Hindawi
Security and Communication Networks
Volume 2021, Article ID 9486949, 23 pages
https://doi.org/10.1155/2021/9486949

Research Article

Network Intrusion Detection Model Based on Improved BYOL Self-Supervised Learning

Zhendong Wang, Zeyu Li, Junling Wang, and Dahai Li

School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China

Correspondence should be addressed to Zeyu Li; 6720200836@mail.jxust.edu.cn

Received 3 August 2021; Accepted 8 October 2021; Published 28 October 2021

Academic Editor: Wei Feng

The combination of deep learning and intrusion detection has become a hot topic in today's network security. In the face of massive, high-dimensional network traffic with uneven sample distribution, how to be able to accurately detect anomalous traffic is the primary task of intrusion detection. Most research on intrusion detection systems based on network anomalous traffic detection has focused on supervised learning; however, the process of obtaining labeled data often requires a lot of time and effort, as well as the support of network experts. Therefore, it is worthwhile investigating the development of label-free self-supervised learning-based approaches called BYOL which is a simple and elegant framework with sufficiently powerful feature extraction capabilities for intrusion detection systems. In this paper, we propose a new data augmentation strategy for intrusion detection data and an intrusion detection model based on label-free self-supervised learning, using a new data augmentation strategy to introduce a perturbation enhancement model to learn invariant feature representation capability and an improved BYOL self-supervised learning method to train the UNSW-NB15 intrusion detection dataset without labels to extract network traffic feature representations. Linear evaluation on UNSW-NB15 and transfer learning on NSK-KDD, KDD CUP99, CIC IDS2017, and CIDDS_001 achieve excellent performance in all metrics.

1. Introduction

With the advent of the information age and the popularity of the Internet, all aspects of our lives have changed greatly. While the Internet has given us significant convenience, it has also brought about a variety of network security issues. How to avoid these security problems has become the focus of the industry. Intrusion detection, as an important part of the network security system, was first proposed by Anderson [1], who defined an intrusion attempt or threat as a potential, premeditated, unauthorized attempt to access information and manipulate information, making the system unreliable or unusable. The earliest intrusion detection model was proposed by Denning and Neumann [2], which focuses on generating a number of profiles about the system based on the audit log data of the host system and monitoring the variance of the profiles to detect intrusions in the system. According to different data sources, intrusion detection systems can be classified as host-based intrusion detection system (HIDS) [3] and network-based intrusion detection system (NIDS) [4]. NIDS observes and analyses real-time network traffic and monitors multiple hosts, aiming to detect intrusions in the network by collecting packet information and viewing its contents [5]. Previous researchers have mostly used pattern-matching algorithms to analyse their data, and feature selection usually includes three schemes, which are filtered approaches (e.g., information gain and correlation coefficient algorithms), encapsulated methods (e.g., genetic algorithms [6] and particle swarm algorithms [7]), embedded methods (e.g., LASSO regression algorithms), and linear transformation methods for feature extraction, such as principal component analysis (PCA) and linear discriminant analysis, as well as nonlinear transformation methods, like kernel-based principal component analysis. However, all of the above methods have certain drawbacks. For example, genetic algorithms are prone to

premature convergence problems. As far as PCA algorithm, the meaning of each feature dimension of the principal components in PCA algorithms is somewhat ambiguous and not as interpretable as the original. Moreover, the interpretability of the samples is not as strong as that of the original samples.

Traditional NIDS also has a large number of problems such as terrible detection rate of unknown attacks, high false alarm rate, and high resource consumption. The machine learning algorithms have lots of advantages such as strong generalization ability, simple implementation, and easy to understand and explain. Traditional machine learning algorithms like support vector machine (SVM), decision tree (DT), and K nearest neighbor (KNN) have been introduced into the field of intrusion detection to improve the efficiency of intrusion detection and reduce the false negative rate and false positive rate in recent years. Nonetheless, the complexity of traditional machine learning algorithms makes their performance and accuracy in dealing with high-dimensional massive data to be far away from deep learning methods. Not only that, traditional machine learning algorithms also rely on feature engineering and we need to design algorithms to extract effective features of network traffic, which greatly increases the computational cost. Deep learning methods do not require human experience to extract feature information but algorithms automatically learn feature information from original data, known as representation learning, which means farewell to task-heavy feature engineering. Furthermore, deep learning methods can extract better feature representations from massive amounts of data to create models with better generalization capabilities. In recent years, convolutional neural network (CNN) and recurrent neural network (RNN) have been widely used in the field of intrusion detection. For example, CNN methods convert one-dimensional network traffic into two-dimensional grayscale images and then use the convolutional kernel to extract effective features of network traffic to improve the detection rate of intrusion detection. However, there are many weak points in intrusion detection models based on supervised learning, the main point being the cost of acquiring labeled data, requiring professional security experts to scrutinize traffic data and decide whether a particular pattern is a new attack, which undoubtedly increases the cost of intrusion detection. Based on the above drawbacks, unsupervised learning methods have recently gained attention in the domain of intrusion detection, where various types of autoencoders (e.g., variational autoencoder, sparse autoencoder, and denoising autoencoder) and generative adversarial neural networks have been applied to reconstruct network traffic samples and learn feature representations of them. Although unsupervised learning methods can learn feature representation without labeled data, the learned feature representations are only applicable to certain datasets and cannot be transferred to other datasets, which definitely limits the generalization ability of the model.

Given the shortcomings of traditional machine learning, supervised learning, and unsupervised learning, this paper proposes a new intrusion detection model based on improved BYOL self-supervised learning. In view of the disadvantages that supervised learning methods require the use of a large amount of manually labeled data and the poor generalization ability of unsupervised learning models, we adopt self-supervised learning, which can be trained without labels, and it can fully exploit its own supervisory information from large-scale unsupervised data and train the network with this constructed supervisory information to learn highly generalizable and valuable data. The essence of deep learning lies in its powerful representation learning capability. The excellent results obtained in the transfer learning experiments of NSL-KDD, KDD CUP99, CIC IDS2017, and CIDDS_001 datasets are enough to prove the strong generalization ability of the self-supervised learning model and the generality of the extracted network traffic feature representations.

In general, the work of this paper has made the following contributions to the intrusion detection domain:

(1) Self-supervised learning is introduced to the field of intrusion detection and the strong potential and scope of self-supervised learning in intrusion detection are validated.

(2) A novel data augmentation strategy for the intrusion detection dataset is proposed to introduce different perturbations to generate samples with different perspectives to enhance the feature representation ability of the model to learn network traffic efficiently.

(3) The BYOL self-supervised learning algorithm is improved by introducing BoTNet with multihead attention mechanism to suppress the features that contribute less to classification and increase the features that contribute more to classification, so as to promote the performance of the model. The BYOL loss function is optimized to make the model training process smoother and the model converge faster, thereby enhancing the stability and robustness of the model.

(4) To verify the effectiveness of the methods and models proposed in this paper, we apply linear evaluation on UNSW-NB15 and transfer learning on NSL-KDD, KDD CUP99, CIC IDS2017, and CIDDS_001 datasets and compare them with various machine learning methods and state-of-the-art deep learning models using several experimental evaluation metrics.

The rest of the paper is organized as follows: Related works are discussed in Section 2. The network intrusion detection model based on improved BYOL self-supervised learning is presented in Section 3. Section 4 describes the dataset used in this paper and the preprocessing of the dataset. In Section 5, we validate the effectiveness of the improved BYOL self-supervised learning intrusion detection model through relevant experiments. Finally, we draw the related conclusions as well as suggest some future works in Section 6.

2. Related Works

Nowadays, with the development of science and technology, new methods such as data mining, machine learning, and deep learning have been applied to the domain of intrusion detection [8–10]. While data mining algorithms usually require a large amount of data to extract feature information, the detection rate is low for those categories which consist of insufficient samples; furthermore, most data mining algorithms are sensitive to noise. If the dataset contains plenty of noisy data, there is no doubt that it will have a huge impact on the algorithm. Currently, researchers apply all kinds of machine learning methods to detect anomalous network traffic. These methods include KNN, DT, SVM, LR, and ensemble learning. Kabir et al. [11] proposed an intrusion detection model based on least square support vector machine (LSSVM), which selects representative samples from randomly divided subgroups of the dataset in order to make it reflect the whole significant features. Nancy et al. [12] proposed a dynamic recursive feature selection algorithm to select an optimal number of features from the dataset, and then, an intelligent fuzzy temporal decision tree algorithm integrated with convolution neural networks was used for classification. The experimental results showed that the new types of attacks could be detected well and it also reduced the network delay and false negative rate. Hurley et al. [13] used PCA for feature extraction, combined with fuzzy techniques to obtain the degree of sample objects belonging to each category, and then used KNN to classify the attack categories, while the accuracy of this algorithm gradually decreases as the amount of data increases. Most traditional machine learning methods are shallow learning and emphasize feature engineering and feature selection. They cannot effectively solve the problem of classifying large-scale intrusion data appearing in the actual web application environment, and the performance and accuracy of the algorithm in dealing with multiple classification problems decrease sharply with the dynamic growth of the dataset. Moreover, shallow learning is not suitable for the prediction requirements of high-dimensional massive data.

Since recent years, deep learning methods such as CNN and RNN have been widely used among the intrusion detection. Riyaz and Ganapathy [14] proposed a new feature selection algorithm called conditional random field and linear correlation coefficient-based feature selection algorithm to select the most contributed features and classify them using the existing convolutional neural network; this new feature selection algorithm not only greatly reduces the training time but also increases the accuracy of the model by eliminating irrelevant features. Yang and Wang [15] proposed a convolutional neural network with cross-layer feature fusion using the structural properties of convolutional neural networks combined with the cross-layer aggregation design concept, and the experimental results showed that the model has a high accuracy, true positive rate, and low false alarm rate in intrusion detection. RNN and LSTM treat the network traffic as sequential data. The authors of [16–18] proposed RNN-based, gated recurrent units-based, and LSTM-based intrusion detection systems that treat data as time series, respectively, and experimented on KDD CUP99 and NSL-KDD datasets with good results. However, there are many drawbacks in the intrusion detection model based on supervised learning, the most important point being it is expensive to obtain labeled data, requiring professional security experts to scrutinize traffic data and decide whether a particular pattern is a new attack, which undoubtedly increases the cost of intrusion detection. Unsupervised learning methods are also gaining importance in the field of intrusion detection. Choi et al. [19] used autoencoder, denoising autoencoder, variational autoencoder, and stacked autoencoder to train datasets and construct thresholds to discriminate traffic types based on the mean and variance of reconstruction errors and upper alpha quantile obtained from each model. Farahnakian and Heikkonen [20] proposed a deep autoencoder trained in a greedy layerwise fashion in order to avoid overfitting and local optima and achieved 94.53% multiclassification accuracy and only 0.42% false alarm rate in KDD CUP99. Sakurada and Yairi [21] proposed a method to perform anomaly detection based on reconstruction error thresholds using autoencoders. In the training phase, the autoencoder model learns to reconstruct its input data, which includes only normal data. In the testing phase, test data are fed into the learned model to output the reconstructed test data. When the reconstruction error is higher than some threshold arbitrarily chosen by the user, the data are determined to be anomalous and vice versa. But, the unsupervised learned features are only applicable to this dataset and cannot be transferred to other datasets, which definitely limits the generalization capability of the model.

In summary, with the development of time and technology, machine learning, deep learning, and unsupervised learning have made good progress in the field of intrusion detection. The study in [22] used Markov models for feature extraction and solved the problem that Bayesian network classifiers were usually trained on data by selecting suboptimal model heuristics, but the evaluation indicators used in their paper were not exhaustive. The work in [23] used information gain algorithm for feature extraction and solved the problem that the KDD CUP99 dataset did not include the current state of cyber attacks, while the F-measure for the unknown attack category was low. The study in [24] solved the problem that the data became more complex and it was difficult to extract better low-dimensional features effectively as the number of features increased and the accuracy of binary classification on NSL-KDD achieved 95.25%. The work in [25] solved the problem that the traditional machine learning techniques could not solve the intrusion detection problem. Furthermore, the binary classification results on UNSW-NB15 achieved excellent performance in all metrics. The study in [26] used the autoencoder for feature extraction and solved the problem that traditional dimension reduction methods have difficulty capturing nonlinear information in the data; however, the model evaluation index is single and only the accuracy used. The study in [27] used the convolution autoencoder for feature extraction and solved the problem that traditional log anomaly detection ignores the temporal pattern of logs as well as a problem of information

loss caused by vector representation, while the F-measure was as low as 73.76%. Unlike existing intrusion detection models, this paper proposes an intrusion detection model based on a self-supervised learning approach. Not only do we take into account the difficulty of acquiring labeled data, but also we attach significant importance to the generalization ability of the model. The model is applied to the intrusion detection benchmark datasets KDD CUP99, NSL-KDD, UNSW-NB15, CIC IDS2017, and CIDDS_001. The intrusion detection dataset used in this paper is relatively complete. We use multiple evaluation indicators such as accuracy, precision, detection rate, F1 score, ROC curve, and AUC value to evaluate the performance of the proposed model, which makes the evaluation of the proposed method more scientific and comprehensive.

