Application of Deep Extreme Learning Machine in Network Intrusion Detection Systems

Li Wuke, Yin Guangluan*, and Chen Xiaoxiao

***Abstract*—Network intrusion detection has become a key technology to identify various network attacks. The traditional shallow methods based intrusion detection faces with the problem of 'curse of dimensionality' when computation happens in high-dimensional feature space. It fails to extract representative and abstract features from the high dimensional input, which reduces the detection accuracy. Therefore, an intrusion detection model based on deep learning framework with multi-layer extreme learning machine (ELM) is proposed. The proposed method is consisted of multiple extreme learning machine based auto-encoder (ELM-AE) in the front hidden layers and one ELM based classifier in the last hidden layer. The multiple ELM-AEs in the front hidden layers are utilized as unsupervised learning to extract deep features from the original input. Then the extracted features are substituted into the ELM in the last hidden layer as supervised learning to identify different types of attacks. The KDD99 dataset is utilized as the training and testing samples in the experiment. The results indicate that the detection accuracy of the proposed method is higher than some shallow methods (support vector machine and ELM), while the time consuming of the proposed method is much lower than the existing deep learning method (stacked auto-encoder).**

***Index Terms*—extreme learning machine, auto-encoder, deep neural network, intrusion detection, KDD99**

I. Introduction

INTRUSION detection technology is an important guarantee of computer network security system, which has been paid much attention by researchers in the field of network information security [1-5]. The purpose of the intrusion detection system is to identify unusual access or attacks on secure internal networks. The modeling of user behavior based on machine learning is an important research topic of the intrusion detection system. The intrusion detection system distinguishes the system normal and abnormal behavior by learning network traffic and host audit records.

Previous researchers have introduced various shallow learning methods into the intrusion detection system, such as neural networks [6, 7], K nearest neighbor algorithm [8, 9], support vector machine (SVM) [10, 11] and so on, all of which have made breakthroughs in intrusion detection system. If the input dimension is large, the aforementioned shallow learning methods fail to extract representative and abstract features from the original input, which may reduce the detection accuracy.

Different from shallow learning algorithms, deep learning algorithms can extract more representative and abstract features from the raw input by itself. Intrusion detection has been realized with deep learning methods in some literatures. Literature [12] uses deep belief network (DBN) to detect intrusion. DBN is utilized to reduce feature dimension. Since DBN is an unsupervised learning algorithm, it is more suitable for feature selection from a large number of unlabeled data. The stacked auto-encoder (SAE) based deep learning machines are proposed for intrusion detection in literature [13]. DBN and SAE use bottom-up unsupervised learning strategies to achieve pre-training, and top-down supervised learning strategies to realize fine-tune. DBN and SAE parameters are learned by back propagation (BP) algorithm. However, BP basically has two weakness: (1) BP based on gradient descent is easy to fall into local optimum; (2) large-scale iterative computation of DBN and SAE results in slow convergence speed (i.e., slow learning speed).

The extreme learning machine (ELM) proposed by Huang et al. [14] in 2006, whose input weights and hidden layer weights are generated by random initialization, has the advantages of fast learning speed and good generalization performance. ELM has been applied in intrusion detection in some literatures [15-17]. Subsequently, the ELM based auto-encoder (ELM-AE) has been proposed in literature [18]. ELM-AE can map the raw input data into another feature space. Refer to the structure of SAE, stacking multiple ELM-AEs layer by layer can extract deeper and more abstract feature from the raw input. Then the extracted features are utilized as the input for ELM to classify the intrusion type.

The rest of this paper proceeds as follows. In Section 2, we briefly review the existing extreme learning machine (ELM) and ELM based auto-encoder (ELM-AE). In Section 3, we propose the deep learning extreme learning machine (DLELM), especially the process of applying the DLELM for intrusion detection in detail. The proposed method is evaluated on actual intrusion dataset in Section 4. Section 5 concludes the work.

II. Brief Review of ELM and ELM-AE

A. Extreme Learning Machine (ELM)

Gradient-based learning algorithm has the disadvantages of slow training speed and poor generalization performance. To solve these problems, Huang et al. [14] proposed the extreme learning machine (ELM) algorithm. ELM consists of

Manuscript received June 15th, 2019; revised January 19th, 2020. This work was supported by Hunan Provincial Department of Education General Project Fund (No. 19C1255).

Li Wuke is with the Hunan University of Arts and Science ,Changde, Hunan Province, 415000, China; (e-mail: 258752552@ qq.com).

Yin Guangluan, the corresponding author, is with the Yongzhou Vocation Technical College,Yongzhou, Hunan Province, 425000, China. (e-mail: 342616427@ qq.com)

Chen Xiaoxiao is with the Hunan University of Arts and Science ,Changde, Hunan Province, 415000, China.

three layers of network, namely input layer, hidden layer and output layer. Given N training samples $\{\boldsymbol{x}_i, \boldsymbol{t}_i\}_{i=1}^{N}$, where $\boldsymbol{x}_i = [x_{i,1}\ x_{i,2}\ \cdots\ x_{i,n}]^{\mathrm{T}} \in \mathbb{R}^n$, $\boldsymbol{t}_i = [t_{i,1}\ t_{i,2}\ \cdots\ t_{i,n}]^{\mathrm{T}} \in \mathbb{R}^m$. Supposing the model contains L hidden layer nodes and m output layer nodes, and the activation function of the hidden layer is $g(\cdot)$, then the hidden layer output can be expressed in equation (1). If the model with L hidden nodes achieve the approximation of the given N training samples with zero error, the output layer is shown in equation (2).

$$\boldsymbol{h} = g(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b}) \tag{1}$$

$$\boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{t}_i^{\mathrm{T}},\ i = 1, 2, \cdots, N \tag{2}$$

where $\boldsymbol{w}$ is input weights, $\boldsymbol{b}$ represents hidden layer bias, $\boldsymbol{\beta}$ defines the output weights.

