







## Research Article

# Deep-Feature-Based Autoencoder Network for Few-Shot Malicious Traffic Detection

Mingshu He <sup>1</sup>, Xiaojuan Wang <sup>1</sup>, Junhua Zhou <sup>2</sup>, Yuanyuan Xi <sup>3</sup>, Lei Jin <sup>1</sup>  
and Xinlei Wang <sup>1</sup>

<sup>1</sup>School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

<sup>2</sup>State Key Laboratory of Intelligent Manufacturing System Technology, Beijing Institute of Electronic System Engineering, Beijing 100854, China

<sup>3</sup>School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm SE-10044, Sweden

Correspondence should be addressed to Xiaojuan Wang; [wj2718@163.com](mailto:wj2718@163.com)

Received 27 December 2020; Revised 1 February 2021; Accepted 15 March 2021; Published 27 March 2021

Academic Editor: Liguozhang

Copyright © 2021 Mingshu He et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the increase of Internet visits and connections, it is becoming essential and arduous to protect the networks and different devices of the Internet of Things (IoT) from malicious attacks. The intrusion detection systems (IDSs) based on supervised machine learning (ML) methods require a large number of labeled samples. However, the number of abnormal behaviors is far less than that of normal behaviors, let alone that the shots of malicious behavior samples which can be intercepted as training dataset are actually limited. Consequently, it is a key research topic to conduct the anomaly detection for the small number of abnormal behavior samples. This paper proposes an anomaly detection model with a few abnormal samples to solve the problem in few-shot detection based on convolutional neural networks (CNN) and autoencoder (AE). This model mainly consists of the CNN-based supervised pretraining module and the AE-based data reconstruction module. Only a few abnormal samples are utilized to the pretrain module to build the structure of extracting deep features. The data reconstruction module simply chooses the deep features of normal samples as training data. There also exist some effective attention mechanisms in the pretraining module. Through the pretraining of small samples, the accuracy of abnormal detection is improved compared with merely training normal samples with AE. The simulation results prove that this solution can solve the above problems occurring in network behavior anomaly detection. In comparison to the original AE model and other clustering methods, the proposed model advances the detection results in a visible way.

## 1. Introduction

Network application plays an indispensable role in people's lives with a large number of devices connected to the Internet; network security is attracting greater attention. With the continuous development of network technology, cyber threats are increasingly complex. According to the statistical prediction, the devices of the Internet of Things will increase from 27 billion to 75 billion by 2025, constituting a huge scale of network access devices [1]. Meanwhile, the number of network attacks will rise rapidly as well, which explains why network attack detection is widely concerned by researchers.

There are some intrusion detection systems especially for identifying malicious behaviors on the Internet and raising alarms [2]. However, the detection process tends to be complicated in the real production network environment. For the time being, most methods, such as ML-based models, always require a large number of labeled samples to accomplish the training process. Nevertheless, only a limited number of malicious behaviors can be intercepted to be the training data. Hence, few-shot detection is urgently needed. More seriously, a new attack pattern can hardly be predicted previously before it appears, which requires good sensitive detection ability of the model for unknown types of malicious

network traffic. Focusing on this problem, this paper also adopts a large number of samples of normal behavior and a few abnormal behavior samples in pretraining, which is applicable to the actual few-shot learning scenario.

There are two major methods for malicious network behavior detection, traditional business approaches and ML-based ones [3]. Generally speaking, both of them need to extract the necessary information or features based on professional awareness. Deep Packet Inspection- (DPI-) based, port-based, behavior-based, and statistic-based methods are several frequently used traditional approaches [4, 5]. These methods rely on strong single service characteristics, being excellent at identifying specific threats accurately. However, if some new threats occurred, the new business features and rules need to be extracted as well, making it difficult for traditional methods to identify malicious behaviors with unstable rules and features. By contrast, ML methods usually take out business features from more various dimensions, which therefore provides better generalization.

However, from a business perspective, there is a limitation that the performance of an ML model always depends on the feature engineering results based on manual analysis, which makes our feature mining task heavy. At the same time, the training data also rely on specialized knowledge concerning cyber security and attacks. As a result, researchers need to be more concentrated on feature engineering work, which means a great challenge to the efficiency of anomaly detection. Besides, some methods try to encrypt in the process of information transmission [6, 7], which makes the feature extraction of encrypted traffic more difficult. Compared with the traditional machine learning algorithm, the deep learning (DL) model has its own advantages in extracting features. For example, the CNN-based model can directly obtain the deep features without manual feature engineering, the data processed by which displays a better spatial distribution. This is one of the reasons why CNN is selected for the pretraining process. In common with all deep learning algorithms, the model proposed in this paper also eliminates the manual process of feature extraction. Except for this, the model will no longer require a mass of abnormal samples to carry out supervised learning. During the detection process, the pretraining model only needs a small number of abnormal samples to achieve a good performance of the subsequent unsupervised anomaly detection. By doing so, the problems of few-shot detection can be worked out as well.

The contributions of this paper can be summarized as follows:

- (1) Proposing a network malicious behavior detection model based on CNN and AE (DFAE), which can not only achieve a good detection result in the case of a small number of abnormal training samples but also provide a certain guiding significance for the real abnormal detection scenario in which the number of normal samples is much larger than that of abnormal samples.
- (2) Improving the AE detection results by a large margin by using the output of deep features from the CNN-based pretraining model as the training data.

- (3) Enhancing the anomaly detection results on precision, recall, and F1-score in three available datasets by adding an effective attention mechanism into each block of CNN.

The remainder of this paper is arranged as follows. Some related works are arranged in Section 2. In Section 3, we introduce the detailed structure of the models and the specific process of the method. There is also some basic knowledge regarding the proposed model. Section 4 analyzes the results produced by the proposed model and some comparative experiments. Section 5 summarizes the paper on the whole.

## 2. Related Work

Network traffic as an essential means to record the network behavior process contains almost all the information of a complete access from the source host to a destination host [8] and can be captured and collected in a packet capture (PCAP) file [9] in most implementations. This is because of its data retention and general integrity [10, 11]. In general, extracting efficacious information and valid business features from source data is the first step in network abnormal behavior detection [12]. Similarly, this paper picks out PCAP files of three different datasets and extracts effective information from files for the following anomaly detection.

At present, there are a number of network traffic anomaly detection methods based on machine learning. Many of them pay attention to business features and adopt the approach of traditional machine learning. Meanwhile, some researchers are inclined to use neural networks to extract deep features for detection without any artificial process [5]. Also, there are studies focusing on distinct anomaly detection results output from different traditional algorithms on dataset NSL-KDD [13, 14]. Negandhi et al. employed the supervised learning algorithm to train a network attacks detection model based on random forests. The model reduces the number of input features under an effective feature selection mechanism, not only improving the running speed but also realizing a high accuracy [15]. Sarker et al. proposed an ML-based multilayered framework for the purpose of promoting the security of network system, which aims at the applicability towards data-driven intelligent decisions and protects network systems and devices from network attacks [16]. Atli proposed a traffic classification method that identifies the normal traffic and encrypted traffic by analyzing network flow based on decision tree (DT) and *K*-Nearest Neighbor (KNN) algorithms [17]. The work by D'hooge and Kayes concluded that the results of anomaly detection with machine learning algorithms as a basis are not ideal among different datasets [18], which provides a research impetus to increase the generalization of the model with limited data. Without exception, these methods cost a lot of time in feature extraction. That is why the methods rooted in the neural networks are extensively utilized in anomaly detection tasks.

