Hindawi
Security and Communication Networks
Volume 2021, Article ID 5533269, 13 pages
https://doi.org/10.1155/2021/5533269

Research Article

FCNN: An Efficient Intrusion Detection Method Based on Raw Network Traffic

Yue Wang, Yiming Jiang, and Julong Lan

Information Technology Institute, PLA Strategic Support Force Information Engineering University, Zhengzhou 450001, China

Correspondence should be addressed to Yue Wang; tomato_wy1996@163.com

Received 14 January 2021; Revised 5 May 2021; Accepted 30 May 2021; Published 14 June 2021

Academic Editor: Zhiyuan Tan

When traditional machine learning methods are applied to network intrusion detection, they need to rely on expert knowledge to extract feature vectors in advance, which incurs lack of flexibility and versatility. Recently, deep learning methods have shown superior performance compared with traditional machine learning methods. Deep learning methods can learn the raw data directly, but they are faced with expensive computing cost. To solve this problem, a preprocessing method based on multipacket input unit and compression is proposed, which takes m data packets as the input unit to maximize the retention of information and greatly compresses the raw traffic to shorten the data learning and training time. In our proposed method, the CNN network structure is optimized and the weights of some convolution layers are assigned directly by using the Gabor filter. Experimental results on the benchmark data set show that compared with the existing models, the proposed method improves the detection accuracy by 2.49% and reduces the training time by 62.1%. In addition, the experiments show that the proposed compression method has obvious advantages in detection accuracy and computational efficiency compared with the existing compression methods.

1. Introduction

At present, the security of cyberspace has been widely concerned by all sectors of society, and ensuring the security of network information and network equipment is the focus of network security. The research of anomaly-based Intrusion Detection System (IDS) is the main research direction in the field of intrusion detection. In recent years, deep learning has been widely used in many fields, including image processing and network traffic analysis [1–3]. Network intrusion detection is a subfield of network traffic analysis and deep learning has also been widely studied in the field. The network intrusion detection method based on deep learning model is considered to be an effective means of network intrusion detection [4].

There are many network intrusion detection methods based on deep learning model, and it is considered to be an effective way of network intrusion detection [5]. Although deep learning has the ability to learn effective features, in order to ensure the computational efficiency of the deep learning model and improve the real-time performance of intrusion detection, most of the studies still rely on well-designed features, and feature extraction is a key step in detection accuracy. The ever-changing attacks make these methods have low resolution to normal network behavior and some network attacks (such as the penetration attacks in CSE-CIC-IDS2018 data set [6]). To meet the needs of the intrusion detection technology of the future network, the dependence on manpower should be reduced as much as possible. In order to overcome the limitation of carefully designed features by experts, a deep learning method using raw traffic data for detection has emerged [7]. However, this kind of method has the following defects: (1) the amount of raw traffic data is large, which leads to the decrease of computational efficiency and the lack of practicability in the real environment where the network traffic increases exponentially. (2) In the process of using the raw flow for preprocessing, a large amount of data will be intercepted in order to ensure the uniform input format, resulting in information loss.

In addition, the detection result lags behind when using the raw traffic data directly, because the input sample flow often contains a lot of data packets or lasts for a long time (for example, many network data flows often last tens of seconds or even minutes). Therefore, it needs to be detected after the whole flow. In this paper, a data preprocessing method based on multipacket is proposed to solve the problem of detection lag. After fully considering the advantages and disadvantages of data packet and flow as data unit, respectively, m data packets are taken as an input unit. In this paper, the influence of the change of parameter m on the performance of the detection method is evaluated.

Some researchers believe that for intrusion detection, it is equally important to improve the computational efficiency of the deep learning model (including training time and model trainable parameters) [8–11]. Therefore, the goal of this paper is to improve the computational efficiency of the intrusion detection deep learning model while ensuring the detection accuracy.

Our previous work tried to directly use the raw traffic for analysis [12]. As an extension of this work, this paper compares multiple compression methods, adds a lot of details of algorithm design and discusses future research directions, and expands the content by 41.4%. This paper proposes to compare the raw traffic and compare and evaluate a variety of compression methods. Training after compressing the raw traffic alleviates the problem of inefficient calculation by using the raw flow data directly. By compressing the raw traffic and the raw data preprocessing method based on multipacket, it can improve the computational efficiency and ensure the detection accuracy while automatically learning the effective feature representation.

The main contributions of this paper are as follows:

(1) For the first time, a method of network traffic data preprocessing based on Naive Compression (NC) and multipacket is proposed, which not only overcomes the disadvantages of manual feature extraction, but also ensures the computational efficiency.

(2) A network intrusion detection model FCNN based on deep learning is designed. In order to further improve the computational efficiency and detection accuracy of the traditional convolution neural network, the weights of some convolution layers are assigned directly by Gabor filter. In addition, the network structure, activation function, and optimizer of the model are optimally designed, and the influence of Gabor filter parameters on accuracy with the change of sample dimension (sample dimension is related to compression ratio) is studied, and the optimal Gabor filter direction parameter θ in different sample dimension ranges is given.

(3) This paper compares the effects of a variety of compression preprocessing methods, including the proposed NC method, Principal Component Analysis (PCA) and Locally Linear Embedding (LLE), and deep learning Autoencoder (AE) compression methods, on the detection accuracy and training time of FCNN and other deep learning intrusion detection models. In this way, the optimal compression methods and related parameters suitable for intrusion detection based on raw traffic under different conditions are found. The experimental results verify the advantages of the NC compression method proposed in this paper.

The organization structure of this paper is as follows. Section 2 gives the related works in the field of intrusion detection based on deep learning. Section 3 describes the proposed method in detail. The experimental comparison results are presented in Section 4. Section 5 summarizes this paper and discusses future research direction.

2. Related Works

2.1. Deep Learning Methods in Intrusion Detection. Some deep learning methods have shown good performance in traffic classification tasks. Kim et al. [13] proposed that using deep learning to learn the high-level representation of data through the combination of a complex network structure and nonlinear transformation can strengthen the research in intrusion detection and obtain a higher detection rate. Yin et al. [14] and Kim et al. [6], respectively, proved that the research on RNN and CNN can promote the development of intrusion detection. Yin et al. studied the performance of RNN in binary and multiclassification and the influence of the number of neurons and different learning rates on the accuracy. By comparing the performance of RNN-IDS with other machine learning methods on the benchmark data set NSL-KDD, it is confirmed that RNN-IDS is very suitable for intrusion detection. Kim et al. developed CNN model for CSE-CIC-IDS2018 data set and compared it with RNN model to evaluate its performance. Experimental results show that when CNN model is applied to CSE-CIC-IDS2018 data set, the performance is better than RNN model. Vinayakumar et al. [15] designed a deep neural network (DNN) model for intrusion detection. By passing IDS data to many hidden layers to learn the abstraction and high-dimensional feature representation of IDS data, the proposed hybrid framework scale-hybrid-IDS-AlertNet can effectively monitor network traffic and host level events in real time. Mirsky et al. [16] present a plug-and-play NIDS which can learn to detect attacks without supervision and in an efficient online manner. This method uses autoencoders to collectively differentiate between normal and abnormal traffic patterns. The above methods promote the development of intrusion detection, but they all need more complex preprocessing operations, resulting in the lack of generality of the model.