Equation (2) can be transformed into

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T} \tag{3}$$

where

$$\boldsymbol{H} = \begin{bmatrix} g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{w}_L \cdot \boldsymbol{x}_1 + b_L) \\ \vdots & & \vdots \\ g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{w}_L \cdot \boldsymbol{x}_N + b_L) \end{bmatrix}_{N \times L} \tag{4}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\beta}_L^{\mathrm{T}} \end{bmatrix}_{L \times m},\ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{t}_N^{\mathrm{T}} \end{bmatrix}_{N \times m} \tag{5}$$

Therefore, training ELM is equivalent to solving the least squares norm solution, and its expression is as follows

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{6}$$

where $\boldsymbol{H}^{\dagger}$ is a Moore-Penrose generalized inverse of matrix $\boldsymbol{H}$.

To improve ELM generalization ability, the target of training ELM is to minimize both the training error $\|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{T}\|^2$ and the norm solution of output weight $\|\boldsymbol{\beta}\|$. Then equation (3) can be solved by the following function

$$\begin{aligned} &\min\ L = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \delta\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 \\ &s.t.\ \ \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{t}_i^{\mathrm{T}} - \boldsymbol{\xi}_i^{\mathrm{T}},\ \ i = 1, 2, \cdots, N \end{aligned} \tag{7}$$

where $\boldsymbol{\xi}_i$ is the training error vector of m output nodes relative to training samples. δ>0 is a regularization factor, which controls the tradeoff between the output weights and the errors. Using the Karush-Kuhn-Tucker (KKT) theorem, the corresponding Lagrange function of the ELM optimization (7) is as follows:

$$L = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \delta\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 - \sum_{i=1}^{N}\alpha_i\left(\boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} - \boldsymbol{t}_i^{\mathrm{T}} + \boldsymbol{\xi}_i^{\mathrm{T}}\right) \tag{8}$$

The following optimality condition of equation (8) should be satisfied:

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = 0 \Rightarrow \boldsymbol{\beta} = \sum_{i=1}^{N}\alpha_i,\ \boldsymbol{h}(\boldsymbol{x}_i) = H\boldsymbol{\alpha} \tag{9}$$

$$\frac{\partial L}{\partial \boldsymbol{\xi}_i} = 0 \Rightarrow \alpha_i = \delta\boldsymbol{\xi}_i, i = 1, 2, \cdots, N \tag{10}$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \Rightarrow \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} - \boldsymbol{t}_i^{\mathrm{T}} + \boldsymbol{\xi}_i^{\mathrm{T}} = 0, i = 1, 2, \cdots, N \tag{11}$$

where $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_N]^{\mathrm{T}}$ is the vector of Lagrange variables. Substitute (9) and (10) into (11), then we have

$$\left(\frac{\boldsymbol{I}}{\delta} + \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\right)\boldsymbol{\alpha} = \boldsymbol{T} \tag{12}$$

where $\boldsymbol{I}$ is the identity matrix. Substitute (12) into (9), then we have

$$\hat{\boldsymbol{\beta}} = \left(\frac{\boldsymbol{I}}{\delta} + \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{T} \tag{13}$$

B. Extreme Learning Machine based Auto-encoder (ELM-AE)

Auto-encoder (AE) is an unsupervised neural network. The input is reconstructed by AE through encoding and decoding the input. Extreme learning machine based auto-encoder (ELM-AE) is a new neural network method, which can reconstruct input data as AE. Similar to ELM, ELM-AE contains input layer, hidden layer and output layer. The structure of ELM-AE is shown in Figure 1. The main difference between ELM-AE and traditional ELM is that ELM is a supervised learning algorithm whose output is the target category. ELM-AE is an unsupervised learning algorithm whose output is its input. Given N training samples $\{\boldsymbol{x}_i\}_{i=1}^{N}$, then the hidden layer output can be expressed in equation (14). The network output is shown in equation (15).

$$\boldsymbol{h} = g(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b}),\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} = \boldsymbol{I},\ \boldsymbol{b}^{\mathrm{T}}\boldsymbol{b} = \boldsymbol{I} \tag{14}$$

$$\boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{x}_i^{\mathrm{T}},\ i = 1, 2, \cdots, N \tag{15}$$