3. Materials and Methods

3.1. Bootstrap Your Own Latent. Typical methods for self-supervised learning include CPC [28], MoCo [29], SimCLR [30], DINO [31], and BYOL [32]. CPC is mainly applied in video and speech fields for processing serialized information and SimCLR and MoCo need lots of positive and negative sample pairs and large batch sizes to train to get excellent feature representations, while Dino uses ViT [33] as a feature extractor. In the field of intrusion detection, a larger batch size means that larger memory is needed to process the data and the large number of parameters in ViT makes it difficult for real-time detection, so this paper adopts BYOL as the intrusion detection model. BYOL is a simple and elegant self-supervised learning framework that does not require positive or negative sample pairs and a large batch size to train a network with sufficiently powerful feature extraction capabilities. Furthermore, BYOL does not require human experience to extract feature information but algorithms automatically learn feature information from original data, which means that there is no need to do feature engineering, so it can save much time and effort to do other things. Furthermore, BYOL can extract better feature representations from massive amounts of data to create models with better generalization capabilities. That is to say, the feature representations extracted from BYOL can be applied to other tasks whose domain is as same as that of the original dataset, and BYOL's goal is to learn a representation y_θ which can then be used for downstream tasks. It uses two interacting and mutually learning asymmetric neural networks, called online network and target network, to train target network feature representations of the same image in different augmented views. The symbols used in this paper are explained in Table 1.

Assuming that its network weight parameters are denoted by θ in the online network, the online network includes encoder f_θ for feature extraction, projector g_θ for feature projection, and predictor q_θ for feature prediction. As for the target network, its weight parameters are denoted by ξ. The target network includes encoder f_ξ for feature extraction and projector g_ξ for feature projection. The specific training process is shown in Figure 1.

Given a set of network traffic X, a grayscale image $x \sim X$ is sampled uniformly from X (we can regard x as a greyscale image after preprocessing and reshaping the network traffic), then we need to apply two different sets of image augmentation operations t and $t\prime$ on x, respectively, and the resulting augmented views are v and $v\prime$, where $v = t(x)$ and $v\prime = t\prime(x)$. From the first augmented view v, the online network outputs a representation $y_\theta \triangleq f_\theta(v)$, projection $z_\theta \triangleq g_\theta(y_\theta)$, and a prediction $q_\theta(z_\theta)$. The target network outputs $y'_\xi \triangleq f_\xi(v\prime)$ and the target projection $z'_\xi \triangleq g_\xi(y'_\xi)$ from the second augmented view $v\prime$. Then, we do l_2-normalization on $q_\theta(z_\theta)$ and z'_ξ. The unit length of the two latent variables is taken, and only their directionality is preserved to pave the way for finding the loss function later.

$$\begin{aligned}\overline{q_\theta}(z_\theta) &\triangleq \frac{q_\theta(z_\theta)}{q_\theta(z_\theta)_2},\\ \overline{z'_\xi} &\triangleq \frac{z'_\xi}{\left\|z'_\xi\right\|_2}.\end{aligned} \tag{1}$$

Finally, the loss function of BYOL is trained with the online network and target network by constraining the similarity of the normalized online predictions and target projections.

$$L_{\theta,\xi} \triangleq \left\|q_\theta(z_\theta) - z'_\xi\right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi\rangle}{\left\|q_\theta(z_\theta)\right\|_2 \cdot \left\|z'_\xi\right\|_2}. \tag{2}$$

We symmetrize the loss $L_{\theta,\xi}$ in equation (2) by separately feeding $\dot{v}$ to the online network and v to the target network to compute $\widetilde{L}_{\theta,\xi}$; then, the loss function of BYOL can be written as

$$L_{\theta,\xi}^{\mathrm{BYOL}} = L_{\theta,\xi} + \widetilde{L}_{\theta,\xi}. \tag{3}$$

At each training step, we perform a stochastic optimization step to minimize $L_{\theta,\xi}^{\mathrm{BYOL}}$ with respect to θ only, but not ξ, as depicted by the stop gradient in Figure 1. BYOL's dynamics are summarized as

$$\begin{aligned}\theta &\leftarrow \mathrm{optimizer}\left(\theta, \nabla_\theta L_{\theta,\xi}^{\mathrm{BYOL}}, \eta\right),\\ \xi &= \tau \cdot \xi + (1-\tau) \cdot \xi,\end{aligned} \tag{4}$$

where optimizer is an optimizer and η is a learning rate.

Also known as the EMA, the weight update approach (exponential moving average), where $\tau \in [0, 1]$, is an artificially hyperparameter. At the end of training, we only keep the encoder f_θ, as in [29]. The full training steps of BYOL algorithm are shown in Algorithm 1.

From Section 3.1, we can draw the conclusion that the intrusion detection model based on improved BYOL self-supervised learning can be divided into four main steps: (1) data augmentation, (2) feature representation, (3) feature projection, and (4) contrastive learning. Next, we will discuss the specific implementation of these four steps and how to integrate improved BYOL self-supervised learning into the intrusion detection domain.

Table 1: Notation.

Symbol description	Description
x	Network traffic sample
X	Network traffic dataset
θ, ξ	Parameters of online and target network
f_θ, f_ξ	Feature extraction operation
g_θ, g_ξ	Feature projection operation
q_θ	Feature prediction operation
$t, t\prime$	Data augmentation operations
$v, v\prime$	Views after data augmentation
y_θ, y'_ξ	Feature vectors of v and $v\prime$
z_θ, z'_ξ	Projection vectors of v and v'
$q_\theta(z_\theta)$	Prediction vector of v
$\overline{q_\theta(z_\theta)}, \overline{z'_\xi}$	Prediction vector of v and projection vector of v' after l_2-normalization
$sg(*)$	Stop gradient
$f_Q(*), f_K(*), f_V(*)$	Convolutional operations with 1×1 kernel size
R_h, R_w	The relative position encoding of the image's height and width
softmax$(*)$	The function of SoftMax
W, b	Weights and bias of the fully connected layer
BN	Batch normalization layer
σ	ReLU activation function

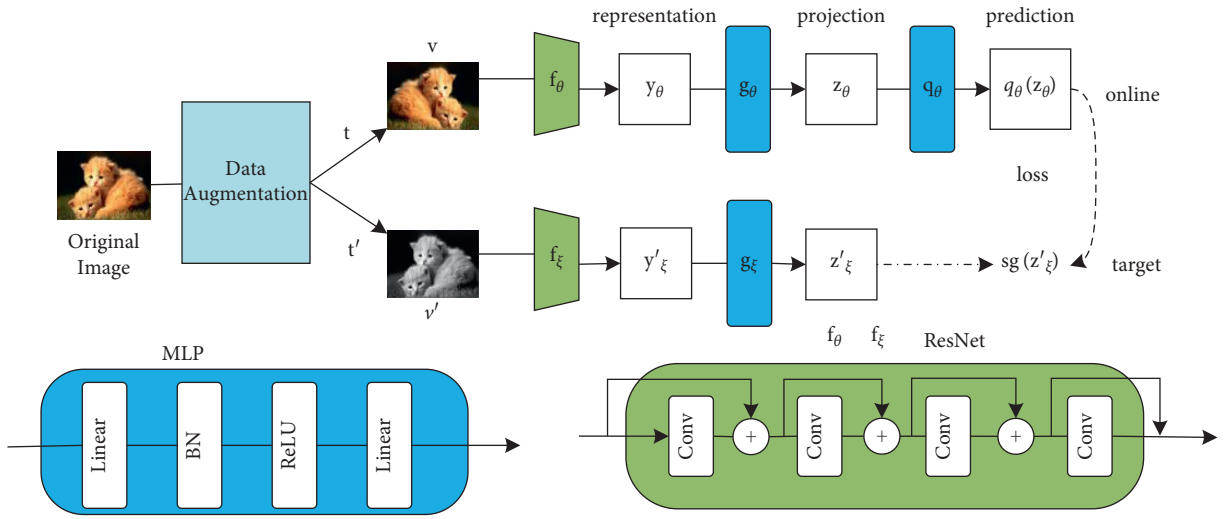


Figure 1: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $sg(z'_\xi)$, where θ and ξ are the trained weights of online and target network and sg means stop gradient. At the end of training, everything, but f_θ, is discarded and y_θ is used as the image representation.

3.1.1. Data Augmentation. A set of data augmentation operations play a crucial role in learning of good data representations. Different data augmentation operations introduce different perturbations and generate samples under different enhancement views. BYOL self-supervised learning can learn the feature representation of network traffic invariance precisely by pulling the different augmentation views of the same image and pushing the augmentation views of different images, thus showing the importance of data augmentation for self-supervised learning. The existing image data augmentation operations are as follows: colour jittering (the brightness, saturation, and contrast transformation), Gaussian blur, colour dropping (a conversion to grayscale), horizontal-vertical flipping, and random cropping. For intrusion detection data, the network traffic has been converted to the grayscale format after preprocessing, so the two data augmentation operations of colour dropping and colour jittering are not needed. At the same time, the network traffic data have been normalized to a value between 0 and 1 after the data normalization in preprocessing, and if the Gaussian fuzzy operation is added, noise will be introduced, which will greatly reduce the effect of feature extraction.

As a result, this paper proposes a new data augmentation operation named *random_shuffle* for intrusion detection data. Given an input set $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$

Inputs:
X, T, and $T\prime$ set of images and combination of transformations
$\theta, f_\theta, g_\theta$, and q_θ initial online parameters
ξ, f_ξ, and g_ξ initial target parameters
optimizer updates online parameters using the loss gradient
S and N total number of optimization steps and batch size
$\{\tau_s\}_{s=1}^{S}$ and $\{\eta_s\}_{s=1}^{S}$ target network updates schedule and learning rate schedule
for $s = 1$ to S do
$B \leftarrow \{x_i \sim D\}_{n=1}^{N}$ // sample a batch of N images
for $x_i \in B$ do
$t \sim T$ and $t\prime \sim T\prime$ // sample image transformations
$z_1 \leftarrow g_\theta(f_\theta(t(x_i)))$ and $z_2 \leftarrow g_\theta(f_\theta(t\prime(x_i)))$// compute projections
$z_1' \leftarrow g_\theta(f_\theta(t'(x_i)))$ and $z_1' \leftarrow g_\theta(f_\theta(t(x_i)))$ // compute target projections
$l_i \leftarrow -2 * ((\langle q_\theta(z_1), z_1' \rangle / \|q_\theta(z_1)\|_2 \cdot \|z_1'\|_2) + (\langle q_\theta(z_2), z_2' \rangle / \|q_\theta(z_2)\|_2 \cdot \|z_2'\|_2)) \ldots$//.compute the loss for.$x_i$
end
$\delta\theta \leftarrow (1/N) \sum_{i=1}^{N} \partial_\theta l_i$.
$\theta \leftarrow$ optimizer$(\theta, \delta\theta, \eta_s)$ // update online network parameters
$\xi \leftarrow \tau_s \xi + (1 - \tau_s)\theta$ // update target network parameters
end
Output: encoder f_θ

Algorithm 1: BYOL (bootstrap your own latent).

representing Nnetwork traffic data. In addition, each sample $x^{(i)}$ is a d-dimensional feature vector and can be described as $x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$, where $x_j^{(i)}$ denotes the j-th feature of the traffic data $x^{(i)}$. It is noted that $x^{(i)}$ is usually a high-dimensional feature vector. We can use the *random_shuffle* function to randomly disrupt the positions of the features to obtain the augmented data $x\prime$. The *random_shuffle* function uses the modern version of the Fisher–Yates algorithm, and we can view network traffic data as an array of $x_1, x_2, \ldots, x_d$; then, the modern version of the Fisher–Yates algorithm pseudo-code is shown in Algorithm 2.

For example, suppose the original array is [9, 6, 7, 2, 4, 5, 1, 3]. Table 2 shows how the modern version of the Fisher–Yates algorithm performs the shuffle operation on this array. Through this table, we will understand better how it works.

The 2D convolutional neural networks are usually used to process images in the two-dimensional (2D) array format. In order to make the network traffic conform to the input format of convolutional neural networks, the augmented data need to be subjected to the reshape operation. For instance, the preprocessed UNSW-NB15 network traffic sample has 196 dimensions, i.e., $x \in \mathbb{R}^{196}$. After reshaping, it was transformed into the grayscale map format, i.e., $x\prime \in \mathbb{R}^{14*14}$, and after that, multiple augmentation operations are selected from the four array augmentation operations of horizontal flip, vertical flip, random crop, and *random_shuffle* proposed in this paper to form a set of data augmentation operations. For example, we can define that $t\prime$ = {horizontal flip, vertical flip, random crop} and$t\prime$ = {vertical flip, random_shuffle, random crop}. After two sets of different data augmentation, the network traffic views v and $v\prime$ are obtained before they can be input to the f_θ and f_ξ for feature extraction. We selected two sets of different network traffic data augmentation comparison images in the UNSW-NB15 dataset for visualization. As shown in Figures 2 and 3, we can find that the network traffic after data augmentation retains the original traffic characteristics while also introducing different disturbances. In this way, the feature representations learned by the model are more generalized and the model can learn the feature representations of the invariance of network traffic.

3.1.2. Feature Representation. After two sets of data augmentation operations, we obtained two different sets of augmented views v and $v\prime$ of the original network traffic samples. According to the BYOL training framework, two different sets of views should be input to encoders f_θ and f_ξ for encoding to extract features at this time. ResNet [34] was selected as the feature encoder backbone to get the image representation in the BYOL paper; however, not every feature will play an effective role in the classification result for intrusion detection data. If we incorporate too many nonessential features, noise will be introduced, which will greatly affect the final classification result. In this paper, BoTNet [35], which uses the attention mechanism, is used as the backbone of the encoder since it can automatically learn and calculate the contribution of the input data to the output classification and can suppress the features that contribute less to the classification and increase the features that contribute more to the classification in the intrusion detection data. The only difference between the two is that BoTNet adds the global multihead self-attention mechanism in the c5 stage, as shown in Figure 4.

