Hindawi
Security and Communication Networks
Volume 2021, Article ID 2843856, 11 pages
https://doi.org/10.1155/2021/2843856

WILEY | Hindawi

Research Article

Network Intrusion Detection with Nonsymmetric Deep Autoencoding Feature Extraction

Zhaojun Gu,[1] **Liyin Wang**,[2] **Chunbo Liu**,[1] **and Zhi Wang**[3]

[1]*Information Security Evaluation Center, Civil Aviation University of China, Tianjin 300300, China*
[2]*College of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China*
[3]*College of Cyber Science, Nankai University, Tianjin 300350, China*

Correspondence should be addressed to Zhi Wang; zwang@nankai.edu.cn

Received 28 October 2021; Accepted 14 December 2021; Published 31 December 2021

Academic Editor: Ilsun You

To address the problems of high reconstruction error and long training time when using Stack Nonsymmetric Deep Autoencoder (SNDAE) feature extraction technology for intrusion detection, Adam Nonsymmetric Deep Autoencoder (ANDAE) is proposed based on SNDAE. The Adam optimization algorithm is used to update network parameters during training so that the loss function can quickly converge to the ideal value. Under the premise of not affecting the effect of feature extraction, the network structure is simplified, and the training time of the network is reduced to realize the efficient extraction of the rapid growth of high-dimension and nonlinear network traffic features. For the low-dimensional prominent features extracted by ANDAE, Random Forest is used for classification to detect intrusion action, and a network intrusion detection model based on ANDAE feature extraction is implemented. The experimental results on the NSL-KDD and the CIC-IDS2017 datasets show that, compared to the SNDAE-based intrusion detection model, the ANDAE model has an average increase of 6.78% in accuracy, an average of 13.06% in recall, and an average of 14.9% in $F1$ scores. Feature extraction time is reduced by 23.1% on average. Thus, the ANDAE model is an intrusion detection solution, which can simultaneously improve detection accuracy and time efficiency.

1. Introduction

The information society is developing rapidly, while more and more network equipment and applications are produced to meet people's life and work needs. Individuals and businesses are facing threats from cyberattacks such as data theft and ransomware. The Stuxnet virus that targets critical infrastructure discovered in 2010 proves that the harm of cyberattacks has reached the national level. Intrusion detection system (IDS), as active defense technology, has gradually become a key technology to ensure network security [1]. However, with the increase in network transmission speed, the popularization of the Internet of Things, and the application of technologies such as cloud computing, the network traffic analysis by IDS is growing rapidly. Many different protocols are used in the transmission of network traffic transmission, and the field values of these protocols often contain many categorical variables, which make network traffic have high-dimensional and nonlinear characteristics. If these high-dimensional data are directly detected, a lot of computing resources will be consumed, and the detection efficiency will be low.

To address the above problems, many studies have adopted methods that combine deep learning and traditional machine learning. That is, the deep neural network is used to unsupervised extract the prominent features of the data distribution, reduce the high-dimensional data to low-dimensional data, and then use traditional machine learning methods to build a classification model for intrusion detection. Feature extraction is different from conventional feature selection in that it generates new features that are more representative of the original data than the original features. The superior layerwise feature learning ability of deep learning can better match the performance of traditional machine learning techniques, especially the ability to capture nonlinear features [2]. The learned features have a

more substantial representation of the original data, which is convenient for visualization or classification.

Deep learning is being applied in many research fields. There are some existing works in the field of network intrusion detection. Yin et al. [3] proposed a deep learning method using Recurrent Neural Network (RNN) for intrusion detection. The performance of the model in two-class classification and multiclass classification and the influence of the number of neurons and different learning rates on the performance of the model are also studied. Man et al. [4] used Federated Learning and Convolutional Neural Networks (CNN) to achieve intrusion detection. Li et al. [5] proposed an image conversion method for network data and used CNN automatically to learn the characteristics of images. Liu [6] proposed a network intrusion detection system based on CNN. Man and Sun [7] used network data to construct CNN with residual blocks to perform intrusion detection. In addition, Ring et al. [8] also investigated the datasets used in the detection of network intrusion.

There are many ways to use deep learning for feature extraction, such as Autoencoder (AE) [9–12], improved Autoencoder [1, 13–15], Long Short-Term Memory (LSTM) neural network [16], and Stacked Nonsymmetric Deep Autoencoder (SNDAE) [17]. Wang and Wang [9] implemented the dimensionality reduction and feature extraction of the original data by introducing AE and used the improved K-means algorithm to cluster the processed data. Xiao et al. [10] used AE and Principal Component Analysis (PCA) to achieve data dimensionality reduction and converted the dimensionality-reduced data into an image format. The image is used to train CNN to obtain optimal features and finally classified by Softmax. Liu and Chung [11] first used AE with two consecutive hidden layers to extract features and then used Random Forest (RF) and Support Vector Machines (SVM) for feature selection. Kunang et al. [12] used AE for feature extraction and SVM as a classifier.

There are also many studies using improved AE for feature extraction. Song et al. [13] achieved adaptive and intelligent feature extraction using an improved Sparse Autoencoder. Yan and Han [1] used Stacked Sparse Autoencoder (SSAE) to extract high-level feature representations of intrusion action information. Low-dimensional sparse features are used to construct different basic classifiers. Yao et al. [14] applied the Variational Autoencoder (VAE) to extract valuable features for unsupervised anomaly detection tasks and used algorithms such as K-Nearest Neighbor (KNN) for anomaly detection. Furthermore, Wang et al. [16] proposed two deep feature extraction algorithms based on LSTM, which is used to learn the prominent features of the data.