The research by Zeng et al. proposed a light-weight framework without manual intervention and private

information but with the aid of deep learning for encrypted traffic classification and intrusion detection [19]. The model in [20] studied by Thapa et al. was based on classification and regression tree (CART) and CNN and performed well in 10-fold cross-validation and independent testing on dataset CIC-IDS2017 [21]. Compared with some other methods, the model brought forward by Javaid employs a self-taught learning technique on NSL-KDD and, as a result, improves the precision and recall rates [22]. Chen et al. proposed a network traffic classification model on the basis of metric learning, which gained a better performance on some available datasets by adding additive angular margin loss embedded on CNN [23]. The traffic classification method based on text convolution neural network was projected by Song et al., in which the model performs better because of adding a new loss function [24]. Zhu et al. proposed a long short-term memory- (LSTM-) based anomaly detection method [25]. Du et al. put forward a packet-based malicious payload detection and identification algorithm based on object detection deep learning network [26]. At the same time, reinforcement learning (RL) [27, 28] which is widely used in resource scheduling is applied to anomaly detection [29]. These methods based on deep learning make a lot of contributions to the network security anomaly detection tasks. However, none of them raised a good solution to the problems mentioned in Section 1, including the lack of abnormal sample markers as well as the recognition of new abnormal behaviors.

In the fields of network anomaly detection and other similar ones, some methods try to solve the class imbalance by modifying the model structure formed by ML algorithms. The problem of too few labeled samples of abnormal behaviors reveals the imbalance between negative samples and positive ones. Buda et al. analyzed the imbalance problem in image classification tasks and investigated some solutions [30]. As for Rodda and Erothi, they exerted the traditional ML methods to evaluate the effect of class imbalance on the benchmark NSL\_KDD dataset [31]. Gu et al. presented a semisupervised weighted  $K$ -means detection method to tackle the problem that supervised learning methods need abundant labeled data [32]. On the foundation of variational autoencoder (VAE), Xu et al. concluded an unsupervised anomaly detection algorithm named Donut, which remarkably advanced the results of anomaly detection [33]. Also, some data augmentation methods are put to use to solve this problem in detection applications [34, 35]. In this paper, under the theory of data augmentation and the idea of unsupervised learning, we propose a network traffic anomaly detection model with supervised pretraining, which uses a few abnormal samples.

### 3. Model and Methods

**3.1. Anomaly Detection Framework.** This model targets attaching a superior anomaly detection result on the network traffic data in the circumstance of a few malicious behavior negative samples. In order to complete the experimental demonstration and explanation, we divide the whole procedure into three parts. As shown in Figure 1,

there are data processing, supervised pretraining, and autoencoder training process. Besides, there are some comparative experiments as well. The similarity-based and  $K$ -means-based methods are explained later. In the figure here, one of the comparative methods, AE method mentioned in the experiment, is the third part training process whose input is the raw data without pretraining. Another one is the whole deep-feature-based autoencoder without any attention mechanisms. We also take advantage of the similarity of raw data to measure the distance between the test sample and benign sample center to identify whether the tested one is abnormal. What is more,  $K$ -means clustering model is used as the comparative method in the same way. The effectiveness of the proposed model is proved by the comparison of its results with the results of the four methods.

The network behavior is usually recorded as PCAP files by the means of network traffic. PCAP files consist of a series of hexadecimal numbers. Generally, it does not come easy to understand original numbers in a short time. The numbers in rigid positioning always represent particular meanings, the business significance of which thus needs to be perceived by some analysis software. Data processing is designed to convert raw data from PCAP into matrixes. The numbers with a certain length will be taken out from PCAPs to shape the data matrixes. After this step, the supervised pretraining process will train a classifier to seek for the feature space with a larger distance among samples from different categories. Then, deep features output from the pretraining module will be fed into autoencoder for data reconstruction with benign data only. When the data reconstruction model converged, samples could be imported into the model to evaluate the reconstruction effect and detect the abnormal samples. As can be seen from Figure 1, the outputs of abnormal samples (the red sample outputs from AE in the figure) are usually more different from inputs than those normal ones (the blue sample outputs from AE in the figure). Because only normal traffic data is used for training in this process, the model learns nothing but the reconstruction ability of normal traffic, which means the reconstruction process for abnormal data cannot be completed. In the autoencoder training process, the reconstruction rate threshold will be set, such as 90%, which is from 0 to 100%. If the sample reconstruction rate is not smaller than the threshold value, the sample will be regarded as a normal one; otherwise, it will be regarded as an abnormal one. These three processes mentioned above are followed by the realization of detection results. Also worth noting is the fact that just a few abnormal samples in the second pretraining process actualize the goal of achieving the anomaly detection with a small number of negative samples.

#### 3.2. Model Structure of Deep-Feature-Based Autoencoder

**3.2.1. Introduction of CNN and AE.** This section will introduce the basic structure of CNN, AE, and the basic net of the proposed model. CNN has outstanding performance in many applications, especially in the image area [36, 37].

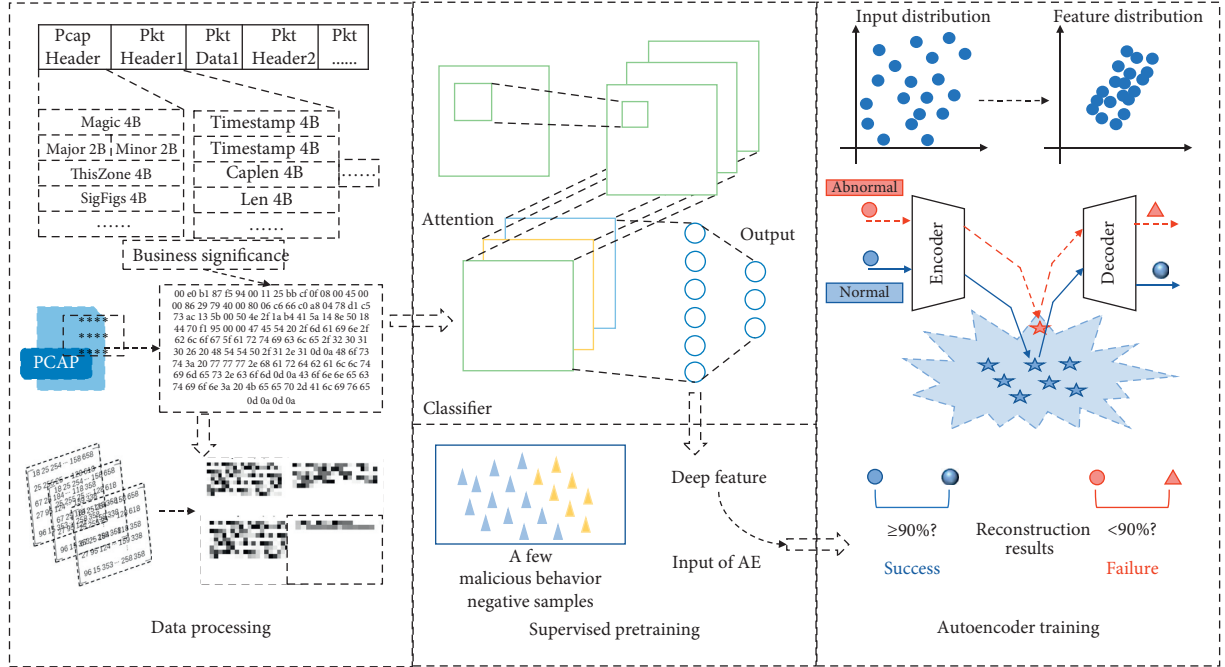


FIGURE 1: Framework of anomaly detection process. Data processing converts raw data with hexadecimal numbers to matrixes. Supervised pretraining accomplishes few-shot training and provides deep features that can be used in autoencoder training process and the data reconstruction process can be completed.

Autoencoder is an unsupervised model and is widely applied in data dimensionality reduction and feature extraction as well as data augmentation [38]. Figures 2 and 3 illustrate the simple structures of CNN and AE. A typical CNN network always contains several convolutional layers, pooling layers, and fully connected layers. Roughly speaking, the main function of a convolutional layer is to extract features, and it can obtain many effective ones without any manual intervention. Convolutional layers always serve as the beginning structure of the whole network. The convolution kernel size can be defined to extract local features with different sizes and different convolution kernels can represent different features. Then the results output from the convolutional layer are mapped by a nonlinear operation, which is usually achieved through the activation function such as rectified linear unit (ReLU), Tanh, and Sigmoid function [39, 40]. The main purpose of the pooling layer is to reduce the dimension of current data, which is actually the process of sampling. Pooling layer not only retains the main features but also greatly reduces the computation of the model. Max pooling and average pooling are two major operations in this layer. One calculates the maximum of local units and the other figures out the average in the feature map. FC layer decides which category is the true result.