Suppose the input image is $x \in \mathbb{R}^{H \times W \times d}$ and $R_h \in \mathbb{R}^{H \times 1 \times d}$and $R_w \in \mathbb{R}^{1 \times W \times d}$ refer to the relative position encoding of the height and width, which represent the relative information in the vertical and horizontal directions of image x. Let the query matrix of the image be q, the key

```
# a[d]: an array of d elements (indices from 0 to d − 1)
for i from d − 1 downto 1 do
j ← random integer such that 0 ≤ j ≤ i
exchange a[j] and a[i]
```

Algorithm 2: Fisher–Yates algorithm.

Table 2: Fisher–Yates algorithm steps.

Range	Roll	Scratch	Result
		9 6 7 2 4 5 1 3	
1-8	6	9 6 7 2 4 3 1	5
1-7	2	9 1 7 2 4 3	6 5
1-6	6	9 1 7 2 4	3 6 5
1-5	1	4 1 7 2	9 3 6 5
1-4	3	4 1 2	7 9 3 6 5
1-3	3	4 1	2 7 9 3 6 5
1-2	1	1	4 2 7 9 3 6 5
			1 4 2 7 9 3 6 5

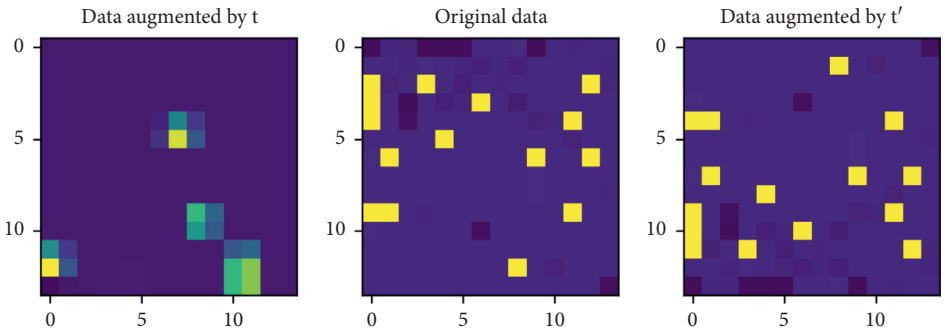


Figure 2: Normal traffic.

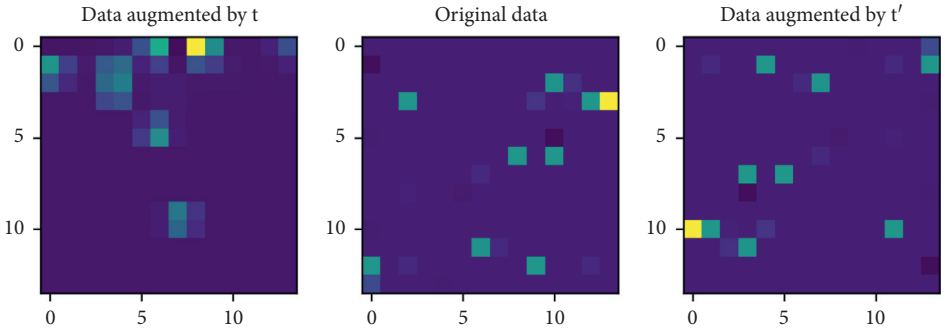


Figure 3: Anomaly traffic.

matrix of the image be k, and the value matrix of the image be v. Then, we can get them by convolving the input image x with three different 1×1 convolution kernels, respectively.

$$\begin{aligned} q &= f_Q(x), \\ k &= f_K(x), \\ v &= f_V(x), \end{aligned} \tag{5}$$

where $f_Q(x)$, $f_K(x)$, and $f_V(x)$ represent convolutional operations on x, $q \in \mathbb{R}^{H \times W \times d}$, $k \in \mathbb{R}^{H \times W \times d}$, and $v \in \mathbb{R}^{H \times W \times d}$.

Through q and k, the content-content encoding of the image can be obtained from formula (5) as follows:

$$\text{content\_content} = qk^T. \tag{6}$$

Given q, R_h, and R_w, the content-position encoding of the image can be calculated from formula (6) as follows:

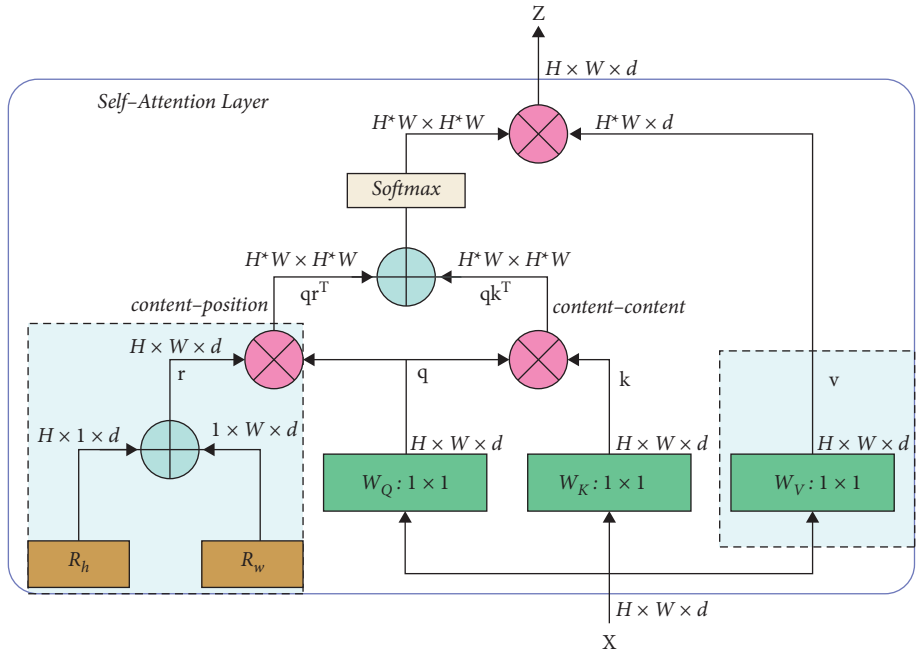


Figure 4: MHSA architecture.

$$\text{content\_position} = q\left(R_h + R_w\right)^T. \tag{7}$$

After obtaining the two encodings, we can obtain the attention matrix of the original image by the following formula:

$$\text{attention} = \text{softmax}\left(\text{content\_content} + \text{content\_position}\right), \tag{8}$$

where $\text{softmax}(*)$ indicates the function of SoftMax.

Finally, the output of the MHSA is generated based on v and attention, which can be expressed as follows:

$$z = \text{attention} \cdot v. \tag{9}$$

The entire MHSA process can also be represented by

$$z = \text{softmax}\left(f_Q(x)\left(R_h + R_w\right)^T + f_Q(x) f_K(x)^T\right) f_V(x). \tag{10}$$

The pseudo-code for the MHSA algorithm steps is shown in Algorithm 3.

The MSHA method is simple but powerful, the convolutional neural network can efficiently learn abstract and low-resolution feature maps of the images, and the global self-attention mechanism can process and summarize the information contained in the feature maps. It is this improvement that allows BoTNet to have a large improvement in accuracy in ImageNet competition [36] and to have 1.2 times fewer model parameters than ResNet50.

3.1.3. Feature Projection. After the f_θ and f_ξ encoding, the network traffic is converted from the grayscale image into vectors of y_θ and y'_ξ, which can be described as follows:

$$\begin{aligned} y_\theta &= f_\theta(v), \\ y'_\xi &= f_\xi(v'), \end{aligned} \tag{11}$$

where both the feature representations y_θ and y'_ξ of the network traffic correspond to the output of the final average pooling layer of BoTNet, v and $v\prime$ are the augmented views obtained after two data augmentation operations, respectively, and $y \in \mathbb{R}^d$, with d being an artificially hyperparameter. Subsequently, the resulting feature representations y_θ and y'_ξ of the network traffic are projected from the high-dimensional feature space to the low-dimensional hidden space by the multilayer perceptrons (MLPs) g_θ and g_ξ consisting of two hidden layers and a batch normalization layer to obtain z_θ and z'_ξ, which can be expressed as follows:

$$\begin{aligned} z_\theta &= W_\theta^{(2)}\left(\sigma\left(BN\left(W_\theta^{(1)} y_\theta + b_\theta^{(1)}\right)\right)\right) + b_\theta^{(2)}, \\ z'_\xi &= W_\xi^{(2)}\left(\sigma\left(BN\left(W_\xi^{(1)} z'_\xi + b_\xi^{(1)}\right)\right)\right) + b_\xi^{(2)}, \end{aligned} \tag{12}$$

where W and b are the weights and biases of the fully connected layer, BN is the batch normalization layer, and σ is the activation function of ReLU. The low-dimensional hidden space can be understood as a feature representation of the network traffic after censoring nonessential feature information (e.g., location information of the image) while reducing the feature dimensionality to decrease the computational resource. The feature projection identifies invariants in the data augmentation. Meanwhile, information that may be useful for downstream tasks such as the colour or orientation of objects in the image after data augmentation can be removed. By using the nonlinear transformations g_θ and g_ξ, more information can be formed and maintained in y_θ and y'_ξ. The feature projection step is

```
#f_q, f_k, f_v: 1 × 1 convolution for q, k and v
#rel_h, rel_w: relative position encodings for height and width
#heads: num of heads for MHSA
for x in loader: // load a minibatch x with N samples
b, C, width, height = x.size() //get batch, channels, width and height of minbatch x
q = f_q(x).view(b, heads, C//heads, −1) // apply convolution operation
k = f_k(x).view(b, heads, C//heads, −1)
v = f_v(x).view(b, heads, C//heads, −1)
content_content = bmm(q.permute(0, 1, 3, 2), k) // get content_content encoding
content_position = (rel_h + rel_w).view(1, heads, C//heads, −1).permute(0, 1, 3, 2)
content_position = bmm(content_position, q) // get content_position encoding
attention = softmax(content_content + content_position) // get attention matrix of x
z = bmm(v, attention.permute(0, 1, 3, 2))
z = z.view(b, C, width, height) // get output z
bmm: batch matrix multiplication
```

Algorithm 3: Pseudo-code of MHSA in a PyTorch-like style.

essential, assuming that without this step, the intrusion detection model will likely experience model collapse; i.e., the online network and the target network can similarize the representation of all network traffic images in both networks by reducing the weights and biases to zero, which will result in that the intrusion detection model does not learn any valid feature information. From the perspective of information bottleneck, the neural network is gradually losing nonessential information for the classification task (e.g., the colour or orientation of the objects in the images after the data augmentation mentioned above, i.e., the data perturbation caused by the data augmentation); while adding the feature projection, the feature space before taking the feature projection will retain more useful information for the classification task, so that the weights and biases in the online network and target network can avoid convergence to zero and thus learn more useful feature information.

3.1.4. Contrastive Learning. After feature projection, the network traffic is projected to the low-dimensional vector space to get z_θ and z'_ξ; at this time, the network traffic after the online network also needs to go through the prediction q_θ and then get the prediction vector $q_\theta(z_\theta)$; the network traffic after the target network does not need this process; the composition of q_θ is similar to g_θ and g_ξ and both are multilayer perceptrons consisting of two hidden layers and a batch normalization layer; therefore, the output of feature prediction can be represented as follows:

$$q_\theta(z_\theta) = W_\theta^{(4)}\left(\sigma\left(BN\left(W_\theta^{(3)} z_\theta + b_\theta^{(3)}\right)\right)\right) + b_\theta^{(4)}, \tag{13}$$

where W and b are the weights and biases of the fully connected layer, BN is the batch normalization layer, and σ is the activation function of ReLU. Therefore, for the prediction vector $q_\theta(z_\theta)$ obtained from the online network, we can interpret z'_ξ as the ground truth about the network traffic generated by the target network. Given $q_\theta(z_\theta)$ and z'_ξ, we should apply l_2-normalization for them to obtain $\overline{q_\theta}(z_\theta)$ and $\overline{z'_\xi}$. In the original paper, we can constrain the similarity between $\overline{q_\theta}(z_\theta)$ and $\overline{z'_\xi}$ as the loss function to train the online network and the target network and thus, the loss function can be obtained from formula (13) as follows:

$$L_{\theta,\xi} \triangleq \left\|q_\theta(z_\theta) - z'_\xi\right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi\rangle}{\left\|q_\theta(z_\theta)\right\|_2 \left\|z'_\xi\right\|_2}. \tag{14}$$

After applying l_2-normalization for $q_\theta(z_\theta)$ and z'_ξ, although the value of each dimension in vectors $\overline{q_\theta}(z_\theta)$ and $\overline{z'_\xi}$ is less than 1, since $\overline{q_\theta}(z_\theta)$ and $\overline{z'_\xi}$ are high-dimensional vectors, performing vector multiplication and then summation will produce relatively large gradient values which will lead to making the training process become more unstable just as shown in Figure 8(a), if the mean square error is used as the loss function; i.e.,

$$iL_{\theta,\xi} \triangleq \frac{1}{2} * \left(\overline{q_\theta}(z_\theta) - \overline{z'_\xi}\right)^2. \tag{15}$$

The mean square error can be avoided by subtracting the vectors and then summing the squares (because the vectors $\overline{q_\theta}(z_\theta)$ and $\overline{z'_\xi}$ are vectors obtained from two different augmented views of the same network traffic after feature extraction and feature projection, the difference between them is small, and subtracting the vectors and then summing them will result in a smaller loss value), which can make the model more stable during the training process. Therefore, we can replace $L_{\theta,\xi}$ in BYOL with $iL_{\theta,\xi}$, so that the f_θ can extract the effective feature information and the training process is more stable. Then, the loss obtained from equation (3) is updated by the gradient descent method for the online network weights, and the weights of the target network are updated by EMA until the two networks converge (the reason for using EMA to update the weights of the target network is that it can effectively retain the weights of the online network and the target network different, thus avoiding model collapse). At this point, discarding the data augmentation operation t, the g_θ, and the q_θ in the online network, we obtain the f_θ to represent the network traffic features and use it as a basis to distinguish the categories of network traffic.

