Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Intrusion detection methods based on integrated deep learning model

Zhendong Wang[a], *Yaodi Liu*[a,*], *Daojing He*[b], *Sammy Chan*[c]

[a] School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou Jiangxi 341000 China
[b] School of Software Engineering, East China Normal University, Shanghai 200000 China
[c] Department of Electrical Engineering, City University of Hong Kong, Hong Kong 999077 China

ARTICLE INFO

Article history:
Received 18 July 2020
Revised 12 November 2020
Accepted 3 January 2021
Available online 7 January 2021

Keywords:
Deep learning
Deep neural network
Feature learning
Mini-batch gradient descent
Intrusion detection

ABSTRACT

Intrusion detection system can effectively identify abnormal data in complex network environments, which is an effective method to ensure computer network security. Recently, deep neural networks have been widely used in image recognition, natural language processing, network security and other fields. For network intrusion detection, this paper designs an integrated deep intrusion detection model based on SDAE-ELM to overcome the long training time and low classification accuracy of existing deep neural network models, and to achieve timely response to intrusion behavior. For host intrusion detection, an integrated deep intrusion detection model based on DBN-Softmax is constructed, which effectively improves the detection accuracy of host intrusion data. At the same time, in order to improve the training efficiency and detection performance of the SDAE-ELM and DBN-Softmax models, a small batch gradient descent method is used for network training and optimization. Experiments on the KDD Cup99, NSL-KDD, UNSW-NB15, CIDDS-001, and ADFA-LD datasets show that SDAE-ELM and DBN-Softmax integrated deep inspection models have better performance than other classic machine learning models.

1. Introduction

With the rapid development of global informatization, network connections are becoming more and more convenient. People can better enjoy the convenience brought by the rapid development of the network, but at the same time, network security is also increasingly threatened. Intrusion detection technology as a proactive defense mechanism has received more and more attention. The process of identifying the behaviors that attempt to invade, invading or have already occurred is called intrusion detection (Tidjon et al., 2019). The technical core of intrusion detection is to detect whether various behaviors in the network are safe by analyzing the collected network data. However, with the increasing complexity of the network structure and the diversity of intrusion behaviors, existing intrusion detection systems gradually show some drawbacks, such as high false positive and false negative rates, difficult data feature extraction, and low data processing efficiency.

Network-based intrusion detection system (NIDS) (Hamed et al., 2018) and host-based intrusion detection system (HIDS) (Marteau, 2019) are two types of intrusion detection systems. NIDS determines possible intrusions by analyzing network traffic and network protocols. In the past, researchers mostly used pattern matching algorithms

* Corresponding author.
E-mail addresses: wangzhendong@hrbeu.edu.cn (Z. Wang), 1589807038@qq.com (Y. Liu), djhe@sei.ecnu.edu.cn (D. He), eeschan@city.edu.hk (S. Chan).
https://doi.org/10.1016/j.cose.2021.102177

(Kim et al., 2009) to analyze it. The matching features mainly include string features, port features and data packet header features, so feature selection of data is one of the effective methods to improve the reliability and timeliness of NIDS. The current feature selection algorithms mainly include particle swarm optimization (Wei et al., 2019), genetic algorithm (Ghamisi and Benediktsson, 2015), gray wolf algorithm (AI-Tashi et al., 2020), cuckoo algorithm (Usman et al., 2020), etc., but they all have some problems, such as the relatively large randomness of the genetic algorithm, the easy trapping in the local optimum in the gray wolf algorithm, etc.; The earliest intrusion detection system applied to the network is HIDS, and the detection module is installed on the system hard disk. HIDS analyzes the log files and audits extracted system operation data to achieve the purpose of intrusion detection. With the increasing complexity of the network environment, detection methods migrated from the rule-based expert system (IIgun et al., 1995) to machine learning based methods. Currently, commonly used machine learning methods include SVM, DT (Teng et al., 2018), LR (Besharati et al., 2019), neural network (Canedo and Romariz, 2019), deep learning (AI-Qatf et al., 2018) and so on.

Deep learning appeared relatively late. In 2006, Hinton first proposed the concept of deep learning (Hinton et al., 2006), which has strong feature learning capabilities with unsupervised training layer by layer. Therefore, it has been widely used in speech (Tu et al., 2019), image (Liu et al., 2018) and natural language processing (Wang et al., 2020). In recent years, with the increase in the number of intrusions, the increase of data dimensions, and the increasingly perfect deep learning theory, researchers have begun to consider applying deep learning to intrusion detection. In intrusion detection, the autoencoder (Sadaf and Sultana, 2020) replaces high-dimensional input layer neurons with hidden layer labeled neurons to achieve the functions of dimensionality reduction and feature extraction; the intrusion data is preprocessed to generate uniform traffic gray map, and convolutional neural network (Nguyen and Kim, 2020) is sued to extract features in the gray image, thereby expanding the number of training samples. Deep belief network (Zhang et al., 2020) using restricted boltzmann machines for unsupervised pre-training and the obtained results as the initial value of the supervised learning training probabilistic model greatly improves the learning performance. Although the deep learning model has achieved good research results in the field of intrusion detection, in most cases, the performance of the model is only verified on the network-based intrusion detection dataset, without considering whether it has the ability to detect the host-based dataset.

Therefore, this paper fully considers the characteristics of different intrusion detection systems, and proposes integrated deep learning models for different intrusion data sources, in order to fully exploit the advantages of the integrated deep learning models to achieve the best detection effect for different intrusion detection data sources. At present, network-based intrusion detection systems use original network packets as data sources. The main problems are that the network data traffic is noisy, the response is not timely, the signature database is not updated in time, and the timeliness is poor. Since the stacked denoising AutoEncoder (SDAE) can reduce the noise of the data very well and has better robustness, and the extreme learning machine (ELM) has fast learning speed, the integration of SDAE and ELM is proposed. The learning model improves the detection speed on the basis of reducing the noise of network data traffic, and can detect network-based intrusion systems in real time. On the other hand, host-based intrusion detection systems use the log files of each host as the main data source. As the number of log files is limited, some intrusion means and ways will not appear in the log files. Considering that deep belief networks (DBN) can fully dig out the features in the data and the Softmax classifier can better identify multiple attack types, we propose an integrated deep learning model of DBN-Softmax, which has high accuracy and is a better detection method for host-based intrusion detection systems.

Overall, this work has made the following contributions to the intrusion detection domain:

1) Considering that the performance of the SDAE model can still be improved, this paper proposes an integrated deep intrusion detection model based on SDAE-ELM. The SDAE can well reduce the noise of the data and has strong robustness, and can better reduce the noise of the network data traffic. The ELM has the advantage of fast training speed and can realize intrusion detection faster.
2) Considering that the Sigmoid activation function in the DBN model is more suitable for the binary classification problem, this paper proposes a method for building a deep learning framework for deep structure DBN and Softmax classifiers, and designs an integrated deep intrusion detection model based on DBN-Softmax to train the network using pre-training and fine-tuning methods.
3) For different data sets, SDAE-ELM and DBN-Softmax are applied to NIDS and HIDS data sets respectively. This is because the NIDS datasets contain huge data and much noise. The SDAE-ELM model can remove the noise of the NIDS datasets and improve the detection performance and speed. The number of log files in the HIDS dataset is limited, and most intrusions will not be recorded by the host. The DBN-Softmax model has better data mining capabilities, thereby improving the detection performance.
4) In order to verify the effectiveness of the methods and models proposed in this paper, we apply the SDAE-ELM and DBN-Softmax models to the network-based intrusion detection datasets KDD Cup99, NSL-KDD, UNSW-NB15, and CIDDS-001, and the host-based intrusion detection dataset ADFA-LD, respectively, and compare them with various machine learning algorithms and deep learning models using several experimental evaluation metrics.

The rest of the paper is organized as follows. Related works are discussed in Section 2. In Section 3, we present the deep learning and intrusion detection models proposed in this paper. Section 4 describes the intrusion detection data sets. The mathematical details of the intrusion detection models and the evaluation criteria of the models are introduced in Section 5. The performance of the models are evaluated in Section 6. Finally, we draw conclusions and suggest some future work in Section 7.

2. Related works

Intrusion detection has become an important part of network security defense. More and more researchers are focusing on intrusion detection and many studies have been conducted. Through literature review, we found that the current research on intrusion detection mainly includes dataset preprocessing methods, optimization of detection models, and detection technologies in different network environments.

Some researchers start with intrusion detection datasets, and use dimensionality reduction techniques to reduce irrelevant and redundant data before model training to reduce the complexity of intrusion detection system design, and improve its performance. Feature selection and feature extraction are two types of dimensionality reduction techniques. Serpen and Anghaei, (2018) used principal component analysis for feature extraction of data in the Linux operating system, and applied it to a host-based abuse intrusion detection system. In general, the abuse detection system can detect and predict the type of attacks. Tama et al. (HFSTE, 2017) considered that it is difficult to distinguish the boundary between normal and attack types in anomaly detection, with a high false alarm rate. Therefore, by combining three algorithms for feature selection and integration of tree-based classifiers, and reducing the error pruning tree for classification, their method has better detection performance than existing methods. Beulah et al. (Beulah and Punithavathani, 2017) proposed a hybrid method for feature selection. The method selects and combines the best features from different feature selection methods. It can be used for feature compression in any application domain. Although hybrid feature selection and feature extraction technologies can reduce irrelevant and redundant data, reduce the dimensionality of the dataset, and save the training time of the model, most of the current hybrid feature selection and feature extraction technologies mainly consider linear combination of the original data but do not consider the potential relationship within the variables, which may delete more important features during the dimensionality reduction process, thus affecting the training effect of the model.

In addition to focusing on the preprocessing of the dataset, researchers are also interested in how to optimize the classifier model to improve the detection performance of the classifier model. For traditional intrusion detection models, most researchers use swarm intelligence algorithms to optimize the parameters of the classifier model, so as to avoid the model falling into the local optimum and improve the classification performance of the model. Ye et al. (2019) proposed the grasshopper algorithm to search for better SVM kernel parameters. It is based on genetic algorithm and particle swarm algorithm in order to avoid the slow convergence speed and local minimum problems of traditional algorithms. They conducted comparative experiments using MATLAB tools, which showed that the method has superior performance in intrusion detection. Duan et al. (2019) used an improved artificial bee colony algorithm to optimize the initial weights and thresholds of the back-propagation (BP) neural network to avoid the model falling into the local optimum and improve the training speed, and this method has good classification capabilities and high intrusion detection capabilities. In addition to optimizing the parameters of the model, researchers also focused on the characteristics of the model itself and proposed a series of integrated models. Teng et al. (2018) proposed an adaptive collaborative intrusion detection method, which uses decision trees and support vector machines to design objects, roles and agents, and establishes an adaptive scheduling mechanism. This method is more effective than a single support vector machine. Shone et al. (2018) proposed a new asymmetric deep autoencoder (NDAE) for unsupervised feature learning, and stacked it with the random forest algorithm to form a new deep learning classification model S-NDAE. Experimentally tested on the benchmark test datasets KDD Cup99 and NSL-KDD, the model incurred less training time and achieved better training results.

In the traditional network environment, the hybrid feature selection technology and the optimization of detection model have achieved good detection results. With the increasing complexity of network environments, cloud computing and fog computing environments appear, but the detection effects of these methods may be slightly worse. In order to be able to adapt the intrusion detection to cloud and fog computing environments, in recent years, researchers have proposed a series of intrusion detection technologies and architectures. de Araujo-Filho et al, (2020) used generative adversarial networks to realize intrusion detection on cyber-physical systems (CPS) in fog environments. Network attack detection on CPS needs to comply with strict delay requirements, and detect and prevent attacks before the system is threatened. Araujo-Filho atomizes the intrusion detection system to make the computing resources closer to the terminal node, which helps to meet the low-latency requirements, and solves the response speed problem by training the encoder that accelerates the reconstruction loss calculation. Prabavathy et al. (2018) used fog computing to detect network attacks in IoT applications in a distributed manner, and used the OS-ELM algorithm to implement intrusion detection on distributed fog nodes. The distributed architecture of fog computing enables distributed intrusion detection mechanism scalable, flexible, and interoperable. In addition, Wang et al. (2019) proposed an effective feature selection method and SVM classification algorithm to construct a cloud intrusion detection system. Experimental results show that the detection system has achieved good results in detecting intrusion detection in cloud computing networks, but the Libsvm classifier cannot effectively identify some new attacks of the test dataset. Aljamal et al. (2019) proposed a network-based anomaly detection system at the Cloud Hypervisor level, which combines K-means and SVM classification algorithms to improve the accuracy of the anomaly detection system. Distributed denial of service (DDoS) is a common form of attack against cloud computing, and new features of cloud computing (e.g., virtualization and virtual machine migration) also bring challenges to cloud security. Therefore, Ibrahim and Zainal,(2018) proposed an adaptive and distributed intrusion detection model (A-D-CIDS) for cloud computing to detect coordinated attacks and solve the problems caused by the migration of virtual machines in the cloud intrusion detection system. From the current research, the intrusion detection technology and structure of cloud computing and fog computing are mainly distributed detection structure. This structure is

based on the existing algorithm and implements intrusion detection on distributed nodes, which can greatly improve the detection speed of the model and achieve the purpose of real-time detection.

In summary, good progress has been made in hybrid feature selection technology, classifier model optimization, and intrusion detection technology in different network environments. Table 1 shows the comparison of different intrusion detection models used by researchers. Different from the existing intrusion detection models, two integrated deep learning models, SDAE-ELM and DBN-Softmax are proposed in this paper based on the deep learning approach, which fully consider the influence of data sources on the detection performance of the models. These two models are applied to the network-based intrusion detection datasets KDD Cup99, NSL-KDD, UNSW-NB15, and CIDDS-001 and the host-based intrusion detection dataset ADFA-LD, respectively. The intrusion detection datasets used in this paper is relatively complete. We use multiple evaluation indicators such as accuracy, precision, true positive rate, false positive rate, F1-Score, P-R curve, ROC curve and AUC value to evaluate the performance of the proposed models. The evaluation is more scientific and comprehensive.

3. Intrusion detection models

As classic methods in deep learning, SDAE and DBN have achieved better results when applied to shallower models of intrusion detection, but there are certain limitations. The BP algorithm is used in the fine-tuning process. With the increase in the number of layers, it will have problems with sparse gradients and local optimization. In SDAE, the BP algorithm has local minimization and requires multiple iterations to determine the network output weights to affect the learning of the network, while the weights and thresholds of the ELM network need only be calculated once by the least square method to obtain the optimal weights and thresholds, without the need to update through the BP algorithm, the model training speed is faster, and it takes less time. In DBN, in the BP fine-tuning process, the Sigmoid function is used as the activation function of the last layer to treat each category as a two-category process. The Sigmoid activation function has a large amount of calculation, and the BP error division involves division operations and is prone to gradient disappearance. Hence, the deep network training cannot be completed.On the other hand, the Softmax activation function can directly implement multi-class classification, and the output categories are mutually exclusive to avoid the existence of multiple possibilities, which can better solve the problem of the Sigmoid function. Based on this, we propose integrated deep intrusion detection models of SDAE-ELM and DBN-Softmax, and apply them to the host-based data sets and the network-based data set, respectively.

The integrated deep intrusion detection models proposed in this paper are mainly divided into three modules, and the structure is shown in Fig. 1.

Data preprocessing module: Preprocessing operations on the data sets, mainly including feature extraction, data conversion, data normalization, and other operations, are to make the data sets meeting the requirements of the input data. The testing sets and training sets are used for model training and model testing, respectively.

Intrusion detection module: It combines the dimensions of the data after preprocessing to determine the number of input and output nodes of the network model, then determines the entire network structure and training parameters according to the hidden layer and other parameters, uses the training sets to train the model, and saves model for testing after training.

Detection and classification module: The test sets use the saved model for testing and display the detection classification results to the user.

3.1. *Integrated deep intrusion detection model based on SDAE-ELM*

3.1.1. *Stacked denoising autoencoder*

If only one layer of Denoising AutoEncoder is used, the coding ability is relatively limited, whereas SDAE consists of multiple Denoising AutoEncoder for feature extraction, multiple Denoising AutoEncoders are stacked together, and the output of the previous Denoising AutoEncoder is used as the input of the latter Denoising AutoEncoder, and is pre-trained using a layer-by-layer initialization strategy. After the training is completed, the overall network parameters are corrected according to the output error of the last layer of the network. Finally, the classification of the sample is determined by the classifier.

3.1.2. *Denoising autoencoder*

Denoising AutoEncoder(DAE) (Kachuee et al., 2019) is a type of extension of AutoEncoder(AE) (Bengio et al., 2013). The Denoising AutoEncoder is based on the AutoEncoder, the training data adds noise, the DAE can remove the noise from the input data during the learning process to obtain data that is not contaminated by noise. This learning method is more generalized than the general AutoEncoder. The DAE schematic is shown in Fig. 2 below.

In Fig. 2, x is the original input data, x' is the damaged data, y is the feature after x' encoding, $\tilde{x}$ is the output obtained by y decoding. The reconstruction error expression is:

$$L(x, g(f(x'))) = \|x - g(f(x'))\|^2 \tag{1}$$

The AutoEncoder uses the reconstruction error to represent the training effect during the training process, and requires the minimum reconstruction error to ensure the maximization of the common features. During the training process of the DAE, $f(\cdot)$ is performed on the damaged data x' feature mapping, therefore, compared to AutoEncoder, DAE increases the robustness of features during training.

The objective function of Denoising AutoEncoder is given by:

$$J(W, b) = \left[\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\|x^{(i)} - g\left(f\left(x^{(i)}\right)\right)\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_{l-1}}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \tag{2}$$

where, $x^{(i)}$ represents the ith sample input of the Denoising AutoEncoder, $W_{ji}^{(l)}$ denotes the connection weight between the ith unit of the lth layer and the $j+1th$ unit of the $l+1th$ layer,

Table 1 – Comparison of intrusion detection systems.

		Model	Dataset	Advantages	Disadvantages	Evaluation index		
						Accuracy (%)	TPR (%)	FPR(%)
Data preprocessing	feature extraction	(Serpen and Angha ei, 2018)	ADFA-LD	Higher detection effectiveness; Feature extraction using Eigentraces	Use of a single data set; No comparison experiment	/	Binary class: 99.9 Multi class: 99.9	Binary class:0.2 Multi class:0.2
	feature selection	(HFSTE, 2017)	NSL-KDD	A mixed group intelligence algorithm for feature selection, which yields a better subset of features. Integrating hybrid feature selection techniques and multiple classifiers	Use of a single data set; Single evaluation indicator	99.7	/	/
		(Beulah and Punithavathani, 2017)	NSL-KDD	The feature selection method can be applied to any field; Able to choose the best attributes;	Use of a single data set	79.66	/	/
Detection model optimization	Parameter optimization	(Ye et al., 2019)	KDD Cup99	Speed up the convergence speed to avoid the model falling into the local optimum;	Use of a single data set; Single evaluation indicator	97.838	/	/
		(Duan et al., 2019)	NSL-KDD	Avoid the model falling into the local optimum; Improved the detection ability of the model	Use of a single data set	98.12	97.23	/
	Integrated model	(Teng et al., 2018)	KDD Cup99	Shorten the training time of the model; Improve the detection effect of the model; Propose an adaptive intrusion detection model	Use of a single data set; Single evaluation indicator	89.02	/	/
		(Shone et al., 2018)	KDD Cup99 NSL-KDD	Building a new NDAE; Model training time is shorter; The classification results are more refined	Only DBN model was used for experimental comparison. The effect was not validated on the newer data set	97.85 85.42	/	2.15 14.58
Different network environment	Fog computing environment	(de Araujo-Filho et al., 2020)	SWaT WADI NSL-KDD	Improve detection rate; The detection speed is at least 5.5 times faster than traditional IDS	Only the reconstruction error is considered	The experimental results are expressed in graphical form		
		(Prabavathy et al., 2018)	NSL-KDD	Improve the detection speed of the model; Reduced the false alarm rate of the model	Use of a single data set	Binary class: 97.36 Multi class: 96.54	Binary class: 97.72	Binary class: 0.37
	cloud computing environment	(Wang et al., 2019)	KDD Cup99 NSL-KDD	Reduce the dimension of data set and shorten the training time; Maintaining algorithm performance	Older data set; Multiple classifications were not considered	99.85 98.64	/	/
		(Aljamal et al., 2019)	UNSW-NB15	Use newer intrusion detection data sets; Cluster first and then classify; K-means has better accuracy	Use of a single data set; SVM accuracy rate is low	Cluster16: 84.6 Cluster32: 84 Cluster64: 84.7	/	/
		(Ibrahim and Zainal, 2018)	Simulate a cloud environment to generate new data sets	Data sets are more representative; The detection effect of the model is better	No comparison experiments were made with other classical algorithms	Normal: 98.6 DoS/DDoS: 98	/	0.07 0.15