The Stacked Nonsymmetric Deep Autoencoder (SNDAE) proposed by Shone et al. [17] is currently one of the prominent technologies in the study of feature extraction using deep learning. It combines the efficiency of AE with the advantages of layerwise learning of the Stacked Autoencoder (SAE). Compared with the simple AE, SNDAE has a better unsupervised layerwise feature learning ability. Meanwhile, compared with SAE, SNDAE does not use the layer-by-layer greedy training method to get significant data representation, which reduces the training time of the model and improves the efficiency of the overall model.

However, current research still has limitations. First of all, despite many optimizations, the time required for training the existing feature extraction methods using deep learning is still too long. Second, it is difficult to reduce the reconstruction error of the neural network to the ideal value when using the network flow for training. The value of the reconstruction error determines whether the model can accurately learn the data distribution and extract the ideal features. Finally, some existing studies are still using outdated datasets, such as the KDD99 dataset collected in 1998, which can no longer represent the current network environment.

This paper proposes an Adam Nonsymmetric Deep Autoencoder (ANDAE) based on the Adam optimization algorithm [18] on the basis of SNDAE, which solves the problem that the reconstruction error remains high and cannot converge to an ideal value during SNDAE training. It saves the time of feature extraction in the meantime. This paper also uses ANDAE feature extraction technology to construct an effective network intrusion detection model and uses Random Forest (RF) [19] algorithm to classify the significant data after feature extraction. Finally, experiments are carried out on the NSL-KDD [20] and the CIC-IDS2017 [21] datasets.

2. Materials and Methods

2.1. Autoencoder and Deep Autoencoder. An Autoencoder (AE) is an unsupervised feature learning method based on a neural network. Its purpose is to generate $\hat{x}$ as similar as possible to the input data x. It is often used for data dimensionality reduction and feature extraction. A simple AE has an input layer, an output layer, and a hidden layer. An example of AE is shown in Figure 1.

AE first uses an encoder $f(x)$ to encode data from high-dimensional to low-dimensional hidden layers, and in this process, data dimensionality reduction is achieved. Then, the decoder $d(x)$ reconstructs the low-dimensional data of the hidden layer. The reconstructed data are used with the input x to calculate the reconstruction error $L(x, d(f(x)))$. Finally, the network is updated through backpropagation. In the process of encoding, due to the decrease of dimensionality, AE will learn as much as possible the features that can more significantly represent the data distribution so that the decoder can more accurately use these new learned features to reconstruct.

The Deep Autoencoder (DAE) consists of two symmetric deep neural networks that are used for encoding and decoding, respectively. The work of Hinton and Salakhutdinov [22] fully proved that the use of DAE for feature extraction can make data better discrimination after dimensionality reduction, so that data that originally cannot be separated can also be separated. A typical DAE is a Stacked Autoencoder (SAE) [23]. SAE adopts a greedy layer-by-layer training method. When a certain layer of network is trained, the parameters of all previous layers are frozen.

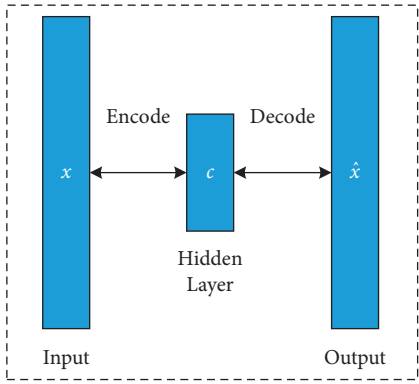

Figure 1: Autoencoder.

After that layer is trained, the next layer is trained in the same way until the entire network is trained. An example of a simple SAE is shown in Figure 2.

Figure 2 shows an SAE with 5 layers of neural network. The hidden layer in the middle is also called the coding layer. The data stored in this layer are called the codes. The code is more discriminative data after dimensionality reduction, which can be used for classification and visualization. The data after feature extraction is the code located in the middle hidden layer.

2.2. Adam Nonsymmetric Deep Autoencoder. When using DAE for feature extraction, the encoder will play an important role. The decoder is only used to reconstruct the hidden layer data to calculate the reconstruction error during the training process. Moreover, after the network is trained, only the coding operation is performed for feature extraction. Compared with the decoder, the performance of the encoder is more important for feature extraction. In addition, the goal of training a neural network is to learn the most knowledge with the fewest neurons, and the streamlined structure of the neural network can also save training time. For this reason, an Adam Nonsymmetric Deep Autoencoder (ANDAE) is proposed that focuses more on the encoder, as shown in Figure 3.

ANDAE sets up two hidden layers to implement encoding operations. Unlike DAE that reconstructs data layer by layer, ANDAE performs only one decoding operation on the hidden layer data to calculate the reconstruction error. This is because as long as the loss function of the neural network during training can converge to an ideal value, the nonsymmetry DAE would extract ideal features. To achieve this goal, optimization of the Adam algorithm is used when training the network. Adam optimization algorithm is currently a very popular neural network optimization scheme, which combines Adaptive Gradient (AdaGrad) [24] and Root Mean Square Propagation (RMSProp) [25]. Compared to other optimization algorithms, it converges faster but requires relatively low memory. Therefore, it is suitable for large-scale datasets.