3.2. Detection Procedures for the Improved BYOL Model. It can be seen from Section 3.1 that the specific steps to the improved BYOL intrusion detection model are as follows:

Step 1: preprocessing for the UNSW-NB15 intrusion detection dataset, mainly including character-based data one-hot encoding processing and data normalization processing.

Step 2: construction of the intrusion detection model based on improved BYOL training as follows:

(1) Initialize model parameters and determine the structure of the network model
(2) Apply two separate data augmentation processes for the UNSW-NB15 dataset
(3) Put two sets of augmented data into the online network and the target network, respectively, and adjust the error of the training process according to the loss obtained from equation (3) until both the online network and the target network models reach convergence
(4) Take out the feature extraction encoder f_θ, get the feature representation of the network traffic, and save the f_θ weights

Step 3: the improved BYOL intrusion detection model is tested by inputting the preprocessed test dataset to the f_θ to obtain the feature representation of each data item in this dataset and then inputting the feature representation to the classifier (one linear layer), which in turn obtains the classification result of each data item.

The overall flow chart of the improved BYOL intrusion detection model is shown in Figure 5.

4. Datasets and Preprocessing

To verify the powerful detection and generalization capabilities of the improved BYOL intrusion detection model, this paper conducts experiments not only on the old intrusion detection datasets like KDD CUP99 [37] and NSL-KDD [38] but also on the new intrusion detection datasets such as UNSW-NB15 [39], CIC IDS2017 [40], and CIDDS_001 [41]. Since UNSW-NB15 contains more comprehensive types of attacks and rich feature information, this paper obtains the feature representation of network traffic by applying the improved BYOL intrusion detection model to UNSW-NB15 and thus performs transfer learning on the KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001 datasets to validate the proposed model's powerful generalization ability. The operating environment of the experimental part is shown in Table 3.

4.1. Datasets Description. The KDD CUP99 dataset was derived from an intrusion detection evaluation project conducted by the U.S. Department of Defense Advanced Planning Agency (DARPA) at MIT Lincoln Laboratory in 1998. Many simulated attacks are added to the network. Network traffic was labeled as normal or anomalous, and the anomaly types were subdivided into 4 major categories (Probe, DoS, U2R, and R2L) for a total of 39 attack types, of which 22 attack types appeared in the training set and another 17 unknown attack types appeared in the test set. Every network traffic sample contains 41 attributes and a category label. Table 4 describes the KDD CUP99 dataset in detail.

NSL-KDD dataset is an improvement of KDD CUP99, which solves the problems of data redundancy and duplicate data in KDD CUP99. The NSL-KDD dataset contains 4 anomaly types, namely, Dos, Probe, U2R, and R2L, and each intrusion record has 42 dimensional features, of which 42 features are composed of 9 basic TCP connection features, 13 content features of TCP connections, 9 time-based network traffic statistics features, 10 host-based network traffic statistics features, and a category label. Table 4 details the NSL-KDD dataset.

The UNSW-NB15 dataset was created by the Australian Centre for Cyber Security (ACCS) in 2015. It is a comprehensive network attack traffic dataset. The dataset contains data with two labels, 1 for the attack category and 0 for the normal category. Specifically, it has a total of 9 different categories of attack, and the data flow is described by a total of 49 features, 47 of which are attack-related features, with a specific attack category label and an attack and normal category label. The detailed description of the UNSW-NB15 dataset is shown in Table 5.

The CIC IDS2017 dataset is a network traffic dataset collected and made public by the Canadian Institute for Cyber Security in 2017, which contains five days of network traffic data collected from Monday to Friday, including normal traffic and anomalous traffic due to common attacks. This paper uses Wednesday-workingHours.csv as the intrusion detection dataset, and Table 6 describes the CIC IDS2017 dataset in detail.

The CIDDS_001 dataset is a tagged traffic-based dataset for evaluating anomaly-based intrusion detection systems. The dataset consists of three log files (attack logs, client configuration, and client logs) and traffic data from two servers, each consisting of four 4-week periods of captured traffic data. Table 7 details the CIDDS_001 dataset.

4.2. Data Preprocessing. For the intrusion detection datasets, the original datasets cannot be directly inputted into the network model for intrusion detection, because the input dataset must conform to the input format of the convolutional neural network. Therefore, the experimental datasets need to be preprocessed in advance by the following steps.

4.2.1. One-Hot Encoding. Taking the NSL-KDD dataset as an example, where the element types of three features, protocol, flag, and service, are symbolic features, it needs to be converted into numerical representations; for example, assuming that protocol contains three categories, UDP, TCP, and ICMP, the protocol categories can be processed as [1, 0, 0], [0, 1, 0], [0, 0, 1], and [0, 0, 1] and other symbolic features are processed similarly. The final length of each network traffic data is 121 dimensions.

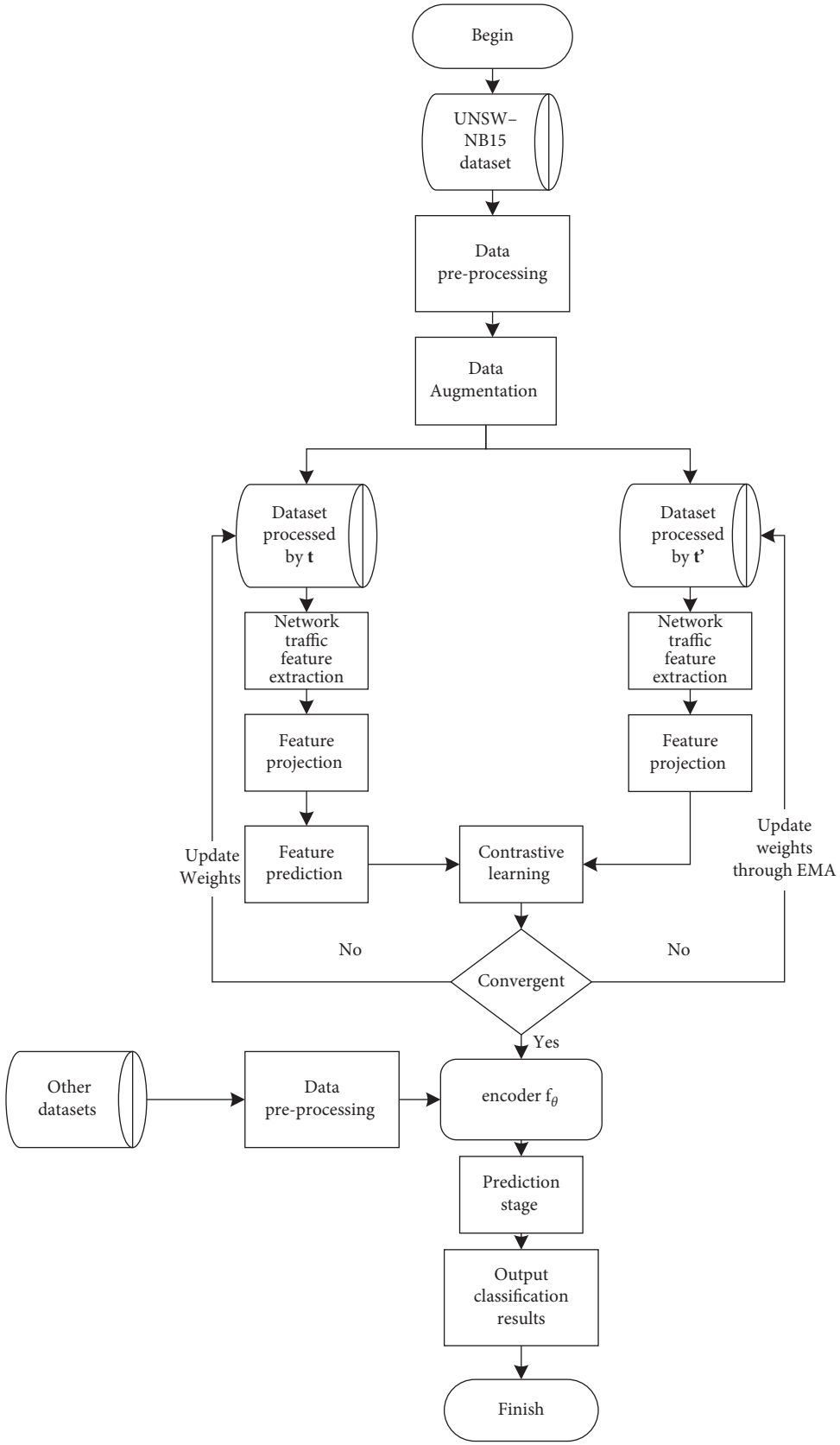


Figure 5: Intrusion detection model flowchart.

Table 3: Experimental environment.

Experimental environment	Environment configuration
Operating system	Windows 10
Programming language	Python 3.7
Deep learning framework	PyTorch 1.7
Machine learning library	Scikit-Learn 0.23.2
Graphics card	RTX 2070

Table 4: Training and testing records of KDD CUP99 and NSL-KDD.

Attack category	KDD CUP99		NSL-KDD	
	Training	Testing	Training	Testing
Normal	97277	60592	67345	9711
Probe	4107	4166	11655	2421
Dos	391438	229825	45926	7458
U2R	52	228	52	200
R2L	1126	16189	995	2754
Total	494000	311000	125973	22544

Table 5: Training and testing records of UNSW-NB15.

Attack category	Training	Testing
Normal	56000	37000
Fuzzers	18184	6062
Analysis	2000	677
Backdoors	1746	583
Dos	12264	4089
Exploits	33393	11132
Generic	40000	18871
Reconnaissance	10491	3496
Shellcode	1133	378
Worms	130	44
Total	175341	82332

Table 6: Training records of CIC IDS2017.

Attack category	Training
Benign	440031
Dos Hulk	231073
Dos GoldenEye	10293
Dos slowloris	5796
Dos Slowhttptest	5499
Heartbleed	11
Total	692703

4.2.2. Data Normalization. In order to cancel the magnitude, make the gradient always in the direction of the minimum as well as accelerate the convergence; it is necessary to do the normalization of the data after feature mapping.

$$x_{norm} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{16}$$

where x is the original data, $x_{\min}$ is the minimum value among the same features, $x_{\max}$ is the maximum value among the same features, and x_{norm} is the result of using maximum-minimum normalization.

Table 7: Training and testing records of CIDDS_001.

Attack category	Training	Testing
Normal	130000	4240
Attacker	10000	2260
Suspicious	430000	7911
Unknown	70000	7932
Victim	8000	907
Total	648000	23241

4.3. Evaluation Metrics. Since the network intrusion detection data are complex, the evaluation of the model cannot be based on the accuracy rate alone as the only evaluation criterion, so this paper will use the accuracy rate (ACC), precision rate (precision), detection rate (DR), and F1 score as the evaluation indexes of intrusion detection and verify the accuracy and stability of the model through a comprehensive comparison of the above indexes. The definition of these metrics can be given as follows:

$$\begin{aligned} \text{ACC} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{DR} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{F1 score} &= \frac{2 * \text{precision} * \text{DR}}{\text{precision} + \text{DR}}, \end{aligned} \tag{17}$$

where true positive (TP) is the number of connection records correctly classified to the normal class, true negative (TN) is the number of connection records correctly classified to the attack class, false positive (FP) is the number of normal connection records wrongly classified to the attack connection records, and false negative (FN) is the number of attack connection records wrongly classified to the normal connection record.

5. Results and Discussion

There are four groups of experiments in this paper, and the experiments mainly verify four aspects:

(1) The correctness of the improved encoder architecture in BYOL proposed in Section 3.1.2 and the effect of hyperparameter d on the anomaly detection of UNSW-NB15 in Section 3.1.3 as well as the stability of the model training after optimizing the BYOL loss function in Section 3.1.4 by performing linear evaluation on UNSW-NB15 are verified.

(2) By conducting linear evaluation on UNSW-NB15, the extracted robust network traffic feature representations using the improved BYOL intrusion detection model and the effectiveness of the data augmentation operation proposed in Section 3.1.1 are verified.

(3) To compare traditional deep learning models like DNN, CNN, and RNN with our model, we

conducted experiments on KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001, so that we can verify the feasibility of using the feature representations extracted by our model to discriminate network traffic.

(4) Experiments are conducted on KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001 datasets for transfer learning and they are compared with other state-of-the-art models taken from the recent literature on network intrusion detection to verify that the extracted feature representation using our model has a strong generalization capability.

5.1. Effectiveness of the Improved BYOL Self-Supervised Learning. Firstly, we verify the correctness of the improved encoder architecture in BYOL proposed in Section 3.1.2 and the impact of hyperparameter d on the accuracy of UNSW-NB15 anomaly detection in Section 3.1.3 as well as the stability of model training after optimizing the loss function of BYOL proposed in Section 3.1.4. Figure 6 shows the impact of different encoder architectures on UNSW-NB15 anomaly detection, and it can be seen that when the feature extraction encoder architecture is BoTNet, accuracy, precision, and other performance indicators of UNSW-NB15 anomaly detection are the highest and the training process is relatively more stable, further verifying that the introduction of attention mechanism in Section 3.1.2 can effectively suppress the features that contribute less to the classification in the intrusion detection data and increase the features that contribute more to the classification, thus increasing the recognition rate of network anomalous traffic. Furthermore, it also verifies the correctness of choosing BoTNet for the encoder architecture in the improved BYOL. The impact of performance metrics on UNSW-NB15 anomaly detection when $d \in \{64, 128, 256, 512, 600\}$is shown in Figure 7, and it can be seen that when d is taken as 512, the accuracy, precision, and other performance indexes of UNSW-NB15 anomaly detection are the highest, so the BoTNet model with a d value of 512 is used for the feature extraction encoder architecture in the subsequent experiments. The loss curve of different loss functions and their performance metrics are presented in Figure 8. We can see that the loss of the training process becomes smoother after using the optimized loss function proposed in this paper and the model converges faster than the loss function proposed in the original BYOL paper from Figure 8(a), and from Figure 8(b), we can see that the accuracy, precision, and other performance indexes obtained by the model are similar to the loss function proposed in the original BYOL paper for UNSW-NB15 anomaly detection. It can be verified that the model training is more stable and converges faster after the optimized BYOL loss function proposed in this paper.