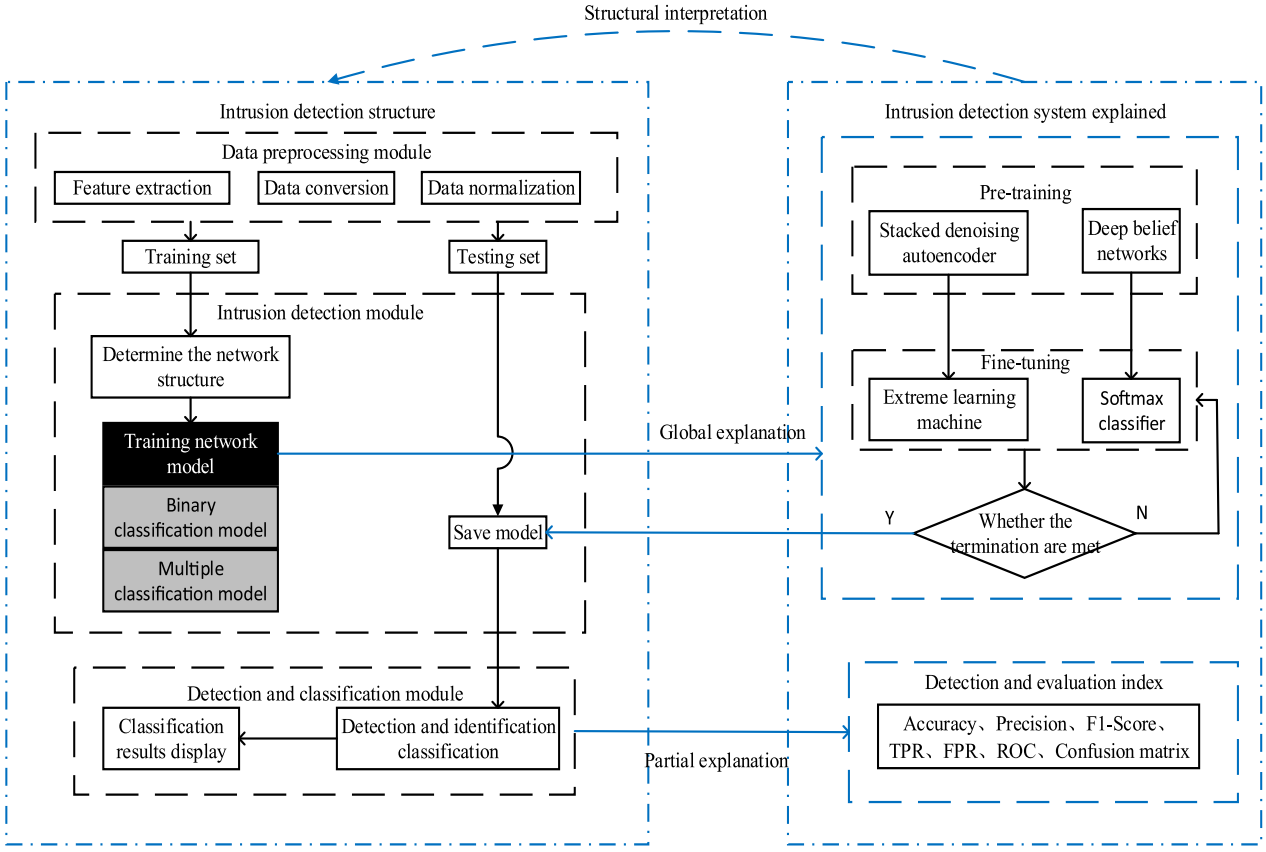


Fig. 1 – Intrusion detection model structure diagram.

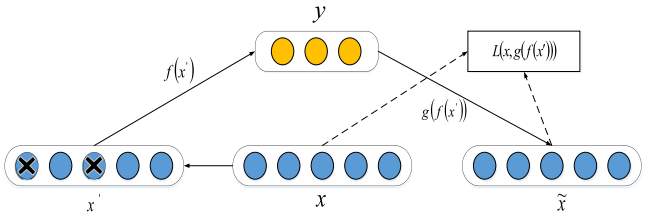


Fig. 2 – Schematic diagram of DAE.

n represents the number of network layers, n_l represents the number of network layers, s_l represents the number of neurons in the lth layer, λ is to reduce the weight amplitude and prevent the regularization coefficient of overfitting.

After determining the optimal objective function of the Denoising AutoEncoder, the error BP algorithm is used to fine-tune the parameters of each layer of the Denoising AutoEncoder, namely:

$$W = W - \alpha \frac{\partial}{\partial W} J_D(W, b) \tag{3}$$

$$b = b - \alpha \frac{\partial}{\partial b} J_D(W, b) \tag{4}$$

where α is the learning rate.

3.1.3. *Extreme learning machine*

Extreme Learning Machine (ELM) is a Single-hidden Layer Feed Forward Network (SLFNS) proposed by Huang (Huang et al., 2014). It can solve the problems of low efficiency and complicated parameters in the BP algorithm. ELM completes the training by the minimum error function, the main feature is the connection weight between the input layer and the hidden layer, and the threshold of the hidden layer node only needs to be calculated once by the least square method, instead of using the traditional iterative method to calculate the parameters. This training method greatly shortens the training time and has a strong generalization ability.

Given N training sample data (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ is the sample input data, $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m$ is the sample output value, for a single-layer network with L hidden layer nodes, and excitation function $g(x)$, then its network output is

$$y_j = \sum_{i=1}^{L} \beta_i g_i\left(\omega_i x_j + b_i\right),\ j = 1, 2, \ldots, N \tag{5}$$

where β_i is the weight vector between the i-th hidden layer node and the output layer node; ω_i is the weight vector between the i-th hidden layer node and the input layer node; b_i is the i-th hidden layer offset; y_j is the output value of the network.

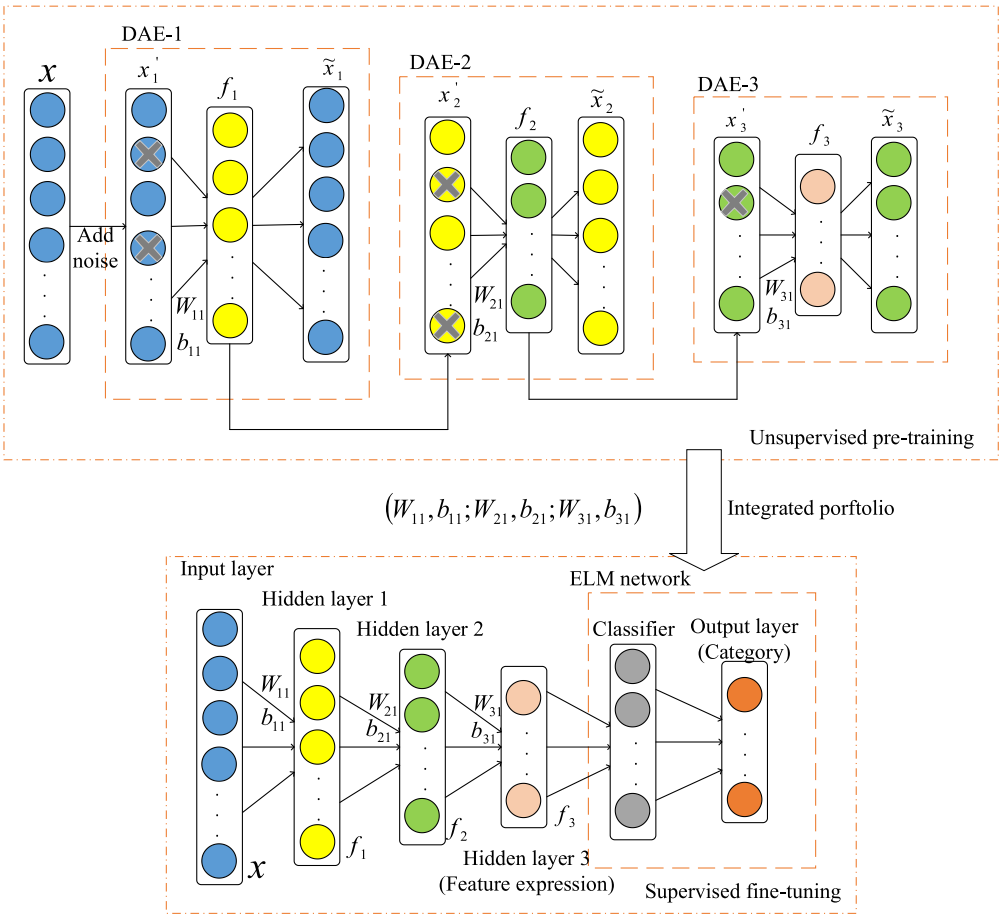


Fig. 3 – SDAE-ELM model structure diagram.

When the activation function can approach any N samples with zero error, that is $\sum_{i=1}^{N} \|y_i - t_i\| = 0$, there is

$$t_j = \sum_{i=1}^{L} \beta_i g_i\left(\omega_i x_j + b_i\right),\ j = 1, 2, \ldots, N \tag{6}$$

The above formula can be expressed as

$$H\beta = T \tag{7}$$

Among them,

$$H = \begin{bmatrix} g_i(w_1 x_1 + b_1) & \cdots & g_i(w_L x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g_i(w_1 x_N + b_1) & \cdots & g_i(w_L x_N + b_L) \end{bmatrix} \beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_L \end{bmatrix} T = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}$$

where H is the output matrix of the hidden layer; β is the output weight; T is the desired output vector.

The connection weights of the hidden layer and the output layer can be obtained by solving the least squares solution of the following system of equations:

$$\min_{\beta} \|H\beta - T\| \tag{8}$$

The least squares solution of the system of equations is:

$$\hat{\beta} = H^{+}T \tag{9}$$

where H^{+} is the generalized inverse matrix of the hidden layer output matrix H.

3.1.4. *SDAE-ELM model and detection procedures*

In this paper, the SDAE-ELM integrated deep model is used as the network-based intrusion detection system. The experimental data sets are all network-based data sets. The model first uses SDAE to learn the features of the data sets, and then, the features learned by the SDAE are input into the ELM algorithm for fine-tuning, to get the trained SDAE-ELM model. Finally, the test sets data are input into the SDAE-ELM model to complete the intrusion detection. The structure of the SDAE-ELM model is shown in Fig. 3. The SDAE-ELM intrusion detection flowchart is shown in Fig. 4.

The specific steps of SDAE-ELM intrusion detection are as follows:

Step 1: Preprocess the intrusion detection data sets, which mainly includes high-dimensional data feature mapping and data normalization processing.

Step 2: SDAE-ELM model training:

1) Initialize model parameters and determine the structure of the network model;
2) Unsupervised algorithm is used to train the first layer of DAE, and the reconstruction error of reconstructed sample

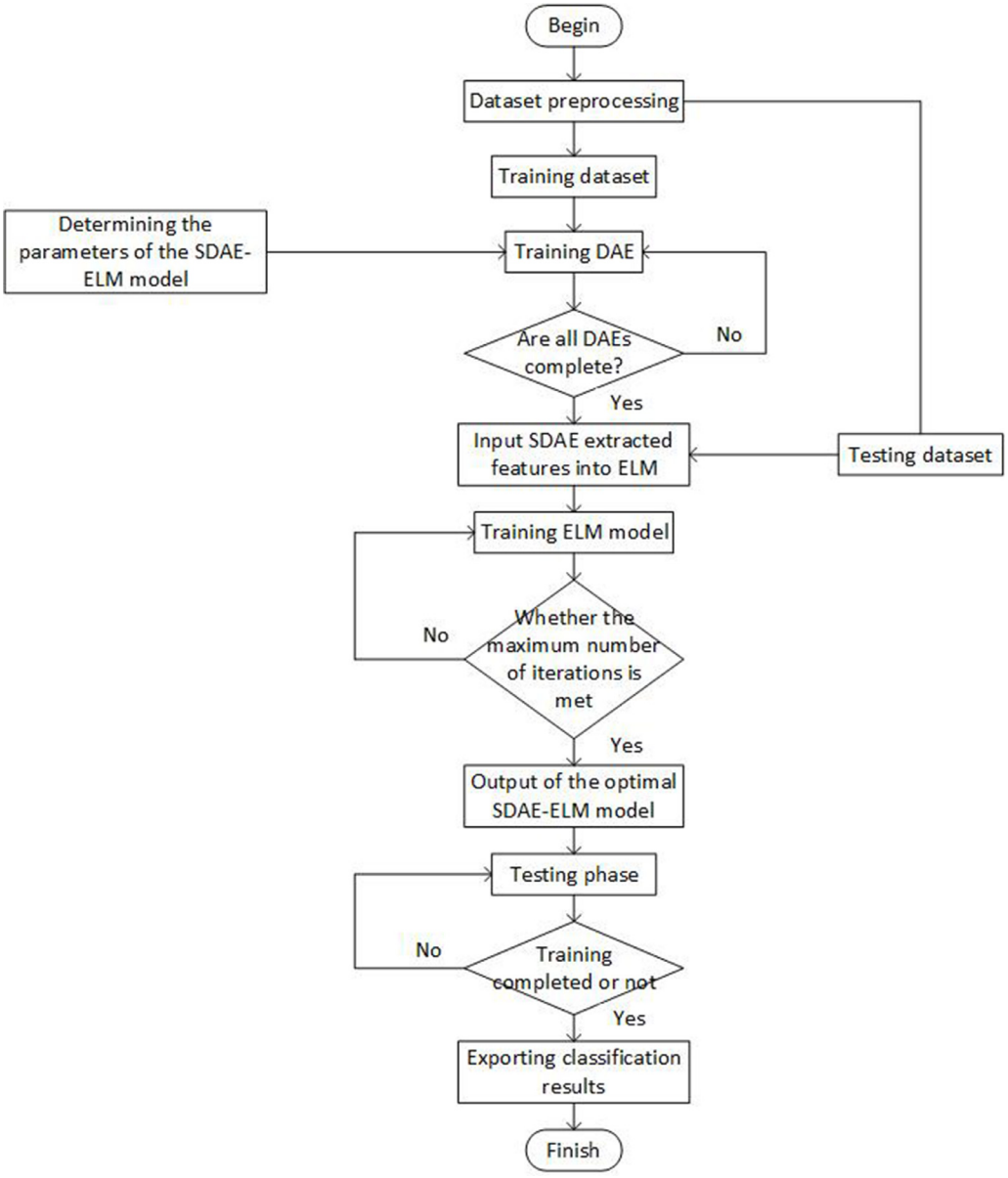


Fig. 4 – SDAE-ELM intrusion detection flowchart.

is controlled within a certain range, its output is used as the input of the next layer of DAE;

3) The output of the above layer of DAE is used as input, and the unsupervised algorithm is used to train the DAE of this layer, so that its reconstruction error is controlled within a certain range;
4) Repeat step 3) until all DAE training is completed;
5) ELM algorithm is used to learn the features extracted by DAE, and the optimal weight and threshold value of the model are determined by the least square method of one order until the specified training times are reached.

Step 3: SDAE-ELM model test, input the test data sets into the trained SDAE-ELM model, and then obtain the classification result of each data set.

3.2. *Integrated deep intrusion detection model based on DBN-Softmax*

3.2.1. *Deep belief network*

DBN is a multi-layer perceptron neural network formed by stacking multiple Restricted Boltzmann machine (RBM) and a layer of BP neural network. The DBN training process includes two parts: pre-training and fine-tuning. The pre-training mainly adopts the unsupervised layer-by-layer training method to train the RBM parameters of each layer. The output of the hidden layer of the low-level RBM is used as the input of the visible layer of the high-level RBM. Abstract feature parameters can be extracted from the original signal data; the BP neural network is used in the fine-tuning phase. The difference between the actual output and the label information is used as a measurement error, and the error is propagated back layer by layer to fine-tune the weight and offset of the

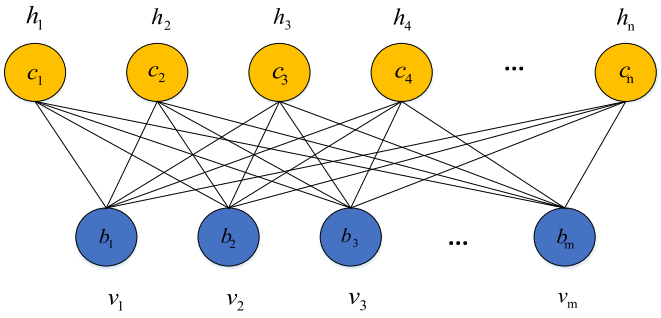


Fig. 5 – Schematic diagram of the restricted Boltzmann Machine.

entire DBN. After multiple iterations, the optimal parameters of the entire DBN can be obtained.

3.2.2. *Restricted Boltzmann machine*

Restricted Boltzmann Machine (Chen et al., 2019) is an energy-based model, which can also be regarded as a special type of Markov random field. Generally, RBM can represent the relationship between random variables. The Restricted Boltzmann Machine is composed of two layers, the visible layer v and the hidden layer h. As can be seen from Fig. 5, the visible layer and the hidden layer of the Restricted Boltzmann Machine are connected with each other (no connection in the layer), the output of the hidden layer unit can obtain the higher-order correlation of the visual unit, that is, the characteristics of the input data.

Let the visible layer unit be $v = \{v_1, v_2, \ldots, v_i\}$, the hidden layer unit be $h = \{h_1, h_2, \ldots, h_j\}$, and the internal parameter vector be $\theta = \{w, a, b\}$, where a and b are the offset value of the visible layer and the hidden layer, w is the weight vector between the visible layer and the hidden layer. Then the RBM energy-based probability model is distributed as follows:

$$p(v, h, \theta) = \frac{1}{Z(\theta)} exp(-E(v, h, \theta)) \tag{10}$$

where $Z(\theta)$ is the regularization factor, given by the sum of the energy functions associated with all visible and hidden units:

$$Z(\theta) = \sum_{v} \sum_{h} exp(-E(v, h, \theta)) \tag{11}$$

The system energy function of RBM can be obtained from the above formula as follows:

$$E(v, h, \theta) = -\sum_{i=1}^{D} a_i v_i - \sum_{j=1}^{F} b_j h_j - \sum_{i=1}^{D} \sum_{j=1}^{F} w_{ij} v_i h_i = a^T v - b^T h - v^T W h \tag{12}$$

where w_{ij} represents the weight between the visible layer unit i and the hidden layer unit j, a_i and b_j are the bias values of the visible layer and hidden layes units, respectively.

In the classification task, the output of RBM is 0 or 1, and the Sigmoid activation function is usually used. At this time, the probability of h and v is calculated as follows:

$$P\left(h_j = 1|v\right) = sigmoid\left(a_i + \sum_i v_i w_{ij}\right) \tag{13}$$

$$P\left(v_j = 1|h\right) = sigmoid\left(b_j + \sum_j h_j w_{ij}\right) \tag{14}$$

In the process of RBM training, Eqs. (13) and (14) are its core. When there is input in the visible layer, the mapping value of the input value from the visible layer to the hidden layer is first calculated by Eq. (13), and the mapping value is input into the Eq. (14) to recalculate the probability of the visible layer. The error between the input data and the reconstructed data is calculated and used to adjust the network parameters, so that the error is reduced to a minimum. At this time, the output data obtained by the hidden layer can be used to represent the visible layer input data. This entire process is the extraction of data features by RBM.

Because Gibbs sampling is time consuming in high-dimensional data features, Hinton proposed Contrast Divergence algorithm(CD-K) (Hinton, 2002) in 2002 to learn RBM. The main steps of the Contrast Divergence method are shown in Algorithm 1.

3.2.3. *Softmax classifier*

The principle of Softmax classifier (Zeng et al., 2014) is very simple. It is an extension of Logistic Regression (LR). The biggest difference between the two is that LR category labels can only take two, while Softmax increases the possibility of multi-category labels, which is more suitable for multi-class classification problems. The Softmax classifier can map input vectors from N-dimensional space to categories, and output the classification results in the form of probabilities. The probability formula is:

$$h_\theta\left(x^{(i)}\right) = \begin{bmatrix} p\left(y^{(i)} = 1\right)|x^{(i)}; \theta \\ p\left(y^{(i)} = 2\right)|x^{(i)}; \theta \\ \vdots \\ p\left(y^{(i)} = K\right)|x^{(i)}; \theta \end{bmatrix} = \frac{1}{\sum_{k=1}^{K} e^{\theta_k^T x}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_K^T x^{(i)}} \end{bmatrix} \tag{15}$$

where $p(y^{(i)} = K)|x^{(i)}; \theta$ represents the probability that $x^{(i)}$ belongs to category K, and the sum of all elements in the vector

Algorithm 1 – CD-k algorithm steps.