ANDAE adopts a deterministic function to gradually map the input vector $x \in R^d$ to the hidden layer $h_i \in R^{d_i}$. The deterministic function is shown as follows:

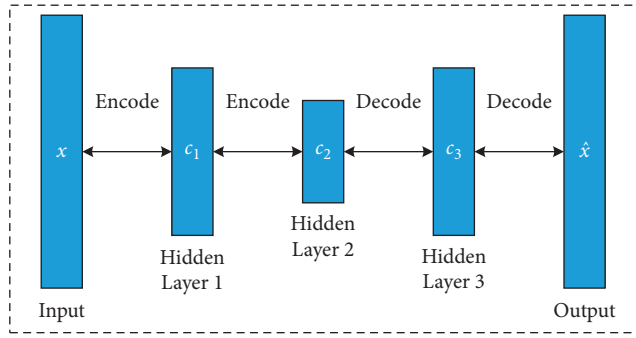

Figure 2: Stacked Autoencoder.

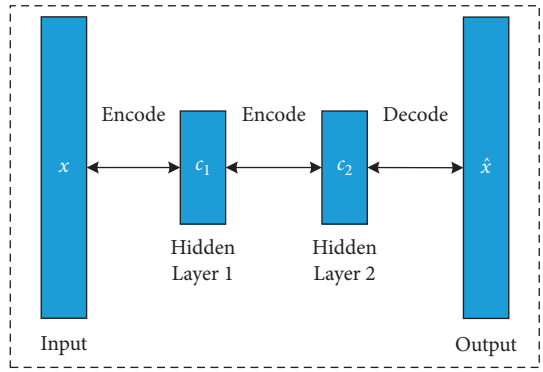

Figure 3: Adam Nonsymmetric Deep Autoencoder.

$$h_i = \sigma\left(W_i.h_{i-1} + b_i\right); \quad i = \overline{1,n}, \tag{1}$$

where $h_0 = x$, d represents the dimension of the vector, n is the number of hidden layers, W is the weight, and b is the bias. ANDAE selects the Sigmoid function as the activation function, which is defined as follows:

$$\sigma(t) = \frac{1}{\left(1 + e^{-t}\right)}. \tag{2}$$

During training, ANDAE employs only one decoding operation to reconstruct the hidden layer data. The output $\hat{x}$ is defined as follows:

$$\hat{x} = \sigma\left(W_{n+1}.h_i + b_{n+1}\right). \tag{3}$$

The purpose of training is to minimize the reconstruction error L of the samples. L is described as follows:

$$L(\theta) = \sum_{i=1}^{m} \left(x_i - \hat{x}_i\right)^2, \tag{4}$$

where $\theta = (W_i, b_i)$ and m represents the number of samples. The process of parameter update is formulated as follows:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\left(\sqrt{\hat{v}_t} + \varepsilon\right)}, \tag{5}$$

where α is the learning rate, t records the steps of parameter update, and ε is a very small number to avoid dividing by zero. $\hat{m}_t$ and $\hat{v}_t$ are the deviation corrections of m_t and v_t, which are defined as follows:

$$\widehat{m}_t = \frac{m_t}{\left(1-\beta_1^t\right)},$$
$$\widehat{v}_t = \frac{v_t}{\left(1-\beta_2^t\right)}, \tag{6}$$

where β_1^t and β_2^t represent β_1 and β_2 to the power of t, respectively. By calculating the exponentially weighted average value m_t of the gradient and the exponentially weighted average value v_t of the gradient square, the local mean value of the parameter can be estimated. So the update of the parameter is related to the value in the past period of time. m_t and v_t are defined as follows:

$$m_t = \beta_1 \cdot m_{t-1} + \left(1-\beta_1\right) \cdot g_t,$$
$$v_t = \beta_2 \cdot v_{t-1} + \left(1-\beta_2\right) \cdot g_t^2, \tag{7}$$

where $\beta_1, \beta_2 \in [0,1)$ are preset hyperparameters used to control the proportion of historical information, $g_t = \Delta_\theta L_t\left(\theta_{t-1}\right)$ is the gradient of the parameter θ at time step t, and g_t^2 is the elementwise square of the gradient.

2.2.1. ANDAE-Based Network Intrusion Detection Model. The ANDAE-based network intrusion detection model (hereafter referred to as the ANDAE model) combines unsupervised feature learning based on deep learning with traditional machine learning methods. The core idea is that the neural network will capture the most significant distribution of the data in the process of reducing the dimensionality of the data. The data will be more distinguishable in the hidden layers, and the distinction between different types of samples will be more obvious. Then intrusion detection on the hidden layer data will be more effective.

After experimental verification of the effects of different traditional machine learning methods on intrusion detection datasets, the Random Forests (RF) model was finally selected as the classifier. RF is an ensemble learning method that makes predictions based on the results of multiple decision trees. RF has been proven to be one of the best learning algorithms for intrusion detection. There are currently many studies using RF for intrusion detection, as summarized by Resende and Drummond [26]. The overall framework is shown in Figure 4.

First, the network data are preprocessed, One-Hot Encoding is used to realize numeralization, and then StandardScaler is used to make the data conform to the standard normal distribution. The preprocessed data are divided into training set and test set. The training set is used to train ANDAE. Minibatch training and backpropagation are used to obtain the ideal weight of ANDAE during training. Then, the trained ANDAE is applied to perform feature extraction on the data of the test set, which extracts the overall data of the test set once and extracts it twice. Finally, the ANDAE model employs RF to classify the extracted data.

Extracting twice refers to stacking two ANDAEs while using the first ANDAE for feature extraction and then passing the extracted data to the next ANDAE. The purpose of doing so is to create a hierarchical deep learning structure to learn the complex and nonlinear relationships among different features. The second feature extraction has an optimizing effect on the first extracted data, which can make the extracted data more significant. The ANDAE network structure is shown in Figure 5.