5.2. Linear Evaluation. After training the UNSW-NB15 dataset using the modified BYOL to obtain the feature representation of network traffic, in order to verify the effectiveness of this feature representation, we use linear evaluation (freezing the weights of the trained BoTNet to train only the last linear layer for network traffic classification) on UNSW-NB15. In the meantime, we also use supervised learning to train BoTNet and some state-of-the-art models for comparison experiments. The experimental results are shown in Table 8 and Figure 9, where "—" represents that the results of this indicator are not given in the paper. As can be seen from Table 8, in most cases, our model achieves better detection performance than other state-of-the-art models. At the same time, the results of various metrics obtained by supervised BoTNet and linear evaluation are similar and the accuracy of UNSW-NB15 anomaly detection achieves 89.97% using only one linear layer, which is just 4.08% worse than the supervised BoTNet with the accuracy of 94.05% and 17.59% greater than SADE-ELM. In the aspect of accuracy, our model is 3.72% and 19.78% greater than VLSTM and SADE-ELM, respectively, and 4.16% and 5.44% worse than MFFSEM and TSIDS, respectively. In terms of detection rate, our model is only 2.54% worse than the highest VLSTM, 0.11% greater than TSIDS, and 14.82% and 7.84% greater than MFFSEM and SADE-ELM, respectively. F1 score is a comprehensive evaluation index of accuracy and detection rate which can better reflect the classification ability of the model. As for F1 score, our model is even 14.7% greater than the SADE-ELM model, 1.71% and 5.77% greater than the VLSTM and MFFSEM, respectively, and only 2.91% worse than the highest BoTNet. The ROC curve is not only easy to understand but also a more stable indicator that can reflect the quality of the model when faced with imbalance of the number of positive and negative samples. Therefore, the ROC curve can reduce the interference brought by different test sets and measure the performance of the model itself more objectively. Figure 9 depicts the ROC curves, and it can be seen from the figure that the AUC of the self-supervised BoTNet is 0.94, which is only 0.6 times higher than that of our model, further verifying the effectiveness of the network traffic feature representation extracted by our model, which can fully and effectively distinguish the categories of network traffic. Combining Table 8 and Figure 9, we can draw a conclusion that our model can effectively distinguish network anomalous traffic and prove that the data augmentation operation *random_shuffle* proposed in this paper enables the improved BYOL intrusion detection model to learn the feature representation of network traffic invariance and then classify network traffic correctly.

5.3. Comparison Experiments of Traditional Deep Learning Algorithms. To verify the feasibility of using improved BYOL to train the feature representations extracted from the UNSW-NB15 dataset with differentiated network traffic, we conducted comparative experiments on KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001 datasets using the traditional deep learning models like DNN, CNN, and RNN and our model for transfer learning. Here, DNN consists of two hidden layers with 128 and 64 neurons, respectively, CNN consists of three convolutional layers with 32, 64, and

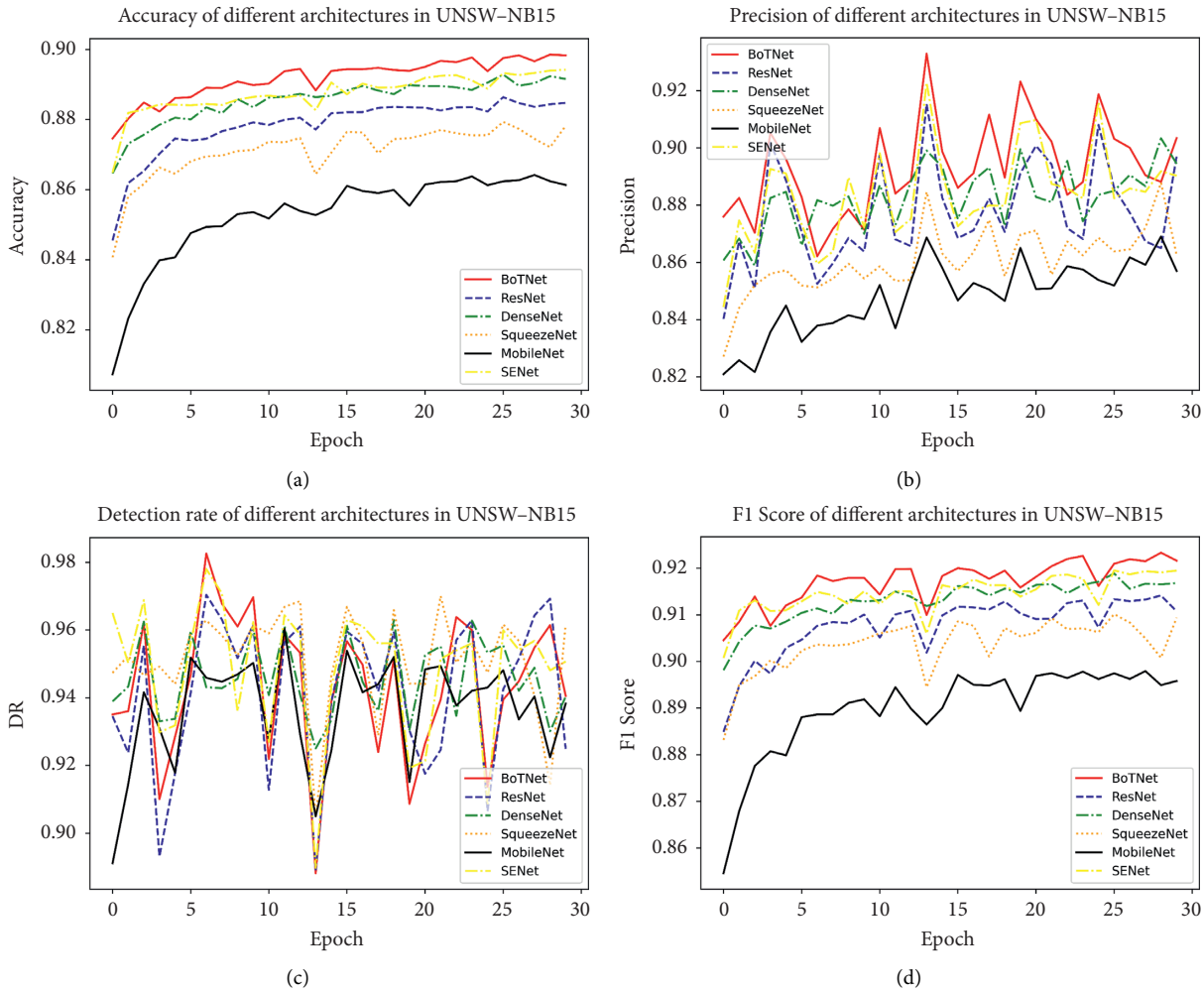


Figure 6: Impact of different encoder architectures on UNSW-NB15 anomaly detection performance metrics: (a) accuracy on different encoders, (b) precision on different encoders, (c) detection rate on different encoders, and (d) F1 score on different encoders.

128 3×3 convolutional kernels, respectively, and RNN consists of a layer of 70 neurons of LSTM. The experimental results are given in Table 9.

Tables 9–12 describe the accuracy, precision, detection rate, and F1 score in detail when DNN, CNN, and RNN and the transfer learning of our model are used for anomaly detection on KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001. Figure 10 shows more visually the differences in the performance metrics of DNN, CNN, and RNN and the transfer learning of our model for anomaly detection on each dataset. From Table 9 and Figure 10(a), it can be seen that all the deep learning models achieve better performance in most cases. Besides, all the metrics can reach above 99%, because the KDD CUP99 dataset is simpler and there is a large amount of data redundancy. As can be seen from Table 10 and Figure 10(b), since the NSL-KDD dataset solves the data redundancy problem existing in the KDD CUP99 dataset, the performance indexes of each model are reduced on the NSL-KDD dataset and the results obtained by our model are slightly worse than of the other three models. This is mainly because our model classifies more normal traffic as abnormal traffic, which leads to the poor performance of the model. From Table 11 and Figure 10(c), it can be seen that DNN performs better on the CIC IDS2017 dataset, with all performance indicators reaching above 99% and CNN, RNN, and our model perform relatively poor, but the performance metrics can still reach more than 95%. From Table 12 and Figure 10(d), it can be seen that the results obtained by DNN, RNN, and CNN are better and all performance indexes can reach more than 99%, while the performance index obtained by our model can reach more than 98%, which can still effectively distinguish the CIDDS_001 dataset from abnormal traffic. In summary, due to the simplicity of the dataset and some problems in the dataset itself, traditional deep learning algorithms such as DNN, CNN, and RNN and our model can achieve good anomaly detection results on the KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001 datasets as well as validate that using the improved BYOL to train the UNSW-NB15 dataset with the extracted feature representations is fully feasible to distinguish network traffic.

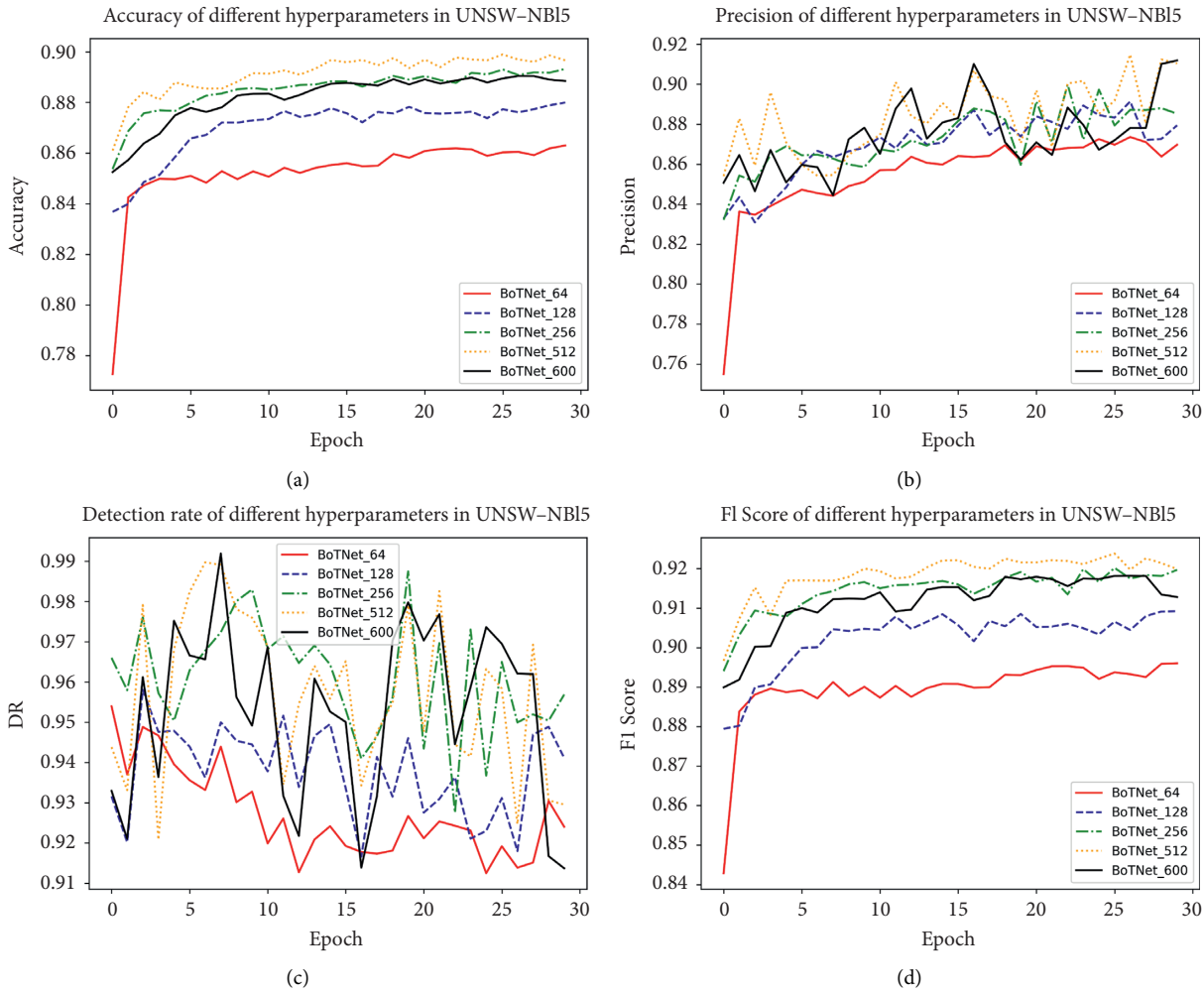
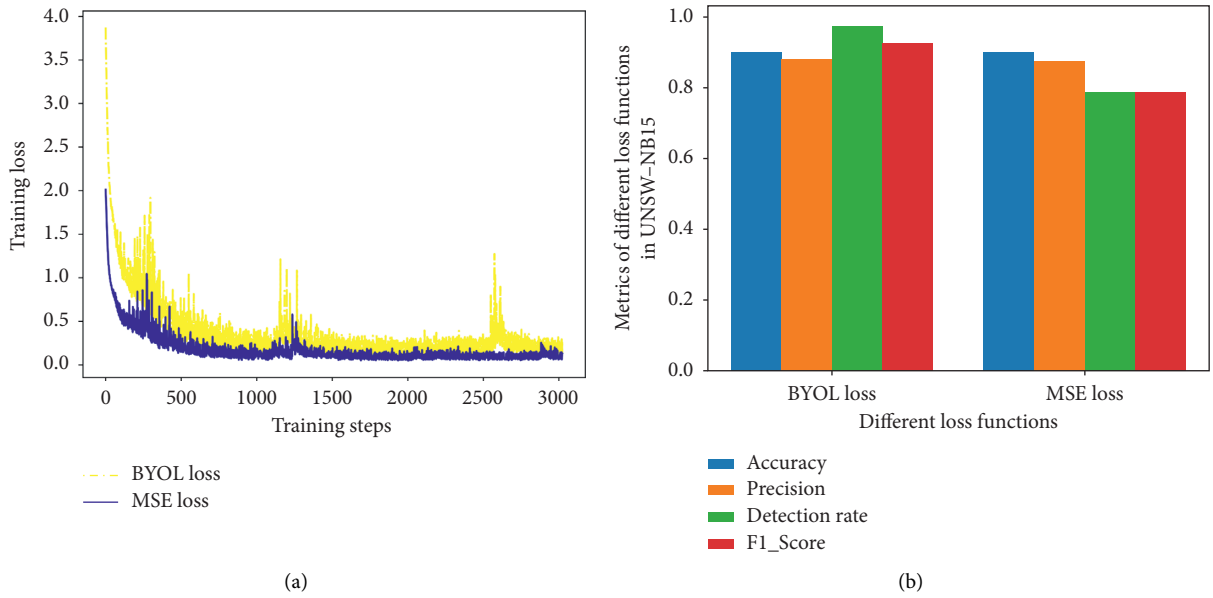


Figure 7: Impact of different values of d on UNSW-NB15 anomaly detection performance metrics: (a) accuracy on different values of d, (b) precision on different values of d, (c) detection rate on different values of d, and (d) F1 score on different values of d.