CD-K algorithm of RBM
Input: $S = \{x_1, x_2, \ldots, x_l\}$ training data,
$v_i(i = 1, 2, \ldots, n)$,$h_j(j = 1, 2, \ldots, m)$ are the visible layer unit and hidden layer unit, respectively. Learning rate is γ, CD algorithm parameter k
Output: Model parameters $\theta = \{w, a, b\}$
1. Initialize model parameters randomly $\theta = \{w, a, b\}$
2. for each of the training samples x_l in S, do
$v^0 \leftarrow v$
For t = 0,1,...,k-1 do
For i = 1,...,n do calculate $h_i^t \sim P(h_i|v^{(t)})$ according to Eq. (13)
For j = 1,...,m do calculate $v_j^{t+1} \sim P(v_j|h^{(t)})$ according to Eq. (14)
For i = 1,...,n,j = 1,...,m
Calculate $\Delta w_{ij} = \gamma \times (P(h_j = 1|v^{(0)})v_j^{(0)} - P(h_j = 1|v^{(k)})v_i^{(k)})$, and update the weight $w_{ij} = w_{ij} + \Delta w_{ij}$
Calculate $\Delta b_j = \gamma \times (v_i^{(0)} - v_i^{(k)})$, and update $b_j = b_j + \Delta b_j$
Calculate $\Delta a_i = \gamma \times (P(h_j = 1|v^{(0)}) - P(h_j = 1|v^{(k)}))$, and update $a_i = a_i + \Delta a_i$
end

is equal to one. For the training sample $x^{(i)}$, we choose the K corresponding to the maximum probability as the final classification result, and the parameter θ is obtained by the cost function, which used in this paper is:

$$(\theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=1}^{K} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{k=1}^{K} e^{\theta_k^T x^{(i)}}}\right] \quad (16)$$

where $1\{\cdot\}$ is an indicative function, when the value is true, it is equal to 1, and when the value is false, it is equal to 0. If $J(\theta)$ is minimized, the parameter θ can be obtained.

3.2.4. *DBN-Softmax model and detection procedures*

In DBN, each layer is a Restricted Boltzmann Machine, that is, the entire network is regarded as a stack of several RBMs. After unsupervised layer-by-layer training, the BP algorithm is used to train the entire network. After the network training, the DBN model uses the Sigmoid classifier to classify the data. However, in the process of data processing, the output results of the Sigmoid classifier are independent of each other, and may have multiple results in parallel. For intrusion detection, we hope to be able to identify the type of intrusion. Therefore, we consider using the Softmax classifier instead of the original Sigmoid classifier. Because the output of the Softmax classifier is interrelated, it can clarify the type of data and improve the accuracy of intrusion detection. The model structure of DBN-Softmax is shown in Fig. 6. The DBN-Softmax intrusion detection flowchart is shown in Fig. 7.

The specific steps of DBN-Softmax intrusion detection are as follows:

Step 1: Preprocess the intrusion detection data set, use the bag of words model to process the data set and normalize the processed data.

Step 2: DBN-Softmax model training:

1) Initialize the model parameters and determine the structure of the network model;
2) The first layer of RBM is trained by unsupervised algorithm. The input of the hidden layer and the output of the visible layer are calculated by Eqs. (13) and (14), and the output of the visible layer is taken as the input of next layer RBM;
3) The output of the upper layer RBM is used as the input of the RBM, and the unsupervised algorithm is used to train the RBM of this layer;
4) Repeat step 3) until all RBM training is completed;
5) Use the BP algorithm and learn the features extracted by the RBM through the Softmax classifier, and use the BP algorithm to adjust the weights and thresholds of the network to reduce the prediction error of the output, so that the results approach the predicted output until the specified training time is reached.

Step 3: DBN-Softmax model test, input the test data set into the trained DBN-Softmax model, and then obtain the classification result of each data set.

3.3. *Model complexity analysis*

Suppose the number of training samples is m, the size of mini-batch samples is *batchsize*, the number of iterations l, the total number of hidden layers is N, the number of neurons in the input and output layers are n_1 and n_3, the number of neurons in the hidden layer is n_{2j}, j is the jth hidden layer. In the original SDAE model, for the SDAE model with a single hidden layer, $n_1 * n_2$ and $n_2 * n_3$ matrix multiplication operations should be carried out respectively in the pre-training stage. Therefore, the time complexity of calculating a sample in the pre-training is $o(n_1 * n_2 + n_2 * n_3)$, and the time complexity of calculating a sample in the adjustment stage and the pre-training stage is the same as $o(n_1 * n_2 + n_2 * n_3)$. SDAE model of the whole time complexity is $o(m * l * 2 * (n_1 * n_{21} + \sum_{j=1}^{N-1} n_{2j} * n_{2j+1} + n_{2N} * n_3))$. SDAE-ELM model on the basis of SDAE was improved, only need to feature extraction and fine-tuning, but it uses mini-batch training, so the SDAE-ELM model of the whole time complexity is $o(batchsize * l * (n_1 * n_{21} + \sum_{j=1}^{N-1} n_{2j} * n_{2j+1} + n_{2N} * n_3))$. Original DBN and SDAE model training process is basically the same, so the DBN model of the whole time complexity is $o(m * l * 2 * (n_1 * n_{21} + \sum_{j=1}^{N-1} n_{2j} * n_{2j+1} + n_{2N} * n_3))$. Compared with DBN, DBN-Softmax only uses the Softmax classifier to replace the original BP classifier, so the time complexity of DBN-Softmax and DBN is basically the same, but DBN-Softmax uses mini-batch training, so the DBN-Softmax overall time complexity is $o(batchsize * l * 2 * (n_1 * n_{21} + \sum_{j=1}^{N-1} n_{2j} * n_{2j+1} + n_{2N} * n_3))$.

4. Data sets description

In order to verify the detection capabilities of the SDAE-ELM model for network-based intrusion detection system, this paper not only performs intrusion detection on the older intrusion detection data sets KDD CUP99 and NSL-KDD, but also on the newer intrusion detection data sets UNSW-NB15 and CIDDS-001 to verify whether the model has the ability to detect new types of network attacks. To verify the detection capability of the DBN-Softmax model for host-based intrusion

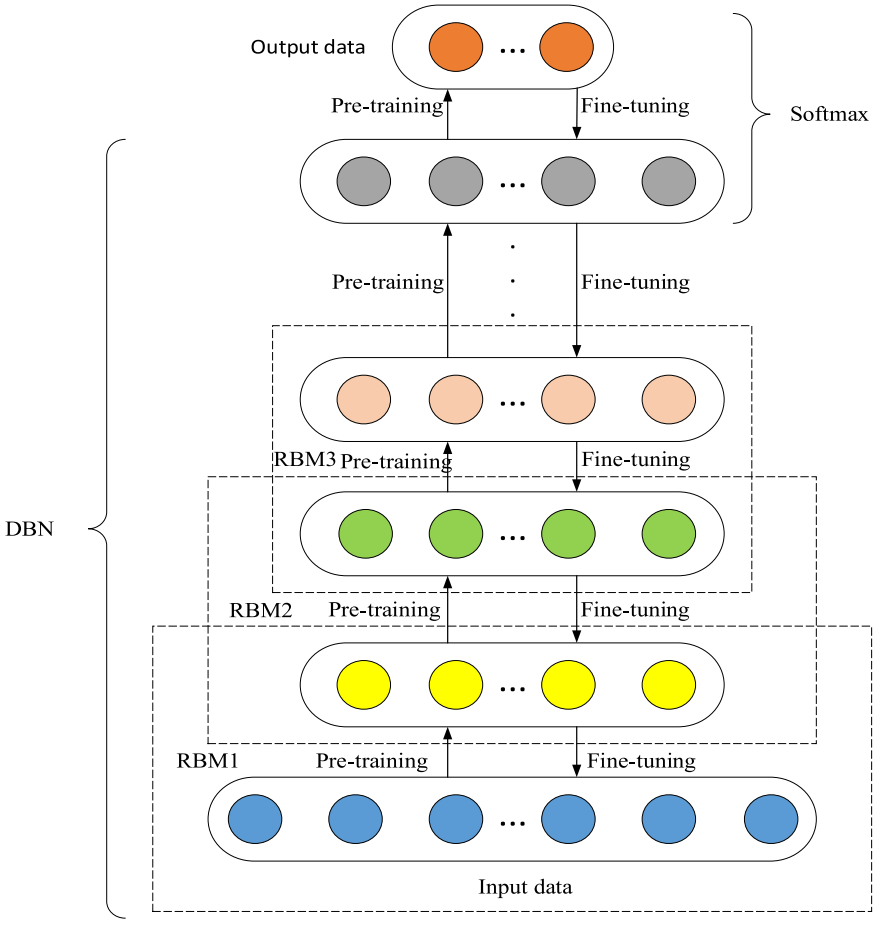


Fig. 6 – DBN-Softmax model structure.

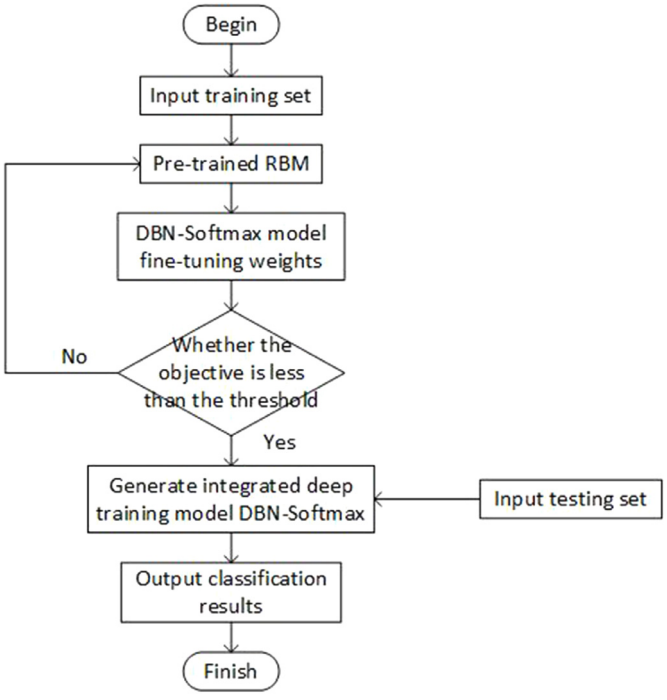


Fig. 7 – DBN-Softmax intrusion detection flowchart.

Table 2 – Training and testing connection records of KDD Cup99 and NSL-KDD.

Attack category	Description	Data instances-10% data		Data instances-20% data	
		KDD Cup99		NSL-KDD	
		Train	Test	Train	Test
Normal	Normal connection records	97,277	60,592	13,357	9690
Probe	Obtaining detailed statistics of system and network configuration details	4107	4166	2289	2421
DoS	Attacker aims at making network resources down	391,438	229,825	9234	7435
U2R	Obtaining the root or super-user access on a particular computer	52	228	11	200
R2L	Illegal access from remote computer	1126	16,189	209	2754
Total		494,000	311,000	25,100	22,500

detection system, we verify it detection ability on the ADFA-LD data set.

4.1. *Network-based intrusion detection data sets*

(1) KDD Cup99 (KDD, 2020): This data set comes from the intrusion detection assessment project of DARPA in 1998. All the network data comes from a simulated US Air Force LAN. Many simulated attacks are added to the network. The training data for the experiment is 7 weeks of network traffic, this network traffic contains about 5 million network connections; the experimental test data is 2 weeks of network traffic, including about 2 million network connections. The data set has two forms, a complete data set and 10 percent dataset. The dataset contains forty-one attributes and one category label. There are five categories of category labels, namely: Normal, Probe, DoS, U2R, R2L. Table 2 describes the KDD Cup99 dataset in detail.

(2) NSL-KDD (NSL-KDD, 2020): This data set is an improvement of the KDD Cup99 data set. The redundant data and duplicated records in the KDD Cup99 data set are deleted. The data set includes the complete data set and the 20 percent data set, and is more suitable for misuse detection than KDD Cup99. Table 2 details the NSL-KDD data set.

(3) UNSW-NB15 (UNSW-NB15, 2020): This data set is generated by the network traffic data collected by the Australian Security Laboratory in 2015. It is a comprehensive network attack traffic data set, including a training data set and test data set. It consists of one normal flow and nine abnormal flows. The data flow is described by forty-two characteristics, plus the final label. The detailed description of the UNSW-NB15 data set is shown in Table 3.

(4) CIDDS-001 (CIDDS-001, 2020): This data set is based on tag streams and used to evaluate anomaly intrusion detection systems. Data from OpenStack and External servers was generated by simulating small businesses. The data set consists of three log files (attack log, client configuration and client log). OpenStack and External servers respectively captured 3.12 million and 60,000 network traffic, including ten characteristic attributes and one category label. Table 4 describes CIDDS-001 data set in detail on External.

4.2. *Host-based intrusion detection data set*

ADFA-LD(Linux Dataset) (ADFA-LD, 2020): This data set is a set of host-level intrusion detection data released by the Australian National Defense Academy, which is a data set that the intrusion event system calls the system sequence (single process, system call api in a time window). The data set mainly contains three types of data. Tables 5 and 6 detail the ADFA-LD data set.

For the network-based intrusion detection data sets, the original data sets cannot be directly inputted into the network model for intrusion detection. We need to preprocess the data sets in advance. The preprocessing includes two steps: (1) Covert the symbolic features in the training and testing sets into numerical representations. (2) Convert category labels to numeric representation. Data set preprocessing is described in detail in Tables 7–9.

For host-based intrusion detection data set, because this data set is generated by the system call time series, we cannot directly input the original data into the detection model. We need to preprocess the data set. The preprocessing mainly includes 2 steps: (1) Use the Bag of Words(BOW) (Bahmanyar et al., 2015) to process the data set. The BOW is mainly used to process the text data set. It does not consider the contextual relationship between the words in the text, only the words of the weight. The weight is related to the frequency of words appearing in the text. So we need to use the BOW to characterize the ADFA-LD data set and convert it into a data set that can be processed by the neural network. (2) Convert category labels to numerical representation. Data set preprocessing is described in detail in Table 10.

We randomly select 20,000 connections from the network-based data sets, and randomly select 2000 connections from the host-based data set and visualize it using the t-SNE method (Van der Maaten and Hinton, 2008). The effectiveness of t-SNE can be seen from Fig. 8(a), the horizontal axis represents the distance and the vertical axis represents the density. For the points of greater similarity, the distance of t distribution in the low-dimensional space needs to be slightly smaller; for the points of low similarity, the distance of t distribution in the low-dimensional space needs to be longer. This just meets our needs, that is, points within the same cluster (closer distance) are more closely aggregated, and points be-

Table 3 – Training and testing connection records of UNSW-NB15.

Attack_cat	Description	Train	Test
Normal	Normal connection records	56,000	37,000
Backdoor	Technology to gain access to programs or systems by bypassing security controls	1746	583
Analysis	An intrusion method to infiltrate web applications through ports, emails, and web scripts	2000	677
Fuzzers	An attack method that attempts to discover security vulnerabilities in a program, operating system, or network, by entering a large amount of random data to crash	18,184	6062
Shellcode	An attack method that controls the target machine by sending code that exploits specific vulnerabilities	1133	378
Reconnaissance	An attack method to collect computer network information to escape security control	10,491	3496
Exploit	A piece of code that controls a target system by triggering a vulnerability (or several vulnerabilities)	33,352	11,132
DoS	An attack method that directly or indirectly exhausts the resources of the attacked object, so that the target computer or network cannot provide normal service or resource access	12,264	4089
Worms	A malicious computer virus that actively spreads through the network	130	44
Genertic	A technique that uses a hash function to collide each block cipher regardless of the configuration of the block cipher	40,000	18,871
Total		175,300	82,332

Table 4 – Training and testing connection records of CIDDS-001-External.

Class	Train	Test
Normal	130,000	4240
Attacker	10,000	2260
Suspicious	430,000	7911
Unknown	70,000	7923
Victim	8000	907
Total	648,000	23,241

Table 5 – Number of system call traces in different category of ADFA-LD.

	Traces	System Calls
Training data	833	308,077
Validation data	4372	2,122,085
Attack data	746	317,388
Total	5951	2,747,550

Table 6 – ADFA-LD dataset attack type.

Attack	Description	Trace Count
Adduser	Client poisoned executable file	91
Hydra_FTP	FTP brute-force cracking	162
Hydra_SSH	SSH brute-force cracking	176
Java_Meterpreter	TikiWiki vulnerability attack	124
Meterpreter	Client poisoned executable file	75
Web_shell	PHP remote file containing vulnerability	118

tween different clusters (longer distance) are more alienated. From Fig. 8(b), 8(c), and 8(d), we can see that all the data sets are non-linearly separable. And from the connection records, we can see that compared to the KDD Cup99 data set, UNSW-NB15 and ADFA -LD data set is more complicated.

5. Experimental setup

The experiments reported in this paper were conducted under the environment of Intel Core i7 dual-core CPU, with 2.5 GHz main frequency, 8GB memory, and Windows 10 operating system. The simulation experiments were carried out using MATLAB R2017b.

5.1. *Hyperparameter setting*

Since the SDAE-ELM and DBN-Softmax models are parameterized, the selection of their parameters greatly affects the classification performance of the models. In this paper, the optimal parameters and network topology of the models are determined only on the KDD Cup99 dataset, and these parameters and network topology are applied to the NSL-KDD, UNSW-NB15, CIDDS-001, and ADFA-LD data sets. We chose a shallow SDAE-ELM and DBN-Softmax model for the experiments. The SDAE-ELM model includes three-layer, an input layer, a hidden layer, and an output layer. For the KDD Cup99 dataset, the input layer contains 41 neurons, the hidden layer can contain 32, 64, 128, 256 and 512 units, and the output layer contains 1 unit, used to distinguish the normal connection of the network and the type of attacks. The input layer and the hidden layer, the hidden layer and the output layer all adopt the full connection mode. For each parameter of the hidden layer, we run 100 times, when the number of hidden layers is 256, the model can detect the most attacks at this time; and when the number of

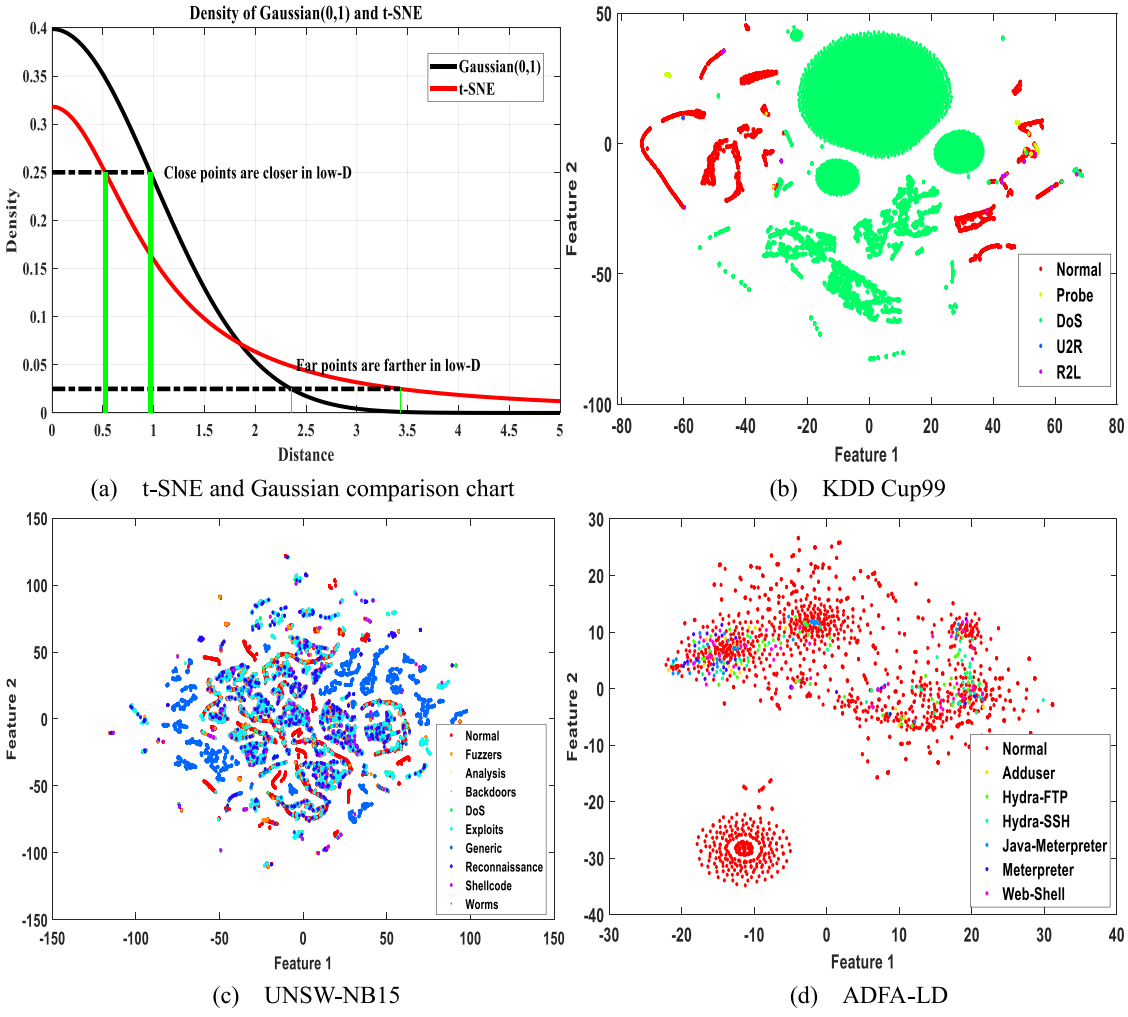


Fig. 8 – Dataset visualization.

