IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SECURITY AND PRIVACY IN EMERGING DECENTRALIZED COMMUNICATION ENVIRONMENTS

Received April 30, 2019, accepted May 12, 2019, date of publication May 17, 2019, date of current version May 30, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2917299*

# Wireless Network Intrusion Detection Based on Improved Convolutional Neural Network

**HONGYU YANG AND FENGYAN WANG**
School of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China
Corresponding author: Hongyu Yang (yhyxlx@hotmail.com)

This work was supported by the Civil Aviation Information Systems Multidimensional Security Situation Assessment and Business Affecting Impact Analysis under Grant U1833107.

**ABSTRACT** The diversification of wireless network traffic attack characteristics has led to the problems what traditional intrusion detection technology with high false positive rate, low detection efficiency, and poor generalization ability. In order to enhance the security and improve the detection ability of malicious intrusion behavior in a wireless network, this paper proposes a wireless network intrusion detection method based on improved convolutional neural network (ICNN). First, the network traffic data is characterized and preprocessed, then modeled the network intrusion traffic data by ICNN. The low-level intrusion traffic data is abstractly represented as advanced features by CNN, which extracted autonomously the sample features, and optimizing network parameters by stochastic gradient descent algorithm to converge the model. Finally, we conducted a sample test to detect the intrusion behavior of the network. The simulation results show that the method proposed in our paper has higher detection accuracy and true positive rate together with a lower false positive rate. The test results on the test set KDDTest + in our paper show that compared with the traditional models, the detection accuracy is 8.82% and 0.51% higher than that of LeNet-5 and DBN, respectively, and the recall rate is 4.24% and 1.16% higher than that of LeNet-5 and RNN, respectively, while the false positive rate is lower than the other three types of models. It also has a big advantage compared to the IDABCNN and NIDMBCNN methods.

**INDEX TERMS** Wireless network intrusion detection, security, convolutional neural network.

## I. INTRODUCTION

With the emergence of new wireless network attack features, the intrusion detection methods with high adaptability, stability and effectiveness are in urgent need. At present, the general wireless network authentication mechanism and firewall technology can basically meet the basic security protection requirements of users, but the protection capability is relatively weak. Once the malicious attacks by professional hackers are encountered, these protection measures are ineffective. At present, the common used intrusion detection methods represented by misuse detection [1], [2] and abnormal detection [3]–[6] have shortcomings such as low detection accuracy, feature extraction efficiency, and high false positive rate. With the application of artificial intelligence methods in intrusion detection system (IDS), detection methods based on artificial intelligence have become one of the hotspots of IDS research.

The artificial intelligence methods currently applied in intrusion detection methods mainly include neural networks, genetic algorithms and immune algorithms, such as intrusion detection based on deep BP neural network [7] and collaborative evolution intrusion detection based on fuzzy systems [8]. However, these methods are prone to loss key information during the process of the data set, which results in ''distortion'' of the sample data, inefficiency of data feature extraction and then contributes to greater volatility of the test results. In order to solve this problem, Chen *et al.* [9] proposed a deep belief network intrusion detection method based on optimized data processing. Qu *et al.* [10] proposed an intrusion detection model based on deep belief network. Yin *et al.* [11] proposed a deep learning methods for intrusion detection based on recurrent neural network. Shone *et al.* [12] proposed a nonsymmetric deep autoencoder (NDAE) based on unsupervised feature learning. Yuan and Lintao [13] using genetic algorithm (GA) and Support Vector Machine (SVM) established a network intrusion detection classifier. Wei *et al.* [14] proposed a multi-group clone selection algorithm, and changed the matching rules of the algorithm to

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaochun Cheng.

