IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 25, 2018, accepted July 18, 2018, date of publication July 23, 2018, date of current version August 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2858277

Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System

BINGHAO YAN AND GUODONG HAN
National Digital Switching System Engineering & Technology Research Center, Zhengzhou 450001, China

Corresponding author: Binghao Yan (ndscybh@qq.com)

This work was supported in part by the National Science Technology Major Project of China under Grant 2016ZX01012101, in part by the National Natural Science Foundation Project of China under Grant 61572520, and in part by the National Natural Science Foundation Innovation Group Project of China under Grant 61521003.

ABSTRACT Classification features are crucial for an intrusion detection system (IDS), and the detection performance of IDS will change dramatically when providing different input features. Moreover, the large number of network traffic and their high-dimensional features will result in a very lengthy classification process. Recently, there is an increasing interest in the application of deep learning approaches for classification and learn feature representations. So, in this paper, we propose using the stacked sparse autoencoder (SSAE), an instance of a deep learning strategy, to extract high-level feature representations of intrusive behavior information. The original classification features are introduced into SSAE to learn the deep sparse features automatically for the first time. Then, the low-dimensional sparse features are used to build different basic classifiers. We compare SSAE with other feature extraction methods proposed by previous researchers. The experimental results both in binary classification and multiclass classification indicate the following: 1) the high-dimensional sparse features learned by SSAE are more discriminative for intrusion behaviors compared to previous methods and 2) the classification process of basic classifiers is significantly accelerated by using high-dimensional sparse features. In summary, it is shown that the SSAE is a feasible and efficient feature extraction method and provides a new research method for intrusion detection.

INDEX TERMS Intrusion detection, deep learning, machine learning, SSAE, feature extraction.

I. INTRODUCTION

With the advent of new Internet technologies such as file sharing, mobile payment and instant messaging, the situation of network security is becoming ever more complicated. At the same time, network attackers become more invisible and attack costs are further reduced, all of which seriously threaten the network security environment. As an active defense technology, intrusion detection has gradually become the key technology to ensure the security of the network system. Intrusion detection system (IDS) is designed for network security proactive protection system, which is based on a certain security strategy to monitor the operation of the network system and found a variety of intrusion behavior, attempt or result, and automatically respond to effectively prevent illegal access or Intrusion. IDS usually includes misuse detection and anomaly detection two kinds of processing methods. The misuse detection system needs to accurately define the intrusive behavior mode in advance, and the intrusion behavior is detected if the attacker's attack pattern exactly matches the pattern library in the detection system. Anomaly detection system considers intrusion activity is unknown, is a subset of abnormal activity. When any behavior deviates from the normal behavior pattern to a certain extent, it is considered as an invasion event.

In order to improve the efficiency and performance of intrusion detection systems, in recent years, scholars have used machine learning methods in the construction of network intrusion detection systems and achieved breakthroughs progress [1]. However, most machine learning algorithms can only achieve satisfactory results in small sample datasets. When these algorithms are actually used in large-scale intrusion detection systems, they usually face the limitations of time complexity and space complexity. The essential reason is attributable to the input data in feature space with high

dimensional and nonlinear characteristics. Therefore, it is an indispensable step in the intrusion detection process to propose or adopt more effective methods to perform dimension reduction on high-dimensional data.

In 2006, Hinton, a professor at the University of Toronto in Canada, published an article on deep learning in Science [2], setting off a wave of research on big data and artificial intelligence. One of the main messages delivered in this article is that deep artificial neural networks (DANN) with many hidden layers have excellent feature-based learning capabilities and the learned features have a more substantive characterization of the original data to facilitate visualization or classification. Moreover, DANN reduce the huge workload in the process of feature extraction and improve the efficiency of feature extraction.

Deep learning is a promising solution to the challenge of intrusion detection because of its outstanding performance in dealing with complex, large-scale data. So we use the stack autoencoder (SAE) model based on deep learning theory to perform unsupervised dimension reduction of intrusion detection samples in this paper, and then we add sparsity constraints to the SAE model to improve the generalization ability and classification accuracy of the model. Compared with the existing methods, SSAE can not only effectively compress the feature dimension of the original data and accelerate the classification process of the classifier, but also the feature extraction capability of SSAE is obviously better than the known methods.

In the following sections, we provide the related work in terms of network intrusion detection methods related to machine learning and deep learning (Section II), our deep learning based approach to feature extraction (Section III), and our test bed, dataset and experimental preparation process (Section IV). Section V highlights SSAE with a discussion about the experimental results and a comparison with a few previous methods. We present in section VI with a summary of conclusions and future work.

