-Bruteforce	0.398	0.9972	0.3237	0.0072	0.4887
	DDoS attacks-HOIC	0.9992	0.9991	0.9999	0.0072	0.9995
	DDoS attacks-LOIC-UDP	0.9838	0.9824	0.9994	0.124	0.9908

CSE-CIC-IDS2018 data set, we set the Brute Force-Web attack as a low-flow attack for training. In the CIC-DDoS2019 data set, we set the UDP-Lag attack as a low-flow attack for training. In the testing data set, we add various attacks and set various new low-flow attacks. The simulation results of intrusion detection for multiple attacks are shown in Tables VII and VIII.

TABLE X

INTRUSION DETECTION RESULT AIMING AT MULTIPLE ATTACKS BASED ON THE CIC-DDOS2019 DATA SET

Method	Attack Type	Accuracy	Precision	Recall	False Alarm	F_Measures
GAN	Overall Data	**0.9853**	**0.9959**	**0.9876**	**0.0326**	**0.9917**
	NTP	0.9517	0.9957	0.9498	0.0326	0.9722
	DNS	0.9879	0.9959	0.9905	0.0326	0.9932
	LDAP	0.9962	0.9959	0.9998	0.0326	0.9978
	MSSQL	0.995	0.996	0.998	0.0326	0.9972
	NetBIOS	0.9963	0.9959	0.9999	0.0326	0.9979
	SNMP	0.9963	0.9959	0.9999	0.0326	0.9979
	SSDP	0.991	0.9959	0.994	0.0326	0.9949
	UDP	0.9964	0.9959	0.9999	0.0326	0.998
	UDP-Lag	0.9963	0.9959	0.9999	0.0326	0.9979
	WebDDoS	0.3766	0.9725	0.2415	0.03	0.3869
	SYN	0.9763	0.9958	0.9774	0.0326	0.9866
	TFTP	0.9624	0.9958	0.9618	0.0326	0.9785
SCAE+SVM	Overall Data	**0.9371**	**0.9897**	**0.939**	**0.078**	**0.9637**
	NTP	0.7019	0.9857	0.6743	0.078	0.8008
	DNS	0.7464	0.9867	0.7245	0.078	0.8355
	LDAP	0.9866	0.9903	0.9947	0.078	0.9925
	MSSQL	0.9881	0.9903	0.9964	0.078	0.9933
	NetBIOS	0.9911	0.9903	0.9998	0.078	0.995
	SNMP	0.9907	0.9903	0.9993	0.078	0.9948
	SSDP	0.9775	0.9902	0.9844	0.078	0.9873
	UDP	0.9912	0.9903	0.9999	0.078	0.9951
	UDP-Lag	0.9908	0.9903	0.9994	0.078	0.9949
	WebDDoS	0.3469	0.9223	0.2164	0.08	0.3506
	SYN	0.9849	0.9903	0.9928	0.078	0.9915
	TFTP	0.966	0.9901	0.9715	0.078	0.9807
VCDL	Overall Data	**0.9685**	**0.9955**	**0.9689**	**0.0352**	**0.9821**
	NTP	0.8763	0.9949	0.8653	0.0352	0.9256
	DNS	0.9244	0.9952	0.9193	0.0352	0.9558
	LDAP	0.9958	0.9956	0.9997	0.0352	0.9977
	MSSQL	0.9939	0.9956	0.9976	0.0352	0.9966
	NetBIOS	0.9958	0.9956	0.9997	0.0352	0.9977
	SNMP	0.9959	0.9956	0.9998	0.0352	0.9977
	SSDP	0.9817	0.9955	0.9838	0.0352	0.9896
	UDP	0.9959	0.9956	0.9999	0.0352	0.9977
	UDP-Lag	0.9956	0.9956	0.9995	0.0352	0.9975
	WebDDoS	0.23	0.9286	0.0592	0.02	0.1113
	SYN	0.9723	0.9955	0.9733	0.0352	0.9842
	TFTP	0.9348	0.9953	0.9311	0.0352	0.9621

In the overall intrusion detection task, the GAN-based model that we designed has higher accuracy, precision, recall, false alarm, and F_measures. Compared with other methods, our method has higher accuracy and lower false alarms in the two data sets and also has higher performance in different environments. At the same time, we select the two optimal models for detailed comparison with our method, and the simulation results are shown in Tables IX and X.

Compared with other methods, our method has higher accuracy and lower false alarms in the two data sets. For new attacks, our method also has a better performance. For multiple types of attacks, our method has high stability and accuracy. However, for the new low-flow attack, i.e., WebDDoS and Brute Force-XSS, our method has low detection accuracy, and there is no significant improvement comparing with the other two methods. On the whole, the GAN-based method designed in this article has better detection performance comparing with the other two methods. Our GAN-based algorithm can be used to capture different types of attacks with high accuracy.