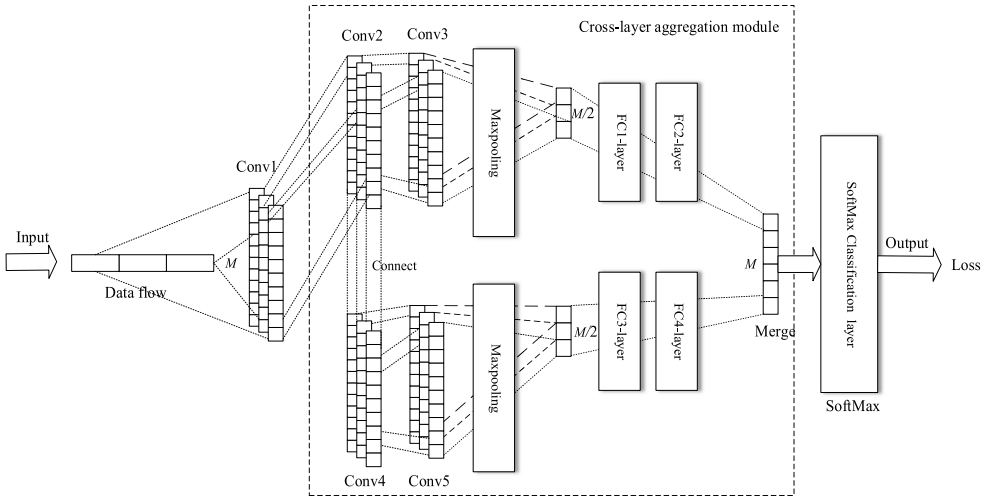



**FIGURE 1.** ICNN structure.

carry intrusion detection simulation experiment. Although the above methods improved the accuracy of intrusion detection in a certain extent, it is difficult to adjust the experiment parameters, as well as facing the problems of low extraction efficiency and the large calculation task. To overcome the above shortcomings, Jia and Kong *et al.* [15] and Wang and Li *et al.* [16] used convolutional neural network to implement Intrusion Detection Algorithm Based on Convolutional Neural Network (IDABCNN) and Network Intrusion Detection Model Based on Convolutional Neural Network (NIDM-BCNN). Compared with other machine learning methods, the wireless intrusion detection method using convolutional neural network significantly improves the accuracy of classification. However, with low convergence speed and poor generalization ability in the model training process, the true positive rate of the method is relatively low and the false positive rate is relatively high.

Although the above research has improved the sample recognition ability and performance, there are some shortcomings such as weak fitting and generalization ability, and the detection accuracy and efficiency need to be improved. In order to avoid the network training over-fitting and improve the generalization ability, an intrusion detection model based on deep learning is proposed in this paper, which based on the structural characteristics of convolutional neural network and the design idea of cross-layer aggregation. The model is trained by using the pre-processed sample dataset. After incessant feature extraction and iterative optimization, the model achieves a good convergence effect, and then the trained classifier is used for classification test. The experimental results verify the feasibility and effectiveness of the method for improving the intrusion detection effect.

## II. IMPROVED CONVOLUTIONAL NEURAL NETWORK

### A. DESIGN IDEAS

Aiming at the traditional classification detection algorithm, a large number of key parameters need to be set manually during the training process, and the key feature information loss and parameter tuning difficulty are easily caused during the training process. This paper considers using the end-to-end semi-supervised network training classifier of convolutional neural network (CNN), and the multi-layer feature of CNN to detect network, learn the sample features and discover the rules in the data training process to simplify the implementation process. To this end, we proposed an improved convolutional neural network (ICNN), which is designed as follows:

1)The cross-layer aggregation network design method is adopted to make the intrusion detection model start from the second convolution operation, and save the convolved result. Then perform convolution, pooling, and full connection operations separately.

2)After performing the same operation on the output result of the third convolution operation, the concat() function in tensorflow is used to perform the merge operation on the output data of the cross-layer aggregation network.

3)The loss value is calculated according to the classification result of SoftMax, and then the back propagation is performed by using the value. The network weights and offsets are continuously optimized through iterative training until a good convergence effect is achieved.

### B. IMPROVED CONVOLUTIONAL NEURAL NETWORK STRUCTURE

The improved convolutional neural network (ICNN) structure is shown in FIGURE 1. The ICNN consists of 5 convolutional layers, 2 pooling layers, 4 fully connected layers and 1 SoftMax layer, where Conv1, Conv2, and Conv3 those 3 convolutional layers use the relu activation function to increase network sparsity. To avoid over-fitting during training, the regularization method dropout is used in the two fully connected layers FC1_layer and FC3_layer of the cross-layer aggregation module.

In the ICNN structure, concat() is used to perform the merge operation, and the SoftMax layer is used to classify the training output results of the intrusion detection model.

### C. MODEL TRAINING

ICNN's model training consists of a forward propagation process and a back propagation process.

#### 1) FORWARD PROPAGATION

In ICNN, the training forward propagation process is designed as follows:

First, training is performed in batches in the network. Each training randomly selects a fixed-size block as input from the pre-processed training data set. The data parameter dimensions entered during training are in accordance with (batch_size, H, W, channel) four-dimensional parameter settings. For each training, a block of size M is selected from the data set. The height and width of the input data are set as 1 and 122 respectively, and the channel is single.

In ICNN (shown in FIGURE 1), Input Data first passes through two convolutional layers (Conv1_Relu and Conv2_Relu) and the convolution operation is performed on all the feature maps of the initial input by using the multi-convolution kernel of the convolutional layer. Each convolution kernel corresponds to a specific feature map for network weight learning. The output of Conv2_Relu is used as the input data of Conv3_Relu and Conv4 respectively under the action of the activation function Relu. The convolution operation and the Relu formula are defined as:

$$y_j^l = \sigma\left(\sum_{i \in M_j} y_i^{l-1} w_{ij}^l + b_i^l\right) \tag{1}$$

$$Relu(y) = \begin{cases} y & (y > 0) \\ 0 & (y \le 0) \end{cases} \tag{2}$$

where $y_j^l$ represents the result of processing $l$ convolutional layers by $j$ convolution kernels, $\sigma$ is the activation function, $w$ represents the weight, and $b$ is the bias.

Then, the output results of Conv2 and Conv3 are processed by the cross-layer aggregation module. The feature map for each convolution kernel output of the convolutional layer (Conv4) is down sampled by the pooling layer (Max_pooling1), and the purpose is to reduce dimensionality and maximize region feature extraction of data, and the output results are under the action of the connection layer FC1_layer and FC2_layer, and an M/2-dimensional data set is obtained. The calculation formula for the pooling layer is:

$$z_j^l = \beta\left(w_j^l down\left(z_j^{l-1}\right) + b_j^l\right) \tag{3}$$

where $down(z)$ represents a down-sampling operation on the matrix element $z$. Pooling operations are similar to ordinary neuron calculations, and it also has weights and bias.