Table 7 – Pretreatment of KDD Cup99 and NSL-KDD.

Symbolic features	protocol_type	tcp=1,udp=2,icmp=3
	service	auth=1,bgp=2,courier=3,cenet_ns=4,ctf=5,daytime=6,discard=7,domain=8,domain_u = 9,echo=10, eco_i = 11,ecr_i = 12,efs=13,exec=14,finger=15,ftp=16,ftp_data=17,gopher=18,hostname=19,http=20, http_443=21,http_8001=22,imap4=23,IRC=24,iso_tsap=25,klogin=26,kshell=27,ldap=28,link=29, login=30,mtp=31,name=32,netbios_dgm=33,netbios_ns=34,netbios_ssn=35,netstat=36,nnsp=37, nntp=38,ntp_u = 39,other=40,pm_dump=41,pop_2 = 42,pop_3 = 43,printer=44,private=45,red_i = 46, remote_job=47,rje=48,shell=49,smtp=50,sql_net=51,ssh=52,sunrpc=53,supdup=54,systat=55, telnet=56,tftp_u = 57,tim_i = 58,time=59,urh_i = 60,urp_i = 61,uucp=62,uucp_path=63,vmnet=64, whois=65,X11=66,Z39_50=67
	flag	OTH=1,REJ=2,RSTO=3,RSTOSO=4,RSTR=5,S0=6,S1=7,S2=8,S3=9,SF=10,SH=11
Category label	label	Normal=1,Probe=2,DoS=3,R2L=4,U2R=5

hidden layers increases from 256 to 512, the model's detection efficiency will deteriorate due to overfitting. Therefore, in the following experiments, the number of hidden layer neurons will be set to 256. In the SDAE-ELM model with fewer parameters, the model can achieve better classification results at the early stage of iteration. In this model, we do not consider the final iteration number of the model; applying the above parameters to the DBN-Softmax model. The model can obtain a better detection effect at the early stage of the iteration. In order to determine the final number of iterations, the remaining algorithms are optimized. After 100 times of model training, a better classification effect can be achieved. Increasing the training number of the model again does not significantly improve the classification effect of the model. Finally, we determine that the training time of the model is 100. The learning rate has a great influence on the training speed of the model.

Table 8 – Pretreatment of UNSW-NB15.

Symbolic features	state	ACC=1,CLO=2,CON=3,ECO=4,FIN=5,INT=6,PAR=7,REQ=8,RST=9,URN=10,no=11
	proto	3pc=1,a/n = 2,aes-sp3-d = 3,any=4,argus=5,aris=6,arp=7,axe.25=8,bbn-rcc=9,bna=10,br-sat-mon=11, cbt=12,cftp=13,chaos=14,compaq-peer=15,cphb=16,cpnx=17,crtp=18,crudp=19,dcn=20,ddp=21, ddx=22,dgp=23,egp=24,eigrp=25,emcon=26,encap=27,etherip=28,fc=29,fire=30,ggp=31,gmtp=32, gre=33,hmp=34,i-nlsp=35,iatp=36,ib=37,icmp=38,idpr=39,idpr-cmtp=40,idrp=41,ifmp=42,igmp=43, igp=44,il=45,ip=46,ipcomp=47,ipcv=48,ipip=49,iplt=50,ipnip=51,ippc=52,ipv6=53,ipv6-frag=54, ipv6-no=55,ipv6-opts=56,ipv6-route=57,ipx-n-ip=58,irtp=59,isis=60,isoip=61,isotp4=62, kryptolan=63,l2tp=64,larp=65,leaf-1 = 66,leaf-2 = 67,merit-inp=68,mfe-nsp=69,mhrp=70,micp=71, mobile=72,mtp=73,mux=74,narp=75,netblt=76,nsfnet_igp=77,nvp=78,ospf=79,pgm=80,pim=81, pipe=82,pnni=83,pri-enc=84,prm=85,ptp=86,pup=87,pvp=88,qnx=89,rdp=90,rsvp=91,rtp=92,rvd=93, sat-expak=94,sat-mon=95,sccopmce=96,scps=97,sctp=98,sdrp=99,secure-rmtp=100,sep=101, skip=102,sm=103,smp=104,snp=105,sprite-rpc=106,sps=107,srp=108,st2=109,stp=110,sun-nd=111, swipe=112,tcf=113,tcp=114,tlsp=115,tp++=116,trunk-1 = 117,trunk-2 = 118,ttp=119,udp=120,unas=121, uti=122,vines=123,visa=124,vmtp=125,vrrp=126,wb-expak=127,wb-mon=128,wsn=129,xnet=130, xns-idp=131,xtp=132,zero=133
	service	dhcp=1,dns=2,ftp=3,ftp-data=4,http=5,irc=6,pop3=7,radius=8,smtp=9,snmp=10,ssh=11,ssl=12,-=13
Category label	Attack_cat	Normal=1,Fuzzers=2,Analysis=3,Backdoors=4,DoS=5,Exploit=6,Generic=7,Reconnaissance=8, Shellcode=9,Worms=10

Table 9 – Pretreatment of CIDDS-001.

Symbolic features	Proto	GRE=1,ICMP=2,TCP=3,UDP=4
	Flags	.....=1,....S.=2,...R..=3,...RS..=4,.A....=5,.A...F = 6,.A..SF=7,.A.R..=9,.A.R.F.=10,.A.RS.=11,.A.RSF= 12,.AP...=13,.AP.S.=14,.AP.SF=15,.APRS=16,.APRSF=17,Ox52=18,Ox53=19,Ox5a=20,Ox5b=21, Oxc2=22,Oxc6=23,Oxd2=24,Oxd3=25,Oxd6=26,Oxd7=27,Oxda=28,Oxdb=29,Oxde=30,Oxdf=31
	attackType	bruteForce=1,portScan=2,—=3
	attackID	—=23
	Attack Description	100 passwords=1,20 passwords=2,nmap args:-sS –T2=3,nmap args:-sU –T2=4,—=5
Category label	class	Normal=1,Attacker=2,Suspicious=3,Unknown=4,Victim=5

Table 10 – Pretreatment of ADFA-LD.

Category label	class	Normal=1,Adduser=2,Hydra_FTP=3,Hydra_SSH=4,Java_Meterpreter=5,Meterpreter=6,Web_Shell=7

Table 11 – Model topology.

Model name		Number of hidden neurons
SDAE-ELM1	DBN-Softmax1	256
SDAE-ELM2	DBN-Softmax2	256, 128
SDAE-ELM3	DBN-Softmax3	256, 128, 64

The learning rate can be obtained directly in [0, 1], so a large number of experiments are required. To avoid repeated experiments, we refer to the literature (Tang et al., 2016) to determine the final learning rate to be 0.0001. Finally, we will use the same optimal parameters and network topology for the SDAE-ELM and DBN-Softmax models. The final model topology determined in this paper is shown in Table 11.

Besides, the unique features of SDAE-ELM and DBN-Softmax in this paper are the loss function, activation function, and the use of Mini-Batch gradient descent to train the model to maximize the learning efficiency of the model.

5.1.1. *Loss function*

The main goal of model training is to minimize the loss function by optimizing the parameters of the neural network, thereby improving the classification effect of the model. There are many types of loss functions, and we need to use different loss functions for different targets. The final output of the output layer of the SDAE-ELM and DBN-Softmax model is the probability distribution of the actual output, so we choose the cross-entropy loss function as the loss function in this paper. The cross-entropy is mainly used to describe the distance between the actual output probability and the expected output probability. The smaller the value of cross-entropy is, the closer the actual output probability and the expected output probability are. Because this paper has carried out both binary classification and multi-class classification on intrusion detection data, we have adopted different loss functions for different classification standards. For binary classification, we use binary_crossentropy loss function; for multi-class classification, we use categorical_crossentropy loss function. Eqs. (17) and (18) are binary_crossentropy loss function

and categorical_crossentropy loss function, respectively.

$$loss = -(y_i log p_i + (1 - y_i) log(1 - p_i)) \tag{17}$$

where y_i is the label of sample data, the positive class is 1, and the negative class is 0; p_i is the probability that the sample i is predicted to be positive.

$$loss = -y_{ic} log p_{ic} \tag{18}$$

where y_{ic} is the indicator variable, if the category is the same as the category of sample i, y_{ic} is 1, otherwise it is 0; p_{ic} is the probability that sample i is predicted to be the category c.

5.1.2. *Activation function*

If no activation function is used in the neural network, no matter how many layers of the network we train, the final model output is a linear combination of inputs. Therefore, we need to use the activation function to calculate the weighted sum between the input and the deviation to determine whether the neuron node can be released. The Rectified Linear Unit(ReLU) activation function (Nair and Hinton, 2020) was proposed by Nair and Hinton in 2010, and compared to the Sigmoid and tanh activation functions, ReLU can retain as many linear models as possible, without saturation area, and gradient disappearance. Moreover, the calculation is simple, the efficiency is very high, and the convergence speed is fast. The formula of ReLU is defined as follows:

$$f(x) = max(0, x) \tag{19}$$

where x is the output of the input layer.

5.1.3. *Mini-Batch gradient descent*

Gradient Descent(GD) is an iterative method to find the global minimum of the objective function in the direction of negative gradient. The minimum loss function is the objective function in deep learning. In traditional deep neural network models, Stochastic Gradient Descent(SGD) (Liu et al., 2019) and Batch Gradient Descent(BGD) (Si et al., 2019) are often used to optimize the objective function. The two methods have their own advantages and disadvantages in different application ranges. So in order to improve the training speed of the model, the Mini-Batch Gradient Descent(MBGD) (Messaoud et al., 2020) is used to train the network model in this paper. BGD needs to update parameters with all samples to obtain the global optimal solution and the loss function each time the weight is updated. However, it will be very tricky to process big data, the training process will be very slow, and unable to continue training because of insufficient memory. SGD updates the parameters every time by training a single sample, which is fast in training. But the gradient of loss function calculated based on a random sample is deviated from the gradient of the loss function calculated based on all samples, which may be a bad gradient direction and may be able to obtain the global optimal solution. MBGD uses a part of samples to update parameters each time, which overcomes the shortcomings of SGD and BGD, and take into account the advantages of both methods. MBGD can update model parameters faster, improve the computational efficiency of the model, and avoid the model falling into local optimum. The Table 12 shows the advantages and disadvantages of SGD, BGD and MBGD.

The size of MBGD is a parameter that is independent of the overall architecture of the network, so we do not need to use the rest of the optimized hyperparameters to optimize the size of MBGD. Finally, we determined the size of the small batch of data as 100. Fig. 9 shows the loss function curves of BGD and MBGD on the SDAE-ELM network in KDD Cup99 dataset. Due to the noise, MBGD oscillates during the learning process. But overall, the loss function value of MBGD is less than that of BGD.

5.2. *Model evaluation criteria*

This paper uses the accuracy, precision, true positive rate, false positive rate, F value, P-R curve, ROC curve, and AUC value to evaluate the classification ability of the models. These values are obtained based on the confusion matrix in Table 13. 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 record, False Negative(FN) is the number of Attack connection records wrongly classified to the Normal connection record.

Indicators for model evaluation (Zaidi et al., 2016; Liang et al., 2020) are defined as follows:

Accuracy: It estimates the ratio of the correctly recognized connection records to the entire test dataset. If the accuracy is higher, the detection model is better($Accuracy \in [0, 1]$). *Accuracy* serves as a good measure for the test dataset that contains balanced classes and is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{20}$$

Precision: It estimates the ratio of the correctly identified attack connection records to the number of all identified attack connection records. If the *Precision* is higher, the detection model is better($Precision \in [0, 1]$). *Precision* is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{21}$$

F1-Score: F1-Score is also called as *F1-Measure*. It is the harmonic mean of *Precision* and *Recall*. If the *F1-Score* is higher, the detection model is better($F1 - Score \in [0, 1]$). *F1-Score* is defined as follows:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{22}$$

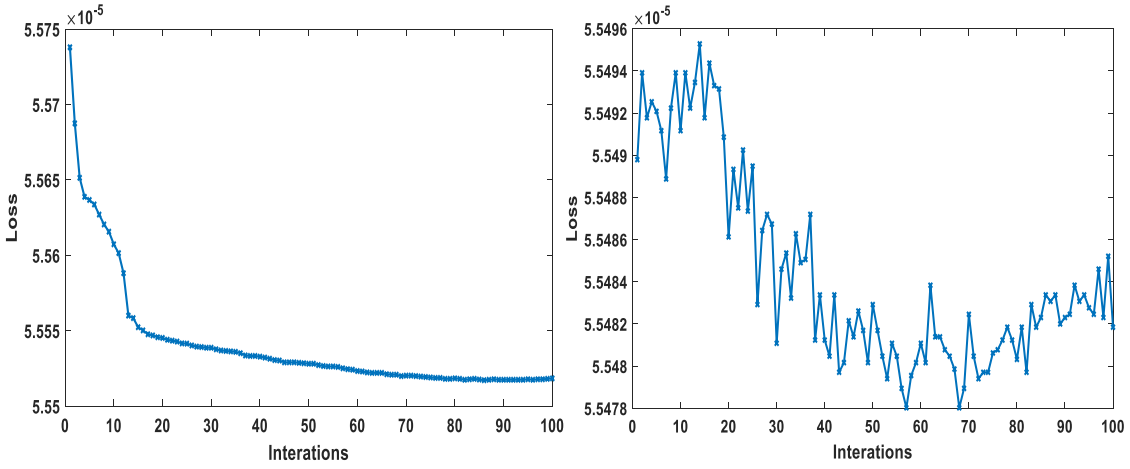
True Positive Rate(*TPR*): It is also called as *Recall*. It estimates the ratio of the correctly classified attack connection records to the total number of attack connection records. If the *TPR* is higher, the detection model is better($TPR \in [0, 1]$). *TPR* is defined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{23}$$

False Positive Rate(*FPR*): It estimates the ratio of the normal connection records flagged as attacks to the total number of

Table 12 – The advantages and disadvantages of GD.

	Advantages	Disadvantages
SGD	fast training every time	decreased accuracy; not necessarily the global optimal solution; not easy to implement in parallel; poor convergence
BGD	global optimal solution; easy to implement in parallel	when the sample data is large, the computation overhead is high and the computation is slow
MBGD	reduced computational overhead; reduced randomness; easy to implement in parallel	



(a) BGD (b) MBGD

Fig. 9 – Loss function.

Table 13 – Confusion matrix.

Actual value	Predictive value	
	Positive example	Counterexample
Positive example	TP (True Positive)	FN (False Negative)
Counter example	FP (False Positive)	TN (True Negative)

normal connection records. If the *FPR* is lower, the detection model is better($FPR \in [0, 1]$). *FPR* is defined as follows:

$$FPR = \frac{FP}{TN + FP} \tag{24}$$

Receiver Operating Characteristics(ROC) curve: ROC is plotted based on the trade-off between the *TPR* on the y axis to *FPR* on the x axis across different thresholds. The area under the ROC curve(*AUC*) is used along with ROC as a comparison metric for the detection model. If the *AUC* is higher, the machine learning model is better.

$$AUC = \int_0^1 \frac{TP}{TP + FN} d\frac{FP}{TN + FP} \tag{25}$$

6. Analysis of experimental results

In the field of intrusion detection, any detection method will eventually be applied to different actual scenarios, but different application environments, data sizes, and network environments are different, so there is no unique standard to measure the superiority of the algorithm. To adopt a unified standard in the experimental environment, the SDAE-ELM model is applied to the network-based KDD Cup99, NSL-KDD, UNSW-NB15, and CIDDS-001 data sets, and the DBN-Softmax model is applied to the host-based ADFA-LD dataset.

6.1. *Experimental results based on network datasets*

6.1.1. *Binary classification experimental results*

Combining the attack types into the abnormal class, the experiment becomes a binary classification problem. Tables 14–17 show the experimental results of the binary classification. From the table, it can be seen that the performance of the algorithms on the binary classification are not much different, but in general, the depth model has better performance than the shallow neural network. Binary classification data sets are an unbalanced data sets, and the amount of data of the attack type far exceeds the amount of data of the normal sample, so only using an accuracy as an evaluation index cannot reflect the detection ability of the model to unbalanced data sets. Therefore, in addition to accuracy, the

Table 14 – KDD Cup99 binary test results.

Algorithm	Accuracy	Precision	Recall	F1-score	Time
AdaBoost	0.9492	0.9425	0.9272	0.9348	7202s
DT	0.9315	0.9806	0.9327	0.9560	14400s
ELM	0.9404	0.9831	0.9318	0.9568	2305s
SVM	0.9305	0.9818	0.9302	0.9553	>17h
LR	0.9301	0.9814	0.9300	0.9551	3409s
SOM	0.2020	0	0	NaN	100s
DNN	0.9379	0.9310	0.9959	0.9624	>12h
DBN	0.9339	0.9867	0.9297	0.9573	>12h
SDAE	0.9332	0.9858	0.9297	0.9569	>12h
SDAE-ELM1	0.9346	0.9867	0.9306	0.9579	4302s
SDAE-ELM2	0.9357	0.9869	0.9319	0.9586	4709s
SDAE-ELM3	0.9356	0.9875	0.9310	0.9584	4878s

Table 15 – NSL-KDD binary test results.

Algorithm	Accuracy	Precision	Recall	F1-score	Time
AdaBoost	0.7475	0.9889	0.5629	0.7175	1801s
DT	0.7621	0.9558	0.6106	0.7451	3600s
ELM	0.7449	0.8667	0.6523	0.7444	576s
SVM	0.7317	0.9522	0.5569	0.7028	>5h
LR	0.7283	0.8942	0.5931	0.7132	825s
SOM	0.4305	0	0	NaN	27s
DNN	0.7392	0.9228	0.5915	0.7209	>3h
DBN	0.7160	0.8972	0.5661	0.6942	>3h
SDAE	0.7293	0.9293	0.5679	0.7049	>3h
SDAE-ELM1	0.7804	0.9599	0.6412	0.7687	1075s
SDAE-ELM2	0.7708	0.9244	0.6507	0.7637	1177s
SDAE-ELM3	0.7664	0.9598	0.6155	0.7500	1220s

Table 16 – UNSW-NB15 binary test results.

Algorithm	Accuracy	Precision	Recall	F1-score	Time
AdaBoost	0.7663	0.7024	0.9887	0.8248	8304s
DT	0.7638	0.7216	0.9612	0.8243	15763s
ELM	0.7651	0.7237	0.9275	0.8130	3098s
SVM	0.6865	0.6513	0.9271	0.7651	>20h
LR	0.6814	0.6605	0.8674	0.7499	4789s
SOM	0.4494	NaN	0	NaN	200s
DNN	0.7342	0.7202	0.8459	0.7780	>17h
DBN	0.6812	0.6594	0.8706	0.7504	>17h
SDAE	0.6816	0.6588	0.8748	0.7516	>17h
SDAE-ELM1	0.7220	0.6693	0.9785	0.7949	5430s
SDAE-ELM2	0.7211	0.6698	0.9736	0.7936	5763s
SDAE-ELM3	0.7238	0.6994	0.8742	0.7771	5943s

model's binary classification ability is evaluated from precision, recall and F1-Score. In each data set, the detection effect of other model is better expect for the poor effect of SOM algorithm.

In most cases, the SDAE-ELM model achieves better detection performance than traditional machine learning models. SDAE-ELM1, SDAE-ELM2, and SDAE-ELM3 can achieve similar detection results, but overall better than DT, ELM, SVM, and DNN. For the KDD Cup99 dataset, the accuracy of the algorithm is greater than 93%; in terms of accuracy and recall, with the increase of model depth, the higher the accuracy and recall of SDAE-ELM, in most cases, the accuracy and recall of SDAE-ELM are better. SDAE-ELM3 has an increase of 0.17% compared to SDAE. Compared with AdaBoost and DNN, the recall of SDAE-ELM is slightly worse, which is mainly because the data set of KDD Cup99 has data redundancy and duplication. Fully learning this data set can achieve better classification results. Although the SDAE-ELM can better remove the noise existing in the data set, compared with AdaBoost and DNN, the data mining ability of the SDAE-ELM is relatively weak, but the model's ability to detect the KDD Cup99 data set has been reduced, which is also expected. For the NSL-KDD

Table 17 – CIDDS-001 binary test results.