In Figure 5, c_1 and c_2 represent the number of neurons in each layer of the network, that is, the dimension of the data. After being digitized by One-Hot Encoding, the data dimension of the NSL-KDD dataset is 122. Two ANDAEs are used to reduce the dimension to 30, while c_1 and c_2 are set to 20 and 30, respectively. Meanwhile, for the CIC-IDS dataset, the dimension of the input data is dropped from 78 to 24, while c_1 and c_2 are set to 14 and 24, respectively. The learning rate, as one of the networks' hyperparameters, is set to 0.01. According to the data dimension after feature extraction, the number of trees in Random Forest is set to 10, and other parameters are set to the default values. In the field of deep learning research, the network structure of the model determines its effect. At present, the network structure of the model is set subjectively in advance. Therefore, in our model, multiple experiments are conducted on public datasets to select the network structure with the best results.

3. Results and Discussion

The experimental equipment is configured with an Intel Core i7-6700 eight-core processor, 16 GB of RAM, and a 64-bit Windows 10 operating system. Python is chosen as the programming language, and ANDAE is implemented with Google's deep learning framework TensorFlow. During training, ANDAE is so efficient that it only takes very little time for large datasets without GPU acceleration.

3.1. Evaluation Measures. The precision, recall, and $F1$ score based on the confusion matrix are used as evaluation metrics. The definition of the confusion matrix is shown in Table 1. Among them, true positive (TP) means correct recognition of attack behavior, true negative (TN) means correct recognition of normal behavior, false positive (FP) means misrecognition of attack behavior, and false negative (FN) means misrecognition of normal behavior.

The precision means the proportion of correctly predicted attacks to all predicted attacks. High precision means that the model produces fewer false positives. It is defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{8}$$

The recall means the proportion of attacks that are correctly predicted to account for all actual attacks. If the goal is to find all attacks for the model, the recall can be used as an evaluation metric. It is defined as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{9}$$

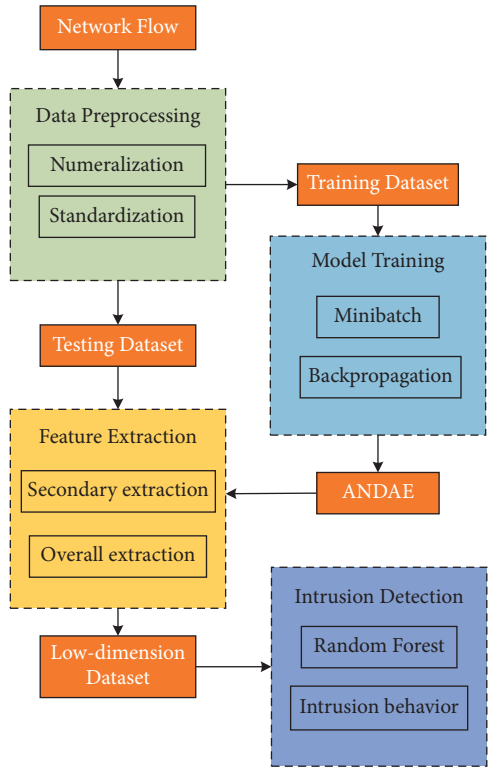

Figure 4: Framework of the ANDAE model.

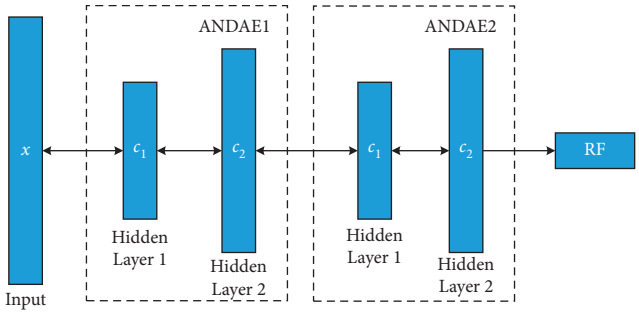

Figure 5: Network structure of the ANDAE model.

Table 1: Confusion matrix.

Sample class		Predicted	
		Normal	Attack
Real	Normal	TN	FP
	Attack	FN	TP

The $F1$ score is the harmonic mean of precision and recall. The larger the value, the better the model. It is defined as follows:

$$F1 = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{10}$$

3.2. Experimental Data. The experiments are implemented on the NSL-KDD and the CIC-IDS2017 datasets, both of which are flow data. Flow data are data packets grouped in time periods, which are the most widely used data source for IDS [27]. These two datasets are both benchmark datasets released by the Canadian Institute of Cyber Security. Many studies have been verified on these two authoritative datasets, so the experimental results on these datasets are more convincing.

The NSL-KDD dataset solves the problems of KDD's original data redundancy, which is an optimized dataset of the latter. This dataset contains 494021 training records and 311029 testing records. There are four types of attacks on these data: Denial of Service (DoS), Detection Attack (Probe), Elevation of Privilege Attack (U2R), and Remote User Attack (R2L). DoS and Probe attack or scan multiple hosts in a short period of time to establish many connections, while R2L and U2R attacks are embedded in the data packets and usually only involve a single connection. Data statistics are shown in Table 2.

The CIC-IDS2017 is a dataset containing flow data stored by time unit, which were captured during 5 consecutive days from Monday to Friday by using a variety of publicly available testing tools to simulate attacks. It simulates the latest common attacks such as DoS, Distributed Denial of Service (DDoS), Heartbleed, Brute Force, Cross-Site Scripting (XSS), Sqlinjection, Infiltration, Portscan, and Botnet. This dataset reflects the current network environment better than some old datasets, especially since it simulates Web attacks, which is lacking in existing public datasets. The statistics of the CIC-IDS2017 dataset are shown in Table 3.

The FTP-Patator and SSH-Patator in Table 3 refer to the use of Patator to brute-force the password of the target

Table 2: Composition of the NSL-KDD dataset.