Similarly, the output result of the third convolutional layer(Conv3_Relu) outputs an M/2-dimensional data set under the action of the convolutional layer(Conv5), the pooling layer(Max_pooling2), and the two fully-connected layers FC3_layer, FC4_layer.

Then use the concat() function in Tensorflow to perform the merge operation on the output results of the FC2_layer and FC4_layer layers to generate a data set of size M. Finally, the data is classified using the SoftMax layer.

#### 2) BACK PROPAGATION

In ICNN, the back propagation process of training is designed as follows:

First, the overall error parameter value loss is calculated by using the SoftMax layer to classify the sample of the training set.

Finally, back propagation is performed based on the loss. The purpose of back propagation is to iteratively adjust the weights and bias of each layer network based on the error between the actual output value and the ideal output value until the model achieves a good convergence effect. In the process, in order to quickly find the optimal weight $w$ and the bias $b$ and the output $f(x)$ of the network can fit all the training inputs x, a loss function $C(w, b)$ is set for finding the optimal combination of parameters to quantify the degree of model fit. The Stochastic Gradient Descent (SGD) algorithm provides a way to minimize this loss function. The loss function is defined as:

$$C\ (w, b) \equiv \frac{1}{2n} \sum_x y^{(x)} - a^2 \tag{4}$$

where $w$ and $b$ represent the set of network weights and offsets respectively, $n$ is the number of training input data, and a represents the vector output when the input is $x$. The parameters $w$ and $b$ are defined as:

$$w \to w_k^{'} \equiv w_k - \eta \frac{\theta C}{\theta w_k} \tag{5}$$

$$b \to b_l^{'} \equiv b_l - \eta \frac{\theta C}{\theta b_l} \tag{6}$$

where $\eta$ is the learning rate, partial derivatives $\theta C/\theta w_k$ and $\theta C/\theta b_l$ represent the change rate of arbitrary weights and arbitrary bias.

The loss function value calculation process is shown in FIGURE 2.

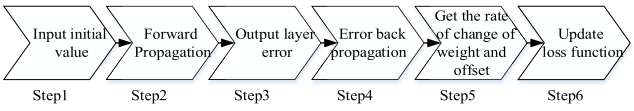



**FIGURE 2.** Flow chart of loss function calculation.

The calculation steps are designed as follows:

Step 1 Set the initial activation value $a^1$ and enter it;

Step 2 Calculate the weighted sum $z^l = w^l a^{l-1} + b^l$ and the activation value of each layer node $a^l = \sigma(z^l)$ where $l = (1, 2, 3, \ldots, L)$, performing forward propagation;

Step 3 Calculate the output layer error $\delta^l = \nabla_a c \odot \sigma' z^L$ of each network and output it;

Step 4 According to each of the obtained output layer errors $\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1} \odot \sigma' z^l\right)$, where $l = (L-1, L-2, \ldots)$, the back propagation is performed;

Step 5 Calculate and output the change rate of arbitrary weight of the loss function $\theta C/\theta w_{jk}^l = a^{l-1} k \delta^l$ and the change rate of arbitrary bias $\theta C/\theta b_j^l = \delta_j^l$;

Step 6 Substituting the results of step 5 into equations (5) and (6) respectively to obtain the weight $w$ and the bias $b$, and then obtaining the loss function value according to formula (4).

ICNN's back propagation process is to calculate the error vector backwards from the last layer. This reverse movement is based on the output of the loss function in the network. In order to grasp the law that the loss function changes with the weight and offset of the previous layer, the stochastic gradient descent method and the repeated use of the chain rule can obtain the desired expression result in reverse, so that the optimal parameter combination can be obtained (i.e., the minimum Loss function).

In the ICNN training process, as the number of network layers increases and the number of convolution kernels increases, the feature information extracted from the network is also increasing. The network itself converges the model through continuous feature selection and parameter optimization, which reflects the effectiveness of ICNN's deep learning process in modeling intrusion detection samples.

## III. WIRELESS NETWORK INTRUSION DETECTION MODEL

### A. WIRELESS NETWORK INTRUSION DETECTION MODEL ARCHITECTURE

Based on ICNN, this paper proposes an ICNN Based Wireless Network Intrusion Detection Model (IBWNIDM), which is mainly composed of data preprocessing module, ICNN model training module, and classification detection module. The architecture of IBWNIDM is shown in FIGURE 3. Shown.

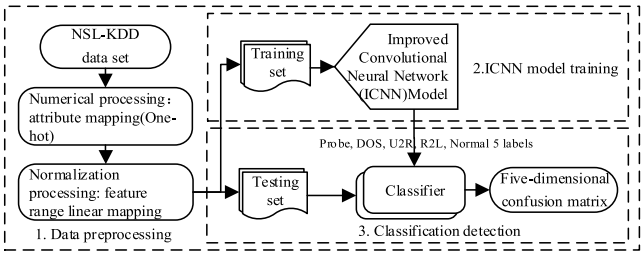



**FIGURE 3.** IBWNIDM architecture.

### B. WIRELESS NETWORK INTRUSION DETECTION MODEL ARCHITECTURE

#### 1) DATA PREPROCESSING

The module includes two operations of numerical processing and normalization. The purpose of the module is to provide a standardized input data set for network training. The processing of the module is designed as follows:

First, the continuous and discrete symbolic data in the data set is numerically processed using the One-hot method.