An AE is made up of two layers, namely, an encoder and a decoder. AE is a genre of unsupervised learning method for dimension reduction and feature extraction. The encoder mainly encodes original data and outputs samples of dimension reduction, while the decoder mainly decodes the encoded vector to restore the original sample. Such two

processes can be regarded as a data reconstruction process. CNN, FC, LSTM, and so forth are feasible to be chosen as the basic network of AE.

**3.2.2. Attention Mechanism.** In this paper, the model that combines CNN with AE has already reached a good result in most testings. However, it does not show outstanding performance in a tiny fraction of datasets, and there is still room for improvement. Therefore, effective attention mechanisms are added to the model. When the attention mechanism is proposed in the DL model, it draws lessons from the human attention mechanism. After observing the whole situation, people tend to pay attention to some more important areas, which is just the skill the model needs to master and improve in a further step under the attention mechanism. In this paper, channel attention is attached to the end of the convolutional block [41]. We try to use three global pooling methods at the beginning of the net, namely, max pooling, average pooling, and max pooling with average pooling. Figure 4 unfolds the operation process of the attention mechanism.

According to the details, the input of a convolutional block can be defined as  $\mathbf{X} \in \mathbf{R}^{H' \times W' \times C'}$  and the output is  $\mathbf{M} \in \mathbf{R}^{H \times W \times C}$ , where  $W'$ ,  $W$ ,  $H'$ , and  $H$  are the widths and heights of the input and output results of the convolutional blocks.  $C'$  and  $C$  denote the input and output channel sets of a block. We use  $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots, \mathbf{v}_c]$  to refer to the learned set of filter kernels, where  $\mathbf{v}_c$  is the parameter of the  $c$ -th filter. Hence, the output can be written as  $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \dots, \mathbf{m}_c]$ , where

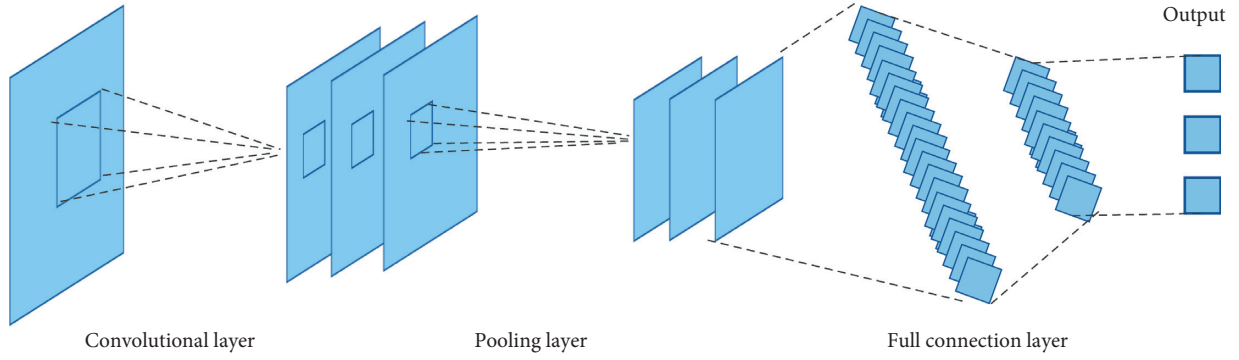


FIGURE 2: CNN basic model.

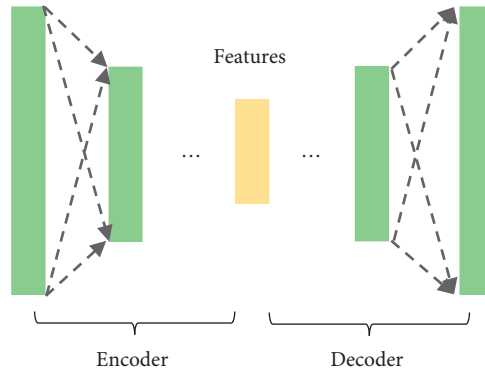


FIGURE 3: AE basic model.

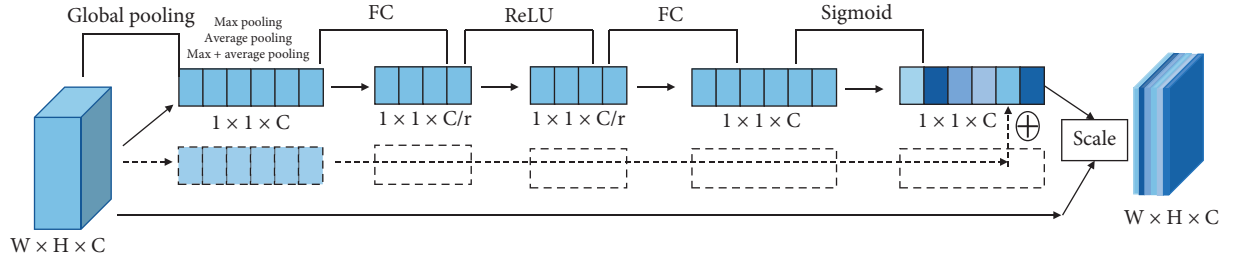


FIGURE 4: Structure of channel attention in the proposed model. Attention mechanism with average pooling adopted in the paper is indicated with solid rectangles and other attention mechanisms with max pooling or max and average pooling not used are indicated with dotted rectangles.

$$\mathbf{m}_c = \mathbf{v}_c * \mathbf{X} = \sum_{s=1}^{C'} \mathbf{v}_c^s * \mathbf{x}^s. \quad (1)$$

The symbol  $*$  represents the convolution;  $\mathbf{v}_c = [v_c^1, v_c^2, v_c^3, \dots, v_c^{C'}]$ . Next, the output of the convolutional block  $\mathbf{M}$  is fed into the attention module. Firstly, the pooling operation converts  $\mathbf{M}$  to size  $1 \times 1 \times C$  as  $\mathbf{Z}_1 = [z_1^1, z_1^2, z_1^3, \dots, z_1^C]$ , where  $z_1^C$  is calculated by

$$z_1^C = \frac{1}{H * W} \sum_{i=1}^H \sum_{j=1}^W \mathbf{m}_c(i, j), \quad (2)$$

$$z_1^{C'} = \text{Max}(\mathbf{m}_{i,j}). \quad (3)$$

Equations (2) and (3) denote two different global pooling methods: average pooling and max pooling. Figure 4 shows the evaluation of the effects under the three different situations of using merely max pooling, using merely average pooling, and using average pooling with max pooling together in experiments. The average pooling is turned out to be the best one. Therefore, the other two methods will not be discussed in detail here. After that, the data will cross two fully connected layers with ReLU and Sigmoid. We define them as  $\mathbf{Z}_2 = [z_2^1, z_2^2, z_2^3, \dots, z_2^{C'/r}]$  and  $\mathbf{Z}_3 = [z_3^1, z_3^2, z_3^3, \dots, z_3^C]$ , where  $r$  is the parameter and it is defined as 16 here. If  $\mathbf{F}_{fc}$  is the fully connected operation,  $\mathbf{F}_{relu}$  denotes ReLU activation process and  $\mathbf{F}_{sigmoid}$  denotes Sigmoid activation process;  $\mathbf{Z}_2$  and  $\mathbf{Z}_3$  can be calculated as

$$\mathbf{Z}_2 = \mathbf{F}_{\text{relu}}(\mathbf{F}_{\text{fc}}(\mathbf{Z}_1)), \quad (4)$$

$$\mathbf{Z}_3 = \mathbf{F}_{\text{sigmoid}}(\mathbf{F}_{\text{fc}}(\mathbf{Z}_2)). \quad (5)$$

Finally, we define  $\mathbf{F}_{\text{scale}}$  as the channel-wise multiplication between the scalar  $z_3^C$  and the input feature map  $\mathbf{m}_c$ . Moreover, the output of the attention module is  $\tilde{\mathbf{M}} \in \mathbf{R}^{H \times W \times C}$ , which can be calculated as

$$\tilde{\mathbf{M}} = \mathbf{F}_{\text{scale}}(\mathbf{Z}_3). \quad (6)$$

Here, we employ the mean squared error (MSE)  $L_{\text{mse}}$  as the loss function, which can be calculated as

$$L_{\text{mse}}(y) = \sum_i (y'_i - y_i)^2, \quad (7)$$

where  $i$  is the prediction probability of  $i$ -th category,  $y_i$  signifies prediction result, and  $y'_i$  indicates the true result. Through attention mechanism, a feature map with channel weight  $\mathbf{Y}$  can be acquired, which can be regarded as a self-attention function on channels whose relationships are not confined to the local receptive field to which the convolutional filters are responsive.

