Study on Intrusion detection model based on improved convolutional neural network

Kaiyan He *

Computer Science and Technology, Xidian University, Xian, 710000, China

*Corresponding author e-mail: hekaiyan@stu.xidian.edu.cn

Abstract. With the rapid development of information technology and cyberspace, information interactions between networks are becoming more frequent. At the same time, cyberattacks are posing more and more threats to network security through intrusions into computer systems or information systems. To address these problems, this paper proposes an improved convolutional neural network-based intrusion detection model (ICNN-IDS) to determine and classify specific types of intrusions after feature extraction and analysis of different network flow. The paper introduces the basic components of neural networks, including the convolutional layer, pooling layer and fully connected layer. Next, the experimental data set acquisition and pre-processing process are introduced, followed by the structural setup, the specific tuning process of the model, and the optimization of the model parameters are evaluated through experiments. In order to optimize the input feature matrix, the model adds a neuron mapping layer before the convolutional layer to convert sample data from 1D to 2D for the improvement of the model. The experimental results show that ICNN-IDS achieves 99.35% detection accuracy and a low false alarm rate of 0.21% on the KDD99 dataset with optimal parameter settings, which has significant improvement over existing detection models.

Keywords: Deep learning, convolutional neural networks, intrusion detection models.

1. Introduction

1.1. Background and significance of the study

Recently, with the rapid development of technology, the interaction of information between networks has become more frequent, while cyberattacks also pose a certain threat to network security through the intrusion of computer systems or information systems. According to the recent security report released by IBM security, it can be found that in 2019, information leakage due to malware intrusion into the network reached four times that of previous years. Therefore, how to effectively defend against network intrusions, such as break-ins and malicious injections has become a concern for network experts.
1.2. Current domestic and international research
The research on network intrusion detection systems at home and abroad has also been in the stage of development. Cisco proposed the Cisco Secure IDS network intrusion detection product, which can report irregular activities in the network in a timely manner; NAI used a layered working approach to develop a hybrid intrusion detection system; deep learning technology has also been continuously used to IDS, such as Huawei NIP IDS, ZEEK and so on[1].

2. Related knowledge

2.1. DEEP Learning and Neural Networks
Convolutional neural network in deep learning is an efficient detection model for classifying network flow and finding potential types of intrusion. CNN mainly contains three components: an input layer, an output layer and a hidden layer. The convolutional layer applies a number of convolutional filters (convolutional kernels) to the image; for each sub-region, the layer performs a set of mathematical operations with specific spatial extent and step values, producing a single value for the output[2]. The activation function then performs a non-linear operation by using some mathematical expressions. The pooling layer performs a dimensionality reduction on the data to reduce the dimensionality of the feature image, retaining the maximum value and discarding the rest. Finally, the fully-connected layer classifies the features where each node in the layer is connected to each node in the previous layer.

2.2. Intrusion detection system
Intrusion detection systems (IDS) are used to detect malicious activity on networks. IDS consists of four components: data source module, analysis module, response module and database. The data source module collects raw data such as logs and network flow to analysis module that determines the type of intrusion. It transmits the results to the response module; which issues alerts or takes active measures. The database contains the data involved in the entire intrusion detection. Traditional IDSs are located at the network boundaries and are based on a flow model. They warn when they detect an abnormal number of accesses or requests per unit of time. IDS can also use neural network as the basic model, using parameters trained from data sets to detect the type of intrusion.

3. Intrusion detection MODEL based on convolutional neural networks
Most of the current deep learning-based intrusion detection utilizes fully connected networks. As the number of layers in the model increases, the parameters increase exponentially and the convergence of the model is relatively much slower. Compared to the number of parameters in a fully connected network, the connections between layers in a CNN are sparser and the number of parameters is smaller. Therefore, the convergence is faster and the model is easier to train. In this paper, an intrusion detection model based on improved convolutional neural network (ICNN-IDS) is proposed for fast classification. This model consists of four main parts: data collection, data processing, model construction, and validation set test.

3.1. Data collection
This experiment uses KDD99 for training and validation. Each network connection in the KDD99 dataset is marked as normal or abnormal (attack), and the abnormal types are subdivided into 4 major categories with 39 attack types; each connection recorded in the KDD99 training data set contains 41 fixed feature attributes and one class identifier, which is used to indicate whether the connection record is normal or a specific type of attack[3].

3.2. Data processing
As each network traffic record consists of 41 fixed features and 1 class identifier, the 41-dimensional features include 38 numeric features and 3 characteristic features. Because the neural network can only process numeric input, the character features are converted to numeric values. All the features are
normalized, and the last class identifier is converted to a vector. The one-hot coding is used here, which is the mapping of discrete eigenvalues to a vector of length n containing only 0s and 1s. All the data sets were divided into two parts, the training set for parameter training and the test set for accuracy validation.