Then, the processed numerical data is linearly mapped to the feature range to obtain a standardized network input data set;

#### 2) ICNN TRAINING

The module includes two processes of forward propagation feature extraction and back propagation iterative optimization.

Forward propagation feature extraction: The pre-processed training data set is forwarded as Input Data, and the feature extraction is performed by ICNN's autonomous learning ability.

Back Propagation Iterative Optimization: The process of error back propagation is conducted according to the loss value obtained by ICNN training, and the parameters are optimized repeatedly until the model performs a good convergence effect.

#### 3) CLASSIFICATION DETECTION

Training classifiers for classification tags based on five sample categories: Probe, DOS, U2R, R2L, and Normal, and the pre-processed test data set is put into the trained classifier as test data. The classifier performs classification detection on the detected samples, and outputs a five-dimensional confusion matrix, which is the detection result.

## IV. DATA FEATURE ANALYSIS AND PREPROCESSING

### A. DATASET SELECTION AND CHARACTERIZATION

In this study, the NSL-KDD CUP data set [17] was selected as the baseline data set for model training and performance testing. Compared with the KDD CUP 99 data set, the NSL-KDD CUP data set deletes a large amount of redundant data, making the proportion distribution of the sample data more balanced and reasonable, and more usable, so it is better able to meet the verification experiment requirements of the test model of this study. The sample data characteristics of the above two data sets are the same, and each intrusion record in the data set has 42-dimensional features, which are detailed into 38-dimensional digital features, 3-dimensional symbol features, and one attack type label. The data types of the NSL-KDD CUP data set mainly include: normal data (Normal) and 4 large attack type data (Probe, DOS, U2R, R2L). The data of the 4 attack types can be specifically subdivided into 39 subclasses.

The NSL-KDD data set mainly includes three sub-data sets which are training set (KDDTrain), test set (KDDTest+) and test set (KDDTest$^{-21}$). The sample category distribution and feature classification are shown in TABLE 1 and TABLE 2 respectively.

### B. DATA PREPROCESSING

Data preprocessing includes two processes which are numerical processing and normalization processing.

**TABLE 1. Sample category distribution table.**

| Category | KDDTrain | KDDTest+ | KDDTest-21 |
|---|---|---|---|
| Probe | 45927 | 2421 | 4342 |
| DOS | 11656 | 7458 | 2402 |
| R2L | 995 | 2754 | 2754 |
| U2R | 52 | 200 | 200 |
| Normal | 67343 | 9711 | 2152 |
| Total | 125973 | 22544 | 11850 |

**TABLE 2. Feature classification table.**

| No. | Features | Types | No. | Features | Types |
|---|---|---|---|---|---|
| 1 | duration | continuous | 22 | is_guest_login | discrete |
| 2 | protocol_type | symbolic | 23 | count | continuous |
| 3 | service | symbolic | 24 | srv_count | continuous |
| 4 | flag | symbolic | 25 | serror_rate | continuous |
| 5 | src_bytes | continuous | 26 | srv_serror_rate | continuous |
| 6 | dst_bytes | continuous | 27 | rerror_rate | continuous |
| 7 | land | discrete | 28 | srv_rerror_rate | continuous |
| 8 | wrong_fragment | continuous | 29 | same_srv_rate | continuous |
| 9 | urgent | continuous | 30 | diff_srv_rate | continuous |
| 10 | hot | continuous | 31 | srv_diff_host_rate | continuous |
| 11 | num_failed_logins | continuous | 32 | dst_host_count | continuous |
| 12 | root_shell | discrete | 33 | dst_host_srv_count | continuous |
| 13 | num_compromised | continuous | 34 | dst_host_same_srv_rate | continuous |
| 14 | root_shell | discrete | 35 | dst_host_diff_srv_rate | continuous |
| 15 | su_attempted | discrete | 36 | dst_host_same_src_port_rate | continuous |
| 16 | num_root | continuous | 37 | dst_host_srv_diff_port_rate | continuous |
| 17 | num_file_creations | continuous | 38 | dst_host_serror_rate | continuous |
| 18 | num_shells | continuous | 39 | dst_host_srv_serror_rate | continuous |
| 19 | num_access_files | continuous | 40 | dst_host_rerror_rate | continuous |
| 20 | num_outbound_cmds | continuous | 41 | dst_host_srv_rerror_rate | continuous |
| 21 | is_host_login | discrete | | | |

#### 1) NUMERICAL PROCESSING

Because the input of ICNN is a digital matrix, the data with symbolic features in the test data set is mapped to a digital feature vector using a one-hot encoding method. This process is mainly for the following three characteristics:

1)For the three types of attributes of the protocol_type type feature: TCP, UDP, and ICMP, which are respectively encoded as binary vectors (1, 0, 0), (0, 1, 0) and (0, 0, 1);

2)Converting 70 symbol attributes included in the service type feature into a 70-dimensional binary feature vector by encoding;

3)The 11 kinds of symbol attributes included in the flag type feature are changed into a 11-dimensional binary feature vector by encoding.