In order to solve this problem, Wang et al. [17] proposed an intrusion detection method combining CNN with LSTM, which is the first time to use a feature learning method based on raw traffic in the field of intrusion detection. However, the structure of CNN-LSTM model is complex, which leads to low computational efficiency. Subsequently, Marín et al. [7] used the incoming byte flow training CNN-Long Short Term Memory (LSTM) model, which can learn both

temporal and spatial features without any preprocessing steps or domain expert intervention, so as to make the method universal and flexible. However, a large number of raw traffic data are directly used as the input of deep model, which will also lead to low training efficiency. In the latest research, Pham et al. [18] proposed a lightweight intrusion detection system, which converts the raw traffic data into two-dimensional image data and then uses the CNN model to learn effective features. This method improves the computational efficiency, but the experimental results show that the detection accuracy is low. Bovenzi et al. [19] propose H2ID, a two-stage hierarchical network intrusion detection approach for IoT scenarios. H2ID performs (i) anomaly detection via a novel lightweight solution based on a Multimodal Deep Autoencoder (M2-DAE) and (ii) attack classification, using soft-output classifiers. Results show false-positive rate is lower and with high efficiency. Compared with the deep learning methods based on feature extraction, the existing deep learning network intrusion detection methods based on raw traffic are more universal, but there is a common problem of high computational efficiency.

The comparison of the related works is presented in Table 1.

2.2. Gabor Convolution Networks. Gabor Convolution Network (GCN) is a deep convolution neural network using Gabor Filter (GoF). GCN is created by manipulating convolution filters learned by Gabor filter banks to generate enhanced feature graphs [20]. Gabor filter is widely used in image processing because it can efficiently extract frequency and direction features and has scale and rotation invariance. Chen [21] applied Gabor filter and CNN to hyperspectral image classification. Sarwar et al. [22] further combined Gabor filter with CNN and replaced some traditional filters of CNN with Gabor filter, so as to improve the training efficiency of CNN model. Similarly, Alekseev and Bobe [23] proposed that using a Gabor filter to replace the traditional filter in the CNN model so as to reduce the training parameters and improve the computational efficiency. The difference is that in the model proposed by Alekseev et al., only the first layer in the Gabor-CNN model is a parameter trainable Gabor filter layer.

Many methods combine Gabor filter with convolution network. Most of these methods solve the problem of image processing and cannot be directly applied to the field of intrusion detection. This paper mainly explores the Gabor network structure with excellent performance for intrusion detection.

The data compression method proposed in this paper is universal when dealing with the raw traffic, and the improved and optimized GCN deep learning method can take into account the computational efficiency and detection accuracy in network intrusion detection.

3. Intrusion Detection Method

The intrusion detection method proposed in this paper includes data preprocessing, model training, and model testing, and the flowchart is shown in Figure 1.

3.1. Data Preprocessing. Marín et al. [7] used packet-based and flow-based raw data as input and designed the corresponding detection models, respectively. The experimental results show that the performance of flow-based detection is better than that of packet-based detection. However, the flow-based detection method also has limitations. In the data preprocessing step, it needs to pad the short flow or intercept additional bytes in the long flow. In fact, the number of bytes contained in some flows is much larger than the interception threshold, which will cause serious information loss and affect the detection accuracy. In addition, because many flows often contain a large number of packets, the prediction can only be made after all the packets have been received, resulting in a lag in the prediction results, and the attacker may have successfully invaded before it is detected.

The method based on multipacket considers not only the fine-grained detection of packets as data units, but also the coarse-grained detection of flows as data units. In data preprocessing, m packets with the same 5-tuple (source IP, destination IP, source port, destination port, and protocol) [24–26] are composed of multipacket input unit. The selection of m value has a great influence on the detection accuracy and calculation efficiency, which will be discussed in the experimental part. The acquisition program has been waiting for m data packets with the same 5-tuple, but when the waiting time exceeds a threshold $T_{\text{time\_out}}$, the data packets received during this period (less m packets) will be used as an input unit. For m packets, their IP data except for IP address and IP port are used as input. In order to keep each input unit equal in length, it is necessary to fill or intercept the multipacket input unit. When the byte length s of the multipacket unit is less than the threshold S, the complement 0 operation is performed, and when the byte length s of the multipacket unit exceeds S, the final (s-S) bytes are intercepted. Through the padding or interception operation, the total length of all input units is guaranteed to be S bytes, as shown in Figure 2.

In order to overcome the problem of low computational efficiency caused by using the raw traffic data as input, the data is compressed before training. This paper proposes a simple and effective compression method, NC, which divides the bytes in the multipacket unit into several parts averagely, each part contains the same number of bytes and averages the ASCII code values of all bytes with the number of *step* in each part; namely,

$$v_i = \frac{\left[\sum_{j=1}^{j=\text{step}} \text{ASCII}\left(B_{i,j}\right)\right]}{\text{step}}, \tag{1}$$

where v_i represents the average ASCII value of all bytes in part i. $B_{i,j}$ represents the jth byte of part i. If the input sample consists of N parts, then after compression, the input sample will become $\{v_1, v_2, \ldots, v_N\}$. Take raw data "27 255 45 108 111 80" as an example. If $step = 2$, then $N = 3$ and $v_1 = (27 + 255)/2 = 141$, $v_2 = (45 + 108)/2 = 76.5$, and $v_3 = (111 + 80)/2 = 95.5$. The data after compressed is "141 76.5 95.5." The data compression method chosen in this paper is simple and efficient, because the compressed data still represents the raw data in most cases. Assuming that the

Table 1: Related work summary.