3.3. Model construction
In CNN, image data has local relevance, which means if a one-dimensional vector is directly converted into a two-dimensional feature matrix, the transformed matrix may destroy the local relevance of the image itself, so the feature extraction in the first step may destroy the basic features of the original input and lead to bias in detection. Therefore, a neuron mapping layer is introduced before input, and the output of this mapping layer is reshaped into a two-dimensional matrix which can optimize the mapping matrix and improve the detection performance of the model. In this mapping layer, the number of neurons is set to 900, and the output is reshaped into a 30*30*1 matrix.

The first convolutional layer uses 64 convolutional kernels of size [3x3], where the step size S=1, padding P=0 and the ReLU activation function. Batch normalization and pooling are then performed, with a pooling window size of 2. The purpose of batch normalization after the convolution operation is to suppress changes in the data distribution due to weight updates during training, which speeds up the convergence of the model. The second convolutional layer is set up with the same parameters as the first layer, and after batch normalization and pooling, the final two fully-connected layers are entered. The activation function of the first fully-connected layer is ReLU, and the output goes through a dropout layer to prevent overfitting. The last fully connected layer (output layer) is set up to output the type scores via a softmax function to predict probabilities for each type.

![Figure 1. Structure of the model](image)

3.4. Validation set
This experiment uses a 5-fold cross-validation method for training. The experimental data set is randomly divided into 5 equal parts, and 4 of them are used as the training set in turn, and the remaining is used as the validation set for the experiment. The mean value of the 5 results is calculated as the final result of experiments.

4. Experimental setup and results

4.1. Experimental environment
The computer used in this experiment is equipped with NVIDIA GeForce 940MX graphics card and i5-6200U processor, based on Win10 operating system platform. The experimental program was written in Python, using the Pandas and Numpy packages for data pre-processing. The neural network is built based on the Tensorflow and Keras frameworks.
4.2. Parameter selection and optimization

In neural networks, the reasonableness of parameter selection has a great impact on the model. To evaluate the model proposed in this paper, several experiments were set up in which some parameters were changed. Parameters can generally be adjusted in terms of the optimization algorithm, loss function, activation function, dropout rate.

For the loss function, cross entropy can be used to reflect the difference between the current predicted probability distribution and the actual true probability distribution, and is generally used in classification problems[4]. The intrusion detection in this paper is a multiclassification problem, so the multiclass logarithmic loss function ‘categorical_crossentropy’ in the cross-entropy loss function is used. At the same time, the number of iterations, as one of the parameters of the model depends on the convergence rate of the neural network when the accuracy is largely unchanged, and the subsequent training can be seen as redundant or can even result in overfitting. The number of iterations in the experiments is 20. Also, the batch_size, as one of the parameters, is related with the frequency of weight updates. If the batch_size is set too small, the convergence may be slow. The batch_size in the experiments is 512.

4.2.1. Experiment 1: Optimizer selection. There are many different optimizers for neural networks, and we mainly focus on the Adam algorithm and the stochastic gradient descent (SGD) algorithm. SGD iterates through each sample and can be iterated to the optimal solution with only a few thousands of samples if the sample size is large. Adam is an adaptive optimizer, where the learning rate is automatically adjusted based on the current situation during training, so that all weights are always updated with the appropriate learning rate.

![Figure 2. Loss and accuracy for the validation set with different optimizer](image)

By observing the loss and accuracy plots for the validation set under both optimizers, it is clear that the model is more accurate, has less loss, and converges faster when the Adam optimizer is chosen.

4.2.2. Experiment 2: Adam parameters. The learning rate is the rate of update of the weights in the model optimization. A small learning rate will result in a slow update of the weights, leading to too many iterations of training and too slow convergence of the model. A large learning rate will easily lead to the model staying at the local minimum and not converging to the global optimum. This experiment is set up to find the optimal learning rate to ensure accuracy with the fastest convergence.

The model uses the Adam optimizer based on Keras, which consists of three parameters: lr, beta1 and beta2. Lr is the learning rate, which controls how quickly the weights are updated. Beta1 is the exponential decay rate for first-order moment estimation, typically set to 0.9. Beta2 is the exponential decay rate for second-order moment estimation, typically set to 0.999. We controlled other parameters the same and set lr to 0.01, 0.001, and 0.005, respectively.
It can be seen that when lr is set to 0.01, the model still has a large loss in the end than the other two, and when lr is set to 0.0005, the model converges slower when the same loss is reached. Therefore, we set lr=0.001, beta1=0.9 and beta2=0.999 in Adam's algorithm.

4.2.3. Experiment 3: dropout connections. The dropout layer is to reduce the number of intermediate features and thus redundancy. It can prevent overfitting by randomly selecting some of the neurons according to probability. The parameter is the connection probability P, which determines the probability that the neurons in that layer will participate in the connection during the training of the model. When P is small, some useful information may be discarded, thus affecting the accuracy of the model. When P is large, the model may overfit. In this experiment, we set the connection probabilities P=0.2, 0.4, 0.6 and 0.8 respectively. The accuracy of the validation set with different P is shown below.