Algorithm	Accuracy	Precision	Recall	F1-score	Time
AdaBoost	0.8182	0.8181	0.9397	0.8747	9432s
DT	0.9374	0.9572	0.9497	0.9534	17987s
ELM	0.8176	0.8176	1	0.8996	3987s
SVM	0.8920	0.9983	0.8693	0.9294	>25h
LR	0.8146	0.8146	0.9806	0.8899	5763s
SOM	0.1824	NaN	0	NaN	390s
DNN	0.8176	0.8176	1	0.8996	>20h
DBN	0.9766	0.9962	0.8696	0.9286	>20h
SDAE	0.8806	0.9983	0.8555	0.9214	>20h
SDAE-ELM1	0.9537	0.9979	0.9454	0.9710	6543s
SDAE-ELM2	0.9238	0.9978	0.9089	0.9512	6879s
SDAE-ELM3	0.9258	0.9951	0.9137	0.9527	7012s

data set, compared to the KDD Cup99 data set, because the number of training sets is greatly reduced, the model learns fewer features, so the detection ability of each algorithm is reduced, but each algorithm can maintain the original detection ability. For the UNSW-NB15 data set, in most cases, SDAE-ELM can achieve a better detection performance, and as the depth increases, the performance of the model is better, and the remaining algorithms can achieve similar detection performance. For the CIDDS-001 data set, compared to other data sets, the detection performance of each algorithm is better, because the CIDDS-001 data set has more training data, each algorithm can better learn the characteristics of the CIDDS-001 data set, so the detection performance of each algorithm is better. In terms of accuracy, SDAE is 4.32% worse than SDAE-ELM; SDAE-ELM is slightly worse than SVM in precision, but it is better than other algorithms; in terms of recall, AdaBoost, ELM, and DNN can reach 100%, that is, all attack data can be detected by three algorithms.

F1-Score is a comprehensive evaluation index of accuracy and recall can better reflect the classification ability of the model. In most cases, the F1-Score of SDAE-ELM is better. In the KDD Cup99 data set, with the increase of model depth, the F1-Score is getting better and better, but the difference is 0.0038 compared with DNN. In the NSL-KDD data set, SDAE-ELM obtains the optimal F1-Score. In the UNSW-NB15 data set, SDAE-ELM can obtain the optimal F1-Score, which is much higher than that of other detection models. In general, in multiple data sets, the SDAE-ELM model has a better detection performance for binary classification of data sets.

Applying the SDAE-ELM model to a network-based intrusion detection system, in addition to ensuring that the model can achieve better detection performance, we also need to consider the time performance of the model. In the process of detecting network-based data sets, we envision that SDAE-ELM can complete intrusion detection faster, which is confirmed by the Time evaluation indicators in Tables 14-17. Although SDAE-ELM requires a longer training time compared to the ELM and SOM, compared with other machine learning and deep learning, the training of time of SDAE-ELM is greatly reduced, which can better meet our requirements for real-time performance.

The confusion matrix uses the heat map to show the differences of data through color difference and brightness, which is easy to understand. It can summarize the records in the data set according to the actual results and prediction results to achieve visualization. The evaluation indexes of the model are also obtained through the confusion matrix. Fig. 10 is the confusion matrix of some algorithms in each data set. It can be seen that in most cases, the classification of the SDAE-ELM model is the most accurate. That is, the values in the first and third quadrants in the confusion matrix is the largest, and that in the second and fourth quadrants in the confusion matrix is the smallest. The confusion matrix of other algorithms such as SVM, ELM, and SDAE are very similar. Although the values in the first and third quadrants are larger, the values in the second and fourth quadrants are also larger. There are false positives in the algorithm. Fig. 11 and 12 show the accuracy curve and the accuracy boxplot of each data set. As can be seen from Fig. 11, in the KDD Cup99 and UNSW-NB15 data sets, AdaBoost, DT, SVM, and LR achieve the optimal accuracy that can be obtained at the beginning of the iteration, and as the number of iterations increases, the accuracy of the model remains unchanged. For DNN, DBN, SDAE, the accuracy of the model increases with the number of iterations, and the accuracy of the model is basically stable at the later stage of the iteration. After 25 iterations of the DBN and SDAE, and 55 iterations of the DNN algorithm, the accuracy of models are basically unchanged. For the SDAE-ELM model, the optimal accuracy can be obtained at the beginning of the iteration. In most cases, the accuracy of the SDAE-ELM model is better than the traditional machine learning. The boxplot can reflect the outliers in the data and the distribution of the data. As can be seen from Fig. 12, it can be seen that for NSL-KDD and CIDDS-001 data sets, AdaBoost, DT, SVM, LR, and SDAE-ELM models have very stable accuracy distribution and no outliers; but for the remaining models, in particular, the ELM algorithm has large fluctuations, especially in the CIDDS-001 data set. For the DNN model, the accuracy fluctuates after several iterations, and outliers appear in the accuracy. This is because the accuracy of the DNN model continuously changes during several iterations, resulting in outliers. Comparing the accuracy of the CIDDS-001 dataset and the NSL-KDD dataset, it is found that the accuracy of each algorithm on the CIDDS-001 dataset is better, indicating that the SDAE-ELM model also has better detection ability for newer attack type. Fig. 13 is the P-R curve of the data set. P-R curve can describe the re-

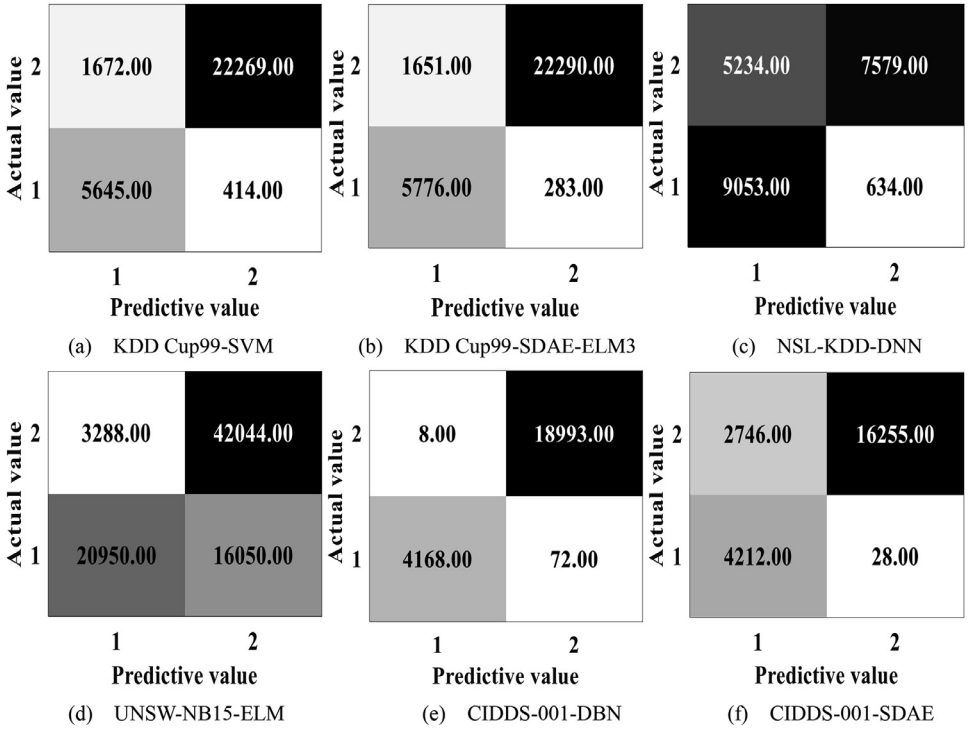


Fig. 10 – Confusion matrix.

lationship between the accuracy and the recall, but compared to the ROC curve, the P-R curve pays more attention to positive samples. As can be seen from Fig. 13, the original deep learning model is directly applied to each data set for intrusion detection, the detection effect is slightly worse than the traditional machine learning model; for each data set, the P-R curve of the SDAE-ELM model is very close. Combining the traditional machine learning and the SDAE-ELM, we can conclude that the P-R curve of the SDAE-ELM is slightly worse than the P-R curve of the DT, but better than the P-R of other models, this is because, in the binary classification algorithm, DT can mine more data features, so the P-R curve of this model is better.

6.1.2. *Multi-class classification experimental results*

When the normal samples and different types of attack data are contained in the intrusion detection data sets, and the experiment becomes a multi-class classification experiment. Tables 18-21 show the experimental results of the multi-class classification of each data set. The multi-class classification data sets alleviate the unbalanced problem of the binary classification data sets to a certain extent, but there are still large differences in the data amount of each type of data. Overall, each model has better detection performance for samples with larger data volume. For multi-class classification, the multi-class classification ability of the model will be evaluated from the true positive rate, false positive rate, and ROC value. As with the binary classification, in each data set, except for the poor multi-class classification ability of the SOM, the other models have a better classification effect.

In most cases, the SDAE-ELM model can achieve better detection results, but when detecting a certain type of data in the data sets, the detection performance may be slightly worse than the traditional machine learning model. As can be seen from Table 18, for "Normal" type data, the comprehensive detection performance of AdaBoost and ELM is the worst, the true positive rate is less than 85%, and the true positive rate of the remaining models are greater than 93%, in particular, the true positive rate of SDAE-ELM can reach 96.82%, this SDAE-ELM can fully learn the characteristics of the "Normal" sample, to obtain a better true positive rate and false positive rate; for the "Probe" attack type, the detection performance of the AdaBoost, DT, ELM, SVM, LR and SDAE-ELM are similar, which are far better than the detection performance of DNN, DBN, and SDAE. For the "DoS" attack type, the detection performance of each algorithm is better, among which the worst rate of the SDAE-ELM2 can also reach 99.93%. In particular, the true positive rate of AdaBoost, DT, SVM, LR, and SDAE can all reach 100%. Because "DoS" has the largest number of attacks, and this paper judges according to the difference between intrusion data and normal data, therefore, for the "DoS" attacks with a large number of samples, each algorithm has a better detection performance; for the "R2L" and "U2R" attack types, in most cases, the performance of AdaBoost, DT, ELM and SDAE-ELM are similar, and the detection performance is poor, SVM, LR, DNN, DBN, and SDAE have the worst detection performance, and basically cannot recognize the "R2L" and "U2R" attack types, but the true positive rate of SDAE-ELM1 is increased by 5.26% compared to SDAE, and the effect is in line with our expectations. There are few types of "R2L" and "U2R" attacks, and most of the attacks are disguised as legit-

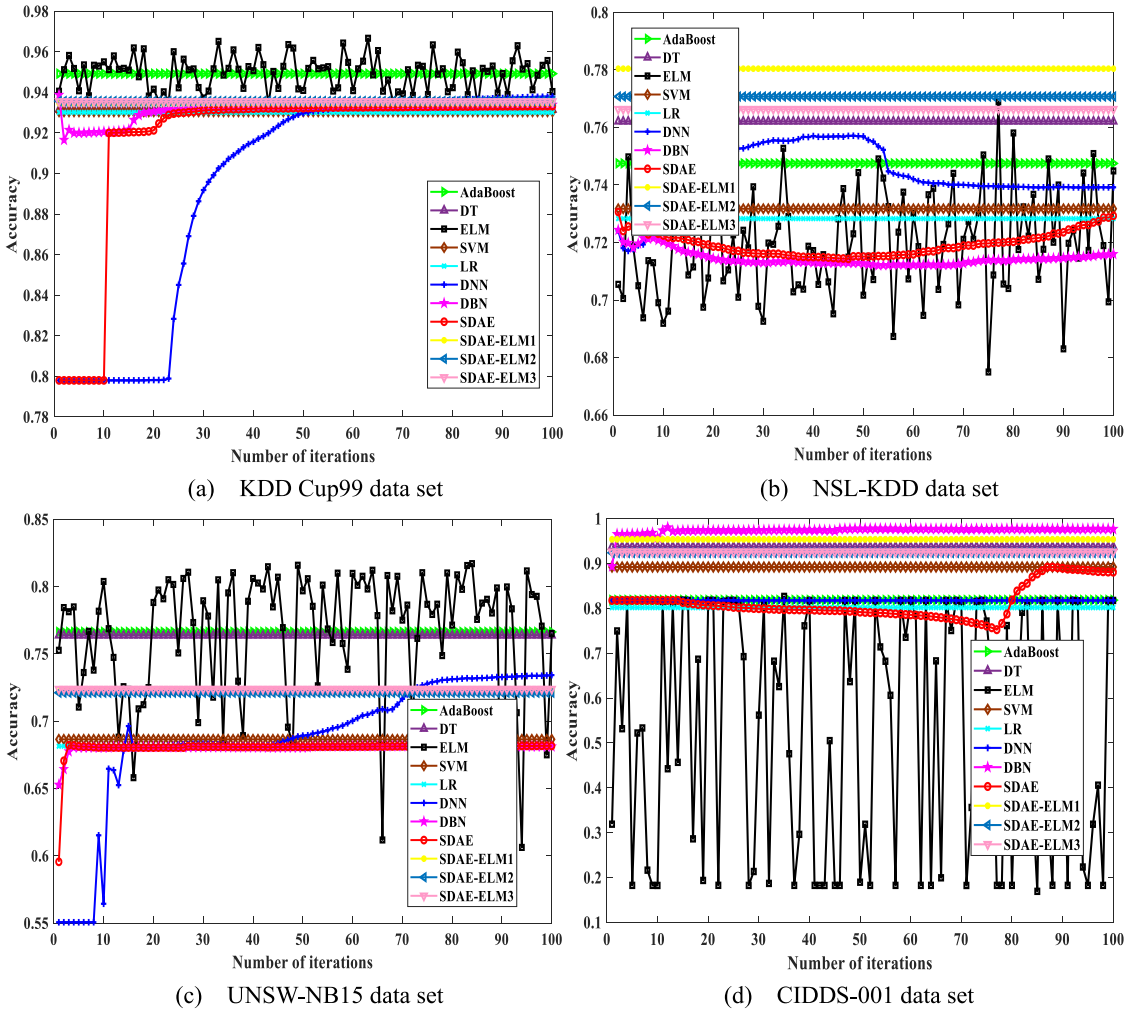


Fig. 11 – Data set accuracy.

imate users. This is to make their characteristics very similar to normal data, which makes it difficult to detect "R2L" and "U2R" attacks. NSL-KDD and KDD Cup99 datasets have the same data type, except for the number of training and testing. Comparing Tables 19 and 18, it can be found that each algorithm can maintain similar detection performance as KDD Cup99 in NSL-KDD, but the detection rate of each detection performance has decreased. Mainly because the training data of NSL-KDD is much less than the training data of KDD Cup99, the features learned by the model are not enough, which leads to the slightly worse detection performance on the NSL-KDD data set.

At present, in the field of intrusion detection, most people still use the KDD Cup99 dataset of the Lincoln Laboratory in the United States for experiments. This dataset was good in a certain period of time, but this dataset was collected more than 20 years ago, and the current complex network cannot be evaluated with this data set. Therefore, we further apply the SDAE-ELM model to the UNSW-NB15 and CIDDS-001 data sets that contain newer attack types. As can be seen from Table 20, the UNSW-NB15 data set has more types of attacks, plus normal sample data totaling 10 types. Besides, this data set also contains the newer types of network attacks that now appear on the network, for example, "Backdoors", "Shellcodes", "Reconnaissance" and "Worm" attack types can better reflect the characteristics of the current network intrusion. Considering all classification results, the SDAE-ELM can obtain a better true positive rate, false positive rate, and AUC value. However, the detection rate of some attack types is slightly worse than that of DT. For example, for the "Worms" attack type, only the true positive rate of the DT is 13.64%, and the true positive rate of the remaining algorithms are 0%; but for the "Generic" attack type, the SDAE-ELM model can obtain the best true positive rate and false positive rate. For the "Analysis", "Backdoors" and "Worms" type attacks, the detection of each algorithm is poor, this is because the number of training samples is too small, data distribution imbalance appears, which affects the classification ability of the algorithm. "Generic" as a general attack, mainly attacks the server, which is very close to the characteristics of the "Exploits" attack type, but there is data imbalance between the two attack types. Therefore, there may be misjudgments in the detection results.

Although there are only four types of attacks in the CIDDS-001 data set, they were captured in the 2017 simulated small-scale business environment, and so far few researchers have

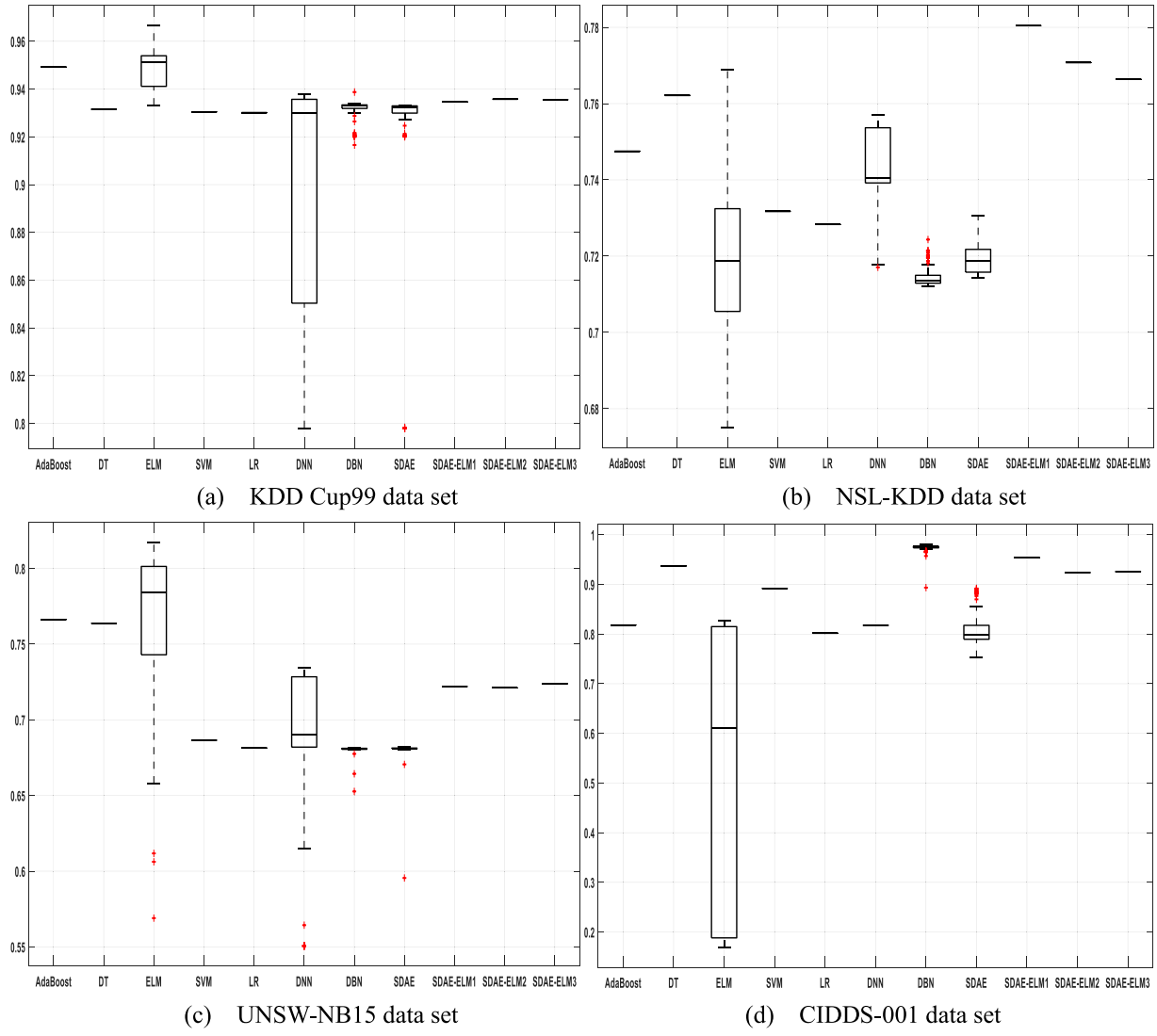


Fig. 12 – Data set accuracy boxplot.

applied it to the field of intrusion detection. Therefore, it is more meaningful for us to apply them to the field of intrusion detection to detect the performance of the models. As can be seen from Table 21, SDAE-ELM can achieve better detection results by integrating all evaluation indicators. For the "Normal" data type, except for ELM, LR, and DNN, the remaining algorithms all can achieve a better comprehensive detection effect, and the true positive rate is greater than 95%, indicating that at this time each algorithm can better learn the characteristics of "Normal" data, so as to achieve a better classification of this type of data; for "Attackers" type of attack, only the AdaBoost can perfectly classify this type of attack. The true positive rate of this algorithm is 100%, the false positive rate is 0%, and the AUC value is 1. The detection performance of the remaining algorithms are poor, and their true positive rates are all 0%, the attack type cannot be identified at all. This is because "Attackers" have similar characteristics with "Suspicious", so the algorithm misinterprets the "Attackers" attack as a "Suspicious" attack during the classification process; for "Suspicious" attack, except for the true positive rate of the SVM algorithm is 87.36%, and the true positive of other algorithms are greater than 93%. Since the number of "Suspicious" attack are the largest, each algorithm can fully learn the characteristics of the "Suspicious" attack. Therefore, for a large number of "Suspicious" attacks, each algorithm has better detection performance; for "Unknown" and "Victim" attack types, the SDAE-ELM can achieve a better detection rate, the algorithm can fully mine the characteristics of "Unknown" and "Victim" attack types, to achieve better classification.