Categories	DL model used in related works	Literature	Advantages	Disadvantages
DL methods based on feature extraction	LSTM	[13]	High computational efficiency	More complex preprocessing operations; lack of generality
	RNN	[14]		
	CNN	[6]		
	DNN	[15]		
	Autoencoder	[16]		
DL methods based on raw traffic	CNN-LSTM	[7, 17]	Strong versatility; high accuracy	Low computational efficiency
	CNN	[18]		
	Autoencoder	[19]		

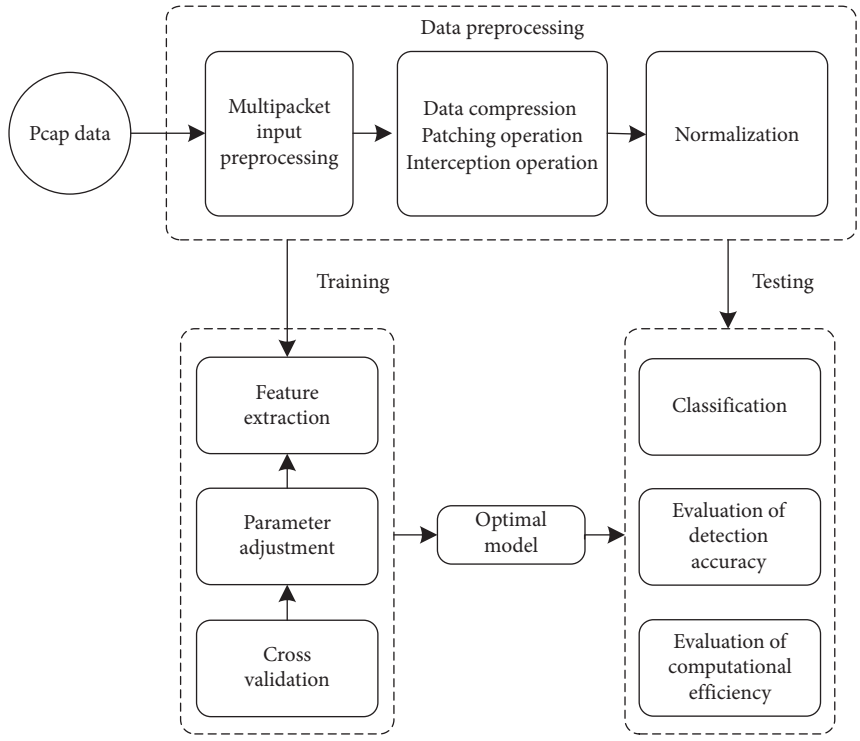

Figure 1: Intrusion detection flowchart.

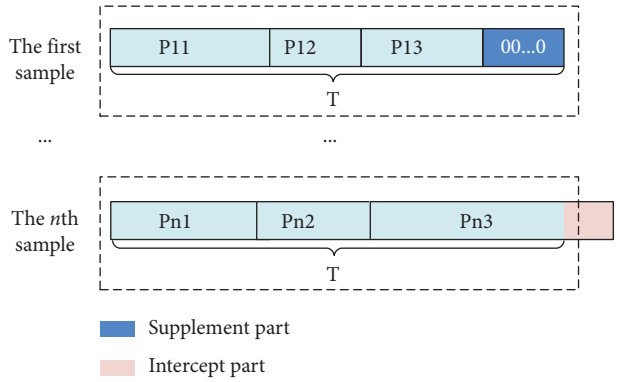

Figure 2: Multipacket preprocessing when $m = 3$.

two different parts are i_1 and i_2, the probability is very small. Furthermore, it is less likely that the compressed strings of two different packets or two different input units are exactly the same.

The min-max input scaling is used in this paper to have the networking focusing on the input component with the higher range. The minimum-maximum scaler of the normalization function is defined as

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \quad (2)$$

3.2. FCNN Model Structure. CNN consists of one or more convolution layers and pooled layers to form a multilayer neural network. The convolution layer shares many weights to detect the local connection of the elements in the upper layer and then samples the output of the convolution layer through the pooling layer to merge semantically similar elements. In this section, we first design the best CNN model for intrusion detection and then introduce in detail how to use GoF to further improve the performance of CNN.

The reasons why we choose CNN to train and predict the data processed by the proposed preprocessing method are as follows. First, because of the shared convolution kernel [27], CNN is very suitable for multidimensional data. Second, compared with other deep networks, CNN requires fewer parameters at the same network depth, which reduces complexity and speeds up the learning process [28]. Third, because of its ability to deal with complex data, CNN has also become a powerful tool for feature extraction and classification in the field of intrusion detection in recent years. The main consumption of CNN training (back propagation) is gradient calculation and weight updating of convolution layer and fully connected layer. In this paper, some weight kernels of CNN are replaced by Gabor filter to reduce the training complexity of CNN and minimize the loss of accuracy. When designing the CNN network structure, the activation function, the number of convolution layers, the size of convolution kernel, and the number of neurons are optimally selected and designed. In addition, multiple dropout layers are added to prevent overfitting.

Use X^l to represent the output of convolution layer l; because X^l has multiple output neurons, then

$$X^l = \left\{x_1^l, \ldots, x_p^l\right\}. \tag{3}$$

In a convolutional neural network, the output expression of each neuron is

$$x_j^l = \varphi\left(\sum_{i=1}^{q} x_i^{l-1} w_{ij}^l + b_j^l\right), \tag{4}$$

where q is the number of convolution neurons in layer 1-1, and w_{ij}^l represents the weight between ith neuron in layer l-1 and the jth neuron in layer l. b_j^l represents the bias of the jth neuron in layer l. $\varphi(g)$ represents the activation function. Like other convolution neural networks, the ReLU activation function is used in the convolution layer, and its expression is as follows:

$$\mathrm{ReLU}(x) = \begin{cases} 0, & x \le 0, \\ x, & x > 0. \end{cases} \tag{5}$$

In the pooled layer,

$$x_j^l = \varphi\left(\beta_j^l \mathrm{down}\left(x_j^{l-1}\right) + b_j^l\right), \tag{6}$$

where $\mathrm{down}(g)$ represents the downsampling function and β_j^l represents the weight of the jth neuron in the pooled layer.

The structure of the CNN model designed in this paper is shown in Table 2. The peculiar choice of the layers and number of filters is mainly based on the consideration of the prediction accuracy and computational efficiency of the model. When the layers and filters are less, the prediction accuracy of the model will be lower. When the layers and filters are more, the improvement of prediction accuracy is very slight but the computational efficiency of model greatly increases. The motivation of the design that the filter number of layer 1, 3 is less than 5, 7 from several existing works, such as [6].

GoF is a narrow-band band-pass filter with direction and frequency selectivity, which has good local performance in both space and frequency domain [21]. The impulse response of GoF can be defined as a sine wave (two-dimensional GoF is a sine plane wave) multiplied by a Gaussian function. Because of the multiplicative convolution, the Fourier transform of the impulse response of the Gabor filter is the convolution of the Fourier transform of the harmonic function and the Fourier transform of the Gaussian function. The filter consists of real part and imaginary part, which are orthogonal to each other. The mathematical expression of the Gabor function is given below:

$$\begin{aligned}
\text{Plural expression: } & g(\lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{\overline{x}^2 + \gamma^2 \overline{y}^2}{2\sigma^2}\right) \exp\left[i\left(2\pi \frac{\overline{x}}{\lambda} + \psi\right)\right], \\
\text{Real part: } & g(\lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{\overline{x}^2 + \gamma^2 \overline{y}^2}{2\sigma^2}\right) \cos\left(2\pi \frac{\overline{x}}{\lambda} + \psi\right), \\
\text{Imaginary part: } & g(\lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{\overline{x}^2 + \gamma^2 \overline{y}^2}{2\sigma^2}\right) \sin\left(2\pi \frac{\overline{x}}{\lambda} + \psi\right), \\
& \overline{x} = x \cos(\theta) + y \sin(\theta), \\
& \overline{y} = -x \sin(\theta) + y \cos(\theta).
\end{aligned} \tag{7}$$

In the formula, each parameter means the following:

(1) Wavelength (λ): its value is related to the size of the input sample, usually greater than or equal to 2, but not greater than 1/5 of the input sample.

