A Fast Deep Learning Method for Network Intrusion Detection Without Manual Feature Extraction

Wang Yue*, Jiang Yiming, Lan Julong
PLA Information Engineering University, Zheng Zhou, China
*li-xin@alu.hit.edu.cn

Abstract—Many deep learning models have been used in network intrusion detection to improve the accuracy of intrusion detection. In the existing deep learning methods, the input sample is either the feature vector extracted from the raw traffic data in advance, or directly the raw traffic data. These two categories of methods have their own limitations. In order to overcome the limitations, this paper proposes a preprocessing method based on multi-packets input unit and the raw traffic data is significantly compressed to improve the computational efficiency. To improve the computational efficiency and the detection accuracy, a deep learning model called F-CNN is designed. Experimental results on the benchmark dataset show that the proposed method has significantly improved detection accuracy and training efficiency compared with existing models with good performance.

1. INTRODUCTION
Deep learning method has been widely employed in the field of network intrusion detection. There are many network intrusion detection methods based on deep learning models, which are considered as effective means of network intrusion detection[1]. Although deep learning methods have the ability to learn effective features, most of them in intrusion detection field still rely on well-designed features. Feature extraction is the key step because it has much impact on the detection accuracy. In order to overcome the limitation of manual feature extraction, some deep learning methods based on the raw traffic data have been proposed. However, this category of methods have the following defects: (1) The large amount of raw traffic data leads to a significant reduction in the computational efficiency, so this method is not practical enough in the real environment where the network traffic data increase exponentially; (2) In the process of using raw traffic for preprocessing, information loss will be caused. In addition, the two categories of methods have a common problem. That is the input sample flow usually contains a lot of packets. Therefore, it is required that the entire flow must be received before it is detected, so that the detection results are deferred.

In order to overcome the limitations of the above two categories of methods, this paper proposes a data preprocessing method that compresses the raw traffic data, the detection accuracy can be improved and the high calculation efficiency is achieved while automatically learning the effective feature representation.

The organization of this paper is as follows: Section 2 summarizes the research works of machine learning methods for intrusion detection and the Gabor convolutional networks (GCN); Section 3 describes the method proposed in this paper in detail; Section 4 presents the experimental comparison results; Section 5 summarizes this paper and gives the future research direction.
2. Related Works
A large number of shallow machine learning algorithms have been applied to the classification of network anomalies. Among them, supervised learning algorithms include Bayesian networks[2], decision trees[3], and support vector machines (SVM)[4]. Using shallow machine learning methods in intrusion detection has the following limitations: 1) lacking of a set of consensus input features to solve specific goals, such as network security, anomaly detection, traffic classification, etc.; 2) continuous changes in network measurement statistics may cause static manual features to fail. Therefore, when intelligent analysis and high-dimensional data learning are required, shallow machine learning usually does not meet the requirements.

Yin C et al.[7] studied the impact of RNN (Recurrent Neural Network) neuron number and learning rate on the accuracy of intrusion detection. Experiments show that the RNN model not only has powerful modeling capabilities for intrusion detection, but also has both binary and multi-class classification performance with high accuracy. However, the input of the RNN model was the feature data extracted from the raw traffic. Vinayakumar R et al.[9] designed a DNN (Deep Neural Networks) model and optimize the design of the network structure and parameters. The above deep learning methods have shown good performance in traffic classification tasks, but they use deep learning models after some preprocessing of the data or the construction of some manual feature sets, which require expert knowledge intervention. Kim J et al.[10] argued that, taking advantage of deep learning, that is, learning advanced representations of data through a combination of complex network structures and non-linear transformations, can obtain higher detection accuracy. Marín G et al.[11] used the incoming byte stream to train the CNN-LSTM (Long Short Term Memory) model, which can learn both temporal and spatial features at the same time, without complex preprocessing steps or domain expert intervention, to make the method universal and flexible. However, a large amount of raw traffic data is directly used as the input of the deep model, resulting in a low training efficiency of the deep model.

3. Method Design
A deep model named F-CNN based on the proposed preprocessing method for raw network traffic is proposed. The key steps in deep learning for data process include data preprocessing, model training, and model testing.