Moreover, our method can detect a variety of new types of attacks by training and learning existing attack types. In summary, our method reduces the amount of calculation and data dimension through feature selection. At the same time, our method can well identify the network environment of multiple types of attacks, and it also has high accuracy for emerging attacks.

V. Conclusion and Future Works

This article studies the intrusion detection problem in CEC-based SIoT. We propose a GAN-based intrusion detection method. By extracting the features of network data, the proposed method can be used to detect various attacks. The proposed method includes three phases. They are feature extraction, an intrusion detection model aiming at a single attack, and an intrusion detection model with various discriminators aiming at multiple attacks. Our method first preprocesses the flow and performs feature extraction. Then, we design an intrusion detection algorithm aiming at a single attack based on GAN. By combining several intrusion detection models aiming at a single attack, we design an intrusion detection algorithm aiming at multiple attacks based on GAN. In addition, we also evaluate our method by implementing it over the CSE-CIC-IDS2018 and CIC-DDoS2019 data sets. From the simulation results, our method can significantly improve the accuracy of intrusion detection comparing with the other two methods.

To improve the accuracy of our method further, we will combine the convolutional neural network and the GAN method for extracting the spatiotemporal features of network data. At the same time, a feature extraction algorithm is necessary to improve the real-time performance of our method.

References

[1] Y. Zhao, X. Dong, and Y. Yin, "Effective and efficient dense subgraph query in large-scale social Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2726–2736, Apr. 2020.

[2] M. A. Azad, S. Bag, F. Hao, and A. Shalaginov, "Decentralized self-enforcing trust management system for social Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2690–2703, Apr. 2020.

[3] H. Ning and Z. Wang, "Future Internet of Things architecture: Like mankind neural system or social organization framework?" *IEEE Commun. Lett.*, vol. 15, no. 4, pp. 461–463, Apr. 2011.

[4] C. Liu, J. Liu, and Z. Jiang, "A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2274–2287, Dec. 2014.

[5] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[6] Z. Ning *et al.*, "Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, early access, Sep. 18, 2020, doi: 10.1109/TMC.2020.3025116

[7] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Jun. 2020.

[8] A. Ghosh and K. Grolinger, "Edge-cloud computing for IoT data analytics: Embedding intelligence in the edge with deep learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2191–2200, Mar. 2021.

[9] Z. Ning *et al.*, "Distributed and dynamic service placement in pervasive edge computing networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1277–1292, Jun. 2021.

[10] T. Dbouk, A. Mourad, H. Otrok, H. Tout, and C. Talhi, "A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1665–1680, Dec. 2019.

[11] Z. Ning *et al.*, "Joint computing and caching in 5G-envisioned Internet of vehicles: A deep reinforcement learning-based traffic control system," *IEEE Trans. Intell. Transp. Syst.*, early access, Feb. 5, 2020, doi: 10.1109/TITS.2020.2970276.

[12] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1840–1852, Mar. 2021.

[13] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Trans. Mobile Comput.*, early access, Jul. 28, 2020, doi: 10.1109/TMC.2020.3012509.

[14] Z. Ning *et al.*, "Intelligent edge computing in Internet of vehicles: A joint computation offloading and caching solution," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 5, 2020, doi: 10.1109/TITS.2020.2997832.

[15] D. Yuan *et al.*, "Intrusion detection for smart home security based on data augmentation with edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[16] Y. Dong, R. Wang, and J. He, "Real-time network intrusion detection system based on deep learning," in *Proc. IEEE 10th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Oct. 2019, pp. 1–4.

[17] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for Web attack detection on edge devices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1963–1971, Mar. 2020.

[18] Z. Ning *et al.*, "Mobile edge computing enabled 5G health monitoring for Internet of medical things: A decentralized game theoretic approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 463–478, Feb. 2021.

[19] W. Wang, X. Du, D. Shan, R. Qin, and N. Wang, "Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine," *IEEE Trans. Cloud Comput.*, early access, Jun. 9, 2020, doi: 10.1109/TCC.2020.3001017.

[20] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9552–9562, Oct. 2020.

[21] A. Mourad, H. Tout, O. A. Wahab, H. Otrok, and T. Dbouk, "Ad hoc vehicular fog enabling cooperative low-latency intrusion detection," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 829–843, Jan. 2021.

[22] R. Heartfield, G. Loukas, A. Bezemskij, and E. Panaousis, "Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1720–1735, 2021.

[23] X. Wang, Z. Ning, and S. Guo, "Multi-agent imitation learning for pervasive edge computing: A decentralized computation offloading algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 411–425, Feb. 2021.

[24] J. Liu, C. Wang, H. Su, B. Du, and D. Tao, "Multistage GAN for fabric defect detection," *IEEE Trans. Image Process.*, vol. 29, pp. 3388–3400, 2020.

[25] H. Yang, Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu, "Deep reinforcement learning-based intelligent reflecting surface for secure wireless communications," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 375–388, Jan. 2021.

[26] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6G networks," *IEEE Netw.*, vol. 34, no. 6, pp. 272–280, Nov. 2020.