**3.2.3. Deep-Feature-Based Autoencoder.** Sections 3.2.1 and 3.2.2 depict the basic structure of the proposed model in detail. We can divide the model into four processing modules. They are supervised deep feature extraction module (SDFE), unsupervised data reconstruction module (USDR), channel attention module (CA), and detection results output module (DROut). Actually, CA is a part of SDFE and DROut is the decision part of USDR. This detailed process is described at length in Figure 5.

As Section 4.2 described, raw data will be converted into a matrix with the data ranging from 0 to 255 whose width is 28 and height is 28. Thus, the input can be written as  $\mathbf{X} \in \mathbf{R}^{28 \times 28}$ . Next, the data will cross the first convolution layer. The output channel is set to 32 and the convolution kernel size is 9. ReLU is the activation function here. Besides, the first pooling kernel size is 2. After all these processes mentioned above, we have  $\mathbf{M}_1 \in \mathbf{R}^{10 \times 10 \times 32}$ . The calculation process of each layer can be generalized as follows:

$$\mathbf{M}_1 = \mathbf{F}_{\text{pooling} \times 2}(\mathbf{F}_{\text{relu}}(\mathbf{F}_{\text{conv} \times 32 \times 9}(\mathbf{X}))), \quad (8)$$

where  $\mathbf{M}_1 \in \mathbf{R}^{10 \times 10 \times 32}$  denotes the results,  $\mathbf{F}_{\text{conv}}$  means a convolution operation, and  $\mathbf{F}_{\text{pooling} \times 2}$  refers to a pooling operation with 2-size pooling kernel, the result of which will be fed into the channel attention, and the corresponding output result is  $\tilde{\mathbf{M}}_1 \in \mathbf{R}^{10 \times 10 \times 32}$  with the channel attention weight depicted in Section 3.2.1.

After that, there is the second convolutional layer with ReLU and the second pooling layer, where the convolution kernel size is 9, the channel size is 64, and pooling kernel size is 2. What is mentioned above can be described as follows:

$$\mathbf{M}_2 = \mathbf{F}_{\text{pooling} \times 2}(\mathbf{F}_{\text{relu}}(\mathbf{F}_{\text{conv} \times 64 \times 3}(\tilde{\mathbf{M}}_1))), \quad (9)$$

where  $\mathbf{M}_2 \in \mathbf{R}^{4 \times 4 \times 32}$  is the output from the second convolution block. The samples then cross the attention layer and

obtain the result  $\tilde{\mathbf{M}}_2 \in \mathbf{R}^{4 \times 4 \times 64}$  with channel attention weight. Finally,  $\tilde{\mathbf{M}}_2$  is entered into a fully connected layer to acquire the supervised classification results. When the supervised classifier training converges, a large amount of benign data will be sent to the supervised model and receive the deep features  $\mathbf{Y} \in \mathbf{R}^{1 \times 1024}$  reshaped from  $\mathbf{M}_2$  for each sample. In SDFE, average cross entropy error (ACE) is used to be the model loss  $L_{\text{ace}}$ , which can be calculated as

$$L_{\text{ace}}(y) = \sum_i y'_i \log(y_i), \quad (10)$$

where  $i$  is the prediction probability of  $i$ -th category,  $y_i$  indicates the prediction result, and  $y'_i$  conveys the true result.

By way of this treatment,  $\mathbf{Y}$  will be imported into USDR. There is an encoder and a decoder in it, as stated in Section 3.2.1. Moreover, three fully connected layers are designed in both encoder and decoder. The sizes of each output feature are 256, 64, and 20 in the encoder and 64, 256, and 1024 in the decoder where the activation functions are ReLU after the first five FC and Sigmoid after the last FC. Then the final reconstructed feature vector can be represented as  $\tilde{\mathbf{Y}} \in \mathbf{R}^{1 \times 1024}$ .

In the output module, the model will evaluate the effect of the reconstruction process. During the training process, we only use benign data as the training data. When the model converges, the model will have a good reconstruction ability for benign samples, which is followed by adding test data including normal and abnormal ones into the trained model. At this time, there will be a significant difference between normal samples and abnormal ones. This paper uses Spearman's rank correlation coefficient and Pearson correlation coefficient to portray the reconstruction probability between the input  $\mathbf{Y}$  and  $\tilde{\mathbf{Y}}$  of USDR. A reasonable threshold will also be set to identify whether the sample is normal or abnormal.

## 4. Experiments Process and Results

**4.1. Data Description.** In this paper, PCAP files are the raw data format from which the model gets the input value. PCAP files can be translated into a group of hexadecimal numbers. Specific hex numbers or their combination at a specific location represents a specific business significance. Generally speaking, a PCAP file consists of a certain number of traffic packets. The structure of a PCAP file and the meaning of its various positions are illustrated in Figure 6.

There is a global header in a PCAP file. As Figure 6 shows, the 4-Byte Magic content represents the beginning and tells the recognition sequence of Byte in this file. The 2-Byte Major content means the major file version number. The 2-Byte Minor content is the minor file version number. The 4-Byte ThisZone refers to the local standard time. The 4-Byte SigFigs is the accuracy of the timestamp. The 4-Byte SnapLen represents the maximum storage length. The 4-Byte LinkType content is the link type. The length of the packet header is 16 Bytes, defining the 4-Byte high-order position of capture timestamp, the 4-Byte low-order position of capture timestamp, 4-Byte data length of currently captured, and 4-Byte actual data length. Accordingly, there is a packet data