3.1 Data Preprocessing
Marín G et al.[11] use packet-based and flow-based raw data as inputs, and design corresponding detection models respectively. The experimental results show that flow-based detection performance is higher than packet-based detection. However, the flow-based detection method also has limitations that are serious information loss and impacts the detection accuracy after being intercepted. The multi-packets-based method considers both fine-grained detection of data packets as data units and coarse-grained detection of flows as data units. In data packets are regarded as one input unit. For each multi-packets unit, it is still like the flow-based method for filling or interception operations, but the information lost by the multi-packets-based interception operation is significantly smaller than the flow-based method. In order to overcome the problem of using raw flow data as input which leads the low calculation efficiency, the data is compressed before training. A simple and efficient compression method is used, that is, the bytes in multi-packets unit are divided into many parts on average, each part contains step bytes, and the ASCII code value corresponding to the step bytes in each part is averaged, namely,

\[ p_i = \left\lfloor \frac{\sum_{j=1}^{\text{step}} \text{ASCII}(B_{i,j})}{\text{step}} \right\rfloor \]

(1)

\( p_i \) denotes the average of the ASCII values of all bytes in i-th part. \( B_{i,j} \) denotes j-th byte in i-th part. If an input sample contains N parts, after compressed, the input sample becomes \( \{p_1, p_2, \ldots, p_N\} \). When \( \text{step}=50 \), the data compression rate is as high as 98%. Min-max scaler linear function normalization is used: the raw data is linearly transformed and the new data is mapped to \([0, 1]\). The normalization function min-max scaler is defined as:
\[ x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \]  

(2)

3.2 Model Design
Because CNN itself is suitable for processing high-dimensional data, the optimized model structure given by the design is not sensitive to the compression parameter step, which greatly enhances the adaptability of the model. The structure of the CNN model designed in this paper is shown in Table 1.

| Layer Number | Description |
|--------------|-------------|
| 1            | Convolutional Layer (48 units, kernel size: 3, activation function: Relu) |
| 2            | Dropout Layer (Dropout Ratio: 0.1) |
| 3            | Convolutional Layer (48 units, kernel size: 3, activation function: Relu) |
| 4            | Pooling Layer (size: 2) |
| 5            | Convolutional Layer (128 units, kernel size: 3, activation function: Relu) |
| 6            | Dropout Layer (Dropout Ratio: 0.1) |
| 7            | Convolutional Layer (128 units, kernel size: 3, activation function: Relu) |
| 8            | Pooling Layer (size: 2) |
| 9            | Flatten Layer |
| 10           | Dense Layer (128 units, kernel size: 3, activation function: Relu) |
| 11           | Dropout Layer (Dropout Ratio: 0.1) |
| 12           | Dense Layer (1 unit, activation function: Sigmoid) |

Gabor filter(GoF) is a narrow band-pass filter with direction and frequency selectivity, which has good local performance in space and frequency domain[13]. GoF is widely used in texture image segmentation, recognition, and other fields[12-15]. The filter consists of real part and virtual part, which are orthogonal to each other. The following is the mathematical expression of the Gabor function:

\[
\begin{align*}
\text{Complex Form:} & \quad g(\lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x^2 + y^2}{2\sigma^2}) \exp(i(2\pi \frac{x}{\lambda} + \psi)) \\
\text{Real Parts:} & \quad g(\lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x^2 + y^2}{2\sigma^2}) \cos(2\pi \frac{x}{\lambda} + \psi) \\
\text{Imaginary Parts:} & \quad g(\lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x^2 + y^2}{2\sigma^2}) \sin(2\pi \frac{x}{\lambda} + \psi) \\
\end{align*}
\]

(3)

In these formulas, the meaning of each parameter is:

\[
\begin{align*}
x & = x \cos(\theta) + y \sin(\theta) \\
y & = -x \sin(\theta) + y \cos(\theta) \\
\end{align*}
\]
1) Wave length (λ): its value is related to the dimension of the input sample, usually greater than or equal to 2. But the value of λ is not greater than one fifth of the dimension of the input sample.

2) Direction (θ): this parameter specifies the direction of parallel stripes of Gabor function, and its value is 0 to 360 degrees.

3) Phase offset (φ): its value range is -180 degrees to 180 degrees.

4) Aspect ratio (γ): the aspect ratio of space determines the shape of Gabor function. When γ = 1, the shape is circular. When γ < 1, the shape is elongated with the direction of parallel stripes.

5) σ is the standard deviation of the Gaussian Factor of the Gabor function.