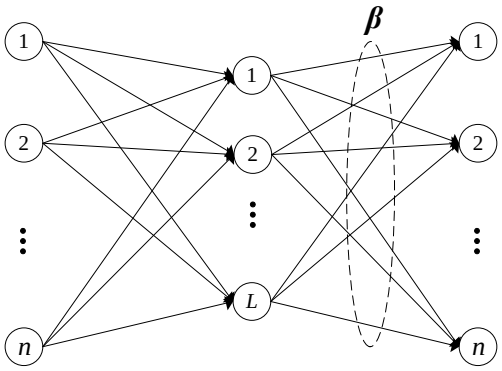


Fig. 1. ELM-AE structure

ELM-AE hidden layer parameters $\boldsymbol{w}$ and $\boldsymbol{b}$ need to be orthogonalized after random generation. In this way, input data can be mapped to random subspace effectively. Compared with ELM random initialization of input weights and hidden layer bias, orthogonalization can capture various edge features of input data better, so that the model can effectively learn the non-linear structure of data. The output weight can be calculated by formula (16).

$$\hat{\boldsymbol{\beta}} = \left(\frac{\boldsymbol{I}}{\delta} + \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{X} \tag{16}$$

In this case where the number of training samples is very large.

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{\mathrm{T}}\left(\frac{\boldsymbol{I}}{\delta} + \boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\right)^{-1}\boldsymbol{X} \tag{17}$$

In this case where the number of training samples is not too large.

III. Intrusion detection based on DLELM

To train the traditional deep learning model, unsupervised learning algorithm is firstly used to train the parameters of each layer, then the network is fine-tuned by supervised learning. However, the fine-tuning process takes a lot of time. In this section, a new deep neural network, deep learning extreme learning machine (DLELM) composed of multi-layers ELM-AEs and one ELM classifier is proposed. Unlike traditional deep learning model, there is no fine-tuning in DLELM, which reduce much more training time. After presenting the DLELM, the detail process applied in intrusion detection is introduced.

A. DLELM

DLELM is a stacked multi-layer models containing multiple pre-layers to extract data features by ELM-AE and one post-layer by ELM to classify the data. The structure of DLELM is shown in Figure 2. Supposing the number of hidden layer is M+1, the weight parameters need to train is $\boldsymbol{W} = \{\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_{M+1}\}$, where $\boldsymbol{W}_i$ is the weight of the ith hidden layer. We define $\boldsymbol{X}_i$ as the input of the ith ELM-AE. The input vector $\boldsymbol{X}$ of DLELM is treated as the input and target output for the first ELM-AE, i.e., $\boldsymbol{T}$=$\boldsymbol{X}_1$=$\boldsymbol{X}$. The output weight $\boldsymbol{\beta}_1$ can be computed according to equation (16). The output $\boldsymbol{H}_1$ of the hidden layer of the first ELM-AE is used as the input and target output for the second ELM-AE, where $\boldsymbol{H}_1 = g(\beta_1^{\mathrm{T}} \boldsymbol{H}_0)$, $\boldsymbol{T}$=$\boldsymbol{X}_2$=$\boldsymbol{H}_1$, $\boldsymbol{H}_0$=$\boldsymbol{X}$. Using the same method, all the network connection weights $\boldsymbol{\beta}_i$ before the Mth hidden layer can be obtained, where $i = 1, 2, \cdots, M$. According to the theory of auto-encoder, the weights of coding layer and decoding layer are transposed with each other. Therefore, the weights of network connection before the first Mth hidden layer are as follows: $\boldsymbol{W}_i = \boldsymbol{\beta}_i^{\mathrm{T}}$, where $i = 1, 2, \cdots, M$. Finally, the Mth hidden layer output $\boldsymbol{H}_k$ is utilized as the input of the M+1th hidden layer, which is classification layer based on ELM. The target output of the M+1th hidden layer is the marked label $\boldsymbol{T}$. The output weight of $\boldsymbol{W}_{M+1}$ can be calculated by equation (13). Thus, we have completed the training process of the DLELM.

B. Intrusion Detection by DLELM

The detail process applied in intrusion detection is described as follows.

Algorithm 1. Intrusion detection model training

Input: Intrusion dataset $\{\boldsymbol{x}_i, t_i\}_{i=1}^{N}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ and $\boldsymbol{t}_i \in \mathbb{R}^m$ are the intrusion feature vector and the corresponding label for sample i, the number of hidden layer M, the number of nodes in each hidden layer L_j, $j = 1, 2, \cdots, M$, the regularization factor δ in ELM.

Output: The predicted intrusion class of the training data.

(1) Supervised learning with ELM-AE on the 1st to M-1th hidden layer

① Set the original input $\{\boldsymbol{x}_i\}_{i=1}^{N}$ as the initial feature vector, i.e., $X_{fea}^{0} = \{\boldsymbol{x}_i\}_{i=1}^{N}$.

② For the hidden layer of the jth ELM-AE, generate L_j hidden nodes with randomly choosing input weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$, where $\boldsymbol{w}^T\boldsymbol{w} = \boldsymbol{I}$, $\boldsymbol{b}^T\boldsymbol{b} = \boldsymbol{I}$.

③ The output X_{fea}^{j-1} of the j-1th hidden layer is used as both input and output for the jth ELM-AE. Then compute the output weight $\boldsymbol{\beta}_j$ based on formula (16).