Attack type	Train	Test
DoS	45927	7460
Probe	11656	2421
R2L	995	2885
U2R	52	67
Normal	67343	9711
Total	125973	22543

Table 3: Composition of the CIC-IDS2017 dataset.

Time	Attack type	Record
Mon	Normal	529918
Tues	Normal	432074
	FTP-Patator	7938
	SSH-Patator	5897
Wed	Normal	440031
	DoS	252661
	Heartbleed	11
Thur am	Normal	168186
	Brute Force	1507
	XSS	652
	Sqlinjection	21
Thur pm	Normal	288566
	Infiltration	36
Fri am	Normal	189067
	Botnet	1966
Fri pm	Normal	97718
	DDoS	128027
Fri pm	Normal	127537
	Portscan	158930

system through FTP and SSH. Brute Force is a cracking attack on Web application. Web attacks and Infiltration attacks were simulated in the morning and afternoon of Thursday, while Botnet, DDoS, and Portscan were simulated on Friday.

3.3. Preprocessing. Using One-Hot Encoding to digitize discrete features will make the distance calculation between features more reasonable. Applying StandardScaler to process the data can make them conform to the standard normal distribution with a mean of 0 and standard deviation of 1. Compared with normalization, standardization better maintains the sample spacing. The transformation function is defined as follows:

$$x^* = \frac{x - \mu}{\sigma}, \tag{11}$$

where μ is the mean of all sample data and σ is the standard deviation of all sample data.

The NSL-KDD dataset has 41 features, among which *protocol_type*, *service*, *flag*, and *label* are categorical variables. *Protocol_type* has 3 categories, *service* has 70 categories, and *flag* has 11 categories. After One-Hot Encoding for these classification variables, the data dimension is increased from 41 to 122. The dataset has been divided into a training set and a test set in advance, both of which are standardized to conform to normal distribution. The dataset is divided into four subsets with only one type of attack in each subset.

The CIC-IDS2017 dataset has 78 features, which are statistically features extracted from the original traffic Pcap file by CICFlowMeter [28]. The extracted features have been defined and interpreted on the CICFlowMeter Web page, which is the best general feature to detect common attacks. The CIC-IDS2017 dataset is stored by day, and the amount of data is very large every day. It takes a lot of time to preprocess the dataset. First, the daily data are counted. We have found that there are infinite values and NaN values for the *Flow Bytes/s* and *Flow Packets/s* features and then replace them with 0. These statistical features are numerical variables, so One-Hot Encoding is not required. The training set and the test set are divided before standardization to avoid the data from the training set affecting the test set during standardization. The test set accounts for 30%.

3.4. Experimental Results of the NSL-KDD Dataset. SNDAE [9] has been proven to be one of the most prominent technologies in current research on intrusion detection using feature extraction. In contrast to existing research, this model has higher accuracy and less training time. Therefore, we choose SNADE model for comparison to prove the effectiveness of the ANDAE model.

In order to show the effect of ANDAE feature extraction more intuitively, the data before and after ANDAE feature extraction are visualized, as shown in Figure 6. We depict the data of the DoS attack because they have the largest number compared with the other three attacks. PCA is used to reduce the data to three dimensions.

Figure 6(a) is visualization before ANDAE feature extraction, and Figure 6(b) is visualization after ANDAE feature extraction. The red dots in Figure 6 are attack samples, and the blue dots are normal samples. It can be seen from the visualization results that the distance between two classes of data will be further after feature extraction, and the difference between classes will be more obvious. It can also be seen that the distribution of data will be more uniform, which is conducive for the model to distinguish different data categories, correctly divide decision boundaries, and identify network attacks.

The loss function curve of the ANDAE and the SNDAE model during training is compared in Figure 7. 64000 pieces of data from the NSL-KDD dataset are selected. During training, the data are reduced from 122 dimensions to 30 dimensions. Minibatch is used in model training. In this method, the dataset is divided into equal subsets. Gradient descent is performed on only one subset at a time; the next subset is trained after updating the parameters; all that is to save running memory. Each subset is called a batch. The size of the batch is generally set to 2^n. The dataset is divided into 1000 subsets; that is, there are 1000 batches, and the size of each batch is 64. 1000 batches means training the model 1000 times. For ease of display, the loss function curve is drawn in units of training times.

In Figure 7, the y-axis is the reconstruction error, and the x-axis is the training times. It can be seen from the figure