After numerical processing, the above three types of symbolic features become 84-dimensional binary feature vectors, and with the 38-dimensional digital features of the data set, the 42-dimensional features of each record in the data set eventually become 122-dimensional binary feature vectors.

#### 2) NORMALIZED PROCESSING

In the data set, the range of values between continuous feature data is significantly different. For example, the value range of the num_root type feature is [0, 7468], and the value range of the num_shells type feature is [0, 5]. It can be seen that the range of the minimum and maximum values of the two is very different. In order to facilitate the arithmetic processing and eliminate the dimension, a normalized processing method is adopted, and the range of values of each feature is uniformly linearly mapped in the interval [0, 1]. The normalized formula is:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{7}$$

where $X_{max}$ and $X_{min}$ represent the maximum and minimum value of the feature separately.

## V. EXPERIMENTS AND RESULTS

### A. EXPERIMENTAL ENVIRONMENT CONFIGURATION AND HYPERPARAMETER SETTINGS

In the research of this paper, the test verification and comparison test of the test model are all carried out on the Linux operating system. Implementing ICNN and IBWNIDM in this article using Python's deep learning library Tensorflow programming. In order to improve the computational efficiency and reduce the training time, Tensorflow-GPU is used for parallel computing acceleration. The experimental software and hardware configuration environment is shown in TABLE 3.

**TABLE 3. Experimental environment configuration.**

| Hardware Configuration | Software Environment |
|---|---|
| Intel Core i7-7700 CPU @ 2.80GHz | Ubuntu16.04 |
| NVidia GeForce GTX 1050 | Python3.5.2 |
| 16.0GB RAM | Pycharm2019 |

After multiple and small-scale parameter combination training, according to the training and test results, determine the super-parameter settings of ICNN training as follows:

The network learning rate is 0.1. The weighting deactivation rate of the regularization method Dropout is set to 0.5, and the total number of experimental iteration training times is 300. Each iteration training of the experimental process is selected from the training set data. The amount of data is 1000 items and the number of iterations for each batch size is 50.

### B. EVALUATION INDICATORS

In this paper, Accuracy (AC), True Positive Rate (TPR) and False Positive Rate (FPR) are used as three key indicators to evaluate the performance of IBWNIDM detection.

Indicator 1 AC: The ratio of the number of correctly classified samples to the total number of samples tested.

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Indicator 2 TPR: The classifier correctly classifies positive samples as the ratio of the number of positive samples to the number of all positive samples.

$$TPR = \frac{TP}{TP + FN} \tag{9}$$

Indicator 3 FPR: The classifier classifies negative sample wrong as the ratio of the number of positive samples to the number of all negative samples.

$$FPR = \frac{FP}{FP + TN} \tag{10}$$

where TP (True Positive) represents the true positive class, which means the number of the sample belonging to the positive class correctly classified as the positive class; TN (True Negative) is the true negative class, indicating the number of the sample belonging to the negative class correctly classified as negative; FP (False Positive) is the false positive class, indicating the number of the sample belonging to the negative class misclassified as the positive class; FN (False Negative) is the false negative class, indicating the number of the sample belonging to the positive class misclassified as negative.

### C. EXPERIMENTAL DESIGN

Three test experiments were performed using the KDDTest+ data set and the NSL-KDD CUP data set. The specific experimental design is as follows:

Experiment 1: The five data labels Normal, Probe, DOS, U2R, and R2L in the test set (KDDTest+) are classified into five types, and the classification detection is performed. According to the five-dimensional confusion matrix (detection result) output by the classification detection, calculating the accuracy, true positive rate, and false positive rate indicators of IBWNIDM for the four types of attacks.

Experiment 2: Applying the NSL-KDD CUP data set to the other three typical neural networks LeNet-5 [18], DBN (Deep Belief Nets) [19] and RNN (Recurrent Neural Network) [20] used in intrusion detection methods. Conduct training and testing, and compare the test results with the detection results of IBWNIDM.

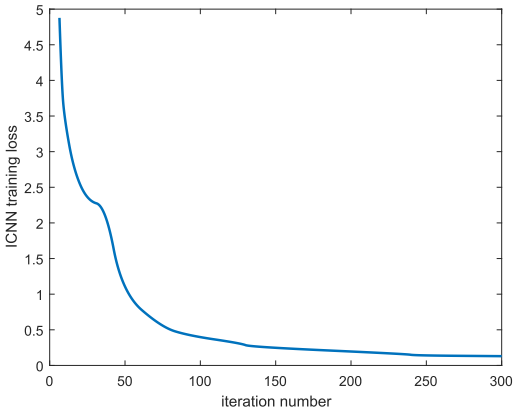



**FIGURE 4.** Relationship between iteration number and ICNN training error.

**TABLE 4.** KDDTest$^{-21}$ test set classification confusion matrix.

| Actual Class \ Predicted Class | Probe | DOS | R2L | U2R | Normal |
|---|---|---|---|---|---|
| Probe | **2322** | 52 | 6 | 0 | 41 |
| DOS | 75 | **6946** | 101 | 0 | 736 |
| R2L | 6 | 0 | **2637** | 7 | 104 |
| U2R | 9 | 0 | 6 | **174** | 11 |
| Normal | 27 | 75 | 176 | 14 | **9419** |

Experiment 3: With the NSL-KDD CUP data set, two typical intrusion detection models IDABCNN and NIDM-BCNN [15], [16] based on CNN intrusion detection method are trained and tested respectively, and the detection effect is compared with the detection effect of IBWNIDM.