④ The output of the jth ELM-AE is obtained by the following formula,

$$\boldsymbol{X}_{fea}^{j} = \boldsymbol{\beta}_j^{\mathrm{T}} \boldsymbol{X}_{fea}^{j-1} \tag{18}$$

(2) Supervised learning with ELM on the Mth hidden layer

① The M-1th ELM-AE output X_{fea}^{M-1} and original intrusion label $\{\boldsymbol{t}_i\}_{i=1}^{N}$ are composed as the new training set $\{(\boldsymbol{X}_{fea}^{M-1}, \boldsymbol{t}_i)\}_{i=1}^{N}$. where $\boldsymbol{X}_{fea}^{M-1}$ is utilized as the input of the ELM based classification model, while $\{\boldsymbol{t}_i\}_{i=1}^{N}$ is used as the corresponding label.

② Compute the output weight $\boldsymbol{\beta}_M$.

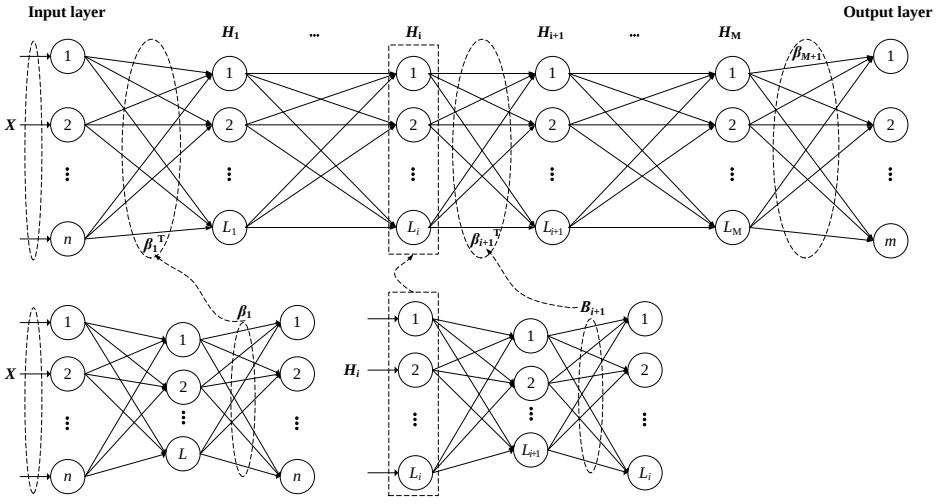


Fig. 2. DLELM structure

Algorithm 2. Intrusion detection with the established model

Model: The trained model according to **algorithm 1**.

Input: Intrusion data $\boldsymbol{x}^*$ to be detected, where $\boldsymbol{x}^* \in \mathbb{R}^n$.

Output: The predicted intrusion class of the data being detected.

① Substitute $\boldsymbol{x}^*$ into the *M*-1 ELM-AEs to extract deeper feature $\boldsymbol{x}_{fea}^{(M-1)*}$.

② Substitute $\boldsymbol{x}_{fea}^{(M-1)*}$ into the following formula to compute $\boldsymbol{h}(\boldsymbol{x}_{fea}^{(M-1)*})$

$$\boldsymbol{h}(\boldsymbol{x}_{fea}^{(M-1)*}) = g(\boldsymbol{w} \cdot \boldsymbol{x}_{fea}^{(M-1)*} + \boldsymbol{b}) \tag{19}$$

③ Substitute $\boldsymbol{h}(\boldsymbol{x}_{fea}^{(M-1)*})$ into the following formula to calculate $f(\boldsymbol{x}^*)$

$$f(\boldsymbol{x}^*) = \boldsymbol{h}(\boldsymbol{x}_{fea}^{(M-1)*})\boldsymbol{\beta}_M \tag{20}$$

where $\boldsymbol{\beta}_M$ is the output weight of the *M*th hidden layer

④ Intrusion type decision formula is as follows,

$$label(\boldsymbol{x}^*) = \arg\max_{i=1,\cdots,m} f_i(\boldsymbol{x}^*) \tag{21}$$

where $f_i(\boldsymbol{x}^*)$ is the *i*th output node, and $f(\boldsymbol{x}^*) = [f_1(\boldsymbol{x}^*) \cdots f_m(\boldsymbol{x}^*)]$.

IV. Illustrative Example

A. Experiment Setup

1 Datasets

KDD99 dataset is the authoritative test data in the field of intrusion detection. It is set up by Lincon Laboratory of Massachusetts Institute of Technology to simulate the LAN environment of the US Air Force. In this experiment, the KDD99 dataset with 10% quantity of the total is used, which contains 494021 training samples and 311029 test samples. The dataset mainly has four types of attack: denial of service (DoS) attack, remote to local (RtL) attack, user to root (UtR) attack and port monitoring/ scanning (PMS) attack. Each sample has forty-two attribute records, of which the first forty-one attributes are features and the last one is classification label. Among the forty-one features, thirty-eight ones are digital, and the rest three ones are symbolic. In order to verify the effectiveness of the proposed method, three groups of training and testing data are randomly selected, shown in Table 1.