(2) Direction (θ): this parameter specifies the direction of the parallel stripes of the Gabor function, with a value of 0 to 360°.

(3) Phase offset (φ): its value ranges from −180° to 180°.

Table 2: CNN model.

Sequence	Description
1	Convolution layer (48 units, kernel size: 3, activation function: ReLU)
2	Dropout layer (dropout rate: 0.1)
3	Convolution layer (48 units, kernel size: 3, activation function: ReLU)
4	Pooled layer (size: 2)
5	Convolution layer (128 units, kernel size: 3, activation function: ReLU)
6	Dropout layer (dropout rate: 0.1)
7	Convolution layer (128 units, kernel size: 3, activation function: ReLU)
8	Pooled layer (size: 2)
9	Flatten layer
10	Dense layer (128 units, kernel size: 3, activation function: ReLU)
11	Dropout layer (dropout rate: 0.1)
12	Dense layer (1 unit, activation function: sigmoid)

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

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

In the FCNN model, GoF is used to directly assign the weight of some convolution layers and no longer perform training in these layers, while the rest of the convolution layers can still be trained normally. In this paper, the network layer in which GoF is used for weight distribution is analyzed, and the influence of each parameter of GoF on the detection accuracy is discussed when the step size of compression parameters changes. The model and parameters of FCNN are shown in Figure 3.

In order to make the parameters in GoF suitable for network anomaly detection and reduce the loss of detection accuracy, each parameter in the test GoF is optimally selected: the wavelength (λ) value varies from 1 to 2, and different sample dimensions (the number of sample dimensions in this paper is related to the compression parameter step) have different optimal direction (θ) ranges (as shown in Table 3). The phase deviation (φ), length-width ratio (γ), and σ are the same as the conventional settings, and the values are 0, 1, and 1, respectively.

3.3. Compression Methods. In addition to the proposed simple compression method NC, this paper also evaluates the performance of other compression methods to compare the raw traffic data. Here, two traditional data dimensionality reduction (compression) methods PCA and LLE, which are commonly used and with relatively low computational complexity, and the deep learning method autoencoder is considered. Among them, PCA is an unsupervised learning method, which does not need to set parameters or interfere with the calculation according to any experience. Compared with the traditional dimensionality reduction methods such as PCA and LDA which focus on sample variance, LLE focuses on maintaining the local linear characteristics of samples when dimensionality reduction.

PCA is a statistical process, which uses orthogonal transformation to project a group of possible related variables into a group of linearly unrelated variables, namely, principal components. This projection follows the definition that the first principal component has the largest possible variance (that is, explaining the variability in the data as much as possible), and each subsequent element or component has the highest possible variance orthogonal to the previous element under constraints. And the number of principal components is less than or equal to the number of raw variables. In the PCA method, one of the most important problems is how to measure the quality of the projection vector. Mathematically, there are three ways to measure the pros and cons of projection. PCA is defined as the orthogonal projection of data on a low-dimensional linear space, which is called the principal subspace (principal subspace), so that the variance of the projected data is maximized, that is, the maximum variance theory. Similarly, it can also be defined as the linear projection that makes the average projection cost most expensive, that is, the minimum error theory.

LLE algorithm believes that each data point can be constructed from the linear weighted combination of its nearest neighbor points. The main steps of the algorithm are divided into three steps: (1) to find the k-nearest neighbor points of each sample point; (2) to calculate the local reconstruction weight matrix of the sample point from the nearest neighbor point of each sample point; and (3) to calculate the output value of the sample point from the local reconstruction weight matrix of the sample point and its nearest neighbor point.

The AE can learn automatically from the data samples. It is a neural network that uses back propagation to make the output value equal to the input value. It consists of two parts: the encoder and the decoder. Among them, the encoder compresses the input into a potential spatial representation, and the decoder reconstructs the output. The purpose of this is to enable the neural network of each layer of the encoder to extract features automatically. The output of the encoder can be regarded as compressed data. When evaluating the compression method, in order to take into account both computational efficiency and information fidelity, an optimal autoencoder model suitable for the raw network traffic is designed.

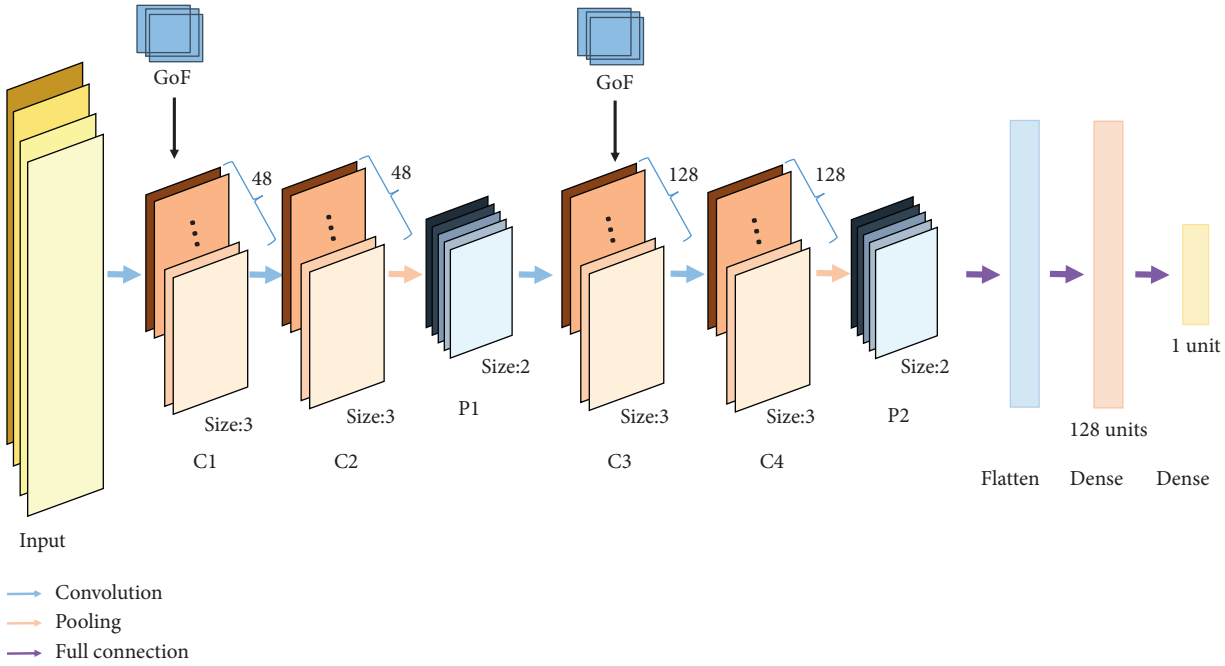

Figure 3: FCNN network structure diagram.