[27] R. Ganesan, S. Jajodia, and H. Cam, "Optimal scheduling of cybersecurity analysts for minimizing risk," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 4, p. 52, 2017.

[28] W. Sha, Y. Zhu, M. Chen, and T. Huang, "Statistical learning for anomaly detection in cloud server systems: A multi-order Markov chain framework," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 401–413, Apr. 2018.

[29] R. Conforti, M. L. Rosa, and A. H. M. T. Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 300–314, Feb. 2017.

[30] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 481–494, Dec. 2019.

[31] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016.

[32] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 924–935, Sep. 2019.

[33] M. Saharkhizan, A. Azmoodeh, A. Dehghantanha, K.-K.-R. Choo, and R. M. Parizi, "An ensemble of deep recurrent neural networks for detecting IoT cyber attacks using network traffic," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8852–8859, Sep. 2020.
[34] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[35] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3540–3552, 2020.
[36] A. Montieri, D. Ciuonzo, G. Aceto, and A. Pescape, "Anonymity services Tor, I2P, JonDonym: Classifying in the dark (Web)," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 3, pp. 662–675, May 2020.
[37] A. Shawahna, M. Abu-Amara, A. Mahmoud, and Y. E. Osais, "EDoS-ADS: An enhanced mitigation technique against economic denial of sustainability (EDoS) attacks," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 790–804, Jul./Sep. 2020.
[38] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.
[39] S. Selvakumar, "Anomaly detection framework for Internet of Things traffic using vector convolutional deep learning approach in fog environment," *Future Gener. Comput. Syst.*, vol. 113, pp. 255–265, Dec. 2020.
[40] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol.*, vol. 3, Dec. 2015, pp. 21–26.

Laisen Nie (Member, IEEE) received the Ph.D. degree in communication and information system from Northeastern University, Shenyang, China, in 2016.

He is currently an Associate Professor with the School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China. His research interests include network measurement, network security, and cognitive networks.

Yixuan Wu was born in Zhejiang, China, in 1994. He received the B.Eng. degree in optoelectronic information science and engineering from Ningbo University, Ningbo, China, in 2017. He is currently pursuing the degree with the School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China.

His research interests include network measurement and anomaly detection.

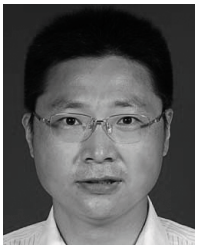
Xiaojie Wang received the M.S. degree from Northeastern University, Shenyang, China, in 2011, and the Ph.D. degree from the Dalian University of Technology, Dalian, China, in 2019.

From 2011 to 2015, she was a Software Engineer with NeuSoft Corporation, Beijing, China. She is currently a Distinguished Professor with the Chongqing University of Posts and Telecommunications, Chongqing, China. Her research interests are wireless networks, mobile edge computing, and machine learning.

Lei Guo (Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006.

He is currently a Full Professor with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical articles in international journals and conferences. His current research interests include communication networks, optical communications, and wireless communications.

Dr. Guo is currently serving as an Editor for several international journals.

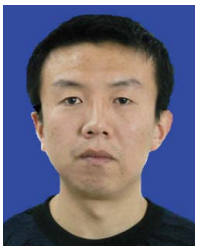


Guoyin Wang (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 1992, 1994, and 1996, respectively.

From 1998 to 1999, he was a Visiting Scholar with the University of North Texas, Denton, TX, USA, and the University of Regina, Regina, SK, Canada. Since 1996, he has been with the Chongqing University of Posts and Telecommunications, Chongqing, China, where he is currently a Professor, the Director of the Chongqing Key Laboratory of Computational Intelligence, and the Vice-President and Dean of the School of Graduate. He had been the Director of the Institute of Electronic Information Technology, Chongqing Institute of Green the and Intelligent Technology, CAS, Chongqing, from 2011 to 2017. His research interests include rough set, granular computing, knowledge technology, data mining, neural network, and cognitive computing.

Xinbo Gao (Senior Member, IEEE) received the B.Eng., M.Sc., and Ph.D. degrees in electronic engineering, signal and information processing from Xidian University, Xi'an, China, in 1994, 1997, and 1999, respectively.

Since 2001, he has been with the School of Electronic Engineering, Xidian University. He is currently a Cheung Kong Professor of Ministry of Education of China, a Professor of Pattern Recognition and Intelligent System of Xidian University and a Professor of Computer Science and Technology of Chongqing University of Posts and Telecommunications. His current research interests include Image processing, computer vision, multimedia analysis, machine learning, and pattern recognition.



Shengtao Li received the M.S. degree in operational research and cybernetics from Ludong University, Yantai, China, in 2010, and the Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2013.

He is currently an Associate Professor with the School of Information Science and Engineering, Shandong Normal University. His research interests include nonlinear systems theory, optimal switch-time control theory of switched stochastic systems, and robust control of systems with time-delay.