2 Preprocessing

The dataset contains three symbolic features: 'protocol_type', 'service' and 'flag'. The symbolic features need be transformed to digital ones. The 'protocol_type' has three types of symbolic value: 'tcp', 'udp' and 'icmp'. The 'tcp', 'udp' and 'icmp' are assignment with numerical value 1, 2 and 3 respectively. The 'service' has seventy types of symbolic value. Similar to 'protocol_type', the numerical value of 'service' is 1 to 70. The 'flag' has eleven types of symbolic value. And the numerical value of 'flag is 1 to 11.

To eliminate the dimension effect among different attributes, normalization is also needed. In this paper, the training data and test data are normalized by the maximum-minimum specification, i.e., the data are normalized to the range of [0,1]. The formula is as follows:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{22}$$

where x is the original value of one feature, $x_{\min}$ is the minimum value of this feature, $x_{\max}$ is the maximum value of this feature.

3 Evaluation standards

In this experiment, training time, testing time, accuracy (AC) of the testing samples, detection rate (DR) and false alarm rate (FAR) are used as evaluation indicators. AC, DR and FAR are defined as follows:

AC = The number of correct recognition / The number of the whole samples × 100%

DR = The number of attack samples of correct recognition / The number of the whole attack samples × 100%

FAR = The number of normal samples of false recognition / The number of the whole normal samples × 100%

B. DLELM Parameters Effect

The dataset Data1 is utilized to evaluate the effectiveness of different parameters in DLELM. The influence of three factors is discussed in this section, which are the regularization factor δ in the last hidden layer (classification layer), the number of nodes L in the last hidden layer.

1 The effect of δ and L in the last hidden layer

The regularization factor δ and the number of nodes L are two parameters in the last hidden layer (classification layer). The DLELM with the structure of 41-1000-1000-L is used as an example, where 41 is the number of input layer nodes (equal to the number of features), 1000 is the number of hidden nodes in the first two ELM-AE layers, L is the number of hidden nodes in the ELM based classification layer. L is chosen from 200 to 2000 with the interval of 200, while δ is selected from exponential sequence $\{2^{-25}, 2^{-24}, \cdots, 2^{25}\}$. Figure 3 shows the accuracy (AC) of the testing samples with different L and δ. It can be seen that L has little effect on the accuracy of testing samples, while δ with small value reduces the accuracy. We choose δ = 1 and L = 500 in this experiment.

TABLE I
TRAINING AND TESTING DATASET

	Training set			Testing set		
	Normal	Abnormal	Total	Normal	Abnormal	Total
Data1	12349	6469	18818	8821	5998	14819
Data2	9311	6863	16174	5881	4145	10026
Data3	6861	4117	10978	4803	2058	6861

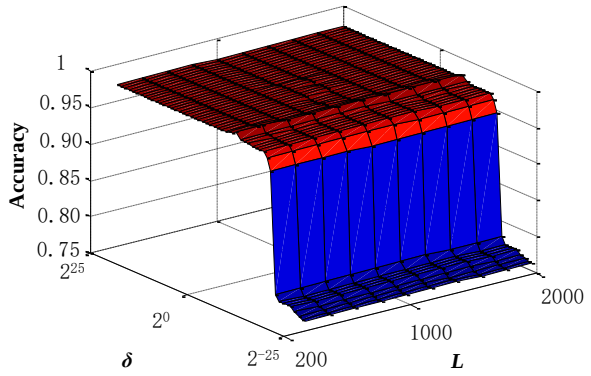


Fig. 3. The accuracy (AC) of the testing samples with different L and δ

2 The influence of the network depth

The depth of DLELM network plays an important role in intrusion detection. With the increase of DLELM layers, the modeling ability is enhanced, and the feature representation ability of deep layer is more abstract. On the other hand, As the number of layers increasing, the number of hidden layer nodes increases, and the training time also increases greatly. Excessive layers easily lead to over-fitting problem, which may reduce the classification accuracy.

DLELM with five different network depths is conducted to compare the accuracy of the testing samples. DLELM^2 has two hidden layers. The structure of DLELM^2 is 41-1000-500, where 41 is the number of input layer nodes, 1000 is the number of hidden nodes in the first hidden ELM-AE layers, 500 is the number of hidden nodes in the ELM based classification layer. Similarly, DLELM^3, DLELM^4 and DLELM^5 represent DLELM with the structure of 41-1000-1000-500, 41-1000-1000-1000-500, 41-1000-1000-1000-1000-500 respectively. The accuracy (AC) and detection rate (DR) of the testing data are shown in Figure 4. Observed from Figure 3, DLELM with three hidden layers has the highest AC and DR.