Table 22 is the time performance of the algorithm on each data set. It can be seen that the larger the data volume of the data set, the longer the training time that the model needs to spend. This is reasonable. In each data set, except SOM and ELM take less time, and the training time of the remaining models is longer. In particular, SVM as a more classic method in machine learning, has also achieved good results in many fields, but its time cost is relatively high,

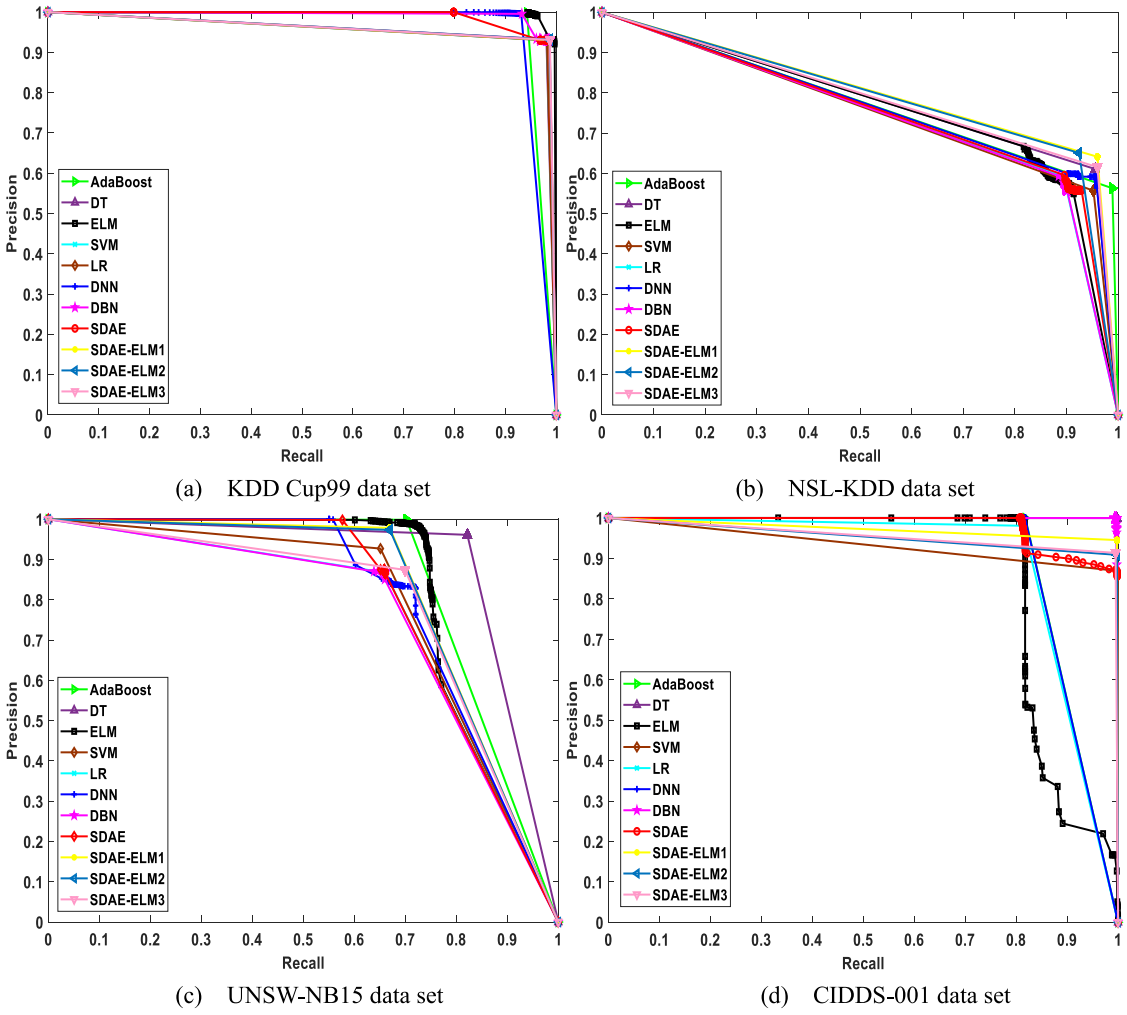


Fig. 13 – P-R graph of data set.

while SDAE-ELM also takes some time to train, but compared with other models, the time performance has been greatly improved.

In the process of implementing intrusion detection, each layer of the neural network is helpful to understand how to classify data into "normal" or "attack" and the specific classification of attacks into attack categories. To understand this process more intuitively, the activation value is passed to t-SNE to visualize it. The KDD Cup99 and CIDDS-001 data sets are shown in Fig. 14(a) and 14(b), respectively. For the KDD Cup99 data set, "Normal", "DoS", and "R2L" features have completely appeared in another cluster; for the CIDDS-001 data set, the "Normal" type feature has appeared in another cluster, but for "Suspicious" and "Unknown" it has appeared in the same cluster. This shows that the algorithm can well identify some types of attacks at this time, but the optimal partition function is not achieved. For the CIDDS-001 data set, the SDAE-ELM1 belongs to the "Suspicious" connection record as shown in Fig. 15(b), which have similar characteristics with "Unknown"; for the NSL-KDD data set, the connection records of the AdaBoost are shown in Fig. 15(a), at this time, different types of data in the data set have different characteristics. Although the AdaBoost has been able to correctly distinguish the attack categories, but for samples with similar features, we need to attach more features to correctly distinguish the data with similar features.

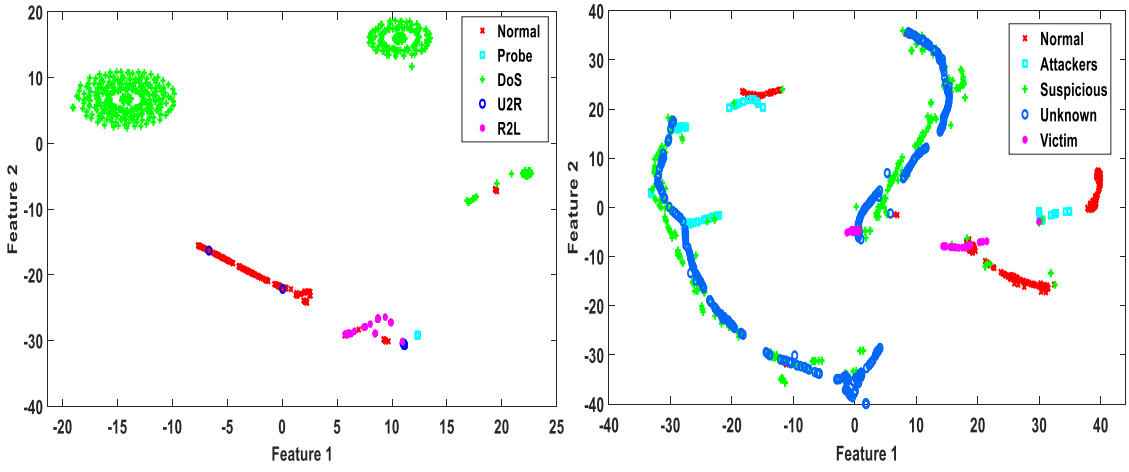
The hidden layer contains more parameters. In deep learning, the parameters of the hidden layer have a great impact on the convergence speed and performance of the model. During the training process, we continuously iterate the parameters of the hidden layer to obtain a better classification model. In general, the parameters of the hidden layer are obtained in the self-learning process of the model. To understand the hidden layer parameters more intuitively, we visualize the weight of the first DAE hidden layer in the model. The visualization results are expressed in grayscale, as shown in Fig. 16.

ROC curve is an easy to understand graphical tool that can be used in all classification models, and it has a huge advantage, when the distribution of positive and negative samples changes, its shape can remain basically unchanged, and it can more objectively measure the performance of the model itself. As can be seen from Fig. 17, for the "Normal" data type in the KDD Cup99 dataset, as the model depth increases, the

Table 18 – KDD Cup99 multi-class classification test results.

Algorithm	Normal			Probe			DoS		
	TPR	FPR	AUC	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.8338	0.0022	0.9158	0.9904	0.0056	0.9924	1	0.32189	0.8301
DT	0.9645	0.0675	0.9485	0.9904	0.0117	0.9894	1	0.0055	0.9973
ELM	0.8487	0.0035	0.9885	0.9832	0.0882	0.9927	0.9999	0.0061	0.9984
SVM	0.9525	0.0705	0.9410	0.9039	0.0072	0.9483	1	0.0427	0.9787
LR	0.9386	0.0702	0.9342	0.8966	0.0064	0.9451	1	0.0634	0.9683
SOM	0	0	0.0093	0.1851	0.9499	0.2183	0	0.9548	0.04517
DNN	0.9668	0.0716	0.9668	0.1899	0.0066	0.5919	0.9995	0.0695	0.9652
DBN	0.9647	0.0719	0.9442	0.1899	0.0065	0.5877	0.9996	0.0814	0.9594
SDAE	0.9512	0.8461	0.9377	0	0	0.5	1	0.1090	0.9455
SDAE-ELM1	0.9647	0.0695	0.9510	0.9928	0.0069	0.9932	0.9997	0.0155	0.9944
SDAE-ELM2	0.9682	0.0697	0.9514	0.9423	0.0072	0.9800	0.9993	0.0158	0.9964
SDAE-ELM3	0.9365	0.0695	0.9512	0.9183	0.0072	0.9885	0.9995	0.0467	0.9947

Algorithm	R2L			U2R		
	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.1140	0.0002	0.5569	0.0012	0.0007	0.5009
DT	0.1974	0.0008	0.5983	0.0037	0.0009	0.5014
ELM	0.0789	0.0001	0.5589	0.0025	0.0014	0.966
SVM	0	0	0.5	0	0.0009	0.499
LR	0	0	0.5	0.0012	0.0003	0.5005
SOM	0.6667	0.8369	0.1397	0	0	0.0259
DNN	0	0	0.5	0.0037	0.0030	0.5003
DBN	0	0	0.5	0.0006	0.0005	0.5001
SDAE	0	0	0.5	0.0056	0.0015	0.5020
SDAE-ELM1	0.0526	0.0002	0.5459	0.0062	0.0036	0.5033
SDAE-ELM2	0.0044	0	0.5066	0.0080	0.0039	0.5022
SDAE-ELM3	0.0102	0	0.5123	0.0080	0.0043	0.5013



(a) KDD Cup99-SAE-ELM2 (b) CIDDS-001-SAE

Fig. 14 – Feature mapping of last hidden layer activation function.

AUC value of the SDAE-ELM is better, but its optimal AUC value is 0.0371 lower than that of ELM; for the "DoS" attack type in the NSL-KDD data set, SDAE-ELM is slightly worse than DT by 0.0073, but compared to other models, SDAE-ELM can obtain the optimal AUC value; for the "Generic" attack type in the UNSW-NB15 data set, the AUC value of each algorithm is better, the worst AUC value of the LR can also reach 0.8228, but the SDAE-ELM is 0.0537 lower than that of DT. For the "Suspicious" attack type of the CIDDS-001 data set, SDAE-ELM can obtain better AUC value, but some algorithms have poor AUC values, such as the AUC value of ELM is 0.3183, the AUC value of DBN is 0.4849. In summary, in most cases, AUC is used as the

Table 19 – NSL-KDD multi-class classification test results.

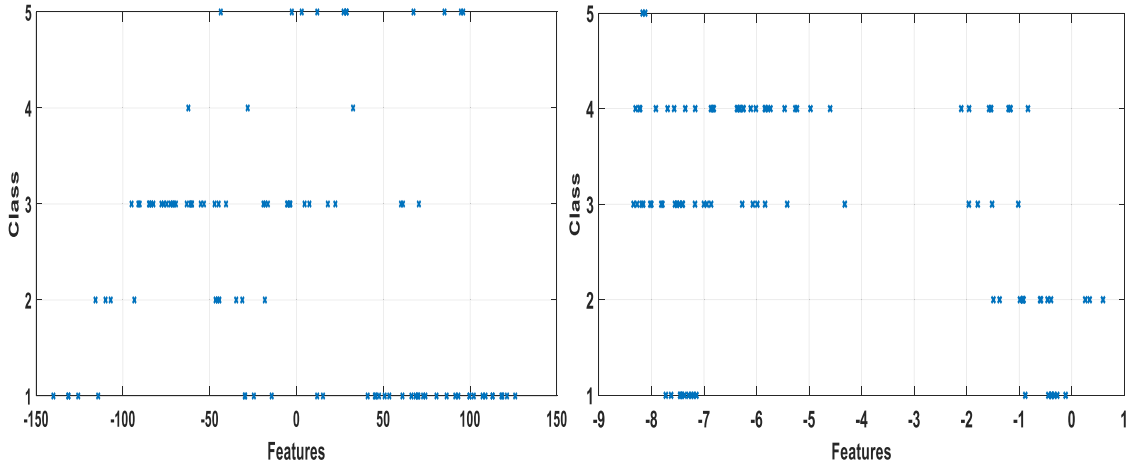
Algorithm	Normal			Probe			DoS		
	TPR	FPR	AUC	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.9726	0.5110	0.7308	0.6126	0.0493	0.7817	0.6109	0.0009	0.8050
DT	0.9624	0.3527	0.8049	0.8030	0.0243	0.8893	0.7658	0.0168	0.8745
ELM	0.9757	0.4166	0.8133	0.6051	0.0279	0.8702	0.7447	0.0465	0.7659
SVM	0.9329	0.5239	0.7045	0.0950	0.0102	0.5424	0.6720	0.2015	0.7352
LR	0.9262	0.5180	0.7041	0.1297	0.0474	0.5412	0.6548	0.1834	0.7357
SOM	0.7567	0.7243	0.5287	0.1404	0.3726	0.4384	0.0793	0.0841	0.3894
DNN	0.9420	0.5214	0.7137	0	0	0.5	0.7147	0.1991	0.7663
DBN	0.9299	0.4988	0.7101	0.2359	0.0602	0.5863	0.6706	0.1357	0.7494
SDAE	0.9319	0.5587	0.6822	0	0	0.5	0.6725	0.1848	0.7335
SDAE-ELM1	0.9333	0.4088	0.7867	0.6468	0.0383	0.8199	0.7294	0.0531	0.8439
SDAE-ELM2	0.9324	0.4334	0.7495	0.5287	0.0574	0.7357	0.6927	0.1083	0.8522
SDAE-ELM3	0.9656	0.4700	0.7478	0.3317	0.0493	0.6412	0.6755	0.1214	0.8672

Algorithm	R2L			U2R		
	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.0300	0	0.5150	0.0007	0.0001	0.5003
DT	0.0350	0.0050	0.5150	0.0835	0.0200	0.5318
ELM	0.0250	0.0001	0.5224	0.0044	0.0029	0.5080
SVM	0	0	0.5	0	0	0.5
LR	0	0	0.5	0	0	0.5
SOM	0	0.1515	0.2003	0.0054	0.3696	0.3062
DNN	0	0	0.5	0	0	0.5
DBN	0	0	0.5	0	0	0.5
SDAE	0	0	0.5	0.0076	0.0028	0.5024
SDAE-ELM1	0	0	0.5	0.0243	0.0209	0.5055
SDAE-ELM2	0	0	0.5	0.0257	0.0001	0.5130
SDAE-ELM3	0	0	0.5	0.0271	6.5867e-5	0.5171

evaluation index compared with the existing model, SDAE-ELM model performed well. This also shows that SDAE-ELM obtained a higher TPR and lower FPR.

6.2. *Experimental results based on host dataset*

Combining the attack types into the abnormal class, the experiment becomes a binary classification problem. Table 23 shows the results of the binary classification experiment of the ADFA-LD data set. When the normal samples and indi-



(a) NSL-KDD-AdaBoost (b) CIDDS-001-SAE-ELM1

Fig. 15 – Connection record of last hidden layer activation function.

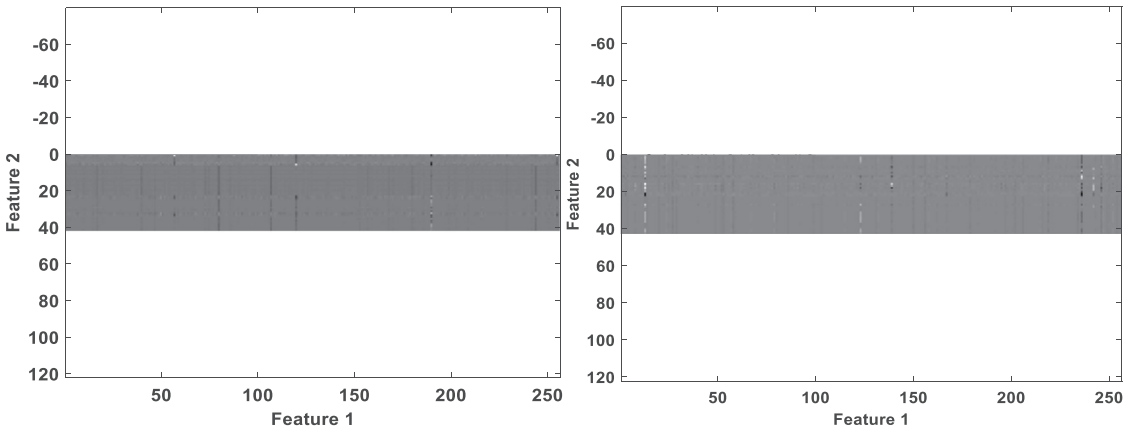
Table 20 – UNSW-NB15 multi-class classification test results.

Algorithm	Normal			Fuzzers			Analysis		
	TPR	FPR	AUC	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.5663	0.0505	0.7579	0.6140	0.2077	0.7032	0	0.0027	0.4987
DT	0.7479	0.0510	0.8485	0.4261	0.1272	0.6495	0.0827	0.0331	0.5248
ELM	0.6469	0.1320	0.8130	0.2801	0.0823	0.7943	0	0.0064	0.4951
SVM	0.8735	0.4559	0.7088	0	0	0.5	0	0	0.5
LR	0.865	0.4561	0.7045	0	0	0.5	0	0	0.5
SOM	0.4240	0.9606	0.1380	0	0	0.5	0	0	0.5
DNN	0.6935	0.3154	0.6639	0	0.0005	0.4996	0	0	0.5007
DBN	0.8698	0.4568	0.7001	0	0	0.5	0	0	0.5
SDAE	0.8729	0.4576	0.7071	0	0	0.5	0	0	0.5
SDAE-ELM1	0.7044	0.2672	0.7186	0.0200	0.0037	0.5081	0	0	0.5
SDAE-ELM2	0.7318	0.3131	0.7094	0.0048	0.0009	0.5019	0	0	0.5
SDAE-ELM3	0.7945	0.3987	0.6979	0.0007	0.0004	0.5002	0	0	0.5

Algorithm	Backdoors			DoS			Exploit		
	TPR	FPR	AUC	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0	0	0.5	0.0037	0.0007	0.5015	0.8824	0.1991	0.8417
DT	0.2487	0.0513	0.5987	0.1289	0.0280	0.5505	0.6387	0.0798	0.7795
ELM	0	0	0.5	0.4444	0.0534	0.7752	0.6288	0.1147	0.8526
SVM	0	0	0.5	0	0	0.5	0.0230	0.0097	0.5066
LR	0	0	0.5	0.0176	0.0031	0.5073	0.0163	0.0121	0.5021
SOM	0	0	0.5	0.2194	0.7366	0.2954	0	0	0.5
DNN	0	0	0.5	0	2.121e-5	0.5008	0.2869	0.1831	0.4427
DBN	0	0	0.5	0.0051	0.0008	0.5023	0.0166	0.0098	0.5032
SDAE	0	0	0.5	0.0465	0.0082	0.5191	0	0	0.5
SDAE-ELM1	0	0	0.5	0.1029	0.0002	0.5124	0.5650	0.2789	0.6430
SDAE-ELM2	0	0	0.5	0.1037	0.0002	0.5128	0.5705	0.1933	0.6536
SDAE-ELM3	0	0	0.5	0.1049	0.0004	0.5122	0.5442	0.0808	0.6317