5.4. Transfer Learning. To verify the feature representations of network traffic obtained by training the UNSW-NB15 dataset using improved BYOL that has strong generalization capability, we perform transfer learning on the KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001 intrusion detection datasets as well as perform comparison experiments with the state-of-the-art models on each dataset. We evaluate our network traffic feature representations on the KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001 datasets to make sure whether the feature representations learned on UNSW-NB15 are generic and thus useful across intrusion detection domains, or if they are UNSW-NB15-specific. The experimental results are shown in Tables 13–16 and Figure 11, where "—" represents that the metric is not given in the paper results.

As can be seen from Table 13 and Figure 11(a), the performance metrics obtained by transfer learning in the UNSW-NB15 dataset for network traffic feature representation are fully comparable to supervised BoTNet on the KDD CUP99 intrusion detection dataset and the difference between them is only a fraction of a percentage point, which is due to the powerful feature extraction capability of the improved BYOL intrusion detection model. Compared with other state-of-the-art models on the KDD CUP99 dataset, the results obtained from transfer learning are even 1%–6% better than those of the supervised learning SADE-ELM model in terms of performance metrics, with only 0.67% difference in accuracy compared to the DT-EnSVM model. From Table 14 and Figure 11(d), it can be seen that compared with other state-of-the-art models on the CIDDS_001 dataset, the results obtained from transfer learning differ from the MLIDS model with the highest accuracy by only 2.37% and are 4.97% higher than those of the SADE-ELM model with the lowest accuracy. In terms of detection rate, the result of our model obtained from transfer learning is 97.82%, which is 2.04% lower than that of the supervised learning BoTNet and MLIDS with the highest detection rate, 0.99% and 0.51% lower than that of DBN and RF, and 6.45% higher than that of SADE-ELM, respectively, which indicates that our model can detect the intrusion data more comprehensively and with fewer faults. As can be seen from Table 15 and Figure 11(b), the result of accuracy obtained from the transfer learning on the NSL-KDD dataset is slightly lower than that of the supervised learning BoTNet



(a) (b)

Figure 8: Different loss functions on UNSW-NB15 anomaly detection performance metrics. (a) Loss curve of different loss functions. (b) Metrics of different loss functions.

Table 8: Experimental results of different models in UNSW-NB15 anomaly detection.

Model	Accuracy	Precision	DR	F1 score
VLSTM [42]	—	0.8600	**0.9780**	0.9070
MFFSEM [43]	0.8885	0.9388	0.8044	0.8664
TSIDS [44]	—	0.9516	0.9515	0.9515
SADE-ELM [45]	0.7238	0.6994	0.8742	0.7771
BoTNet (supervised)	**0.9405**	**0.9618**	0.9447	**0.9532**
Linear evaluation	0.8997	0.8972	0.9526	0.9241

The bold values are the maximum value among the metric indicator.

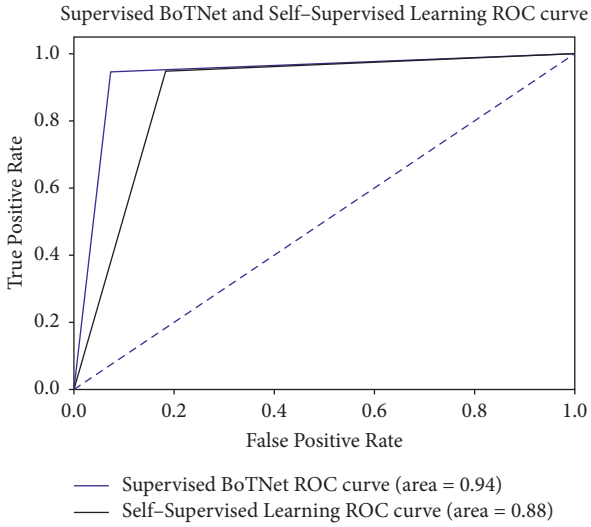


Figure 9: ROC curve of UNSW-NB15.

because of the increase in the complexity of the dataset, which is nearly 7% lower. However, compared with other state-of-the-art models on the NSL-KDD dataset, the transfer learning results are still better than those of other models in all metrics in most cases, even higher than the SADE-ELM model in the accuracy by nearly 16%, but slightly lower than its precision by 3.5%. In terms of F1 score, our model achieves 0.9227, which is 7.07% lower than that of

Table 9: Experimental results of different models in KDD CUP99 anomaly detection.

Model	Accuracy	Precision	DR	F1 score
DNN	**0.9995**	0.9997	**0.9997**	**0.9997**
CNN	0.9994	**0.9998**	0.9994	0.9996
RNN	0.9993	0.9995	0.9995	0.9995
Our model	0.9922	0.9983	0.9919	0.9951

Table 10: Experimental results of different models in NSL-KDD anomaly detection.

Model	Accuracy	Precision	DR	F1 score
DNN	**0.9904**	0.9867	**0.9934**	**0.9900**
CNN	0.9488	**0.9967**	0.8958	0.9436
RNN	0.9901	0.9874	0.9919	0.9896
Our model	0.9299	0.9324	0.9199	0.9261

Table 11: Experimental results of different models in CIC IDS2017 anomaly detection.

Model	Accuracy	Precision	DR	F1 score
DNN	**0.9968**	**0.9917**	**0.9995**	**0.9956**
CNN	0.9900	0.9821	0.9904	0.9863
RNN	0.9847	0.9713	0.9869	0.9791
Our model	0.9670	0.9500	0.9596	0.9548

Table 12: Experimental results of different models in CIDDS_001 anomaly detection.

Model	Accuracy	Precision	DR	F1 score
DNN	**0.9975**	**0.9981**	**0.9988**	**0.9984**
CNN	0.9970	0.9974	**0.9988**	0.9981
RNN	0.9974	**0.9981**	0.9986	**0.9984**
Our model	0.9813	0.9816	0.9953	0.9884

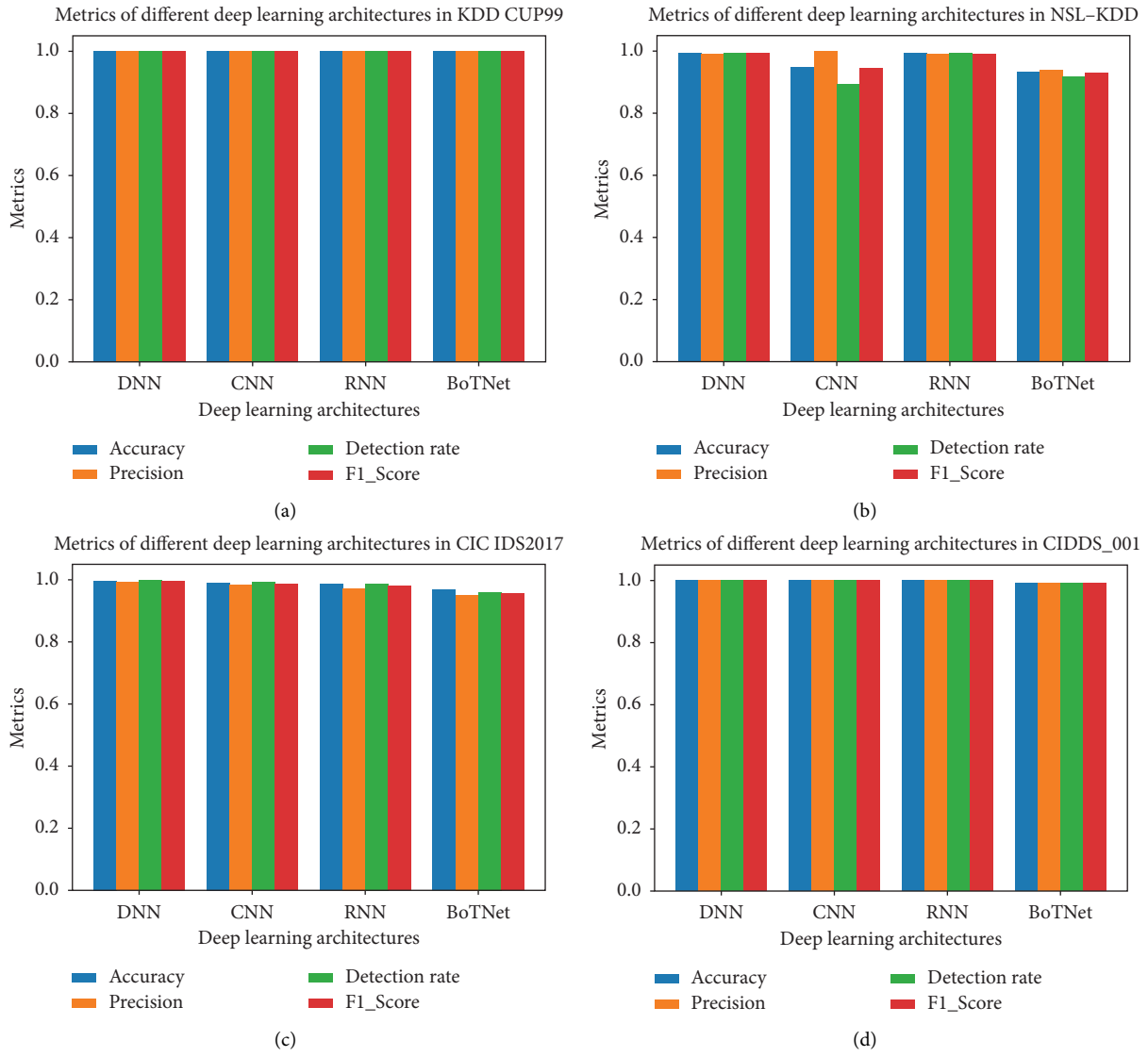


Figure 10: Anomaly detection performance metrics for each algorithm in each dataset: (a) KDD CUP99, (b) NSL-KDD, (c) CIC IDS2017, and (d) CIDDS_001.

the supervised learning BoTNet and 17.27%, 11.49%, 7.4%, and 8.1% higher than of SADE-ELM, LCVAE, FL-NIDS, and IGAN-IDS, respectively, which indicates that the performance of our model is more comprehensive without serious drawbacks. We compare the performance of our model against state-of-the-art models in detail on CIC IDS2017, as shown in Table 16 and Figure 11(c). The results obtained from transfer learning are slightly lower than those of other models in terms of accuracy, precision, detection, and F1 score. In terms of F1 score, our model is 4.29%, 3.26%, and 2.95% worse than IGAN-IDS, DBN, and LSTM-RNN, respectively. And, in terms of precision, our model is 4.72% and 4.6% worse than DBN and LSTM-RNN, respectively. Our model is 3.75%, 1.89%, and 0.53% worse than NB-SVM, DBN, and LSTM-RNN in terms of detection rate, respectively. Our model is 3.09%, 2.22%, and 1.17% worse than IGAN-IDS, NB-SVM, and DBN in terms of accuracy, which indicates that the model is slightly weak in feature

Table 13: Experimental results of different models in KDD CUP99 anomaly detection.