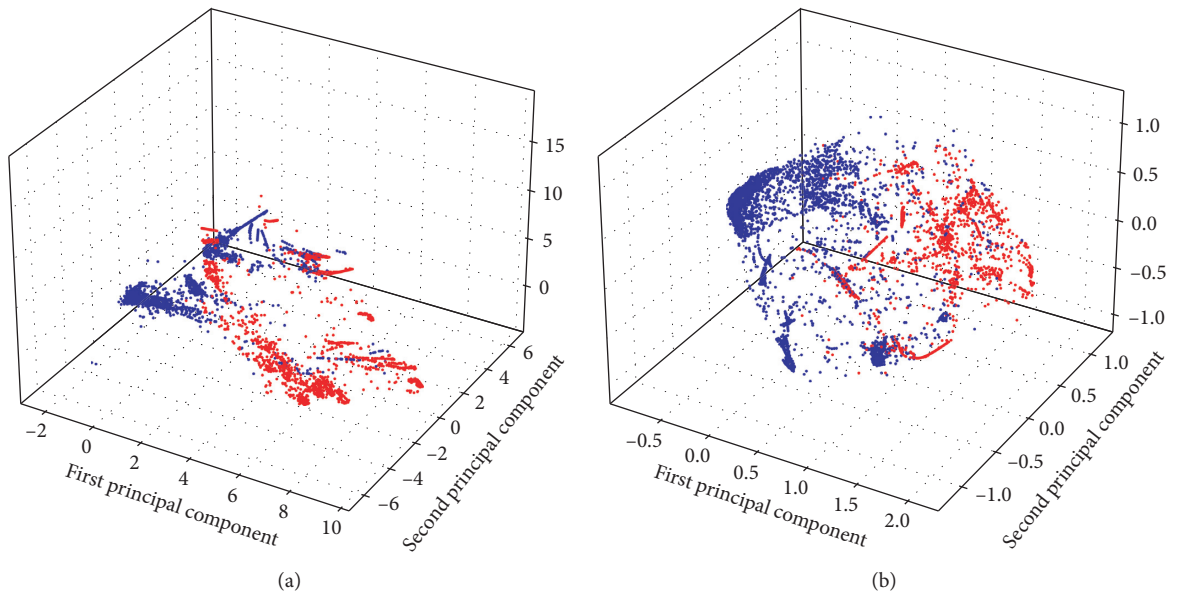

Figure 6: Comparison of visualization before and after ANDAE feature extraction. (a) Visualization before ANDAE feature extraction. (b) Visualization after ANDAE feature extraction.

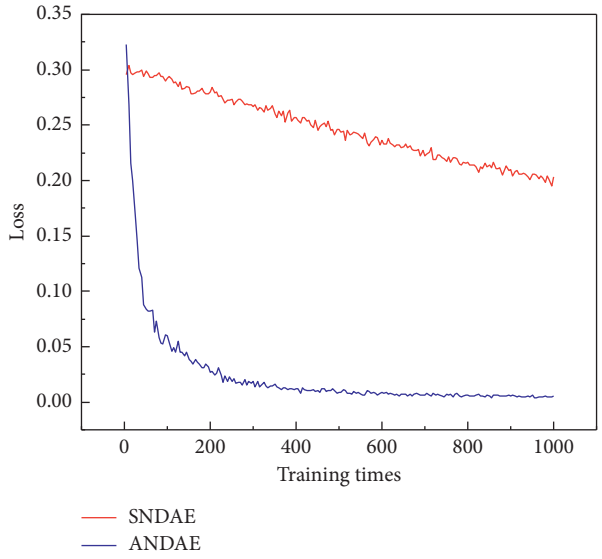

Figure 7: Comparison of loss function curves.

that, in 1000 pieces of training, the reconstruction error of ANDAE first decreases rapidly to 0.05 and then slowly tends to 0 as the training times increase. It means that ANDAE can reconstruct $\hat{x}$ well, so the extracted low-dimensional features are ideal classification data. It can also be seen that the line goes up and down, and the reconstruction error fluctuates. This is because only one subset is trained at a time, not the entire dataset. The difference in data leads to such a result. But it has no effect on feature extraction because the overall is declining. The loss function curve of SNDAE is always above 0.2, and the overall downward trend is slow. The nonconvergence of the loss function means that there is a large difference between the samples learned by SNDAE and the actual samples, which leads to the extracted features being not ideal, so the classification accuracy of the classification algorithm will be affected.

Reference [9] of the SNADE model provides three evaluation values of its four attacks on the NSL-KDD datasets, which are directly cited in this paper for comparison. The number of neurons in the two hidden layers of the ANDAE network is 20 and 30, respectively. The learning rate is set to 0.01. The number of trees in Random Forest is set to 10, and other parameters are set to the default values. Table 4 shows the values of three evaluation metrics of the two models against four attacks in the NSL-KDD dataset. As can be seen in the table, the ANDAE model is higher than the

Table 4: Experimental results of 4 attacks in NSL-KDD dataset.

Attack type	Precision (%)		Recall (%)		$F1$ (%)	
	SNDAE	ANDAE	SNDAE	ANDAE	SNDAE	ANDAE
DoS	**100.00**	99.23	94.58	**99.39**	97.22	**99.31**
Probe	**100.00**	97.55	94.67	**95.58**	**97.26**	96.55
R2L	**100.00**	93.45	3.82	**90.91**	7.36	**92.16**
U2R	**100.00**	92.78	2.70	**55.07**	5.26	**68.81**

The bold values are the larger of the evaluation metrics of the two models.

SNDAE model in recall and $F1$ score, which more reflects overall performance. It indicates that the comprehensive performance of the ANDAE model is better than that of the SNDAE model. Although the SNDAE model achieves high precision, low recall and $F1$ score mean that the model will have false positive in detection, which is unfavourable to network intrusion detection system. Individually, for DoS attacks, the recall and $F1$ scores of the ANDAE model are higher than those of the SNDAE model. For Probe, the scores of the ANDAE model and the SNDAE model are close. It should be noted that the recall and $F1$ scores of R2L and U2R attacks are significantly improved by the ANDAE model compared with the SNDAE model.

Figure 8 shows the Recursive Feature Elimination Cross-Validation (RFECV) curve of the ANDAE model for four types of attack detection. The RFECV curve first selects different numbers of features for cross validation and then selects the feature combination with the highest score under a fixed number. This method has a large amount of calculation and takes a long time to run once, but it can help to verify the performance of new features extracted from the ANDAE model. In this paper, cross-validation $F1$ score is selected as the evaluation metric of cross validation.