Based on the optimized CNN model, the GoF is used to directly assign the weights of some convolutional layers, and training is no longer performed in these layers, while the remaining convolutional layers are still trained normally. Which layer the GoF use for weight assignment is analyzed, and the influence of each parameter of the GoF on the detection accuracy is discussed as the compression parameter step changes. The model and parameters of F-CNN finally are shown in Table2.

| Layer Number          | CNN                                               | F-CNN                                           |
|-----------------------|---------------------------------------------------|-------------------------------------------------|
| 1st Convolutional Layer | Convolutional Layer (48 units, kernel size: 3, activation function: Relu) | The weight value is set by the parameters generated by GoF, and this layer does not participate in the training. The remaining parameters are the same as CNN |
| 2nd Convolutional Layer | Convolutional Layer (48 units, kernel size: 3, activation function: Relu) | same as CNN                                      |
| 3rd Convolutional Layer | Convolutional Layer (128 units, kernel size: 3, activation function: Relu) | The weight value is set by the parameters generated by Gabor filter, and this layer does not participate in the training. The remaining parameters are the same as CNN |
| 4th Convolutional Layer | Convolutional Layer (128 units, kernel size: 3, activation function: Relu) | same as CNN                                      |

### 3.3 Training And Testing

In the training phase, the training data set is divided into a training data subset and a verification data subset. These samples are as the input of the training model and training step stops after training $e$ epochs. Different data sets have different values of $e$, and the appropriate $e$ is obtained through testing.

In the test phase, the test sample data is input into the training model outputted in the training phase for classification prediction. It is worth emphasizing that all of the above sample subsets are randomly selected from the total sample set and are not artificially selected. All deep learning models mentioned in this paper use an efficient adaptive learning rate optimizer: Adam$^{[2]}$. Adam designs independent adaptive learning rates for different parameters by calculating the first and second moment estimates of the gradient. The learning rate value is set to 0.0001, beta 1 = 0.9, beta 2 = 0.999. The loss functions of all deep learning models are binary cross entropy.

### 4. EXPERIMENTAL EVALUATION

The hardware environment of the experiment includes: Intel Xeon (Cascade Lake) Platinum 8269 2.5 GHz/3.2 GHz 4-core CPU, 8GB memory. On the public network data set IDS2018$^{[13]}$, F-CNN, CNN-LSTM$^{[11]}$, IDS-DNN$^{[9]}$ and literature$^{[11]}$ based on the raw traffic data are compared. For F-CNN, CNN-LSTM and IDS-DNN, input samples are the sample data of the raw traffic processed by
the preprocessing method proposed in this paper. The detection accuracy and training time performances are compared and analyzed.

### TABLE 3. IDS2018 three data subsets

| Data subset | Collecting Time | Attack Type | Total sample size |
|-------------|----------------|-------------|-------------------|
| Sub_DS1     | Wednesday-14-02-2018 | Benign | 663,808 |
|             |                 | FTP-BruteForce | 193,354 |
|             |                 | SSH-Bruteforce | 187,589 |
| Sub_DS2     | Thursday-15-02-2018 | Benign | 988,050 |
|             |                 | DoS-GoldenEye | 41,508 |
|             |                 | DoS-Slowloris | 10,990 |
| Sub_DS3     | Thursday-01-03-2018 | Benign | 235,778 |
|             |                 | Infiltration | 92,403 |

#### 4.1 Dataset

IDS2018 is a data set containing a large number of network traffic and system logs. Through analyzing the data set of IDS2018 by the feature generation tool CICFlowMeter-V3[18], about 80 types of feature data can be generated, which represent the network traffic and the activity behavior of data packets. According to the detection accuracy in the existing research, these selected data subsets are representative to verify the proposed model, because of their various detection accuracies. The three data subsets are described in Table 3.

#### 4.2 Measurable Metrics

In order to measure the performance of deep learning models and draw on the related metrics in the existing research, this paper mainly uses the global accuracy rate Acc, FPR (false positive rate), TPR (true positive rate) and the model training time to compare and analyze the involved models[9]. The definitions of Acc, FPR, and TPR are given as follows:

\[
\text{Acc} = \frac{(\text{True Positives} + \text{True Negatives})}{\text{Total Samples}}
\]

\[
\text{FPR} = \text{False Positives}/(\text{False Positives} + \text{True Negatives})
\]

\[
\text{TPR} = \text{True Positives}/(\text{True Positives} + \text{True negatives})
\]

#### 4.3 Experimental Comparisons

In order to make a fair comparison, the default parameters remain fixed. We compare the detection accuracy of F-CNN, CNN-LSTM, and IDS-DNN algorithms in IDS2018 data set (as shown in Fig.2). From Fig.1, we can see that F-CNN’s Accs on Sub-DS1 Sub-DS2, and Sub-DS3 are always higher than that of CNN-LSTM and IDS-DNN. Compared with CNN-LSTM, F-CNN improves the Acc up to 2.49%.