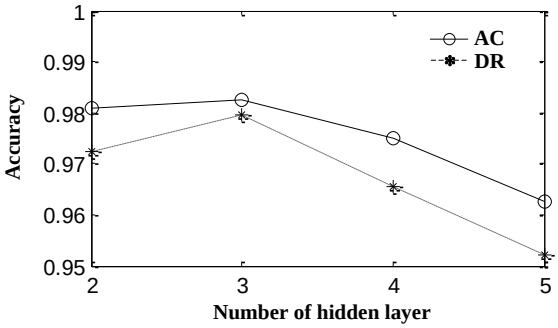
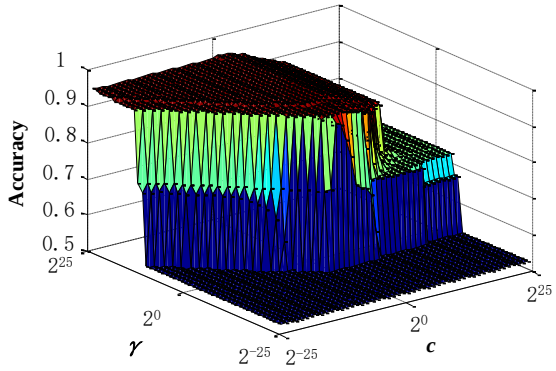


Fig. 4. AC and DR of DLELM with different depths

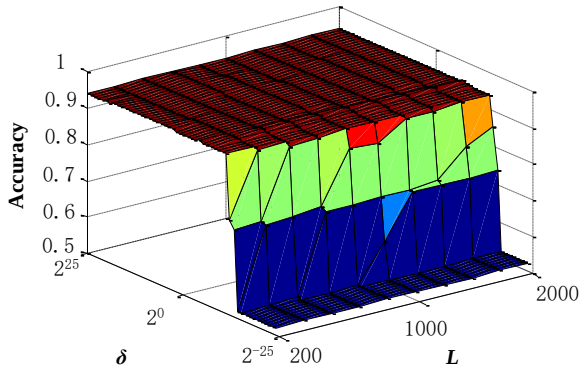
C. Comparison with other algorithms

In order to test the effectiveness of DLELM, three kinds of data driven algorithms are selected as comparison: support vector machine (OCSVM), the standard extreme learning machine (ELM) and stacked auto-encoder (SAE). SVM and ELM are two classification model with single hidden layer, while SAE is the classification model with multiple hidden layers. The libsvm toolbox is implemented as the SVM classifier [19]. The process of ELM is shown in section 2.1. All experiments are conducted in Matlab 2011b, PC with 2.5 GHz CPU and 4 GB memory.

As mentioned in section 4.2, the optimal parameters of DLELM is with the structure of 41-1000-1000-500, and δ = 1. For SVM, γ in kernel function and cost parameter c are both chosen from $\{2^{-25}, 2^{-24}, \cdots, 2^{25}\}$. For ELM, the number of hidden layer nodes L is chosen from 200 to 2000 with the interval of 200, while δ is selected from exponential sequence $\{2^{-25}, 2^{-24}, \cdots, 2^{25}\}$. The accuracy (AC) of the testing samples from Data1 with different parameters for SVM and ELM is shown in Figure 5. Seen from Figure 5, the highest accuracy of SVM and ELM is 95.15% and 95.46% respectively, shown in Table 2. In addition, it can be seen from Figure 5 that two parameters γ and c determine the accuracy of SVM, while only parameter δ determines the accuracy of ELM. L has little effect on the accuracy for ELM. It is more different to optimize two parameters comparing with only one parameter, which is another advantage for ELM.



(a) SVM

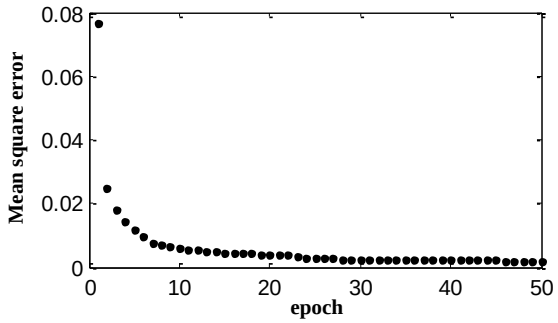


(b) ELM

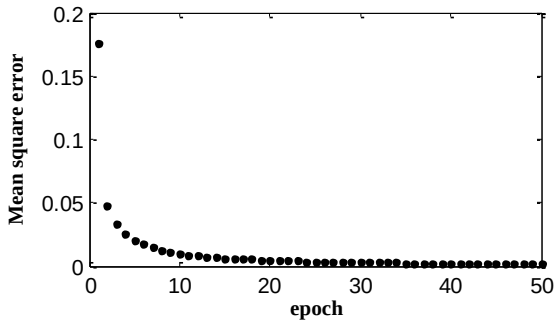
Fig. 5. The test accuracy of different parameters for SVM and ELM

For SAE, the network structure is the same as DLELM, namely 41-1000-1000-500, the learning rate is 0.1, the number of epochs is 30, the size of batch is 1, the model parameters are adjusted based on back propagation (BP) algorithm, where the mean squared error is used as the lost function. The mean squared error on training set of Data1 of the first hidden layer (with 1000 hidden nodes), the second hidden layer (with 1000 hidden nodes) and the last hidden layer (with 500 hidden nodes) are shown in Figure 5. It can be

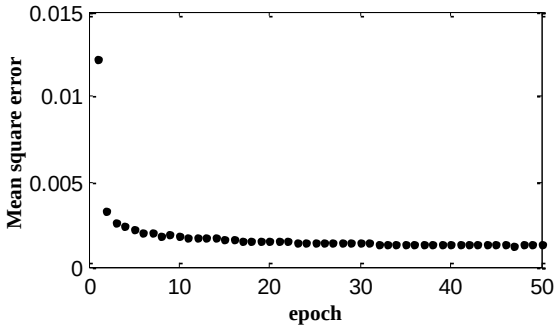
seen from Figure 6 that the training result is convergent. Using the trained model parameters, the test accuracy of SAE is 98.13%, shown in Table 2.