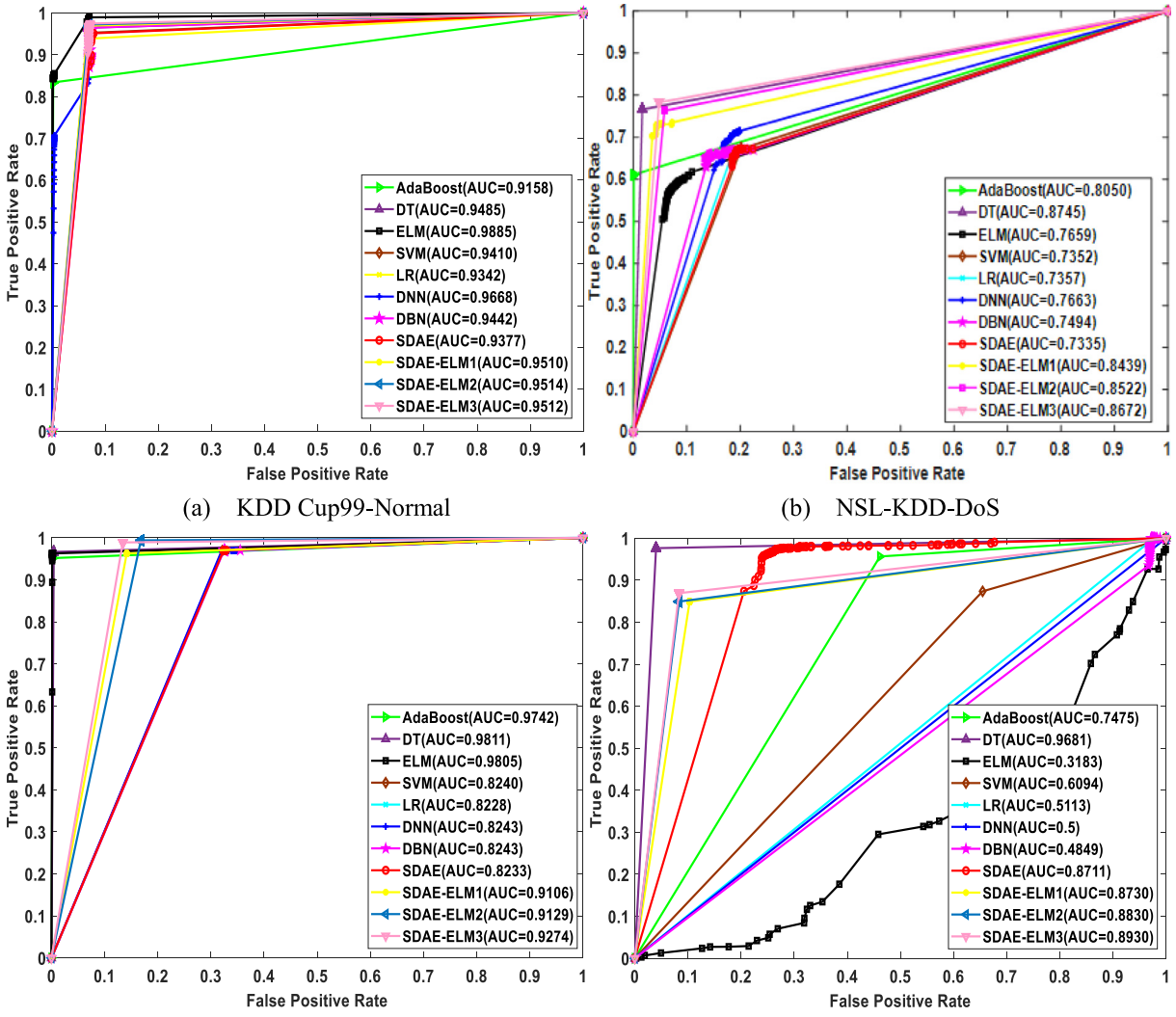
Algorithm	Generic			Reconnaissance			Shellcode		
	TPR	FPR	AUC	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.9518	0.0036	0.9742	0.5712	0.0227	0.7743	0.0714	0.0020	0.5347
DT	0.9665	0.0043	0.9811	0.7589	0.0154	0.8717	0.5238	0.0111	0.7564
ELM	0.9627	0.0015	0.9805	0.7203	0.1321	0.8828	0	0	0.5
SVM	0.9693	0.3214	0.8240	0	0	0.5	0	0	0.5
LR	0.9694	0.3239	0.8228	0	0	0.5	0	0	0.5
SOM	0	0	0.5	0	0	0.5	0	0	0.5
DNN	0.9693	0.3481	0.8243	0	0	0.4999	0	0	0.4588
DBN	0.9695	0.3244	0.8224	0	0	0.5	0	0	0.5
SDAE	0.9695	0.3235	0.8233	0	0	0.5	0	0	0.5
SDAE-ELM1	0.9629	0.1416	0.9106	0.0023	0	0.5134	0	0	0.5
SDAE-ELM2	0.9640	0.1482	0.9129	0.0031	0	0.5178	0	0	0.5
SDAE-ELM3	0.9691	0.1342	0.9274	0.0043	0	0.5213	0	0	0.5

Algorithm	Worms		
	TPR	FPR	AUC
AdaBoost	0	0	0.5
DT	0.1364	0.0010	0.5677
ELM	0	0	0.5
SVM	0	0	0.5
LR	0	0	0.5
SOM	0	0	0.5
DNN	0	0	0.5
DBN	0	0	0.5
SDAE	0	0	0.5
SDAE-ELM1	0	0	0.5
SDAE-ELM2	0	0	0.5
SDAE-ELM3	0	0	0.5



(a) NSL-KDD-DBN (b) CIDDS-001-DBN-Softmax1

Fig. 16 – First DAE visualization weights.



(a) KDD Cup99-Normal (b) NSL-KDD-DoS

(c) UNSW-NB15-Generic (d) CIDDS-001-Suspicious

Fig. 17 – ROC curve.

Table 21 – CIDDS-001 multi-class classification test results.

Algorithm	Normal			Attackers			Suspicious		
	TPR	FPR	AUC	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.9693	0.0406	0.9644	1	0	1	0.9565	0.4615	0.7475
DT	0.9929	0.0002	0.9964	0.1128	0	0.5564	0.9963	0.0200	0.9681
ELM	0	0.0069	0.3653	0	0	0.4997	0.9936	1	0.3183
SVM	0.9941	0.2676	0.8632	0	0	0.5	0.8736	0.6547	0.6094
LR	0	0.0092	0.4954	0	0	0.5	0.9927	0.9700	0.5113
SOM	0.4854	0.8828	0.0557	0	0	0.5	0	0	0.5
DNN	0	0	0.5	0	0	0.5	1	1	0.5
DBN	0.0101	0.1365	0.4363	0	0	0.5	0.9350	0.9671	0.4849
SDAE	0.9689	0.0023	0.9905	0	0	0.4916	0.9879	0.3325	0.8711
SDAE-ELM1	0.9948	0.0548	0.97	0.8487	0.0024	0.9231	0.9493	0.2199	0.8730
SDAE-ELM2	0.9599	0.0301	0.9649	0.8379	0.0002	0.9499	0.9731	0.2579	0.8830
SDAE-ELM3	0.9653	0.1860	0.8897	0.8543	0.0012	0.9378	0.9300	0.1986	0.8930

Algorithm	Unknown			Victim		
	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.1106	0.0028	0.5539	1	0	1
DT	0.9821	0.0048	0.9886	1	0.0850	0.9575
ELM	0	0.0003	0.2759	0	0	0.4007
SVM	0.0617	0.0002	0.5308	0	0	0.5
LR	0.0579	0.0003	0.5388	0	0	0.5
SOM	0	0	0.5	1	0.8672	0.1373
DNN	0	0	0.5	0	0	0.5
DBN	0.0553	0.0003	0.5287	0	0	0.5
SDAE	0.7621	0.0158	0.9467	0	0.0001	0.5823
SDAE-ELM1	0.1934	0.0017	0.5958	0.9945	0.0136	0.9905
SDAE-ELM2	0.3155	0.0298	0.6429	0.9932	0.0124	0.9912
SDAE-ELM3	0.3222	0.0341	0.6111	0.9950	0.1089	0.9872

Table 22 – Time performance evaluation index.

Algorithm	KDD Cup99	NSL-KDD	UNSW-NB15	CIDDS-001
AdaBoost	4604s	3809s	5123s	6732s
DT	7800s	4328s	8763s	9087s
ELM	1039s	789s	2432s	4321s
SVM	>20h	>10h	>24h	>28h
LR	3421s	2319s	5342s	6123s
SOM	100s	78s	140s	200s
DNN	>7h	>4h	>12h	>15h
DBN	>7	>4h	>12h	>15h
SDAE	>7h	>4h	>12h	>15h
SDAE-ELM1	3908s	2987s	4321s	5891s
SDAE-ELM2	4109s	3034s	4498s	5987s
SDAE-ELM3	4530s	3298s	4510s	6032s

Table 23 – ADFA-LD binary test results.

Algorithm	Accuracy	Precision	Recall	F1-score
AdaBoost	0.8742	0.3654	0.0896	0.1439
DT	0.8606	0.4531	0.4458	0.4494
ELM	0.8724	0.7143	0.0766	0.1384
SVM	0.8775	0.8	0.0755	0.1379
LR	0.8590	0.25	0.0268	0.0484
SOM	0.4024	0.0952	0.4355	0.1563
DNN	0.8790	NaN	0	NaN
SDAE	0.8770	NaN	0	NaN
DBN	0.8721	0.3214	0.0892	0.1396
DBN-Softmax1	0.8790	0.8108	0.1230	0.2136
DBN-Softmax2	0.8790	0.8113	0.1241	0.2153
DBN-Softmax3	0.8831	0.8243	0.1342	0.2308

vidual type of attack data are contained in the intrusion detection dataset, and the experiment becomes a multi-class classification experiment, Table 24 gives the experimental results of multi-class classification of each data set. Comparing Tables 23 and Table 24, it can be seen that the overall detection effect of binary classification is better than that of multi-class classification, because for the detection of binary classification, if the attack type "Hydra_FTP" is mistakenly detected as the "Hydra_SSH" attack type, the detection result of the binary classification is still correct, but the detection result of the multi-class classification is wrong. From the results of the binary classification experiment and the multi-class classification experiment, it can be seen that the overall detection effect of the DBN-Softmax is better than other detection models, and the overall false positive rate of the multi-class classification is lower than other detection models, but in the multi-class classification, the detection performance of each algorithm on the attack type is not very good, mainly because of the 5951 sample data, the maximum number of "Hydra_SSH" attacks are only 176, and the minimum number of "Meterpreter" attacks are only 75, and the total amount of data is too

Table 24 – ADFA-LD multi-class classification test results.

Algorithm	Normal			Adduser			Hydra_FTP		
	TPR	FPR	AUC	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0.9988	1	0.4994	0	0.0012	0.4994	0	0	0.5
DT	0.9226	0.9150	0.5038	0.0313	0.0226	0.5043	0.0385	0.0368	0.5008
ELM	1	1	0.5039	0	0	0.5	0	0	0.5099
SVM	1	0.9962	0.5019	0	0	0.5	0	0	0.5
LR	0.9894	0.9876	0.5009	0	0	0.5	0.02	0.0053	0.5073
SOM	0.0228	0.0469	0.4625	0	0.4382	0.2760	0.4906	0.9007	0.2967
DNN	1	1	0.5	0	0	0.5	0	0	0.5
SDAE	0.9786	0.9694	0.5046	0	0	0.5	0.0702	0.0242	0.5
DBN	1	1	0.5	0	0	0.5	0	0	0.5
DBN-Softmax1	0.9641	0.9157	0.5813	0.0432	0	0.5356	0.0413	0	0.5223
DBN-Softmax2	0.9743	0.9212	0.5798	0.1032	0	0.5373	0.0798	0	0.5267
DBN-Softmax3	1	1	0.5	0.1143	0	0.5390	0.0942	0	0.5312

算法	Hydra_SSH			Java_Meterpreter			Meterpreter		
	TPR	FPR	AUC	TPR	FPR	AUC	TPR	FPR	AUC
AdaBoost	0	0	0.5	0	0.0012	0.4994	0	0.0006	0.4997
DT	0.0702	0.0322	0.5190	0.0857	0.0075	0.5391	0	0.0145	0.4926
ELM	0	0	0.5081	0	0	0.4997	0	0	0.5379
SVM	0.0213	0.0047	0.5083	0	0	0.5	0	0	0.5
LR	0.0147	0.0041	0.5053	0	0.0018	0.4991	0.0417	0.0036	0.5191
SOM	0.6140	0.9148	0.3516	0	0.2908	0.33	0	0.1525	0.3992
DNN	0	0	0.5	0	0	0.5	0	0	0.5
SDAE	0	0	0.5	0.0667	0.0146	0.5257	0	0	0.5
DBN	0	0	0.5	0	0	0.5	0	0	0.5
DBN-Softmax1	0.2341	0	0.6168	0.0270	0.0150	0.5044	0.1132	0.0018	0.5410
DBN-Softmax2	0.2432	0	0.6179	1	1	0.5	0.1243	0	0.5611
DBN-Softmax3	0.2498	0	0.6234	0.1043	0	0.5976	0.1034	0	0.5543

算法	Web_Shell		
	TPR	FPR	AUC
AdaBoost	0	0	0.5
DT	0.1035	0.0154	0.5440
ELM	0	0	0.4997
SVM	0	0	0.5
LR	0	0.0018	0.4991
SOM	0	0.7738	0.1114
DNN	0	0	0.5
SDAE	0	0	0.5
DBN	0	0	0.5
DBN-Softmax1	0.0432	0.0319	0.5675
DBN-Softmax2	0.0791	0.0432	0.5700
DBN-Softmax3	0.0913	0.0443	0.5708

small, so the detection rate is poor, but it proves the feasibility of DBN-Softmax in general.

It can be seen from Table 23 that the DBN-Softmax can obtain the best accuracy. Except for the SOM, the accuracy of the other algorithms is all greater than 85.9%. From the perspective of accuracy, as the depth of the DBN-Softmax increases, the accuracy of the algorithm is better. Compared with other algorithms, the accuracy of DBN-Softmax is the best, the optimal is 82.43%, which is 57.43% higher than the accuracy of LR. From the perspective of recall, DBN-Softmax is slightly worse than DT and SOM, but they are better than other algorithms. In most cases, the F1-Score of DBN-Softmax is better. The F1-Score of DBN-Softmax is second only to DT, which is 0.2186 worse than DT.

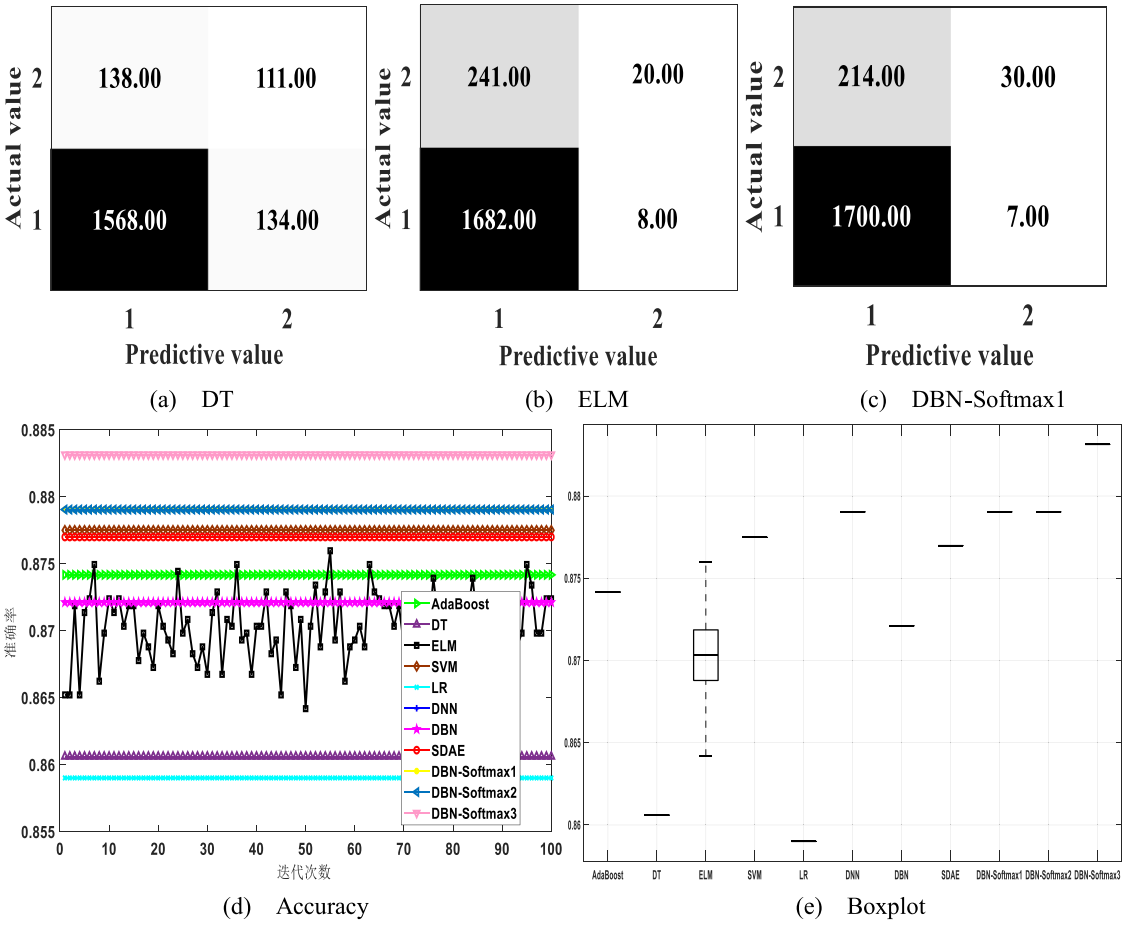


Fig. 18 – Binary confusion matrix, accuracy and boxplot.

It can be seen from Table 24 that the detection performance of the SOM is always poor. Overall, DBN-Softmax has achieved a good detection rate. For the "Normal" type, the true positive rate of each algorithm is better, and the true positive rate is greater than 92.26%, the true positive rate of some algorithms is even 100%, but each algorithm also has a high false positive rate. For the "Normal" type, which leads to a poor comprehensive evaluation index of the model, because the samples of other types of attacks are relatively few, which cannot accurately learn the characteristics of each attack type, causing each model to falsely report the attack sample as a normal sample, resulting in a high false positive rate; for the attack type, the detection effect of each algorithm is poor, in most cases, compared with other algorithms, the detection effect of DBN-Softmax is significantly improved, and as the depth of the model increases, the detection effect of the algorithm is better. Although there are attack types in ADFA-LD data set that do not appear in the network-based data set, the number of various types of samples is too small, which results in the detection performance of the algorithm being slightly worse than the detection performance of the algorithm on the network-based data set.

Fig. 18 is the ADFA-LD data set binary classification confusion matrix, accuracy, and boxplot. Among them, Fig. 18(a)-18(c) is the confusion matrix of some algorithms, and Fig. 18(d) is the accuracy, Fig. 18(e) is a boxplot of accuracy. As can be seen from Fig. 18(a)-18(c), the normal sample data occupies a considerable proportion in the test data set, and each algorithm has a good performance for the normal sample, the monitoring effect can be seen from the larger values of the third quadrant of each algorithm, but because the attack samples are too small, the prediction ability on the attack sample is poor. Fig. 18(d) and Fig. 18(e) show that the accuracy of the ELM has been in a fluctuating state during multiple iterations, the remaining algorithms can obtain the optimal accuracy at the early stage of the iteration, and it can be seen from the figure that the accuracy of DBN-Softmax3 is significantly better than the other algorithms.

To intuitively understand the distribution of the activation function values of the hidden layer of the model, we use t-SNE to visualize it. The feature mapping of the AdaBoost and DBN-Softmax1 are shown in Fig. 19(a) and Fig. 19(b), respectively. In the two algorithms, some of the "Normal" type features have appeared in a cluster, but the remaining "Normal" features and the remaining attack types have appeared in another cluster, which shows that the algorithm can identify some normal samples well at this time. For ELM and DBN-Softmax3, the connection records belonging to "Normal" are shown in Fig. 19(c) and 19(d), respectively. These connection records contain some features of other attack types, which

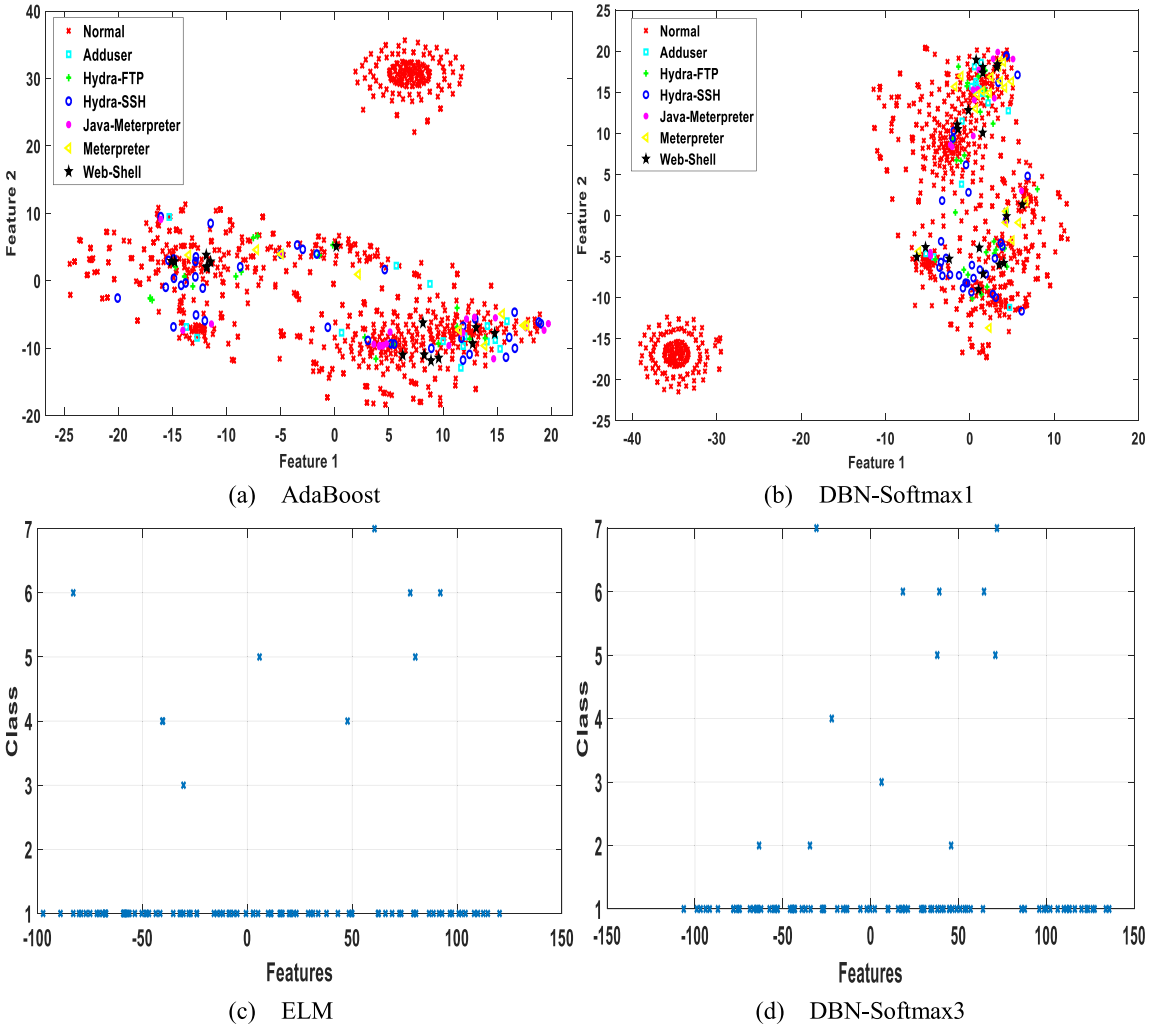


Fig. 19 – Feature mapping and connection record of last hidden layer activation function of multi-class classifications.