Fig.2 shows the changes of TPR when FPR changes from 0 to 1 on Sub-DS3 for the three methods (the value of step is 35 and remains unchanged), namely the RoC (Receiver Operating Characteristic) curves of these methods are compared. Fig.5 shows the changes of Acc on Sub-DS3 as the step of the compression parameter changes from 5 to 50. From these two figures, we can see that when the step changes between 5 and 35, the three algorithms mentioned in this paper have little impact on Acc; when the step is greater than 35, the detection accuracies of the three algorithms are significantly affected negatively.
Figure 1. Comparison of Acc of different models

Figure 2. Comparison of RoC curves

Figure 3. The influence of sample compression parameter step on detection accuracy

Compared with CNN-LSTM, when the step changes from 5 to 50, the training time of F-CNN is always lower than that of CNN-LSTM when the Acc is almost unchanged. As the Fig.3 shows, Compared with CNN-LSTM, F-CNN reduces the training time up to 23.82%. The reason is that F-CNN uses GoF to initialize the weight values of two convolution layers in CNN model, which saves part of the training time. When step is less than or equal to 15, the training time of IDS-DNN and F-CNN are almost the same, but the accuracy of IDS-DNN model is lower than F-CNN; when step is greater than 30, the training time of IDS-DNN is much higher than F-CNN and CNN-LSTM, while the Acc of IDS-DNN model is not significantly improved.

5. CONCLUSION

In order to address the limitations in the existing deep learning methods of network intrusion detection, this paper proposes a data preprocessing method based on multi-packets input unit and compressing the raw traffic data. To improve the computational efficiency and the accuracy of detection, optimal deep learning network models are designed. The comparative experiments show the advantages of the proposed method in the public data set. How to improve the proposed preprocessing methods, such as choosing better data compression methods, is our main research direction in the future.
REFERENCES

[1] N. Shone, T. Ngoc, V. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” IEEE transactions on emerging topics in computational intelligence, vol. 2, no. 1, pp. 41-50, 2018.

[2] A. Onik, N. Haq, and W. Mustahin, “Cross-breed type Bayesian network based intrusion detection system (CBNIDS),” in 18th International Conference on Computer and Information Technology (ICICT), pp. 407-412, 2015.

[3] S. Sahu and B. Mehtre, “Network intrusion detection system using J48 Decision Tree,” in 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2023-2026, 2015.

[4] B. Senthilnayaki, K. Venkatalakshmi, and A. Kannan, “Intrusion detection using optimal genetic feature selection and SVM based classifier,” in 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), pp. 1-4, 2015.

[5] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, “Deep learning approach for intelligent intrusion detection system,” IEEE Access, vol. 7, pp. 41525-41550, 2019.

[6] J. Kim, J. Kim, H. Thu, and H. Kim, “Long short term memory recurrent neural network classifier for intrusion detection,” International Conference on Platform Technology and Service (PlatCon). IEEE, pp. 1-5, 2016.

[7] G. Martin, P. Casas, and G. Capdehourat, “DeepSec meets RawPower-Deep Learning for Detection of Network Attacks Using Raw Representations,” ACM SIGMETRICS Performance Evaluation Review, vol. 46, no. 3, pp. 147-150, 2019.

[8] S. Luan, C. Chen, B. Zhang B, J. Han, and J. Liu, “Gabor convolutional networks,” IEEE Transactions on Image Processing, vol. 27, no. 9, pp. 4357-4366, 2018.

[9] Y. Chen, L. Zhu, P. Ghamisi, X. Jia, G. Li, and L. Tang, “Hyperspectral images classification with Gabor filtering and convolutional neural network,” IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 12, pp. 2355-2359, 2017.

[10] S. Sarwar, P. Panda, and K. Roy, “Gabor filter assisted energy efficient fast learning convolutional neural networks,” IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). IEEE, pp. 1-6, 2017.

[11] A. Alekseev, and A. Bobe, “GaborNet: Gabor filters with learnable parameters in deep convolutional neural network,” International Conference on Engineering and Telecommunication (EnT). IEEE, pp. 1-4, 2019.

[12] D. Kingma, and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[13] CSE-CIC-IDS2018 on AWS, https://www.unb.ca/cic/data sets/ids-2018.html.

[14] CICFlowMeter, https://www.unb.ca/cic/research/applications.html#CICFlowMeter