Table 3: Relationship between input sample dimension and θ range.

Sample dimension	θ range
<100	$0 \sim 0.5\pi$
100 ~ 500	$0 \sim \pi$
500 ~ 1920	$0 \sim 2\pi$

In order to compare the performance of compression algorithms more intuitively, in the next section, we keep the data dimensions of all the compressed methods the same, then evaluate the impact of compression methods on the performance of FCNN, and discuss the optimal compression methods under different conditions.

3.4. Training and Testing. In the training phase, the training data set is divided into a subset of training data and a subset of verification data. The training set and test set are randomly selected from the complete data set of IDS2018. In order to ensure that the samples contained in the training set and test set do not repeat as much as possible, the number of samples in the training set and test set is much smaller than the total sample number (for example, the total sample number of the Sub_DS1 subset is 1044751; the sample number of the training set and the test set is 3000 and 1800, respectively). At this stage, the training set and test set will be randomly selected for many times (usually 50 times), so the training set and test set can fully reflect the characteristics and distribution of the overall sample. Because the total sample is large enough to represent most of the possibilities of normal and aggressive behavior, and the training set and test set can better reflect the overall sample, the model has the ability to detect future attacks to a certain extent. These samples are used as inputs to the training model, and the training steps are stopped after the training period e. Different data sets have different e values, and the appropriate e values are obtained through testing.

In the testing phase, the test sample data are input into the training model output in the training phase for classification prediction. It is worth emphasizing that all the above sample subsets are randomly selected from the total sample set, not artificially selected. All the deep learning models mentioned in this article use an efficient adaptive learning rate optimizer: Adam [29]. Adam designs independent adaptive learning rates for different parameters by calculating the first and second moment estimates of the gradient. The learning rate is set to 0.0001, $beta1 = 0.9$, $beta2 = 0.999$, and the loss function is binary cross-entropy. The expression of binary cross-entropy L is

$$L = -\frac{1}{n}\sum_{i}^{n} y_i \log \overline{y_i} + (1 - y_i)\log(1 - \overline{y_i}), \tag{8}$$

where n is the total number of samples, $\overline{y_i}$ is the predicted value of the neural network to the sample i, and y_i is the label of the sample i.

4. Results and Discussion

4.1. Data Set. CSE-CIC-IDS2018 is a mixed data set of a large number of network traffic and system log, which contains 10 days of data, and the daily data form a subset of data with a total size of more than 400G. The data set includes 7 attack types and 16 attack subtypes, including brute force attacks, DoS attacks, surveillance network attacks, and penetration attacks. However, there are few deep learning intrusion detection methods for this data set [8]. By using the feature generation tool CICFlowMeter-V3 [30] to analyze the data

set of CSE-CIC-IDS2018, about 80 types of feature data can be generated, representing the activity behavior of network traffic and packets. On the basis of related research, two data subsets with high detection accuracy (Sub_DS1 and Sub_DS2) and a data subset with low detection accuracy (Sub_DS3) are selected as the evaluation data set of this experiment. According to the detection accuracy of the data set in the existing research, the three data subsets are representative, because the detection accuracy of the three data subsets is quite different in the existing research, and the performance of the model can be evaluated from different angles. Table 4 describes these three subsets of data.

4.2. Metrics. In order to measure the performance of deep learning models, global accuracy Acc, model training time, and normalized compressed data time are used to compare and analyze the models involved [13].

The relevant indicators are defined as follows:

(1) Acc = (True positive + True negative)/total number of samples.

(2) Normalized training time = training time/maximum training time.

(3) Normalized compression time = compression time/maximum compression time.

Training time is usually positively correlated with other indicators of computational efficiency. Here, normalized training time is used as an easy-to-measure index of computational efficiency. He and Sun [31] analyzed the time complexity of CNN model in detail and mentioned that training time is an important index to test the real-time performance of the model.

4.3. Experimental Comparison. First of all, when the compression method is NC, the performance of the deep learning detection model is evaluated. Then, when the detection model is FCNN, the effects of the four compression methods involved in this paper on the detection accuracy and computational efficiency are compared. The Keras implementation algorithm of FCNN is shown in Appendix.

The hardware environment of the experiment has an Intel Xeon (Cascade Lake) Platinum 8269 GHz/3.2 GHz 4-core CPU and 8 GB memory. On the common network data set CSE-CIC-IDS2018 [25], the proposed FCNN is compared with the hybrid model CNN-LSTM [6] and IDS-DNN [13] with good performance in the current study. Through experiments, the detection accuracy and training time of the three models are compared and analyzed. In the CNN-LSTM model, CNN uses the same optimal network structure and parameters as the CNN model proposed in this paper. In addition, an LSTM layer with 70 units is added after the last convolution layer. IDS-DNN mainly consists of some dense layers; the number of dense layers in the fully connected layer and the number of units in each dense layer depend on the sample size (the sample size is related to the compression ratio). For different compression ratios, select the best number of dense layers and the number of each dense unit.

4.3.1. Performance Evaluation of Deep Learning Detection Model with Compression Method NC. In order to make a fair comparison, keeping the default parameters unchanged, the detection accuracy of FCNN, CNN-LSTM, and IDS-DNN algorithms are compared in CSE-CIC-IDS2018 data set. As you can see from Figure 4, the Acc of FCNN on Sub-DS1, Sub-DS2, and Sub-DS3 is always higher than that of CNN-LSTM and IDS-DNN. Compared with CNN-LSTM, FCNN increases Acc by 2.49%.

As can be seen from Figure 5, the detection accuracy of the method of not compressing the raw traffic and using the first 120 bytes of the packet is lower than that of compressing the raw traffic (compression parameter step = 35) and using the method of the first 1600 bytes of the packet. In the case of compressing the raw traffic data, the Acc increased by about 0.81%, while the training time was reduced by about 62.1%. The experimental results show that the data preprocessing method proposed in this paper greatly improves the computational efficiency on the premise of ensuring the accuracy.

Figures 6 and 7 show the changes in Acc and training time on Sub-DS3 as the compression parameter step changes from 5 to 50. As can be seen from these two figures, when the step varies between 5 and 35, the three algorithms mentioned have little impact on Acc. When the step is greater than 35, it will have a significant negative impact on the detection accuracy of the three algorithms. For FCNN, compared with step = 5, the accuracy of step = 35 is improved by about 0.20% and the training time is reduced by about 85.2%. For CNN-LSTM, compared with step = 5, the Acc of step = 35 is reduced by about 0.73%, and the training time is reduced by about 84.46%.