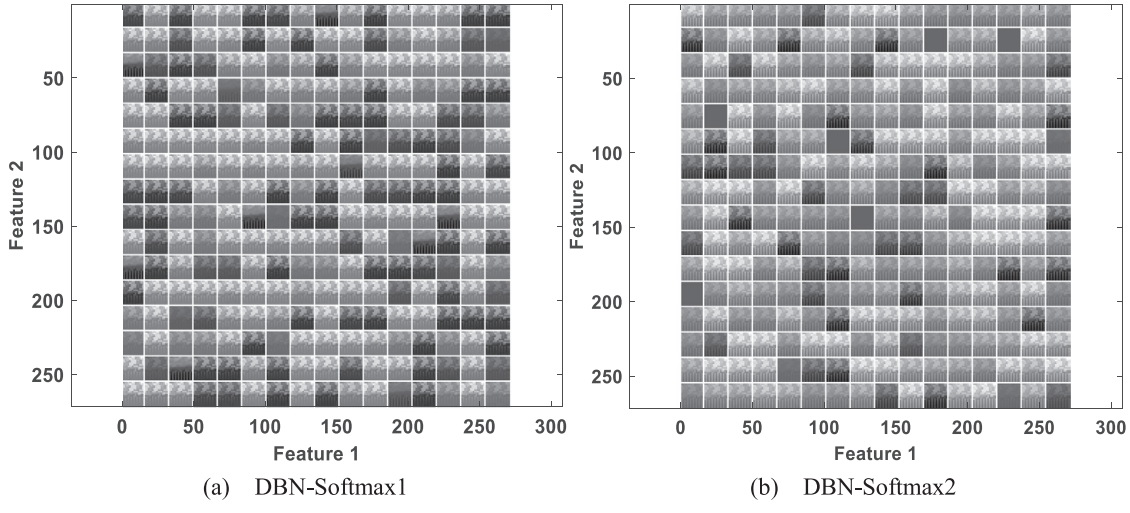
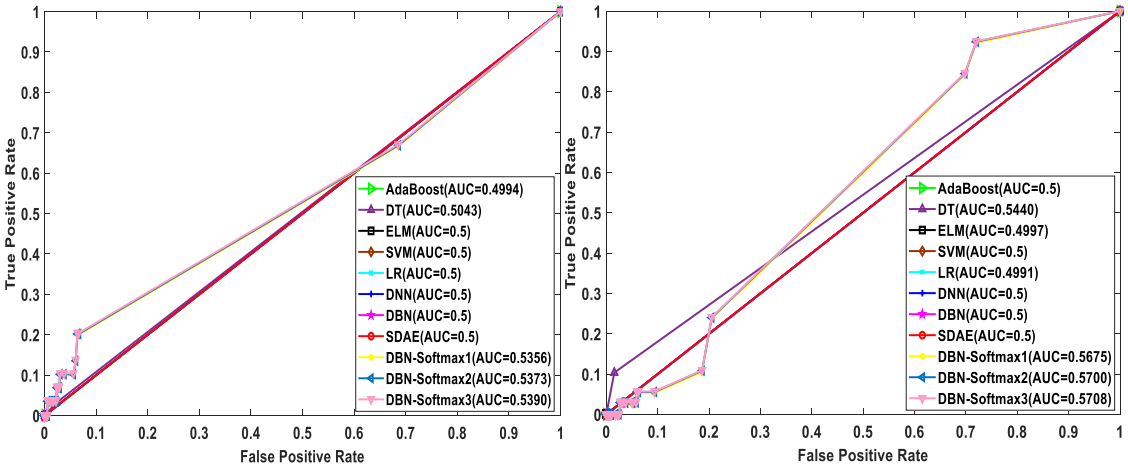


Fig. 20 – First RBM visualization weights of multi-class classifications.



(a) Adduser (b) Web-Shell

Fig. 21 – ROC curve of multi-class classifications.

indicates that although each algorithm can distinguish some normal samples, more features need to be added to better distinguish the types of attacks. In addition to visualizing the activation function value and connection weight of the hidden layer, we also visualize the hidden layer weights of the first RBM, and the size of the hidden layer weights can be clearly seen from Fig. 20.

The ROC curve is a comprehensive indicator for evaluating TPR and FPR. It is easy to understand. In the face of the imbalance of the number of positive and negative samples, the ROC curve is a more stable indicator that can reflect the quality of the model. Fig. 21 depicts the ROC curves. It can be seen from the figure that compared with other algorithms, the AUC value of DBN-Softmax is significantly improved, and as the number of layer increases, the AUC value of DBN-Softmax is better; for the of "Adduser" attack type, the AUC value of DBN-Softmax3 is 0.0396 higher than that of AdaBoost; for the "Web-Shell" attack type, the worst AUC value of ELM is 0.4997.

7. Conclusion and future work

Based on the deep Denoising AutoEncoder and Deep Belief Network, this paper has proposed the integrated deep intrusion models SDAE-ELM and DBN-Softmax. SDAE-ELM uses the distributed deep learning model of SDAE, which can handle real-time data, analyze large-scale data, and reduce the noise in the data set. The distributed deep learning model of DBN is used to deeply mine the features in the data set and improve the classification accuracy of the model. At the same time, in order to avoid the problems that the BP algorithm is prone to gradient sparseness, local optimization, and the original classifier is not suitable for multi-class classification during the fine-tuning process, we have used the ELM algorithm and Softmax classifier to optimize the SDAE and DBN models, respectively. In addition, in order to quickly update the parameters and improve the model computation efficiency, SDAE-ELM and DBN-Softmax are trained using the Mini-Batch gradient descent method. The SDAE-ELM and DBN-Softmax models have been verified using the network-based and host-based intrusion detection data sets, respectively. The results have demonstrated that no matter it is a binary classification or a multi-class classification, the detection performance of the SDAE-ELM and DBN-Softmax models on their respective data sets is better than the traditional machine learning models.

Compared with the traditional machine learning models, SDAE-ELM and DBN-Softmax models have achieved better detection results in intrusion detection. However, the data mining ability of SDAE-ELM model is effective, and the detection effect of small datasets is poor. In addition, the DBN-Softmax model has the disadvantage of long training time for large datasets, and cannot realize real-time detection of intrusions. In the future, we will consider using hybrid feature extraction technique to reduce the dimensionality of the dataset and reduce the training time of the model under the premise of ensuring the accuracy of the intrusion detection target. In addition, due to the complex structure of the deep intrusion detection model and the large number of parameters, we will consider to improve the model neurons and calculation methods, simplify the network structure, and improve the model efficiency.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Zhendong Wang: Writing - review & editing. **Yaodi Liu:** Validation, Formal analysis, Visualization, Supervision, Data curation, Writing - original draft. **Daojing He:** Writing - review & editing. **Sammy Chan:** Writing - review & editing.

Acknowledgements

This work is supported by National Natural Science Foundation of China (61562037, 61562038, 61563019, 61763017, U1936120, U1636216), the National Key Research and Development Program of China (2017YFB0802805 and 2017YFB0801701), the Natural Science Foundation of Jiangxi Province (20171BAB202026, 20181BBE58018).

REFERENCES

ADFA-LD 2020 dataset[*Online*], available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-IDS-Datasets/.

AI-Qatf M, Lasheng Y, AI-Habib M, AI-Sabahi K. Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access 2018;6:52843–56. doi:10.1109/ACCESS.2018.2869577.

AI-Tashi Q, Abdulkadir SJ, Rais HM, Mirjalili S, Mlhussian H, Ragab MG, Alqushaibi A. Binary multi-objective grey wolf optimizer for feature selection in classification. IEEE Access 2020;8:106247–63. doi:10.1109/ACCESS.2020.3000040.

Aljamal I, Tekeoglu A, Bekiroglu K, Sengupta S. Hybrid intrusion detection system using machine learning techniques in cloud computing environments. In: 2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications(SERA); 2019. p. 84–9. doi:10.1109/SERA.2019.8886794.

Bahmanyar R, Cui S, Datcu M. A comparative study of bag-of-words and bag-of-topics models of EO image patches. IEEE Geosci. Remote Sens. Lett. 2015;12(6):1357–61. doi:10.1109/LGRS.2015.2402391.

Bengio Y, Courville A, Vincent P. Representation Learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. 2013;35(8):1798–828. doi:10.1109/TPAMI.2013.50.

Besharati E., Naderan M., Namjoo E. LR-HIDS: logistic regression host-based intrusion detection system for cloud environments. 2019, 10(9): 3669–3692. 10.1007/s12652-018-1093-8.

Beulah JR, Punithavathani DS. A hybrid feature selection method for improved detection of wired/wireless network intrusions. Wirel. Pers. Commun. 2017;98(2):1853–69. doi:10.1007/s11277-017-4949-x.

Canedo DRC, Romariz ARSR. Intrusion detection system in Ad Hoc networks with artificial neural networks and algorithm K-means. IEEE Latin Am. Trans. 2019;17(7):1109–15. doi:10.1109/TLA.2019.8931198.

Chen CLP, Zhang C, Chen L, Gan M. Fuzzy restricted boltzmann machine for the enhancement of deep learning. IEEE Trans. Fuzzy Syst. 2019;23(6):2163–73. doi:10.1109/TFUZZ.2015.2406889.

CIDDS-001 2020 dataset[*Online*], available: https://www.hs-coburg.de/index.php?id=927.

de Araujo-Filho PF, Kaddoum G, Campelo DR, Santos AG, Macedo D, Zanchettin C. Intrusion detection for cyber-physical systems using generative adversarial networks in fog environment. IEEE Internet Things J. 2020. doi:10.1109/JIOT.2020.3024800.

Duan LT, Han DZ, Tian QT. Design of intrusion detection system based on improved ABC_elite and BP neural networks. Comput. Sci. Inf. Syst. 2019;16(3):773–95. doi:10.2298/CSIS181001026D.

Ghamisi P, Benediktsson JA. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. IEEE Geosci. Remote Sens. Lett. 2015;12(2):309–13. doi:10.1109/LGRS.2014.2337320.

Hamed T, Dara R, Kremer SC. Network intrusion detection system based on recursive feature addition and bigram technique. Comput. Sec. 2018;73:137–55. doi:10.1016/j.cose.2017.10.011.

HFSTE. Hybrid feature selections and tree-based classifiers ensemble for intrusion detection system. IEICE Trans. Inf. Syst. 2017;E100D(8):1729–37. doi:10.1587/transinf.2016ICP0018.

Hinton GE, Osindero S, The YW. A Fast learning algorithm for deep belief nets. Neural Comput. 2006;18(7):1527–54.

Hinton GE. Training products of experts by minimizing contrastive divergence. Neural Comput. 2002;14(8):1771–800. doi:10.1162/089976602760128018.

Huang G, Song S, Gupta JND, Wu C. Semi-supervised and unsupervised extreme learning machines. IEEE Trans. Cybern. 2014;44(12):2405–17. doi:10.1109/TCYB.2014.2307349.

Ibrahim NM, Zainal A. A model for adaptive and distributed intrusion detection for cloud computing. In: 2018 Seventh ICT International Student Project Conference(ICT-ISPC); 2018. p. 1–6. doi:10.1109/ict-ispc.2018.8523905.

IIgun K, Kemmerer RA, Porras PA. State transition analysis: a rule-based intrusion detection approach. IEEE Trans. Softw. Eng. 1995;21(3):181–99. doi:10.1109/32.372146.

Kachuee M, Darabi S, Moatamed B, Sarrafzadeh M. Dynamic feature acquisition using denoising autoencoders. IEEE Trans. Neural. Netw. Learn. Syst. 2019;30(8):2252–62. doi:10.1109/TNNLS.2018.2880403.

KDD 2020 Cup99 dataset[*Online*], available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Kim H, Hong H, Kim H-S, Kang S. A memory-efficient parallel string matching for intrusion detection systems. IEEE Commun. Lett. 2009;13(12):1004–6. doi:10.13140/RG.2.2.16174.82241.

Liang W, Li K, Long J, Kui X, Zomaya AY. An industrial network intrusion detection algorithm based on multifeature data clustering optimization model. IEEE Trans. Ind. Inf. 2020;16(3):2063–71. doi:10.1109/TII.2019.2946791.

Liu J, Pan Y, Li M, Chen ZY, Tang L, Lu CQ, Wang JX. Applications of deep learning to MRI images: a survey. Big Data Mining Anal. 2018;1(1):1–18. doi:10.26599/BDMA.2018.9020001.

Liu Y, Huangfu W, Zhang H, Long K. An efficient stochastic gradient algorithm to maximize the coverage of cellular networks. IEEE Trans. Wireless Commun. 2019;18(7):3424–36. doi:10.1109/TWC.2019.2914040.

Marteau P. Sequence covering for efficient host-based intrusion detection. IEEE Trans. Inf. Forensics Secur. 2019;14(4):994–1006. doi:10.13140/RG.2.2.16174.82241.

Messaoud S, Bradai A, Moulay E. Online GMM clustering and mini-batch gradient descent based optimization for industrial. IEEE Trans. Ind. Inf. 2020;16(2):1427–35. doi:10.1109/TII.2019.2945012.

Nair V, Hinton GE. Rectified linear units improved restricted Boltzmann machines Vinod Nair. Proceedings of the 27th International Conference on Machine Learning(ICML-10), June 21-24, 2020.

Nguyen MT, Kim K. Genetic convolutional neural network for intrusion detection systems. Fut. Gener. Comput. Syst. 2020;113:418–27. doi:10.1016/j.future.2020.07.042.

NSL-KDD 2020 dataset[*Online*], available: http://users.cis.fiu.edu/~lpeng/Datasets_detail.html.

Prabavathy S, Sundarakantham K, Shalinie SM. Design of cognitive fog computing for intrusion detection in internet of things. J.Commun. Netw. 2018;20(3):291–8. doi:10.1109/JCN.2018.000041.

Sadaf K, Sultana J. Intrusion detection based on autoencoder and isolation forest in fog computing. IEEE Access 2020;8:167059–68. doi:10.1109/ACCESS.2020.3022855.

Serpen G, Anghaei E. Host-based misuse intrusion detection using PCA feature extraction and KNN classification algorithms. Intell. Data Anal. 2018;22(5):1101–14. doi:10.3233/IDA-173493.
Shone N, Ngoc TN, Phai VD, Shi Q. A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Comput. Intell. 2018;2(1):41–50. doi:10.1109/TETCI.2017.2772792.
Si Z, Wen S, Dong B. NOMA codebook optimization by batch gradient descent. IEEE Access 2019;7:117274–81. doi:10.1109/ACCESS.2019.2936483.
Tang TA, Mhamdi L, McLernon D, Zaidi SAR, Chogho M. Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Coference on Wireless Networks and Mobile Communications(WINCOM); 2016. p. 258–63. doi:10.1109/WINCOM.2016.7777224.
Teng S, Mu N, Zhu H, Teng L, Zhang W. SVM-DT-based adaptive and collaborative intrusion detection. IEEE/CAA J. Automat. Sinica 2018a;5(1):108–18. doi:10.1109/JAS.2017.7510730.
Teng SH, Wu NQ, Zhu HB, Teng LY, Zhang W. SVM-DT-based adaptive and collaborative intrusion detection. IEEE-CAA J. Automat. Sinica 2018b;5(1):108–18. doi:10.1109/JAS.2017.7510730.
Tidjon LN, Frappier M, Mammar A. Intrusion detection systems: a cross-domain overview. IEEE Commun. Survey Tutor. 2019;21(4):3639–81. doi:10.1109/COMST.2019.2922584.
Tu Y, Du J, Lee C. Speech enhancement based on teacher-student deep learning using improved speech presence probability for noise-robust speech recognition. IEEE/ACM Trans. Audio Speech Lang Process 2019;27(12):2080–91. doi:10.1109/TASLP.2019.2940662.
UNSW-NB15 2020 dataset[*Online*], available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.
Usman AM, Yusof UK, Naim S. Filter-based multi-objective feature selection using NSGA III and cuckoo optimization algorithm. IEEE Access 2020;8:76333–56. doi:10.1109/ACCESS.2020.2987057.
Van der Maaten L, Hinton G. Visualizing Data using t-SNE[J]. J. Mach. Learn. Res. 2008;9(11):2579–625. doi:10.1007/s10846-008-9235-4.
Wang D, Su J, Yu H. Feature extraction and analysis of natural language processing for deep learning english language. IEEE Access 2020;8:46335–45. doi:10.1109/ACCESS.2020.2974101.
Wang W, Du X, Wang N. Building a cloud IDS using an efficient feature selection method and SVM. IEEE Access 2019;7:1345–54. doi:10.1109/ACCESS.2018.2883142.
Wei B, Zhang W, Xia X, Zhang Y, Yu F, Zhu Z. Efficient feature selection algorithm based on particle swarm optimization with learning memory. IEEE Access 2019;7:166066–78. doi:10.1109/ACCESS.2019.2953298.
Ye Z, Sun Y, Sun S, Zhan S, Yu H, Yao Q. Research on network intrusion detection based on support vector machine optimized with grasshopper optimization algorithm. In: 2019 10th IEEE International Conference on Intelligent Data Acquistion and Advanced Computing Systems: Technology and Applications(IDAACS); 2019. p. 378–83. doi:10.1109/IDAACS.2019.8924234.
Zaidi K, Milojevic MB, Rakocevic V, Nallanathan A, Rajarajan M. Host-based intrusion detection for vanets: a statistical approach to rogue node detection. IEEE Trans. Veh. Technol. 2016;65(8):6703–14. doi:10.1109/TVT.2015.2480244.
Zeng R, Wu J, Shao Z, Senhadji L, Shu H. Quaternion softmax classifier. Electron. Lett. 2014;50(25):1929–31. doi:10.1049/el.2014.2526.
Zhang H, Li Y, Lv Z, Sangaiah AK, Huang T. A real-time and ubiquitous network attack detection dased on deep delief detwork and support vector machine. IEEE/CAA J. Automat. Sinica 2020;7(3):790–9. doi:10.1109/JAS.2020.1003099.

Zhendong Wang [S'06, M'09] received the B.E (2006) and M. Eng. (2009) degrees from Changchun University of Science and Technology and Harbin University of Science and Technology respectively, and a Ph.D. degree in computer applied technology from Harbin Engineering University in 2013. Since 2014, he has been with the Department of Information Engineering, Jiangxi University of Science and Technology, P.R. China, where he is currently an associate professor. His-research interests include wireless sensor network, artificial intelligence and network security.

Yaodi Liu received the B.S. degree in Information and Computing Science from Jiangsu Ocean University, in 2018. She is currently pursuing the M.S. degree with Jiangxi University of Science and Technology. Her main research interests include network security and group intelligence optimization algorithms.

Daojing He [S'07, M'13] received the B.Eng. (2007) and M. Eng. (2009) degrees from Harbin Institute of Technology (China) and the Ph.D. degree (2012) from Zhejiang University (China), all in computer science. He is currently a professor in the School of Computer Science and Software Engineering, East China Normal University, P.R. China. His-research interests include network and systems security. He is on the editorial board of international journals such as IEEE Communications Magazine and IEEE Network.

Sammy Chan [S'87, M'89] received his B.E. and M.Eng. Sc. degrees in electrical engineering from the University of Melbourne, Australia, in 1988 and 1990, respectively, and a Ph.D. degree in communication engineering from the Royal Melbourne Institute of Technology, Australia, in 1995. Since December 1994 he has been with the Department of Electronic Engineering, City University of Hong Kong, where he is currently an associate professor.