It can be seen that when P=0.6, the convergence rate faster and the accuracy is highest, so the dropout connection probability should be set to 0.6.

4.2.4. Experiment 4: activation function. The activation function introduces a non-linear factor to the neurons, allowing the neural network to approximate any non-linear function, so that it can be applied to a multitude of non-linear models. Common activation functions include Sigmoid, Relu and Tanh. In this experiment, Sigmoid, Relu and Tanh are chosen respectively on the loss curve of the validation set.
Figure 5. Loss for validation set with different activation

It can be seen that Relu activation function has a smaller loss compared to the Sigmoid and Tanh activation functions with almost the same rate of convergence. Therefore, we set Relu as the activation function for this model.

4.3. Analysis of experimental results

Using the KDD99 dataset, the 5-fold cross-validation method was used for training. ICNN-IDS was compared with two existing typical network intrusion detection models (CNN, LeNet-5). The evaluation index was based on the Accuracy, which defines the ratio of the number of correctly predicted samples to the total number of samples, and the higher Accuracy is, the better the detection model is. We introduce DR(detection rate) and FAR(false alarm rate). FAR refers to the total number of samples where the normal type is predicted to be the attack type as a proportion of the total number of samples where the true type is the normal type. DR refers to the ratio of the number of samples classified as attack type to the actual type of samples classified as attack type. The higher DR is and the lower FAR is, the better the model performs. The mathematical expressions of Accuracy, DR, FAR are shown below.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \\
\text{DR} = \frac{TP}{TP + FN} \\
\text{FAR} = \frac{FP}{FP + TN}
\]

TN is the number of connections classified as normal data. FN is the number of connections classified as normal data for attack data. FP is the number of attacks classified as attack data for normal data. TP is the number of attacks classified as attack data for attack data.

Using the above three parameters as judgement criteria, the experimental results are shown below.

| Model     | ICNN-IDS | CNN   | LeNet-5 |
|-----------|----------|-------|---------|
| Accuracy(%) | 99.35    | 98.65 | 99.06   |
| FAR(%)        | 0.21     | 0.27  | 0.34    |
| DR(%)        | 99.26    | 98.86 | 99.12   |

It can be seen that the accuracy of the model is partially improved compared to existing models for a similar detection time, and the false alarm rate is also reduced to a certain extent, which is promising for practical applications. In addition, we can observe the accuracy on Dos and Probe type for comparison.
Table 2. Results on Dos type of different models

| Model       | ICNN-IDS | CNN     | LeNet-5 |
|-------------|----------|---------|---------|
| Accuracy (%)| 99.54    | 99.47   | 99.28   |
| FAR (%)     | 0.29     | 0.36    | 0.34    |

Table 3. Results on Probe type of different models

| Model       | ICNN-IDS | CNN     | LeNet-5 |
|-------------|----------|---------|---------|
| Accuracy (%)| 99.15    | 98.24   | 99.04   |
| FAR (%)     | 0.56     | 0.52    | 0.78    |

It can be seen that the detection accuracy of the model proposed in this paper is better on Dos type and Probe type, and the false alarm rate has been improved to a certain extent on both. However, it can be found that the accuracy of the proposed model on U2R type and R2L type is not significantly improved compared with the existing classical models. The lower accuracy of the U2R type is probably due to the low number of training instances, which makes the model not fit well on this type of attack during training.

In summary, the INCC-IDS model has higher accuracy, detection rate and lower false alarm rate than the CNN and LeNet-5 model, and performs better on both Dos and Probe attack types. Due to the uneven distribution of the data, the accuracy for the U2R and R2L type is low and could be improved.

5. Summary
This paper presents an improved convolutional neural network-based intrusion detection model (ICNN-IDS) to detect types of intrusions in different network flow and the KDD99 data is used for model training. The model consists of data collection, data processing, model construction and data validation. In order to optimize the input feature matrix, a neuron mapping layer is added before the convolutional layer. The optimization process is shown in terms of optimizer selection, optimizer hyperparameters, dropout rate and activation function. Accuracy, FAR and DR are evaluated in the model compared with existing intrusion detection models. The experimental results show that ICNN-IDS can achieve 99.35% detection accuracy and a low false alarm rate of 0.21% on the KDD99 dataset with the optimal parameter settings, which is a significant improvement over existing intrusion detection models.

References
[1] Wang Yuchao. Research on network intrusion detection based on deep learning[D]. Qinghai Normal University, 2020.
[2] Sinh-Ngoc Nguyen, Van-Quyet Nguyen, Jintae Choi, Kyungbaek Kim. "Design and implementation of intrusion detection system using convolutional neural network for DoS detection", Proceedings of the 2nd International Conference on Machine Learning and Soft Computing - ICMLSC '18, 2018
[3] Jiayu Sun, Xinzhou Wang, Naixue Xiong, Jie Shao. "Learning Sparse Representation With Variational Auto-Encoder for Anomaly Detection", IEEE Access, 2018
[4] "Artificial Intelligence and Security", Springer Science and Business Media LLC, 2020