When step changes from 5 to 50, when Acc is almost unchanged, the training time of FCNN is always shorter than that of CNN-LSTM. Compared with CNN-LSTM, FCNN reduced training time by 23.82%. The reason is that FCNN uses GoF to initialize the weight values of the two convolution layers in the CNN model, thus saving some training time. When step is less than or equal to 15, the training time of IDS-DNN and FCNN is almost the same, but the accuracy of IDS-DNN model is lower than that of FCNN. When step is greater than 30, the training time of IDS-DNN is much longer than that of FCNN and CNN-LSTM, but the Acc of IDS-DNN model is not significantly improved.

The maximum number of packets in a single input sample is m.

Figure 8 shows the influence of the maximum number of packets m contained in each sample data on the detection accuracy. With the increase of m from 2 to 6, the accuracy of the three algorithms is gradually improved. The result is consistent with the assumption that the more the number of maximum packets is in each sample data, the easier it is to distinguish between positive and negative samples. However, the increase of the number of packets will lead to the increase of anomaly detection delay, so it is necessary to select the appropriate m according to different application scenarios or combining a small number of large packet samples with a large number of large packet samples. With the continuous change of the number m, the detection

Table 4: CSE-CIC-IDS2018 data subset.

Data subset	Collection time	Attack type	Total number of samples
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
		Penetration attack	92,403

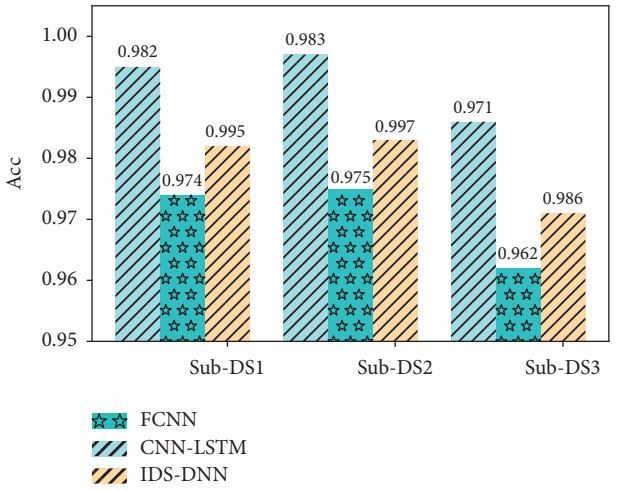

Figure 4: Comparison of Acc of different models on three data subsets of CSE-CIC-IDS2018.

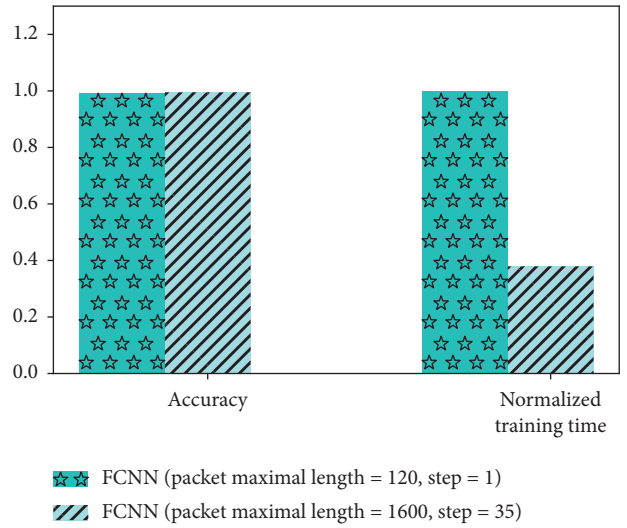

Figure 5: Comparison of accuracy and computational efficiency of raw traffic data before and after compression of the FCNN model.

accuracy of FCNN is always significantly higher than that of the other two models.

Figure 9 shows the Receiver Operating Characteristic (ROC) curves of the three methods (*step* = 35, unchanged). It can be seen that the TPR of FCNN with the change of FPR is always higher than the other two methods, and the AUC (Area under ROC curve) of FCNN is also higher than the other two methods.

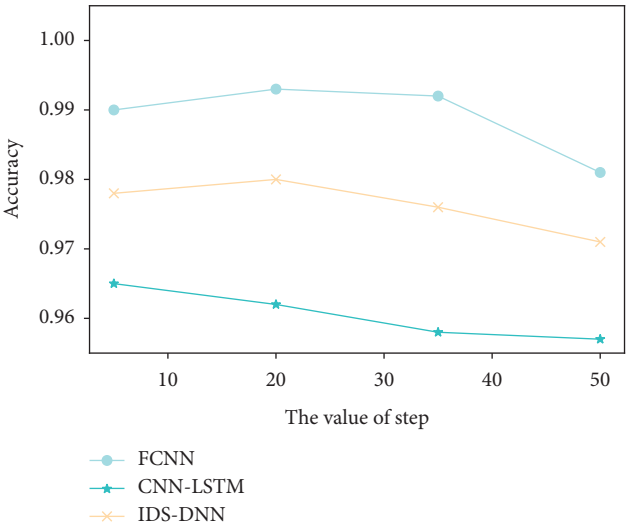

Figure 6: Influence of sample compression parameters on detection accuracy.

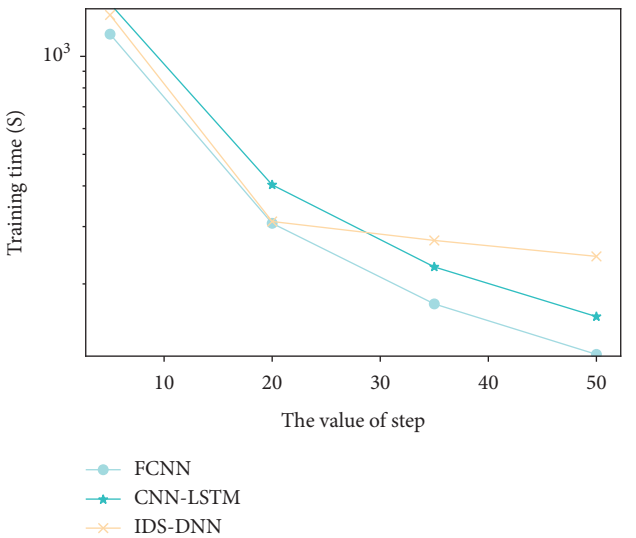

Figure 7: Influence of sample compression parameters on training time.