### D. EXPERIMENTAL RESULTS

#### 1) CLASSIFICATION TESTING AND RESULTS

The experiment included two processes of training and testing. In the training process, the relationship between the training error and the number of iterations of IBWNIDM is shown in FIGURE 4. It can be seen from FIGURE 4 that as the number of iterations increases, the training error decreases gradually; When the number of iterations is 50, the error value is the smallest, which indicates that the super-parameter setting of the ICNN model structure design and model training is reasonable and can meet the detection requirements.

During the test, the five-dimensional confusion matrix output by the KDDTest+ and KDDTest$^{-21}$ test set after the classification test is shown in TABLE 4 and TABLE 5. These two matrices are the test classification result with five types of label samples, where the bold numbers represent the number of each sample category which is identified correctly.

The classification evaluation index results of the four attack types in the test set KDDTest+ and KDDTest$^{-21}$ are shown in FIGURE 5 and FIGURE 6 respectively.

**TABLE 5.** KDDTest$^{-21}$ test set classification evaluation indicator results.

| Predicted Class / Actual Class | Probe | DOS | R2L | U2R | Normal |
|---|---|---|---|---|---|
| Probe | **3986** | 156 | 92 | 0 | 108 |
| DOS | 96 | **1997** | 128 | 0 | 171 |
| R2L | 77 | 0 | **2219** | 94 | 364 |
| U2R | 17 | 0 | 12 | **148** | 23 |
| Normal | 52 | 104 | 183 | 33 | **1780** |

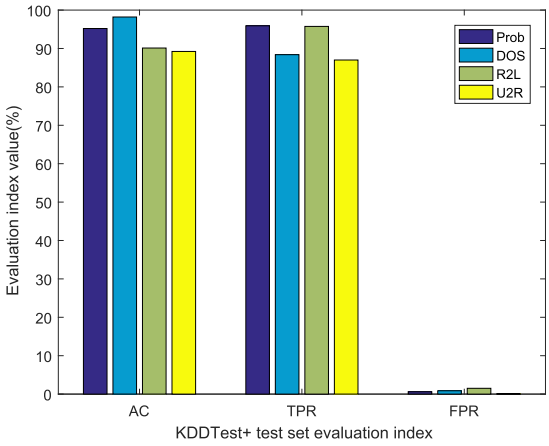



**FIGURE 5.** NSL KDD of NSL KDDTest+ test set evaluation index.

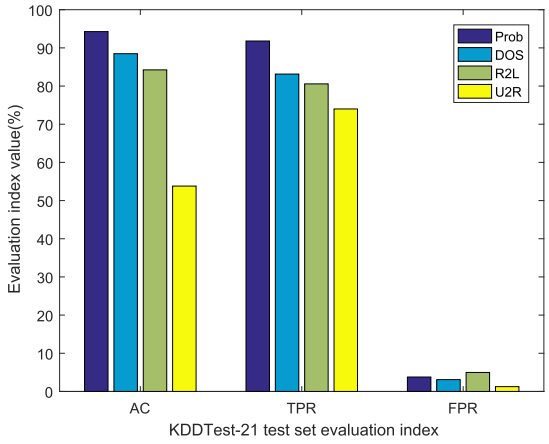



**FIGURE 6.** NSL KDD of NSL KDDTest$^{-21}$ test set evaluation index.

As can be seen from FIGURE 5 and FIGURE 6, the detection accuracy of IBWNIDM for U2R data is lower than the other three types. During model training, because the sample data volume of U2R is too small, feature extraction is less, so the data detection capability of classifier is weak during testing.

In order to more intuitively evaluate the IBWNIDM test experiment, the ROC curves of FPR and TPR were drawn by using the sample data after the classified test of test set KDDTest+ and KDDTest$^{-21}$. The ROC curves are shown in FIGURE 7 and FIGURE 8.