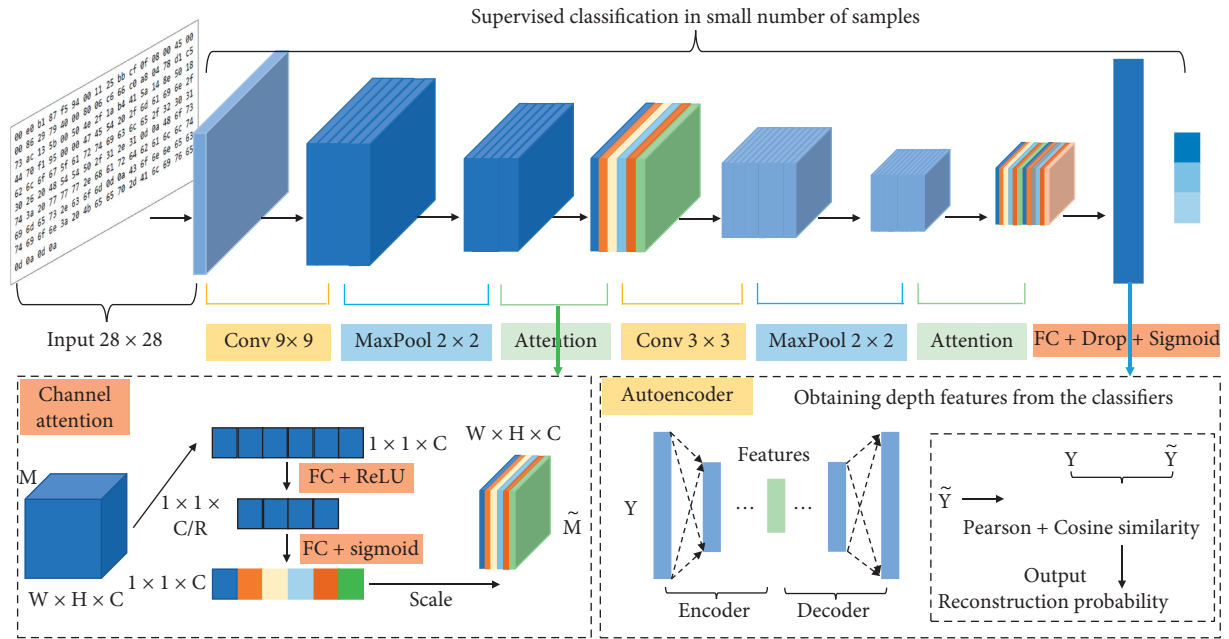


FIGURE 5: Model structure details of deep-feature-based autoencoder.

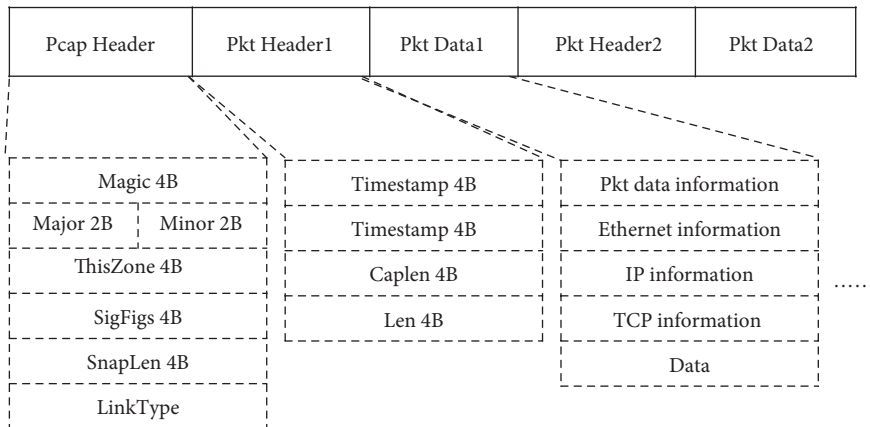


FIGURE 6: Structure of a PCAP file. It always consists of a file header, some packet headers, and some packet data.

area that always includes Internet Protocol (IP) information and Transmission Control Protocol (TCP) information and will be splicing of a certain number of traffic packets.

PCAP files with many traffic packets are selected as the original data source in this paper. There are three different open-source network attack datasets throughout the whole experiment, namely, CIC-IDS2017, CIC-IDS2012, and USTC-TFC2016, and they are all available through open-source addresses. In detail, CIC-IDS2017 contains over 2.2 million pieces of benign flow data and about 550 thousand pieces of malicious behavior flow data containing five major attacks: Denial of Service (DoS) attack, web attack, infiltration attack, port scan attack, and brute-force (BF). CIC-IDS2012 contains about 1.5 million pieces of benign flow data and no more than 50,000 pieces of abnormal flow data with 4 types: brute-force, infiltration, Hyper Text Transfer Protocol (HTTP) DoS, and Distributed Denial of

Service (DDoS). At the same time, USTC-TFC2016 merges about 300,000 pieces of benign and 600,000 pieces of 10 types of Trojan horse attack abnormal flow data. As for these abnormal types, DoS attack is an attack that causes denial of service whose purpose is to make the computer or network unable to provide normal services. Web attack often attacks websites or web applications and causes the application not to work normally. Infiltration attack is a systematic and progressive comprehensive attack mode whose attack target is often clear, but the purpose is not so single. Port scan is the common way to search target computer by hackers that can find out some potential intrusion ports. Brute-force attack means that hackers use the password dictionary to guess the user's password by exhaustive method, which is one of the most widely used attack methods. HTTP DoS and DDoS are two special DoS attacks.



**4.2. Data Processing.** It is described in Section 4.1 that PCAP files are original data of experiments. But the neural network model cannot identify these data types. Traffic packets need to be converted into vectors with the same length. It can be seen from Figure 7 that the data processing covers four steps of data split, data clean, data transfer, and data trimming. Once this process is finished, the data can be sent to the model as an input value.

**Data Split:** A PCAP file records the network behavior information spanning for a period of time, which means that a file contains a large number of different access connections. In general, people take a combination of source IP address, source port, destination IP address, destination port, and transport protocol as a single access behavior, which is also called 5-tuple. Therefore, it is necessary to divide a large PCAP file into a number of small PCAP files that only contain one 5-tuple data. This small PCAP file is named a traffic flow as well.

**Data Clean:** Almost all the data generated by practical applications will inevitably have redundancy, and so does network traffic. In this step, empty flows, duplicate flows, and interferential flows are picked out and deleted to produce the available data.

**Data Transfer:** DL model cannot analyze PCAP files, so they need to be transferred into text files with hexadecimal number as content. This step will not lose any information, just the conversion of the storage form.

**Data Trimming:** This step will determine the length of hex data. Normally, the length depends on business experience, packets average length, flows average length, and suchlike factors. In this paper, we select 784 bytes of data and reshape them to a  $28 \times 28$  matrix. If a PCAP file is larger than 784 bytes, it will be trimmed to 784 bytes. Or if it is smaller than 784,  $0 \times 00$  will be increased into the end of the flow.

### 4.3. Results and Analysis

**4.3.1. Detection Results Output from the Model.** As same as in other classification tasks, we collect four evaluation indexes to evaluate the effect of the model, macro-f1, weighted-f1, recall, and precision, which are written as  $I_{mf1}$ ,  $I_{wf1}$ ,  $I_{recall}$ , and  $I_{pre}$  and are explained as follows:

$$I_{recall} = \frac{1}{N} \sum_{i=1}^N \frac{TP}{TP + FN}, \quad (11)$$

$$I_{pre} = \frac{1}{N} \sum_{i=1}^N \frac{TP}{TP + FP}, \quad (12)$$

$$I_{mf1} = \frac{1}{N} \sum_{i=1}^N \frac{2 * I_{pre} * I_{recall}}{I_{pre} + I_{recall}}, \quad (13)$$

$$I_{wf1} = \frac{1}{N} \sum_{i=1}^N w \frac{2 * I_{pre} * I_{recall}}{I_{pre} + I_{recall}}, \quad (14)$$

where TP represents the number of correctly identified positive samples, TF represents the number of correctly identified negative samples, FP denotes the number of wrongly identified positive samples, and FN denotes the number of wrongly identified negative samples.  $N$  refers to the category number in data and  $w$  is the weight of this category to the total data quantity.

Under the experiment, the model results are verified on three datasets described in Section 4.1. At the same time, we set up four groups of comparative experiments on each dataset. In a further step, the training set and test set use the same data to ensure the effectiveness and the reference value of the four comparison experiments shown in Tables 1–3.

**(1) Similarity Measure Method.** This method exerts the Pearson correlation coefficient to calculate the distance between the test sample and the center samples of benign data in the training set. Only benign data are employed to find the center of the training samples. When calculating, we choose  $n$  samples randomly and calculate the nearest center vector to them as the center sample. The abnormal behavior is detected by the similarity distance between the test sample and the normal data center obtained via the training process. This method attempts to use the most primitive data distribution to distinguish outliers. Due to the complex distribution of the original data, it can be predicted that its accuracy will not be ideal.