In Figure 8, with the increasing number of selected features, the $F1$ score of ANDAE against DoS, Probe, and R2L attacks first increases rapidly to an inflection point and then increases slowly. The $F1$ score does not reach the highest value until there are 30 features. This slow rise is very small, so it is not obvious in the figure. The RFECV curves of four types of attack prove that the 30 new features are ideal and can well represent the original data. It can also be seen from the figure that the detection effect of the ANADE model on U2R is still not satisfactory, and the reason is the lack of U2R attack samples as training data. The above results also prove that the quality of feature extraction is related to the reconstruction error during training. If the loss function during training converges to the ideal value, the ideal feature can be extracted, and the detection accuracy can be improved; that is, the reconstruction errors of the feature extraction network during training have an impact on the accuracy of intrusion detection.

3.5. Experimental Results of the CIC-IDS2017 Dataset. Three evaluation metrics of the ANDAE model and the SNADE model for 11 attacks in the CIC-IDS2017 dataset are provided in Table 5. At this time, the number of neurons in the two hidden layers of the ANDAE network is 14 and 24, respectively. The parameter values of Random Forest and learning rate are the same as those set on the NSL-KDD dataset.

As shown in Table 5, the ANDAE model and the SNDAE model have achieved similar high detection results against DoS, DDoS, and Portscan attacks. For FTP-Patator, SSH-Patator, Brute Force, and XSS attacks, the ANDAE model have achieved an average $F1$ score of 98.01%, which is higher than that of the SNDAE model. The average values of precision and recall are also higher than those of the SNDAE model for these 4 attacks. For Infiltration and Botnet, the ANDAE model has a significant increase in detection score compared to the SNDAE model. In general, the ANDEA model has achieved ideal detection results, which is substantially improved compared with the SNDAE model.

DoS, DDoS, and Portscan attacks last for a long time. Each of them can attack multiple hosts and establish many connections. As a result, the attack data generated is large and continuous. It can be seen from the dataset statistics that there are more DDoS and Portscan attack data than normal data, and DoS attack data is more than half of normal data. Continuous and excessive attack data makes the SNDAE model and the ANDAE model fully trained, so the two models get high scores for the detection of these three attacks.

The data volume of FTP-Patator and SSH-Patator attacks conforms to the normal proportion, accounting for about 2% of the total data. When using a normal proportion of attack data training, the detection results of the ANDAE model for these two attacks are higher than those of the SNDAE model, which is due to the accurate feature learning ability of ANDAE. It is particularly noteworthy that the ANDAE model has achieved good results in the detection of Brute Force and XSS attacks on Web applications. Web service attack is a popular attack method at present. Companies now basically use Web applications. When the intranet is closed, Web applications that need users to access and must connect to the Internet become the preferred target of attackers. The ANDAE model is a feasible intrusion detection method for Web attacks.

Due to the lack of training data, the evaluation metrics of Heartbleed and Sqlinjection attacks are 0. Heartbleed attack only has 11 records, and Sqlinjection attack only has 21 records. ANDAE's detection results of Infiltration and Botnet are not satisfactory, but they are better than those of the SNDAE model. Infiltration is an attack that uses vulnerable software to set up a back door on the victim's computer and infiltrate the interior of the network through the back door. Botnet refers to a network formed by many hosts infected with the bot virus.

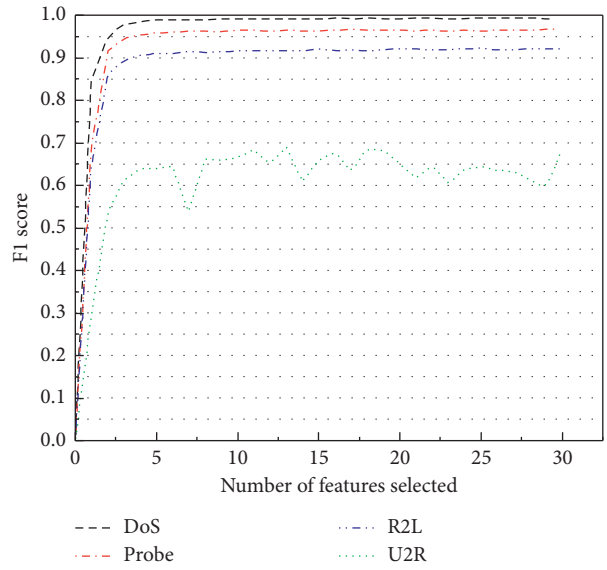

Figure 8: RFECV curves of the ANDAE model.

Table 5: Experimental results of 11 attacks in the CIC-IDS2017 dataset.

Attack type	Precision (%)		Recall (%)		*F*1 (%)	
	SNDAE	ANDAE	SNDAE	ANDAE	SNDAE	ANDAE
FTP-Patator	97.1	**99.87**	98.53	**98.95**	97.81	**99.4**
SSH-Patator	96.02	**99.42**	95.13	**97.85**	95.56	**98.63**
DoS	**99.83**	99.79	99.66	**99.67**	**99.74**	99.73
Heartbleed	0	0	0	0	0	0
Brute Force	91.81	**99.31**	91.58	**96.67**	91.64	**97.97**
XSS	97.24	**99.47**	87.77	**92.85**	92.21	**96.04**
Sqlinjection	0	0	0	0	0	0
Infiltration	0	**83.33**	0	**33.33**	0	**47.43**
Botnet	73.01	**92.56**	71.81	**75.92**	72.32	**83.42**
DDoS	**99.97**	99.96	99.89	**99.93**	99.93	**99.95**
Portscan	99.97	**99.99**	99.92	**99.97**	99.52	**99.98**

The bold values are the larger of the evaluation metrics of the two models.

Table 6: Comparison of feature extraction time.