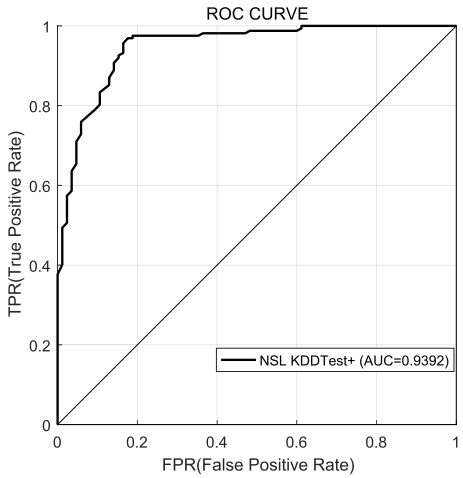



**FIGURE 7.** ROC CURVE of NSL KDDTest+ test set.

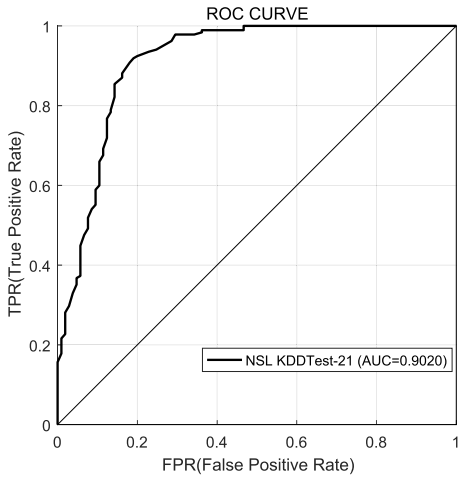



**FIGURE 8.** ROC CURVE of NSL KDDTest$^{-21}$ test set.

As can be seen from FIGURE 7 and FIGURE 8, the ROC curve fitting degree of IBWNIDM five-classification test is good. The AUC values of KDDTest+ and KDDTest$^{-21}$ obtained through calculation are 0.9392 and 0.9020 respectively, indicating that the classification model trained by ICNN has a strong sorting ability for positive samples.

### 2) COMPARATIVE EXPERIMENTS AND RESULTS

In order to further verify the effectiveness of IBWNIDM in wireless network intrusion detection, three typical neural network LeNet-5, RNN and DBN detection models applied in IDS were tested. Five classification test training and detection tests were performed on IBWNIDM and the above three models.

The test results of LeNet-5, RNN, DBN and IBWNIDM are compared as shown in TABLE 6. It can be seen from TABLE 6 that compared with the other three models, the accuracy of IBWNIDM detection is higher than that of LeNet-5 and DBN, which is slightly lower than that of RNN. The true positive rate is slightly lower than DBN

**TABLE 6.** Comparison of test results.

| Data set | Model | AC(%) | TPR(%) | FPR(%) |
|---|---|---|---|---|
| KDDTest+ | IBWNIDM | 95.36 | 95.55 | 0.76 |
| | LeNet-5 | 86.54 | 91.31 | 2.03 |
| | DBN | 92.45 | 95.68 | 0.85 |
| | RNN | 93.08 | 94.39 | 0.92 |
| $KDDTest^{-21}$ | IBWNIDM | 90.91 | 92.42 | 3.96 |
| | LeNet-5 | 82.36 | 87.93 | 3.24 |
| | DBN | 88.69 | 89.48 | 4.46 |
| | RNN | 79.47 | 80.58 | 1.75 |

**TABLE 7.** Test results of 3 models.

| Data set | Model | AC(%) | TPR(%) | FPR(%) |
|---|---|---|---|---|
| KDDTest+ | IBWNIDM | 95.36 | 95.55 | 0.76 |
| | IDABCNN | 92.78 | 90.61 | 0.95 |
| | NIDMBCNN | 97.34 | 91.33 | 0.82 |
| $KDDTest^{-21}$ | IBWNIDM | 90.64 | 91.74 | 3.87 |
| | IDABCNN | 89.33 | 87.49 | 3.63 |
| | NIDMBCNN | 88.16 | 83.71 | 1.02 |

and slightly higher than LeNet-5 and RNN, and the false positive rate are lower than the other three models. In the test set $KDDTest^{-21}$, the detection accuracy and true positive rate of IBWNIDM are higher than the other three types of models, the false positive rate is lower than DBN, slightly higher than LeNet-5 and RNN, which indicates that ICNN has a strong ability to identify intrusion detection data samples.

In order to verify the validity of IBWNIDM better, model training and testing experiments were carried out on IDABCNN [15] and NIDMBCNN [16] respectively. In order to ensure the comparability of the experimental results, the NSL-KDD data set was used for the test samples, and the test results are shown in TABLE 7.

As can be seen from TABLE 7, in the test set KDDTest+, the detection accuracy of IBWNIDM is lower than NIDMBCNN and slightly higher than IDABCNN, while the true positive rate is slightly higher than that of IDABCNN and NIDMBCNN, and the false positive rate is lower. This indicates that the NSL-KDD data set is more helpful for improving the experimental detection effect. In the test set $KDDTest^{-21}$, the detection accuracy and true positive rate of IBWNIDM are higher than the other two models, but the false positive rate is slightly higher than the latter two, so IBWNIDM has further room for improvement. In summary, the research shows that the cross-layer aggregation network structure design method of IBWNIDM can ensure that the model can better extract feature information in the training process, and also reflects that the fitting degree of IBWNIDM is relatively ideal, and the trained network model has a good classification effect on sample data.

## VI. CONCLUSION

Aiming at problem of wireless network intrusion detection technology based on deep learning method that has low detection efficiency and is prone to face over-fitting and generalization issues in the model training process. This paper proposes a wireless network intrusion detection based on improved convolutional neural network. The classification training and test experiments are carried out in IBWNIDM using the pre-processed training set and test set data. The experimental results show that the accuracy and true positive rate of intrusion detection of IBWNIDM are higher and the false positive rate is lower. Overall, IBWNIDM leverages the advantage of the deep learning model for feature extraction of sample data.