**(2) K-Means Method.**  $K$ -means method applies the unsupervised clustering to the detection of anomalies directly. The unsupervised process means that a lot of labeled data is not required in this process.  $K$ -means is a distance-based algorithm selecting the centroids of each category and comparing the distance between the test samples to each centroid. The sample will eventually be classified into the category with the nearest centroid. Assuming that we have 2 categories, normal and abnormal, there are 2 cluster centers. According to the principle of minimum distance from the initial center point, all observations are divided into the categories, where each center point is located. Then the mean value of the observation points in each category is calculated as the center of the next iteration until convergence. The unsupervised method is always sensitive to data. Because there is a lot of information redundancy in traffic data, it will have a greater impact on the clustering process.

**(3) AE Method.** As part of the proposed model, AE reconstructs the data directly with the original data resource, the advantage of which is that it needs no abnormal behavior data and simply makes use of the normal traffic data. But the effects are often decided by the original distribution of data.

**(4) CNN with AE.** CNN with AE also belongs to the proposed model without adding the attention mechanism. Section 3.2 and Figure 5 concretely reveal these treatment processes. Only 2,000 malicious samples produce a marked effect in the supervised classifier of each dataset, by virtue of which the unsupervised autoencoder can reach a state of



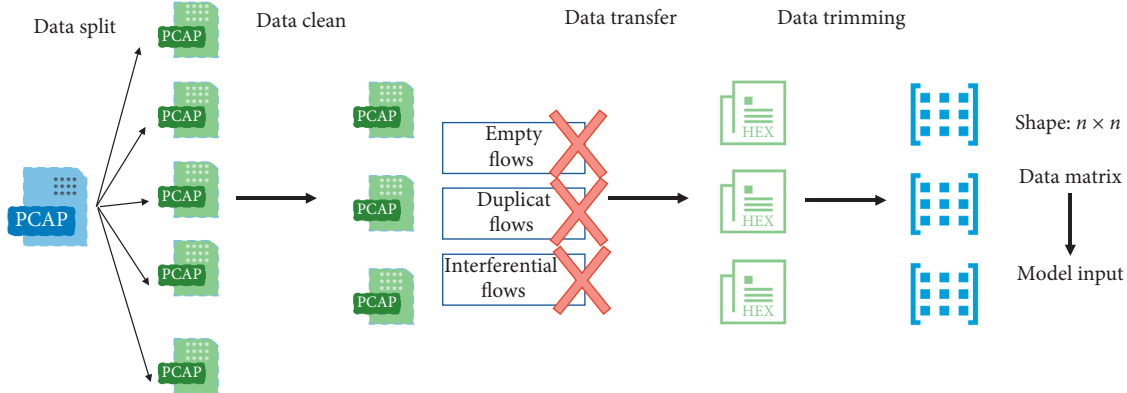
FIGURE 7: Data processing outputs data matrixes with shape  $n$ .

TABLE 1: Experiments results of dataset CIC-IDS2017.

Methods	With unknown categories			Known categories		
	Recall	Precision	Macro-f1	Recall	Precision	Macro-f1
Similarity measure	0.5233	0.5134	0.5103	0.4134	0.4230	0.4071
K-means cluster	0.6428	0.5554	0.3228	0.6163	0.6962	0.4721
AE	0.6226	0.5929	0.6042	0.8510	0.8148	0.8196
CNN with AE	0.6718	0.6399	0.6539	0.8626	0.8262	0.8323
<b>DFAE</b>	<b>0.8593</b>	<b>0.8738</b>	<b>0.8579</b>	<b>0.9519</b>	<b>0.9508</b>	<b>0.9513</b>

perfection. More than 80,000 test samples are put into use, including 30,000 benign samples and other abnormal ones.

Tables 1–3 present the experiment results among three datasets. Each method adopts two types of test data. One only contains samples of known categories trained in the surprised classifier and the other also embodies quite a lot of unknown categories. It can be concluded that there is an obvious improvement in every dataset with the proposed model. It is proved that the pretraining with a small number of malicious samples can significantly improve the subsequent data reconstruction process. As for USTC-TFC2016 and CIC-IDS2012, the supervised training process enhances the recall, precision, and f1-score over 10%. The improvement of CIC-IDS2017 is lightly inferior to others. Meanwhile, the effect of the attention mechanism on CIC-IDS2017 is more obvious by reforming the results over 10%, especially on the test samples with unknown categories. From the results of three datasets, we can see that the proposed method is very effective. As for better distributed dataset such as USTC-TFC2016, the detection accuracy is more than 99%. As for the dataset with general experimental results such as CIC-IDS2017, its accuracy also reached 95%. This also shows that the model has made a significant improvement on different distribution of data. The proposed method has strong generalization ability and provides the possibility for the application in the actual data.

*4.3.2. Some Other Discussion.* Apart from the enhancement of the overall effect, we also observed the classification of normal and abnormal samples. Figure 8 manifests the confusion matrix of the results output from DFAE on each dataset.

In view of the effect of each category, the detection accuracy of the normal flow is better than that of the abnormal ones. From the perspective of confusion matrix results, the accuracy is relatively balanced between positive and negative samples. It is particularly valuable to achieve this result in few-shot learning tasks. Our experiment emerged the imbalance that the number of normal samples is much larger than abnormal ones. However, the proposed model can well solve this problem. The findings suggest that the detection rate performs efficaciously in both categories.

In order to investigate the effect of the model more deeply, we analyze the reconstruction results of the original data with and without the deep features. From Figure 9, we can see the reconstruction rate of different datasets. On the red arrow's left side is the data reconstruction of original data based on AE. On the right side of the red arrow is the data reconstruction process using the deep features output from a supervised classifier based on CNN. It can be easily discovered that the reconstruction rate of the right figure is obviously better than that of the left one. There is a clear reconstruction rate between normal samples and abnormal samples in terms of the right figure. Every row in the graph is the result of a dataset. In each dataset, 1,000 samples are randomly picked up to compare the reconstruction result in the figure. The green rectangles represent the rate of direct reconstruction using the original samples and the blue ones denote the reconstruction rate of deep-features-based results. We can see an apparent dividing line between green and blue in the figure to the right of the red arrow. It can be safely concluded that the proposed method has verified the feasibility of the model in the small sample anomaly detection task from both theory and effect.

TABLE 2: Experiments results of dataset CIC-IDS2012.

Methods	With unknown categories			Known categories		
	Recall	Precision	Macro-f1	Recall	Precision	Macro-f1
Similarity measure	0.5545	0.5567	0.5555	0.4996	0.4996	0.4996
K-means cluster	0.2394	0.3367	0.2798	0.4595	0.3238	0.3799
AE	0.8066	0.8713	0.8335	0.7908	0.8372	0.8068
CNN with AE	0.9455	0.9522	0.9488	0.9418	0.9519	0.9465
<b>DFAE</b>	<b>0.9490</b>	<b>0.9668</b>	<b>0.9576</b>	<b>0.9480</b>	<b>0.9649</b>	<b>0.9557</b>

TABLE 3: Experiments results of dataset USTC-TFC2016.