Model	Accuracy	Precision	DR	F1 score
SADE-ELM [45]	0.9356	0.9875	0.9310	0.9584
DAE-IDS [20]	0.9653	—	0.9565	—
DT-EnSVM [46]	**0.9992**	—	0.9993	—
ImCNN [47]	0.9842	—	0.9872	0.9907
BoTNet (supervised)	0.9990	**0.9992**	**0.9996**	**0.9994**
Our model	0.9925	0.9974	0.9933	0.9953

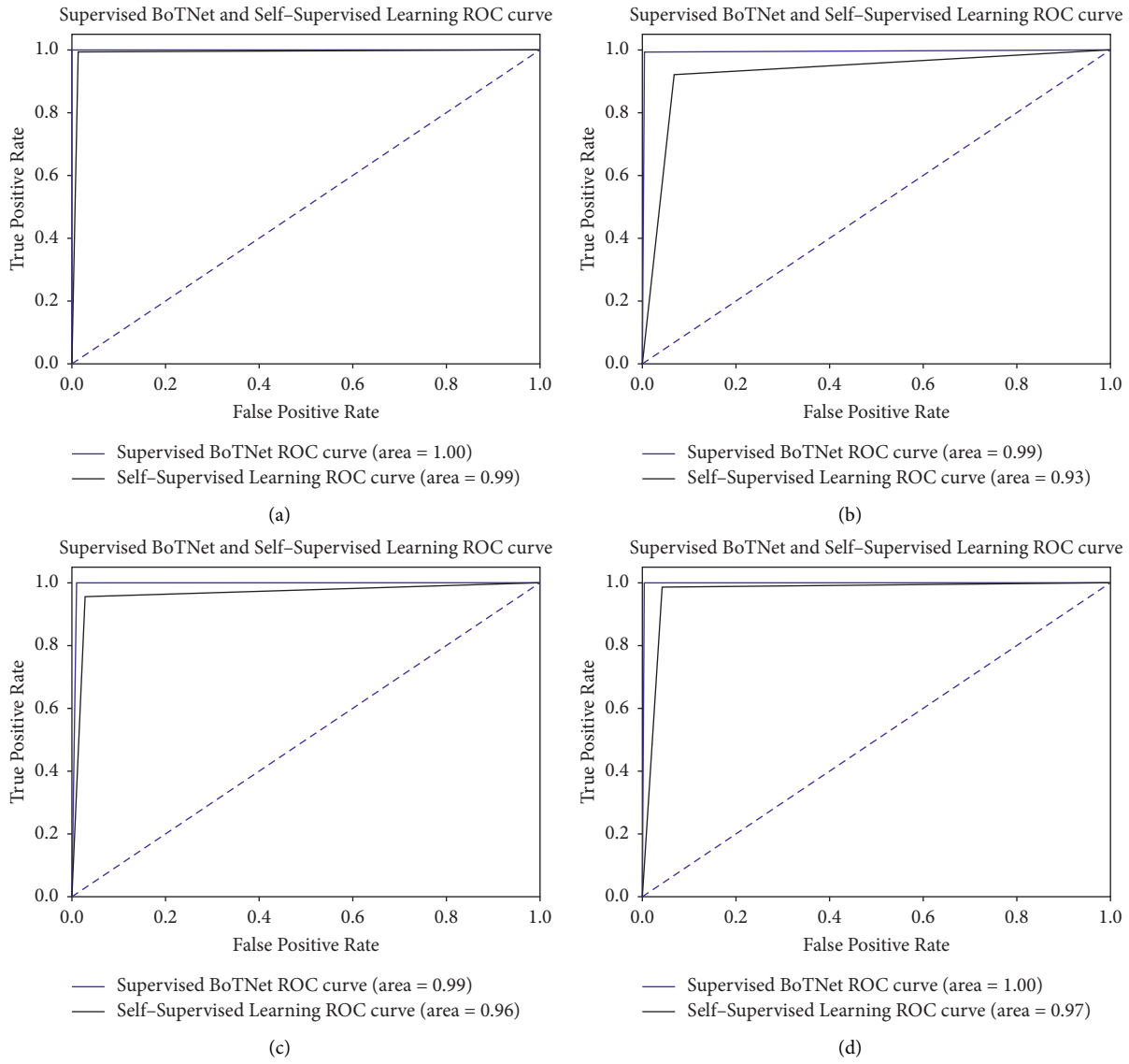


Figure 11: ROC curve of each dataset: (a) KDD CUP99, (b) NSL-KDD, (c) CIC IDS2017, and (d) CIDDS_001.

generalization on CIC IDS2017 dataset and the model generalization capability can continue to be improved. In general, each algorithm is able to achieve a wonderful score for each performance index in the intrusion detection of KDD CUP99, NSL-KDD, CIC IDS2017, and CIDDS_001 datasets, which indicates that all types of algorithms can effectively detect network intrusion data, but the results obtained from transfer learning are significantly better than those of other models in most cases, which fully proves that the network traffic feature representation extracted by the improved BYOL has powerful network traffic discrimination ability.

To better visualize the sample distribution of the intrusion detection dataset after processing with the improved BYOL intrusion detection model, we randomly select 5,000 records from the KDD CUP99, NSL-KDD, CIC

Table 14: Experimental results of different models in CIDDS_001 anomaly detection.

Model	Accuracy	Precision	DR	F1 score
SADE-ELM [45]	0.9258	0.9951	0.9137	0.9527
MLIDS [53]	**0.9992**	0.9887	**0.9986**	0.9900
DBN [54]	0.9894	0.9892	0.9881	0.9813
RF [55]	—	0.9814	0.9833	0.9823
BoTNet (supervised)	0.9979	**0.9988**	**0.9986**	**0.9987**
Our model	0.9755	0.9910	0.9782	0.9846

Table 15: Experimental results of different models in NSL-KDD anomaly detection.

Model	Accuracy	Precision	DR	F1 score
SADE-ELM [45]	0.7664	0.9598	0.6155	0.7500
LCVAE [48]	0.8551	0.9761	0.6890	0.8078
FL-NIDS [49]	0.8568	—	0.8614	0.8487
IGAN-IDS [50]	0.8445	—	—	0.8417
BoTNet (supervised)	**0.9937**	**0.9948**	**0.9920**	**0.9934**
Our model	0.9267	0.9248	0.9206	0.9227

Table 16: Experimental results of different models in CIC IDS2017 anomaly detection.

Model	Accuracy	Precision	DR	F1 score
IGAN-IDS [50]	**0.9979**	—	—	**0.9979**
NB-SVM [51]	0.9892	—	0.9951	—
DBN [52]	0.9787	**0.9996**	0.9765	0.9876
LSTM-RNN [52]	0.9662	0.9984	0.9629	0.9845
BoTNet (supervised)	0.9884	0.9705	**0.9987**	0.9844
Our model	0.9670	0.9524	0.9576	0.9550

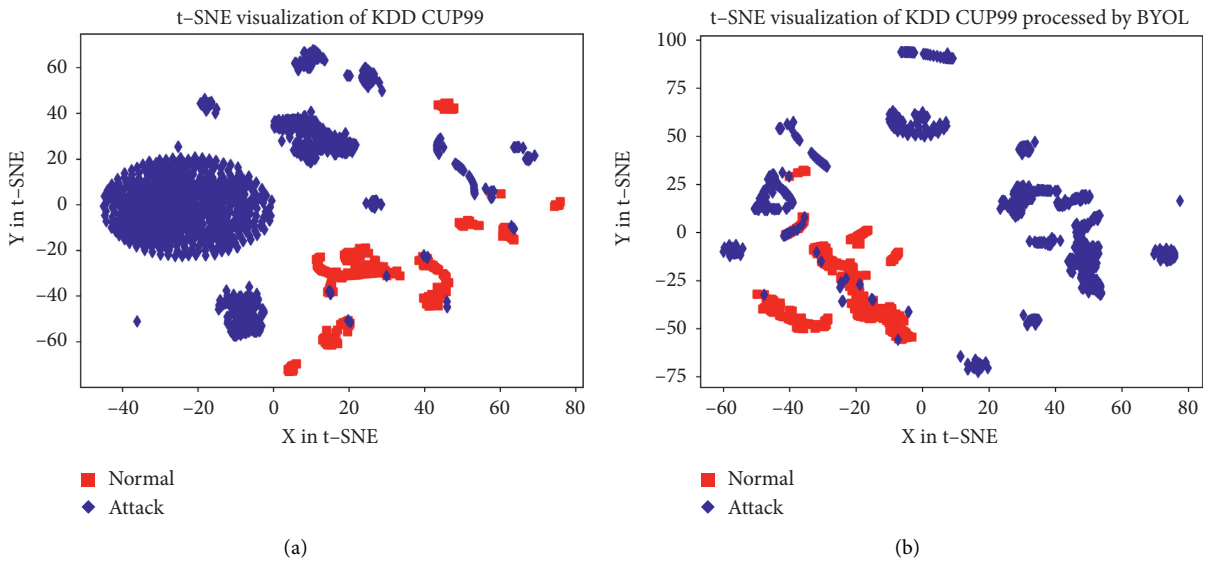


Figure 12: t-SNE visualization of (a) unprocessed KDD CUP99 and (b) processed KDD CUP99.

IDS2017, and CIDDS_001 datasets without processing and after processing, respectively. Then, we used the t-SNE algorithm [56] to reduce the dimension of these records and visualized them. Figures 12(a), 13(a), 14(a), and 15(a) show the visualized images of 10,000 unprocessed records of KDD CUP99, NSL-KDD, CIC IDS 2017, and CIDDS_001 datasets, respectively. As can be seen from the figures, the data in all datasets are linearly indistinguishable and the NSL-KDD and CIDDS_001 datasets are significantly more complex and difficult to distinguish than the

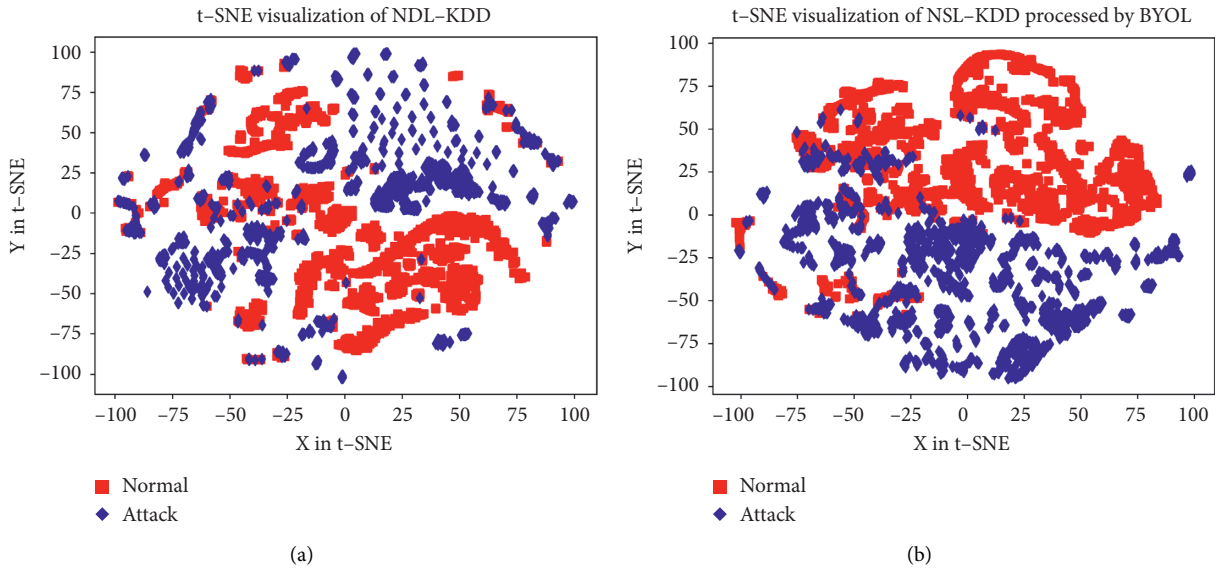


Figure 13: t-SNE visualization of (a) unprocessed NSL-KDD and (b) processed NSL-KDD.

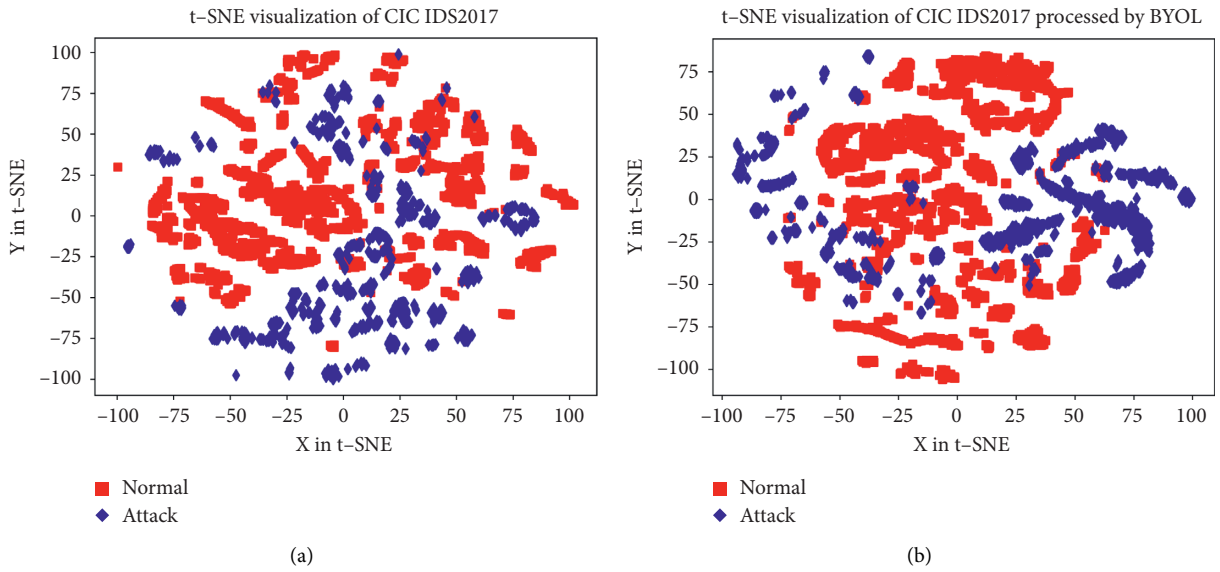


Figure 14: t-SNE visualization of (a) unprocessed CIC IDS2017 and (b) processed CIC IDS2017.