Dataset	Dimension	Extraction time (s)		Time saving (%)
		SNDAE	ANDAE	
NSL-KDD	20	3.91	3.28	16.11
	30	4.17	3.04	27.09
	40	5.14	3.95	23.1
CIC-IDS2017	20	12.64	9.64	23.73
	30	11.82	8.52	27.91
	40	11.26	8.93	20.69

3.5.1. Comparison of Feature Extraction Time. We also compare the feature extraction time of the two methods (see Table 6). This is achieved by extracting the features of 148156 records from the NSL-KDD dataset and 692703 records from the DoS attack within CIC-IDS2017. The time consumed to extract the data to 20, 30, and 40 dimensions is recorded, respectively, and the time-saving rate is calculated. It can be seen from the table that ANDAE saves 23% of the extraction time on average compared with SNDAE.

4. Conclusions

In this paper, an Adam Nonsymmetric Deep Autoencoder (ANDAE) for feature extraction is proposed, and it is combined with the traditional machine learning algorithm Random Forest (RF) to construct a network intrusion detection model. The model has achieved a high recall rate and *F*1 score for the four types of attack detection on the NSL-KDD dataset. It is also proven that the reconstruction error

of the feature extraction network during training has an impact on the accuracy of intrusion detection. As long as the loss function converges to the ideal value during training, the ideal features can be extracted, and the detection accuracy can be improved. The model also achieves high scores in the detection of multiple attacks on the CIC-IDS2017 dataset, especially Web attacks. Compared with the SNDAE model, the ANDAE model improves the detection accuracy and reduces the time of feature extraction. It is a feasible and efficient method that adopts deep feature extraction technology for network intrusion detection. The follow-up work will combine ANDAE with other anomaly detection methods to improve the detection accuracy of small sample abnormal behavior.

Data Availability

The research data supporting the results of this study can be available from https://www.unb.ca/cic/datasets/index.html.

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 under Grants 61872202 and 61601467; the Civil Aviation Safety Capacity Building Foundation of China under Grants PESA2019073, PESA2019074, and PESA2020100; the Natural Science Foundation of Tianjin under Grant 19JCYBJC15500; and Key Research Program of the Chinese Academy of Sciences, Grant no. KFZD-SW-440.

References

[1] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41238–41248, 2018.

[2] S. Hou, A. Saas, L. Chen, and Y. Ye, "Deep4maldroid: a deep learning framework for android malware detection based on linux kernel system call graphs," in *Proceedings of the 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*, pp. 104–111, Omaha, NE, USA, October 2016.

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

[4] D. Man, F. Zeng, W. Yang, M. Yu, J. Lv, and Y. Wang, "Intelligent intrusion detection based on federated learning for edge-assisted Internet of Things," *Security and Communication Networks*, vol. 2021, Article ID 9361348, 11 pages, 2021.

[5] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion Detection Using Convolutional Neural Networks for Representation Learning," in *Proceedings of the International conference on neural information processing*, pp. 858–866, Guangzhou, China, November 2017.

[6] P. Liu, "An intrusion detection system based on convolutional neural network," in *Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*, pp. 62–67, Perth, Australia, February 2019.

[7] J. Man and G. Sun, "A residual learning-based network intrusion detection system," *Security and Communication Networks*, vol. 2021, Article ID 5593435, 9 pages, 2021.

[8] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[9] X. Wang and L. Wang, "Research on intrusion detection based on feature extraction of autoencoder and the improved k-means algorithm," in *Proceedings of the 2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2, pp. 352–356, Hangzhou, China, December 2017.

[10] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, Article ID 42210, 2019.

[11] J. Liu and S. S. Chung, "Automatic Feature Extraction and Selection for Machine Learning Based Intrusion Detection," in *Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation(SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 1400–1405, Leicester, UK, August 2019.

[12] Y. N. Kunang, S. Nurmaini, D. Stiawan, and A. Zarkasi, "Automatic Features Extraction Using Autoencoder in Intrusion Detection System," in *Proceedings of the International Conference on Electrical Engineering and Computer Science (ICECOS)*, pp. 219–224, Pangkal, Indonesia, October 2018.

[13] Y. Song, B. Hou, and Z. Cai, "Network intrusion detection method based on deep learning feature extraction," *Journal of Huazhong University of Science and Technology (Nature Science Edition)*, vol. 49, pp. 115–120, 2021.

[14] R. Yao, C. . Liu, L. Zhang, and P. Peng, "Unsupervised anomaly detection using variational auto-encoder based feature extraction," in *Proceedins of the IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–7, San Francisco, CA, USA, June 2019.

[15] S. N. Mighan and M. Kahani, "Deep learning based latent feature extraction for intrusion detection," in *Proceedings of the Electrical Engineering (ICEE), Iranian Conference on*, pp. 1511–1516, Mashhad, Iran, May 2018.

[16] A. Wang, X. Gong, and J. Lu, "Deep feature extraction in intrusion detection system," in *Proceedings of the 2019 IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 104–109, Tokyo, Japan, December 2019.

[17] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, pp. 41–50, 2018.

[18] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," December 2014, https://www.arxiv-vanity.com/papers/1412.6980/.

[19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] 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*, pp. 1–6, Ottawa, ON, Canada, July 2009.

[21] 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, vol. 1, pp. 108–116, Funchal, Madeira, Portugal, January 2018.

[22] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[23] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-wise Training of Deep Networks," *Advances in Neural Information Processing Systems*, pp. 153–160, 2007.

[24] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.

[25] T. Tieleman and G. Hinton, "Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning," Technical report, University of Toronto, Toronto, Canada, 2012.

[26] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Computing Surveys*, vol. 51, no. 3, pp. 1–36, 2018.

[27] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: a survey," *Applied Sciences*, vol. 9, no. 20, p. 4396, 2019.

[28] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pp. 253–262, Mumbai, India, December 2017.