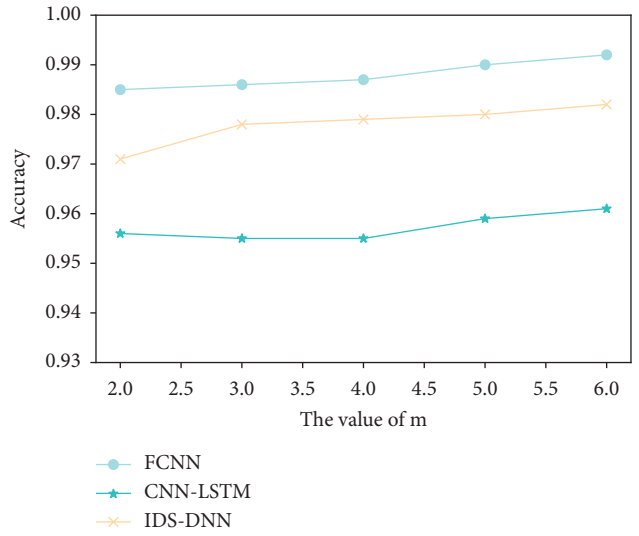

Figure 8: Influence of the maximum number of packets of each sample on the detection accuracy.

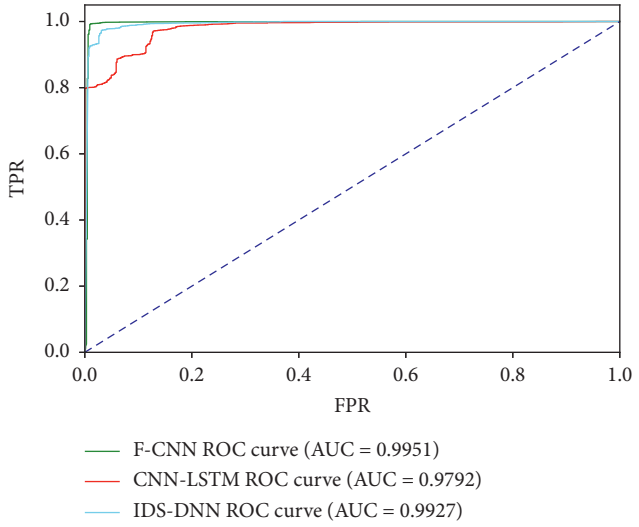

Figure 9: Comparison of ROC curves.

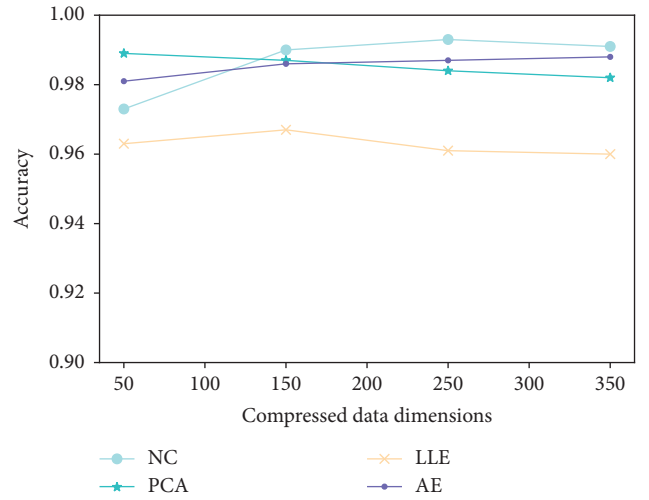

Figure 10: Comparison of detection accuracy of compression methods when the detection model is FCNN.

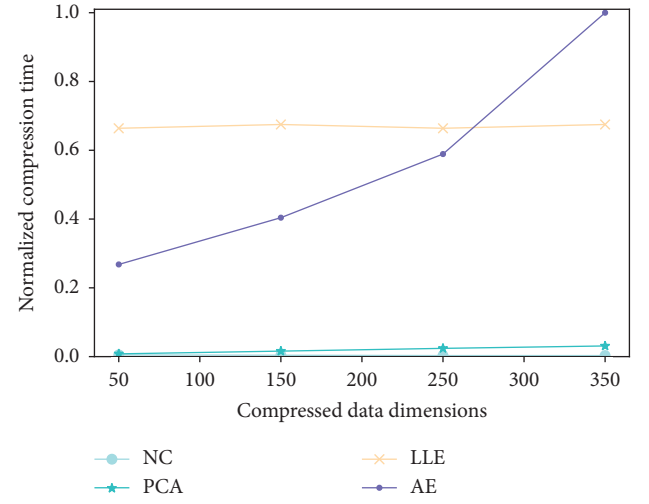

Figure 11: Comparison of compression time of the compression method.

4.3.2. Effect of Compression Methods on Performance When the Deep Learning Detection Model Is FCNN. This section compares the computational efficiency of the proposed compression method NC with the other three compression methods PCA, LLE, and AE and their impact on the detection accuracy of the detection model FCNN.

Figure 10 compares the detection accuracy of the four compression methods as the data dimensions change after compression. As can be seen from the figure, when the compressed data dimension is 50, the detection accuracy of PCA is the best. However, when the compressed data dimension is greater than or equal to 150, the detection accuracy of the NC method proposed in this paper is always the best. Compared with the PCA, the NC compression method can improve the detection accuracy up to 2%. In the process of increasing the dimension value of data after compression, the detection accuracy of AE method is increasing. This is because when the dimension value of the compressed data is very large, the compressed data of the AE method includes more information. However, in the process of increasing the dimension of the data after compression, the detection accuracy of the PCA method continues to decrease. This is because the useful information of PCA method is contained in the top principal components, while the components at the bottom contain more interference information.

Figure 11 compares the compression time of the four compression methods as the data dimensions change after compression. It can be seen that with the increase of data dimension after compression, the compression time of AE, PCA, and NC methods decreases gradually, while the compression time of LLE compression method remains almost unchanged. AE and LLE methods are time-consuming, and the compression time of NC method is always less than that of other methods.

```
Algorithm: FCNN training algorithm of raw traffic intrusion detection model based on Gabor network
Input : custom_gabor, custom_gabor2 # gabor filter
   X_train # training set
   Y_train # training label
   X_val # validation set
   Y_val # verify tag
   epochs = loop_num # iterations
   batch_size = 10 # batch size
   path = "model/fcnn" # training model save path
Output: model FCNN
(1) FCNN = Sequential ()
(2) FCNN.add (Convolution1D (48, 3, trainable = False,
      kernel_initializer = custom_gabor,
         border_mode = "same",
         activation = "relu",
         input_shape = (featureNum, 1)))
(3) FCNN.add (Convolution1D (48, 3, border_mode = "same", activation = "relu"))
(4) FCNN.add (MaxPooling1D (pool_length = (2)))
(5) FCNN.add (Convolution1D (128, 3, trainable = False,
      kernel_initializer = custom_gabor2,
         border_mode = "same",
         activation = "relu"))
(6) FCNN.add (Convolution1D (128, 3, border_mode = "same", activation = "relu"))
(7) FCNN.add (MaxPooling1D (pool_length = (2)))
(8) FCNN.add (Flatten ())
(9) FCNN.add (Dense (128, activation = "relu"))
(10) FCNN.add (Dropout (0.1))
(11) FCNN.add (Dense (1, activation = "sigmoid")) # the construction process of FCNN model
(12) FCNN.compile (loss = "binary_crossentropy", optimizer = "Adam", metrics = ["accuracy"])
(13) FCNN.fit (X_train, y_train,
   validation_data = (X_val, y_val),
   batch_size = 10,
   epochs = loop_num) # training FCNN
(14) FCNN.save (path + "FCNN_model.hdf5") # Save the model
```

Algorithm 1: FCNN training algorithm.