Methods	With unknown categories			Known categories		
	Recall	Precision	Macro-f1	Recall	Precision	Macro-f1
Similarity measure	0.4525	0.4502	0.4514	0.2693	0.2630	0.2660
K-means cluster	0.7965	0.6003	0.5557	0.6535	0.6861	0.5465
AE	0.8105	0.9654	0.8692	0.8573	0.9050	0.8751
CNN with AE	0.8765	0.8410	0.8576	0.9565	0.9675	0.9617
<b>DFAE</b>	<b>0.9552</b>	<b>0.9675</b>	<b>0.9613</b>	<b>0.9983</b>	<b>0.9965</b>	<b>0.9973</b>

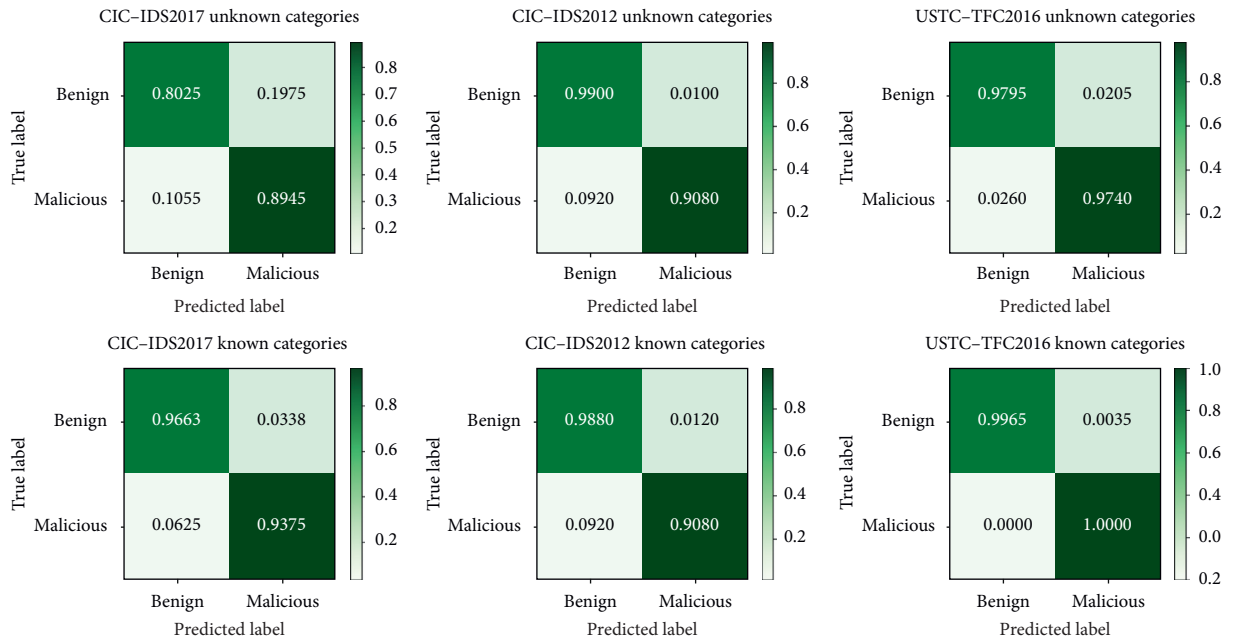


FIGURE 8: Confusion matrix of experiment results on each dataset.

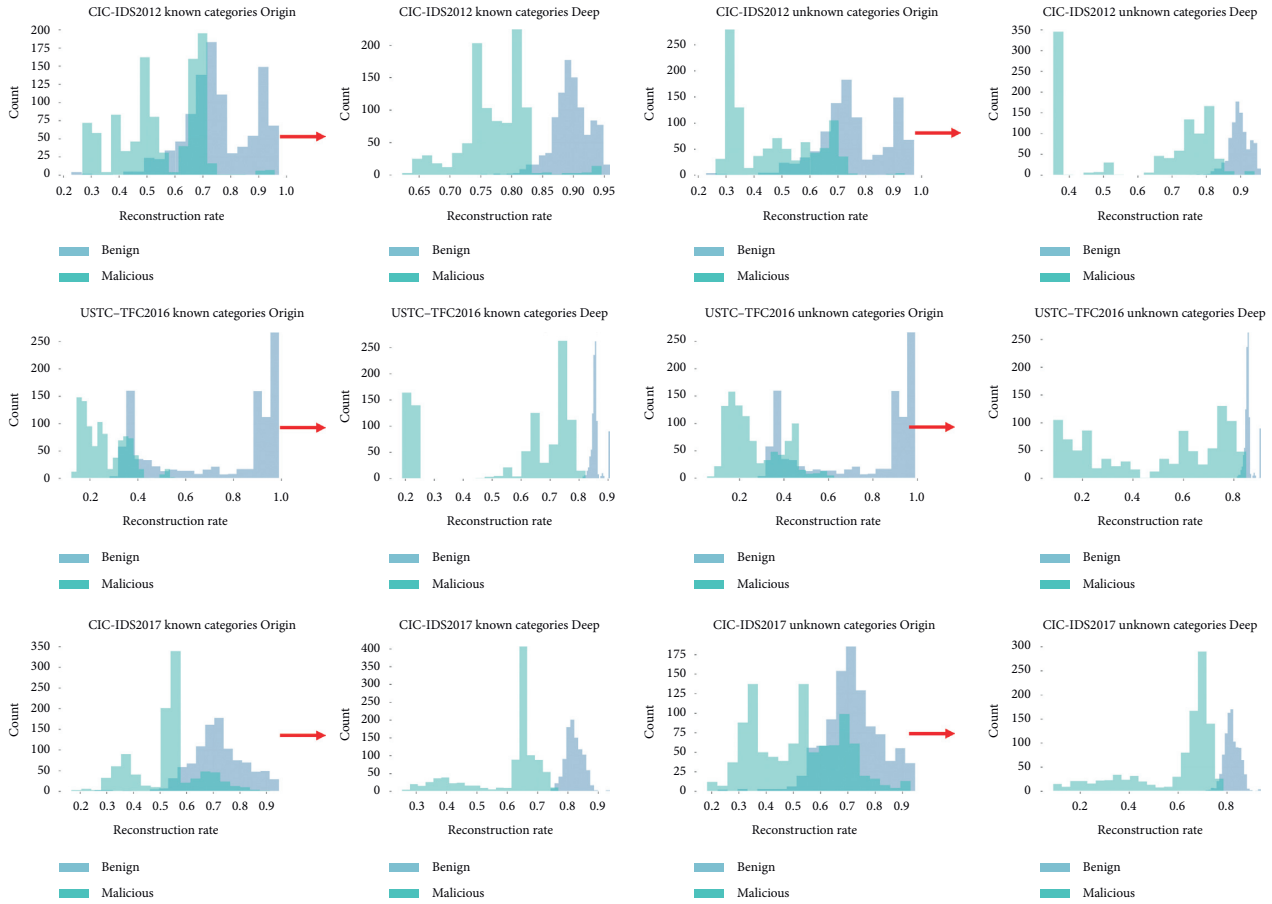


FIGURE 9: Reconstruction rate of different datasets. The image on the left of the red arrow is the result of reconstruction using AE only and the image on the right of the red arrow is the result of data reconstruction using the proposed method.

### 5. Conclusion

This paper proposes an anomaly detection model with a few abnormal samples to solve the problem in few-shot detection based on CNN and AE. This is a very common scenario in the application of network abnormal behavior detection. In the actual production and application, the number of normal behaviors is far greater than the number of abnormal behaviors. The model proposed in this paper can solve this problem and improve the detection results. As described above, the results demonstrate the improvements under this module of detection recall, precision, and f1-score in three datasets. What is more, the experiment proves that, through a small number of malicious samples for pretraining, the data reconstruction task will become easier, and the few-shot detection effect can also be improved in an obvious way. Sufficient comparative experiments verified the effectiveness of the proposed method. With regard to the network traffic detection, it is not easy to detect the anomaly directly through an unsupervised model, which can be easily

reflected from comparative experiments. Therefore, it is necessary to deal with the limited amount of labeled abnormal behavior samples. The few-shot malicious traffic detection task plays a crucial role in practical applications.

In the foreseeable future, we will continue to raise the few-shot malicious traffic detection results by different methods and to increase the possibility of applying this model into the actual and real-time network traffic. As for actual network traffic anomaly detection, it is always difficult to find out malicious behaviors when a series of new attacks occur. It is unrealistic to focus on training and learning new categories. But few-shot detection is meaningful here. Most of the time, we are concerned about the normal data and the new types of abnormal samples are only trained in the set intervals by few-shot learning. This is also the reason why this method is more feasible in the actual application scenarios. Meanwhile, we will tend to utilize fewer training samples to achieve a higher outcome on different datasets and production data. Detection efficiency will also be one of the focuses in future work.

## Data Availability

The datasets used in this paper are available at <https://www.unb.ca/cic/datasets/>, and they are also available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that there are no conflicts of interest.