(a) Layer 1



(b) Layer 2



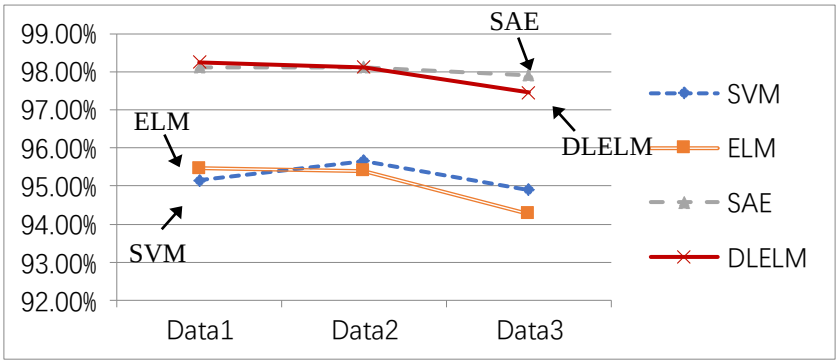
(c) Layer 3

Fig. 6. The mean squared error on training set of SAE

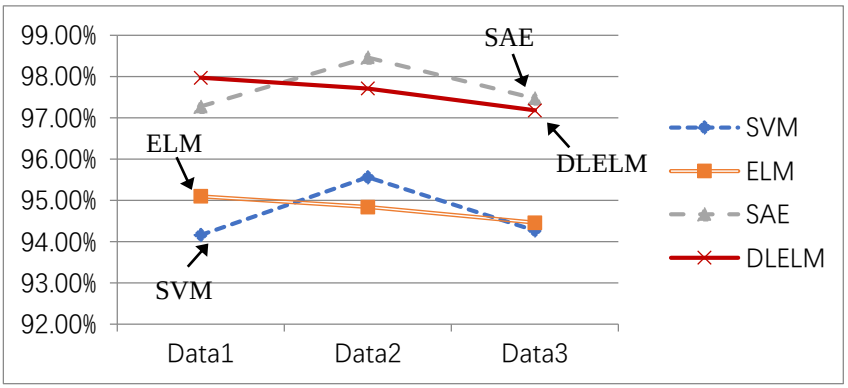
The accuracy (AC), detection rate (DR) and false alarm rate (FAR) of the testing samples for Data1, Data2 and Data3 are shown in Table 2 and Figure 7. As can be seen from Table 2 and Figure 7, the accuracy and detection rate obtained by SVM and ELM are lower than SAE and DLELM. The accuracy and detection rate of SAE is similar to DLELM. While the false alarm rate of DLELM is the lowest. Training and testing time are shown in Table 2. For intrusion detection system, the cost of testing time is more important, because less time consuming means faster intrusion detection. Seen from Table 2, SVM and ELM spend the lowest time for training and testing. Time consuming of DLELM is higher than SVM and ELM. However, SAE spends much more training and testing time than DLELM.

TABLE II
TESTING RESULTS OF DIFFERENT METHODS

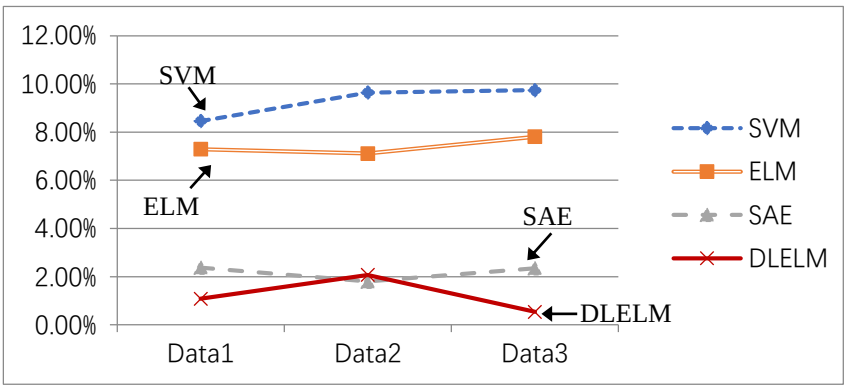
		Data1	Data2	Data3
SVM	AC	95.15%	95.66%	94.90%
	DR	94.16%	95.56%	94.27%
	FAR	8.46%	9.64%	9.74%
	Training time (s)	0.6	0.5	0.3
	Testing time (s)	0.4	0.3	0.2
ELM	AC	95.46%	95.40%	94.27%
	DR	95.10%	94.84%	94.46%
	FAR	7.29%	7.11%	7.81%
	Training time (s)	1.9	1.5	1.0
	Testing time (s)	0.6	0.5	0.3
SAE	AC	98.13%	98.32%	97.92%
	DR	97.27%	98.46%	97.47%
	FAR	2.38%	1.80%	2.35%
	Training time (s)	1314.8	1162.4	722.2
	Testing time (s)	8.5	7.5	4.6
DLELM	AC	98.25%	98.13%	97.46%
	DR	97.97%	97.71%	97.18%
	FAR	1.08%	2.07%	0.54%
	Training time (s)	3.5	2.7	1.9
	Testing time (s)	2.0	1.8	1.1