KDD CUP99 and CIC IDS 2017 datasets, just as reflected by the experimental results obtained from transfer learning. While Figures 12(b), 13(b), 14(b), and 15(b) are the visualized images of 5,000 records after processing with the improved BYOL intrusion detection model, comparing the visualized images of 5,000 processed records with those of unprocessed records shows that the samples of different categories show an aggregation trend in the feature space and can be almost separated linearly, which is sufficient to show that the features obtained in the UNSW-NB15 dataset for network traffic representation has a strong generalization ability and can effectively distinguish various types of network anomaly traffic.

6. Conclusions and Future Work

In this paper, we propose a new data augmentation strategy for intrusion detection data and an intrusion detection model based on label-free self-supervised learning. Using the improved BYOL self-supervised learning model to extract network traffic feature representations, in order to avoid the poor generalization ability of the model due to the fusion of too many invalid features, we introduce a multihead attention mechanism to suppress the features in the intrusion detection data that contribute less to the classification and increase the features that contribute more to the classification.

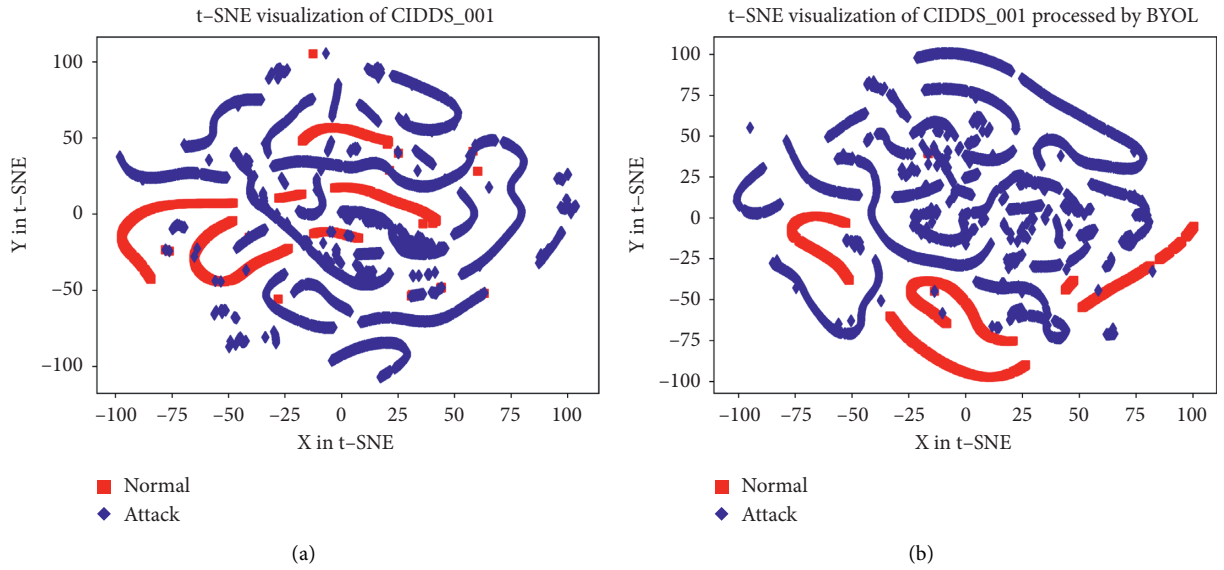


Figure 15: t-SNE visualization of (a) unprocessed CIDDS_001 and (b) processed CIDDS_001.

Training and testing on the intrusion detection benchmark datasets KDD CUP99, NSL-KDD, UNSW-NB15, CIC IDS2017, and CIDDS_001 show that the proposed model has a strong ability to identify network traffic and a better generalization ability. In addition, the proposed model achieved good performance in terms of detection accuracy with 99.25%, 92.67%, 96.70%, and 97.55% in testing datasets on experiments of transfer learning, respectively, which is comparable to the results obtained by supervised learning.

Compared with the state-of-the-art models in recent years, the improved BYOL intrusion detection model achieves superior detection results on intrusion detection datasets. However, there are still some gaps between the improved BYOL intrusion detection model and supervised learning methods when using datasets with complex data distribution, various attack types, and data imbalance. In view of the fact that the Mahalanobis distance is not affected by the magnitude and can take into account the connection between various features while excluding the interference of correlation between features, we will consider to use the Mahalanobis distance to calculate the similarity of the output features of the two networks to reduce the gap between our model and the supervised learning methods. In addition, because the encoder network architecture of feature extraction is slightly complex and has more parameters, compared with traditional deep learning DNN, CNN, and RNN, the training time of our model is a little longer and cannot be detected in real time on large datasets, and we will consider to improve the model neurons and calculation methods to simplify the network structure and improve the efficiency of the model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Science Foundation of China (grant numbers 62062037 and 61763017).

References

[1] J. P. Anderson, "Computer security threat monitoring and surveillance," PA 19034, p. 4, James P. Anderson Company, Washington, DC, USA, 1980.

[2] D. E. Denning and P. G. Neumann, "Requirements and model for IDES a real-time intrusion detection expert system," Technical report,13369043, Computer Science Laboratory, SRI International, Menlo Park, CA, USA, 1985.

[3] B. Rebecca, "An introduction to intrusion detection & assessment," in *Proceedings of the International Conference on Software Architecture(ICSA)*, pp. 89–96, Honolulu, Hawaii (USA), March 1998.

[4] T. Hamed, R. Dara, and S. C. Kremer, "Network intrusion detection system based on recursive feature addition and bigram technique," *Computers & Security*, vol. 73, pp. 137–155, 2018.

[5] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," *ECML PKDD 2018 Workshops*, Springer, Berlin, Germany, pp. 149–158, 2019.

[6] A. J. Obaid, K. A. Alghurabi, S. A. K. Albermany, and S. Sharma, "Improving ExtremeLearning machine accuracy utilizing genetic algorithm for intrusion detection purposes," *Advances in Intelligent Systems and Computing*, Springer, vol. 1254Berlin, Germany, , 2021.

[7] B. Wei, W. Zhang, X. Xia, Y. Zhang, F. Yu, and Z. Zhu, "Efficient feature selection algorithm based on particle swarm optimization with learning memory," *IEEE Access*, vol. 7, Article ID 166066, 2019.

[8] M. Azalmad and Y. Fakir, "Data mining approach for intrusion detection,". Business intelligence. CBI 2021," *Lecture Notes in Business Information Processing*, Springer, vol. 416Berlin, Germany, , 2021.

[9] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: datasets and comparative study," *Computer Networks*, vol. 188, Article ID 107840, 2021.

[10] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, Beijing, China, 6 June 2016.

[11] E. Kabir, J. Hu, H. Wang, and G. Zhuo, "A novel statistical technique for intrusion detection systems," *Future Generation Computer Systems*, vol. 79, pp. 303–318, 2018.

[12] P. Nancy, S. Muthurajkumar, S. Ganapathy, S. V. N. S. Kumar, M. Selvi, and K. Arputharaj, "Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks," *IET Communications*, vol. 14, no. 5, pp. 888–895, 2020.

[13] T. Hurley, J. E. Perdomo, and A. Perez-Pons, "HMM-based intrusion detection system for software defined networking," in *Proceedings of the IEEE International Conference on Machine Learning &Applications(ICMLA)*, Anaheim, CA, USA, 20 Dec. 2016.

[14] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Computing*, vol. 24, no. 22, pp. 17265–17278, 2020.

[15] H. Yang and F. Wang, "Network intrusion detection model based on improved convolutional neural network," *Journal of Computer Applications*, vol. 39, no. 9, pp. 2604–2610, 2019.

[16] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, Article ID 21954, 2017.

[17] A. F. M. Agarap, "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," in *Proceedings of the 10th Int Conf on Machine Learning and Computing*, pp. 26–30, ACM, New York, NY, USA, February 2018.

[18] J. Kim, J. Kim, H. Le Thi 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)*, Jeju, Republic of Korea, 17 Feb. 2016.

[19] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *The Journal of Supercomputing*, vol. 75, no. 9, pp. 5597–5621, 2019.

[20] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Proceedings of the International Conference on Advanced Communications Technology(ICACT)*, Chuncheon, Republic of Korea, 14 Feb. 2018.

[21] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pp. 4–11, ACM, Gold Coast, Australia, December 2014.

[22] L. Xiao, Y. Chen, and C. K. Chan, "Bayesian model averaging of bayesian network classifiers for intrusion detection," in *Proceedings of the International Computer Software and Applications Conference Workshops (COMPSACW)*, pp. 128–133, Vasteras, Sweden, 25 July 2014.

[23] Y. Y. Aung and M. M. Min, "Hybrid intrusion detection system using K-means and K-nearest neighbors algorithms," in *Proceedings of the ACIS International Conference on Computer and Information Science (ICIS)*, pp. 34–38, Singapore, 8 June 2018.

[24] T. Aldwairi, D. Perera, and M. A. Novotny, "An evaluation of the performance of restricted Boltzmann machines as a model for anomaly network intrusion detection," *Computer Networks*, vol. 144, pp. 111–119, 2018.

[25] B. H. Yan and G. D. Han, "Combinatorial intrusion detection model based on deep recurrent neural network and improved SMOTE algorithm," *Chinese Journal of Network and Information Security*, vol. 4, no. 7, pp. 48–59, 2018.

[26] M. Y. Qi, M. Liu, and Y. M. Fu, "Research on PCA-based SVM network intrusion detection," *Information Network Security*, vol. 15, no. 2, pp. 15–18, 2015.

[27] G. Osada, K. Omote, and T. Nishide, "Network intrusion detection based on semi-supervised variational auto-encoder," *Computer Security - ESORICS 2017*, Springer, Berlin, Germany, pp. 344–361, 2017.

[28] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, https://arxiv.org/abs/1807.03748, Article ID 03748.

[29] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9729–9738, Seattle, WA, USA, 19 June 2020.

[30] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning, PMLR 119*, pp. 1597–1607, Vienna, Austria, July 2020.

[31] M. Caron, H. Touvron, I. Misra et al., "Emerging properties in self-supervised vision transformers," 2021, https://arxiv.org/abs/2104.14294.

[32] J.-B. Grill, F. Strub, F. Altché et al., "Bootstrap your own latent: a new approach to self-supervised Learning," 2020, https://arxiv.org/abs/2006.07733, Article ID 07733.

[33] A. Dosovitskiy, L. Beyer, K. Alexander et al., "An image is worth 16x16 words: transformers for image recognition at scale," 2020, https://arxiv.org/abs/2010.11929.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, 30 June 2016.

[35] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, "Bottleneck transformers for visual recognition," 2021, https://arxiv.org/abs/2101.11605, Article ID 11605.

[36] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei, "ImageNet: a large-scale hierarchical image database," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 25 June 2009.

[37] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, Canada, 10 July 2009.

[38] L. Dhanabal and Dr. 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, 2015.

[39] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proceedings of the Military Communications and Information Systems Conference (MilCIS)*, p. 16, 12 Nov. 2015.

[40] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, Funchal, Madeira, Portugal, January 2018.

[41] M. Ring, S. Wunderlich, D. Gruedl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, pp. 361–369, ACPI, Chester, UK, June 2017.

[42] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM enhanced anomaly detection for industrial big data," *IEEE Transactions On Industrial Informatics*, vol. 17, no. 5, 2021.

[43] H. Zhang, J.-L. Li, X.-M. Liu, and C. Dong, "Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection," *Future Generation Computer Systems*, vol. 122, pp. 130–143, 2021.

[44] Q. Tian, D. Han, M.-Y. Hsieh, K.-C. Li, and A. Castiglione, "A two-stage intrusion detection approach for software-defined IoT networks," *Soft Comput*, vol. 25, 2021.

[45] Z. Wang, Y. Liu, D. He, and S. Chan, "Intrusion detection methods based on integrated deep learning model," *Computers & Security*, vol. 103, Article ID 102177, 2021.

[46] J. Gu, L. Wang, H. Wang, and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," *Computers & Security*, vol. 86, pp. 53–62, 2019.

[47] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, 2019.

[48] X. Xu, J. Li, Y. Yang, and F. Shen, "Toward effective intrusion detection using log-cosh conditional variational autoencoder," *IEEE Internet of Things Journal*, vol. 8, no. 8, 2021.

[49] M. Mulyanto, M. Faisal, S. W. Prakosa, and J.-S. Leu, "Effectiveness of focal loss for minority classification in network intrusion detection systems," *Symmetry Plus*, vol. 13, no. 1, p. 4, 2021.

[50] S. Huang and K. Lei, "IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks," *Ad Hoc Networks*, vol. 105, Article ID 102177, 2020.

[51] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with na¨ıve Bayes feature embedding," *Computers & Security*, vol. 103, Article ID 102158, 2020.

[52] W. Elmasry, A. Akbulut, and A. H. Zaim, "Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic," *Computer Networks*, vol. 168, Article ID 107042, 2020.

[53] Z. Chiba, N. Abghour, K. Moussaid, A. El omri, and M. Rida, "Intelligent approach to build a Deep Neural Network based IDS for cloud environment using combination of machine learning algorithms," *Computers & Security*, vol. 86, pp. 291–317, 2019.

[54] G. N. Nguyen, N. H. L. Viet, M. Elhoseny, K. Shankar, B. B. Gupta, and A. A. A. El-Latif, "Secure blockchain enabled Cyber–physical systems in healthcare using deep belief network with ResNet model," *Journal of Parallel and Distributed Computing*, vol. 153, pp. 150–160, July 2021.

[55] D. Gonzalez-Cuautle, A. Hernandez-Suarez, G. Sanchez-Perez et al., "Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusion-detection-system datasets," *Applied Sciences*, vol. 10, no. 3, p. 794, 2020.

[56] L. Van Der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.