## Acknowledgments

The authors acknowledge ICN&CAD Laboratory of School of Electronic Engineering, Beijing University of Posts and Telecommunications, for the experimental environment. Special thanks are due to Lv Tianqi, Xiao Yabo, Wu Lingrui, Fan Yiqi, Keiven, and Zhang Yu for their great help in this work. This work was supported by the National Natural Science Foundation of China (Grant no. 62071056).

## References

- [1] Statista Research Department, "IoT: number of connected devices worldwide 2012–2025, Statista," 2019, <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>.
- [2] S. A. R. Shah and I. Biju, "Performance comparison of intrusion detection systems and application of machine learning to snort system," *Future Generation Computer Systems*, vol. 80, no. 1, pp. 157–170, 2017.
- [3] A. Beaugnon and P. Chifflier, "Machine learning for computer security detection systems: practical feedback and solutions," 2018, <https://www.ssi.gouv.fr/uploads/2018/11/machine-learning-for-computer-security-abeaugnon-pchifflier-anssi-.pdf>.
- [4] K. Sailesh and S. Dharmapurikar, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 339–350, 2006.
- [5] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2014.
- [6] Z. G. Qu, S. Y. Chen, and X. J. Wang, "A secure controlled quantum image steganography algorithm," *Quantum Information Processing*, vol. 19, no. 380, pp. 1–25, 2020.
- [7] C. Qian, X. Li, N. Sun, and Y. Tian, "Data security defense and algorithm for edge computing based on mean field game," *Journal of Cyber Security*, vol. 2, no. 2, pp. 97–106, 2020.
- [8] D. Plonka, "FlowScan: a network traffic flow reporting and visualization tool," in *Proceedings of the 14th Conference on Systems Administration (LISA 2000)*, New Orleans, LA, USA, December 2000.
- [9] E. Kenigsberg, M. Gopshtein, and N. Ioffe, "Collecting network-level packets into a data structure in response to an abnormal condition," 2012, <https://www.freepatentsonline.com/8135979.pdf>.
- [10] J. Liu, F. Liu, and N. Ansari, "Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop," *IEEE Network*, vol. 28, no. 4, pp. 32–39, 2014.
- [11] M. Z. N. Saavedra and W. E. S. Yu, "A comparison between text, parquet, and PCAP formats for use in distributed network flow analysis on Hadoop," *International Conference on Networking and Information Technology*, vol. 5, no. 2, pp. 59–64, 2017.
- [12] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 712–717, Da Nang, Vietnam, January 2017.
- [13] R. Bala and R. Nagpal, "A review on KDD CUP99 and NSL-KDD dataset," *International Journal of Advanced Research in Computer Science*, vol. 10, no. 2, p. 64, 2019.
- [14] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.
- [15] P. Negandhi, Y. Trivedi, and R. Mangrulkar, "Intrusion detection system using random forest on the NSL-KDD dataset," in *Proceedings of the Emerging Research in Computing, Information, Communication and Applications*, pp. 519–531, Singapore, 2019.
- [16] I. H. Sarker and A. S. M. Kayes, "Cybersecurity data science: an overview from machine learning perspective," *Journal of Big Data*, vol. 7, no. 1, pp. 1–29, 2020.
- [17] B. G. Atli, Y. Miche, A. Kalliola, I. Oliver, S. Holtmanns, and A. Lendasse, "Anomaly-based intrusion detection using extreme learning machine and aggregation of network traffic statistics in probability space," *Cognitive Computation*, vol. 10, no. 5, pp. 848–863, 2018.
- [18] L. D'Hooge and A. S. M. Kayes, "Inter-dataset generalization strength of supervised machine learning methods for intrusion detection," *Journal of Information Security and Applications*, vol. 54, no. 1, p. 102564, 2020.
- [19] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "Deep-Full-Range: a deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, no. 1, pp. 45182–45190, 2019.
- [20] N. Thapa, Z. Liu, D. B. Kc, B. Gokaraju, and K. Roy, "Comparison of machine learning and deep learning models for network intrusion detection systems," *Future Internet*, vol. 12, no. 10, p. 167, 2020.
- [21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Intrusion detection evaluation dataset (CIC-IDS2017)," in *Proceedings of the of Canadian Institute for Cybersecurity*, Fredericton, Canada, 2018.
- [22] A. Javaid, Q. Niyaz, and W. Sun, "A deep learning approach for network intrusion detection system," in *Proceedings of the the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, pp. 21–26, New York City, NY, USA, December 2016.
- [23] M. Chen, X. Wang, M. He, L. Jin, K. Javeed, and X. Wang, "A network traffic classification model based on metric learning," *CMC-computers Materials & Continua*, vol. 64, no. 2, pp. 941–959, 2020.
- [24] M. Song, J. Ran, and S. Li, "Encrypted traffic classification based on text convolution neural networks," in *Proceedings of the IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 432–436, Dalian, China, October 2019.
- [25] H. Zhu, F. Meng, S. Rho et al., "Long short term memory networks based anomaly detection for KPIs," *Computers, Materials & Continua*, vol. 61, no. 2, pp. 829–847, 2019.
- [26] C. Du, S. Liu, L. Si, Y. Guo, and T. Jin, "Using object detection network for malware detection and identification in network

- traffic packets,” *Computers, Materials and Continua*, vol. 64, no. 3, pp. 1785–1796, 2020.
- [27] Y. Wei, F. R. Yu, M. Song et al., “User scheduling and resource allocation in HetNets with hybrid energy supply: an actor-critic reinforcement learning approach,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680–692, 2017.
- [28] Y. Wei, F. R. Yu, M. Song et al., “Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2061–2073, 2018.
- [29] G. Caminero, M. Lopez-Martin, and B. Carro, “Adversarial environment reinforcement learning algorithm for intrusion detection,” *Computer Networks*, vol. 159, no. 1, pp. 96–109, 2019.
- [30] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” *Neural Networks*, vol. 106, no. 1, pp. 249–259, 2018.
- [31] S. Rodda and U. S. R. Erothi, “Class imbalance problem in the network intrusion detection systems,” in *Proceedings of the 2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)*, pp. 2658–2688, Chennai, India, March 2016.
- [32] Y. Gu, K. Li, Z. Guo, and Y. Wang, “Semi-supervised  $K$ -means DDoS detection method using hybrid feature selection algorithm,” *IEEE Access*, vol. 7, no. 1, pp. 64351–64365, 2019.
- [33] H. Xu, W. Chen, N. Zhao et al., “Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications,” in *Proceedings of the 2018 World Wide Web Conference*, pp. 187–196, Lyon, France, April 2018.
- [34] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.
- [35] D. Yuan, K. Ota, M. Dong et al., “Intrusion detection for smart home security based on data augmentation with edge computing,” in *Proceedings of the 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, Dublin, Ireland, June 2020.
- [36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [37] Y. Xiao, D. Yu, X. Wang et al., “SPCNet: spatial preserve and content-aware network for human pose estimation,” in *Proceedings of the 24th European Conference on Artificial Intelligence*, pp. 2776–2783, Santiago de Compostela, Spain, August 2020.
- [38] C. S. N. Pathirage, J. Li, L. Li, H. Hao, W. Liu, and R. Wang, “Development and application of a deep learning-based sparse autoencoder framework for structural damage identification,” *Structural Health Monitoring*, vol. 18, no. 1, pp. 103–122, 2019.
- [39] G. E. Dahl, T. N. Sainath, and G. E. Hinton, “Improving deep neural networks for LVCSR using rectified linear units and dropout,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal*, pp. 8609–8613, Vancouver, Canada, May 2013.
- [40] M. Lau and K. Lim, “Investigation of activation functions in deep belief network,” in *Proceedings of the 2nd International Conference on Control and Robotics Engineering (ICCRE)*, pp. 201–206, Vancouver, Canada, April 2017.
- [41] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, Salt Lake City, UT, USA, June 2018.