(a) Accuracy



(b) Detection rate



(c) False alarm rate

Fig. 7. The accuracy (AC), detection rate (DR) and false alarm rate (FAR) of the testing samples

V. CONCLUSIONS

The deep learning extreme learning machine (DLELM) is a new data-driven method for intrusion detection. The proposed method can extract more representative features and improve intrusion detection accuracy. Firstly, the input data are mapped into deep feature space by the multiple stacked ELM-AEs. Then the mapped features are classified to normal or intrusion by the last hidden layer based on ELM. The proposed DLELM based intrusion detection is verified on KDD99 dataset. The experimental results show that the detection accuracy and detection rate of DLELM are higher than the shallow learning methods (SVM and traditional ELM), and the false alarm rate of DLELM is lower than SVM and ELM. The detection accuracy, detection rate and false alarm rate of DLELM are similar to the deep learning method SAE, but the training time and testing time of DLELM are much lower than SAE. DLELM model improves the accuracy and speed of intrusion detection. It is a feasible and efficient intrusion detection model, which provides a new research idea for intrusion detection.

REFERENCES

[1] M. Češka, V. Havlena, L. Holík, O. Lengál and T. Vojnar. "Approximate Reduction of Finite Automata for High-Speed Network Intrusion Detection," *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* 2018, pp. 155-175.

[2] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis and R Atkinson, "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey,", 2017.

[3] C. Ioannou, V. Vassiliou and C.Sergiou, "An Intrusion Detection System for Wireless Sensor Networks," *International Conference on Telecommunications* 2017, pp. 253-259.

[4] J. Kizza, F. M. Kizza, "Intrusion Detection and Prevention Systems," *Transactions of the Institute of Electrical Engineers of Japan A*, vol. 118, no. 2, pp. 502-508, 2018.

[5] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41-50, 2018.

[6] I. Benmessahel, K. Xie and M. Chellal, "A new evolutionary neural networks based on intrusion detection systems using multiverse optimization," *Applied Intelligence*, vol.48, no. 8, pp. 2315-2327, 2018.
[7] G. Wang, J. X. Hao, J. Ma and L. H. Huang, "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering," *Expert Systems with Applications An International Journal*, vol. 37, no. 9, pp. 6225-6232, 2010.
[8] A. H. Farooqi and A. Munir, "Intrusion Detection System for IP Multimedia Subsystem using K-Nearest Neighbor classifier," *Multitopic Conference, 2008 INMIC 2008 IEEE International 2008*, pp. 423-428.
[9] S. Malhotra, V. Bali and K. K. Paliwal, "Genetic programming and K-nearest neighbour classifier based intrusion detection model," *International Conference on Cloud Computing, Data Science & Engineering – Confluence* 2017, pp. 42-46.
[10] F. J. Kuang, W. H. Xu and S. Y. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Applied Soft Computing Journal*, vol. 18, no. C, pp. 178-184, 2014.
[11] S. H. Teng, N. Q. Wu, H. B. Zhu, L. Y. Teng and W. Zhang, "SVM-DT-based adaptive and collaborative intrusion detection," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 108-118, 2018.
[12] B. Y. Wang, S. Sun and S. M. Zhang, "Research on Feature Selection Method of Intrusion Detection Based on Deep Belief Network," *Proceedings of the 2015 3rd International Conference on Machinery, Materials and Information Technology Applications* 2015, pp. 1-5.
[13] O. Kaynar, A. G. Yüksek, Y. Görmez and Y. E. Işik, "Intrusion detection with autoencoder based deep learning machine," *Signal Processing and Communications Applications Conference* 2017.
[14] G. B. Huang, Q. Y. Zhu and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489-501, 2006.
[15] Y. Li, R. Qiu, S. Jing, "Intrusion detection system using Online Sequence Extreme Learning Machine (OS-ELM) in advanced metering infrastructure of smart grid," *Plos One*, vol. 13, no. 2, 2018.
[16] C. R. Wang, R. F. Xu, S. J. Lee and C. H. Lee, "Network Intrusion Detection Using Equality Constrained-Optimization-Based Extreme Learning Machines," *Knowledge-Based Systems*, vol. 147, 2018.
[17] Y. Yi, L. K. Song and Q. He, "A new network intrusion detection algorithm: DA-ROS-ELM," *Ieej Transactions on Electrical & Electronic Engineering*, vol. 13, no.1, 2018.
[18] K. Sun, J. S. Zhang, C. X. Zhang and J. Y. Hu, "Generalized extreme learning machine autoencoder and a new deep neural network," *Neurocomputing*, vol. 130, 2016.
[19] C. C. Chang, C. J. Lin, "Libsvm: A Library for Support Vector Machines," Available from: https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Li Wuke acts as a lecturer in Hunan University of Arts and Science. Her interest is network information security and database technology.
Yin Guangluan is a lecturer in the Yongzhou Vocation Technical College. Her interest includes cloud computing technology and application.
Chen Xiaoxiao works as a lecturer in Hunan University of Arts and Science. Her interest is Communication network technology and data mining.