The next step in the improvement of IBWNIDM focuses on the following two aspects: 1) For the stochastic gradient descent algorithm (SGD), the gradient dispersion is easy to occur in the model training process, and the loss function is easy to fall into the local optimal solution. Consider using simulated annealing. Group intelligent optimization algorithms such as particle swarm optimization or ant colony algorithm for parameter tuning. 2) Try to use other data sets for training and testing verification experiments to improve the algorithms and methods according to the experimental results, and further improve the generalization ability and effectiveness of the intrusion detection model.

## REFERENCES

[1] K. Zheng *et al.*, "Algorithms to speedup pattern matching for network intrusion detection systems," *Comput. Commun.*, vol. 62, pp. 47–58, May 2015. doi: 10.1016/j.comcom.2015.02.004.

[2] S. Kim, "Pattern matching acceleration for network intrusion detection systems," in *Proc. Int. Conf. Embedded Comput. Syst., Archit., Modeling, Simulation*, Berlin, Germany, Jul. 2005, pp. 289–298.

[3] P. Li and W. H. Zhou, "Hybrid intrusion detection algorithm based on k-means and decision tree," *Comput. Modernization*, vol. 37, no. 6, pp. 12–16, Dec. 2019.

[4] Y. M. Qi *et al.*, "Research on SVM network intrusion detection based on PCA," *Netinfo Secur.*, no. 2, pp. 15–18, Feb. 2015. doi: 10.3969/j.issn.1671-1122.2015.02.003.

[5] Y. Xu and H. Zhao, "Intrusion detection alarm filtering technology based on ant colony clustering algorithm," in *Proc. 6th Int. Conf. Intell. Syst. Design Eng. Appl.*, Washington, DC, USA, May 2016, pp. 470–473.

[6] X. Wang, "Design of temporal sequence association rule based intrusion detection behavior detection system for distributed network," *Modern Electron. Techn.*, vol. 41, no. 3, pp. 108–114, Jan. 2018. doi: 10.16652/j.issn.1004-373x.2018.03.025.

[7] S. S. Roy, A. Mallik, R. Gulati, M. S. Obaidat, and P. V. Krishna, "A deep learning based artificial neural network approach for intrusion detection," in *Proc. Int. Conf. Math. Comput.*, Berlin, Germany, Jan. 2017, pp. 44–53.

[8] M. A. Yong, "A network intrusion detection schemer based on fuzzy inference and Michigan genetic algorithm," *Electron. Des. Eng.*, vol. 24, no. 11, pp. 107–110, Jun. 2016. doi: 10.14022/j.cnki.dzsjgc.2016.11.032.

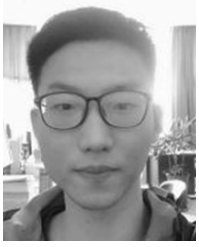
[9] H. Chen, G. X. Wan, and Z. J. Xiao, "Intrusion detection method of deep belief network model based on optimization of data processing," *J. Comput. Appl.*, vol. 37, no. 6, pp. 1636–1643, Jun. 2019. doi: 10.11772/j.issn.1001-9081.2019.06.1636.

[10] N. Gao, L. Gao, Q. Gao, and Hai Wang, "An intrusion detection model based on deep belief networks," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, Kunming, China, Nov. 2014, pp. 247–252.

[11] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21594–21961, 2017. doi: 10.1109/ACCESS.2017.2762418.

[12] 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.
[13] Q. Yuan and L. T. Lv, "Network intrusion detection method based on combination of improved ant colony optimization and genetic algorithm," *J. Chongqing Univ. Posts Telecommun.*, vol. 29, no. 1, pp. 85–89, Jan. 2019.
[14] M. J. Wei, Y. Y. Wang, and J. G. Jin, "An intrusion detection design based on improved immune algorithm," *J. Xidian Univ.*, vol. 43, no. 2, pp. 126–131, May 2016. doi: 10.3969/j.issn.1001-2400.2016.02.022.
[15] Y. Liu, S. Liu, and X. Zhao"Intrusion detection algorithm based on convolutional neural network," *Trans. Eng. Technol. Res.*, vol. 37, no. 12, pp. 1271–1275, Dec. 2019. doi: 10.12783/dtetr/iceta2017/19916.
[16] M. Wang and J. Li, "Network intrusion detection system based on convolutional neural network," *Netinfo Secur.*, vol. 3, no. 11, pp. 990–994, Nov. 2019. doi: 10.3969/j.issn.2096-1057.2019.11.005.
[17] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.
[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[19] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[20] J. Chung, C. Gulcehre, C. Gulcehre, and Y. Bengio, "Gated feedback recurrent neural networks," in *Proc. 2nd Int. Conf. Int. Conf. Mach. Learn.*, New York, NY, USA, Jul. 2015, pp. 2067–2075.

**HONGYU YANG** was born in Changchun, China, in 1969. He received the Ph.D. degree in computer science and technology from Tianjin University, China, in 2003. He is currently a Professor with the School of Computer, Civil Aviation University of China. His main research interest includes network information security.



**FENGYAN WANG** received the B.S. degree in software developing from Zhengzhou University, Zhengzhou, China, in 2017. He is currently pursuing the M.Eng. degree with the School of Computer, Civil Aviation University of China. His main research interest includes network information security.