Comparing Figures 10 and 11, it is found that the detection accuracy of NC is always better than other compression methods when the compressed data dimension value is greater than or equal to 150; the detection accuracy of PCA is the best when the compressed data dimension value is less than 150. The compression time of NC is always the shortest. The experiments in this section show that the NC compression method proposed in this paper performs best in both detection accuracy and computational efficiency in most cases.

5. Conclusions

To address the limitation that the existing network intrusion detection methods rely on manual design features and the efficiency bottleneck of deep learning models when dealing with high-dimensional massive raw traffic data, this paper proposes a data preprocessing method based on multipacket input unit to compare the raw traffic and designs a deep network intrusion detection model with superior performance by using the powerful feature extraction ability of Gabor filter. Through comparative experiments, it is proved that this method can effectively take into account both computational efficiency and detection accuracy in the benchmark data set and the proposed FCNN model is better than the other two deep learning algorithms which perform well in the field of intrusion detection.

Our main research directions in the future include how to improve the proposed preprocessing methods and explore better data compression methods and improvement of the universality of F-CNN, such as make F-CNN be suitable for the Internet of things, industrial Internet, and wide area network with many protocols and various attack types or the backbone network of high-speed switches or routers and other communication equipment. Future research will continue to apply the cutting-edge achievements of deep learning to improve the generalization ability of detection methods, such as using transfer learning to accelerate training speed and improve prediction accuracy. In image processing and natural language processing, transfer learning can significantly improve the prediction accuracy and speed up the training speed. That is to say, when the model is trained on large data sets, it can improve the performance after moving to small data sets by fine-tuning. There are few researches on the application of transfer learning in network anomaly detection. In the future, we will

introduce transfer learning into the field of network anomaly detection and conduct in-depth research, train a network anomaly detection model based on large data sets, and publish it as a pretraining model.

Appendix

The Keras implementation algorithm of the proposed approach is presented in Algorithm 1.

Data Availability

The data used in this study are available at https://www.unb.ca/cic/datasets/ids-2018.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was sponsored by the National Key Research and Development Program of China (Grants nos. 2018YFB0804002 and 2017YFB0803204).

References

[1] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[2] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[3] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Toward effective mobile encrypted traffic classification through deep learning," *Neurocomputing*, vol. 409, pp. 306–315, 2020.

[4] K. Scarfone and P. Mell, *Guide to Intrusion Detection and Prevention Systems (IDPS)*, National Institute of Standards & Technology, Gaithersburg, MD, USA, 2007.

[5] N. Shone, T. N. Ngoc, V. D. 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.

[6] J. Kim, Y. Shin, and E. Choi, "An intrusion detection model based on a convolutional neural network," *Journal of Multimedia Information System*, vol. 6, no. 4, pp. 165–172, 2019.

[7] G. Marín, 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] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-Del-Rincon, and D. Siracusa, "LUCID: a practical, lightweight deep learning solution for DDoS attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[9] S. Khan: Lightweight Deep Learning Framework to Detect Botnets in IoT Sensor Networks by Using Hybrid Self-Organizing Map," 2020.

[10] Y. Dong, R. Wang, and J. He, "Real-time network intrusion detection system based on deep learning," in *Proceedings of the IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, September 2019.

[11] M. Tan, A. Iacovazzi, N. M. M. Cheung, and Y. Elovici, "A neural attention model for real-time network intrusion detection," in *Proceedings of the 2019 IEEE 44th Conference on Local Computer Networks (LCN)*, Osnabrück, Germany, October 2019.

[12] Y. Wang, Y. Jiang, and J. Lan, "A fast deep learning method for network intrusion detection without manual feature extraction," *Journal of Physics: Conference Series*, vol. 1738, no. 1, Article ID 12127, 2021.

[13] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proceedings of the 2016 International Conference on Platform Technology and Service (PlatCon)*, pp. 1–5, Jeju, South Korea, Feburay 2016.

[14] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, no. 99, pp. 21954–21961, 2017.

[15] R. Vinayakumar, M. Alazab, K. P. 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.

[16] Y. Mirsky, T. Doitshman, Y. Elovici et al., "Kitsune: an ensemble of autoencoders for online network intrusion detection," 2018, https://arxiv.org/abs/1802.09089.

[17] W. Wang, Y. Sheng, J. Wang et al., "HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2017.

[18] V. Pham, E. Seo, and T. M. Chung, "Lightweight convolutional neural network based intrusion detection system," *Journal of Communications*, vol. 15, no. 11, pp. 808–817, 2020.

[19] G. Bovenzi, G. Aceto, D. Ciuonzo et al., "A hierarchical hybrid intrusion detection approach in Iot scenarios," in *Proceedings of theGLOBECOM 2020-2020 IEEE Global Communications Conference*, Taipei, Taiwan, December 2020.

[20] S. Luan, C. Chen, B. Zhang, J. Han, and J. Liu, "Gabor convolutional networks," *IEEE Transactions on Image Processing*, vol. 26, no. 2, p. 1, 2017.

[21] 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.

[22] S. S. Sarwar, P. Panda, and K. Roy, "Gabor filter assisted energy efficient fast learning convolutional neural networks," in *Proceedings of the 2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Taipei, Taiwan, July 2017.

[23] A. Alekseev and A. Bobe, "GaborNet: gabor filters with learnable parameters in deep convolutional neural networks," in *Proceedings of the 2019 International Conference on Engineering and Telecommunication (EnT)*, Dolgoprudny, Russia, November 2019.

[24] G. He, Z. Lu, B. Xu, and H. Zhu, "Detecting repackaged android malware based on mobile edge computing," in *Proceedings of the 2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, Lanzhou, China, August 2018.

[25] S. Miller and C. Busby-Earle, "The impact of different botnet flow feature subsets on prediction accuracy using supervised and unsupervised learning methods," *International Journal of*

Internet Technology and Secured Transactions, vol. 5, no. 2, pp. 474–485, 2016.

[26] E. Arestrom and N. Carlsson, "Early online classification of encrypted traffic streams using multi-fractal features," in *Proceedings of the IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France, May 2019.

[27] C. C. J. Kuo, "Understanding convolutional neural networks with A mathematical model," *Journal of Visual Communication & Image Representation*, vol. 41, 2016.

[28] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, 2012.

[29] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

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

[31] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015.