II. RELEVANT WORK

The initial intrusion detection methods include misuse detection [3] and anomaly detection [4]. Although these two detection methods have achieved good results, there are still inherent flaws, among which misuse detection cannot determine whether an unknown behavior is safe, abnormal detection has a disadvantage of high false alarm rate, and the deactivation of alarms relies heavily on the domainspecialist knowledge. With the development of artificial intelligence theory, intrusion detection technology based on machine learning makes up for the defects of the original methods and becomes a new research hotspot. In [5], the authors combined artificial neural network and fuzzy clustering algorithm to solve the problem of low detection rate of low-frequency attacks and help IDS achieve higher detection rate, less false positive rate and stronger stability. Bamakan *et al.* [6] introduced time-varying chaos particle swarm optimization into IDS to adaptively select the parameters of support vector machine (SVM), which has the characteristics of high detection rate and low false alarm rate. In addition, other representative machine learning algorithms such as decision trees [7], Bayesian classification [8], and K-nearest neighbors [9] are also used in the construction of IDS. Furthermore, in order to avoid the existence of defects in a single classifier, the ideas of hybrid classifiers [10], [11] and ensemble classifiers [12] are also applied in IDS and the efficiency of classification is generally better than single classifier models.

Although the above methods have strong adaptability and scalability, researchers believe that while considering the detection rate and false alarm rate performance, IDS also needs to meet the requirements of systems for real-time capability and low power consumption. Ambusaidi *et al.* [13] combined flexible mutual information feature selection with least square SVM, which contributes to lower computational costs and faster detection speeds. Osanaiye *et al.* [14] combined the output of four filter methods into an ensemble-based multi-filter feature selection method to achieve an optimum selection and shown encouraging performance in DDOS detection. However, the above-mentioned feature selection algorithms only sort the manually designed features according to the difference of the detection targets of the IDS, and remove the features that are more redundant in comparison, resulting in the loss of part of the information.

The popularity of deep learning technology has further promoted the progress of intrusion detection systems. On the one hand, deep learning algorithms can uncover deeper relationships between input data, which cannot be achieved with traditional shallow machine learning algorithms. On the other hand, deep learning algorithms have more powerful feature extraction and representation capabilities while retaining data information as much as possible. The deep learning methods used in intrusion detection mainly include Deep Convolutional Neural Network (DCNN) [15], Recurrent Neural Network (RNN) [16], Deep Belief Network (DBN) [17] and their improved models, and the main purpose of these model is also to improve the detection rate and false alarm rate. Deep learning models used in intrusion detection rarely focus on feature extraction. Therefore, in this paper, we use stacked sparse autoencoder (SSAE) as feature extraction method in IDS. Compared with previous works, we use the SSAE for feature extraction rather than for sample anomaly detection.

III. MATHODOLOGY

A. ANTOENCODER

Autoencoder(AE) is an unsupervised three-layer neural network, including an input layer, a hidden layer, and an output layer (also referred to as reconstruction layer). The typical structure of AE is shown in Figure 1, and the representation is shown in Figure 2.

The AE can gradually transform specific feature vectors into abstract feature vectors, which can well realize the nonlinear transformation from high dimensional data space to low dimensional data space. The working process of the

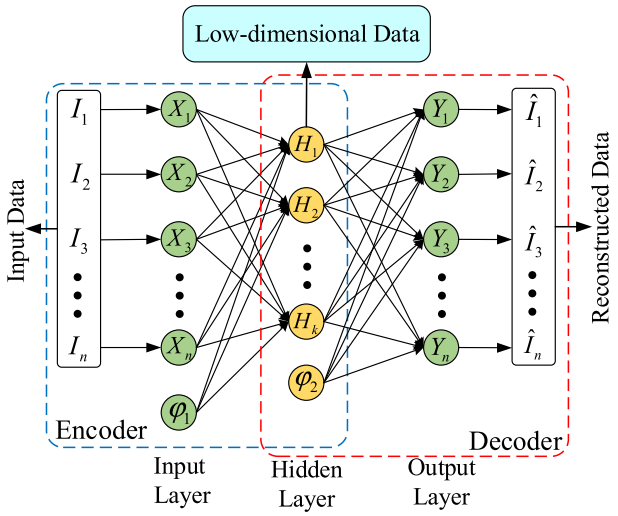


FIGURE 1. The structure of basic Autoencoder.

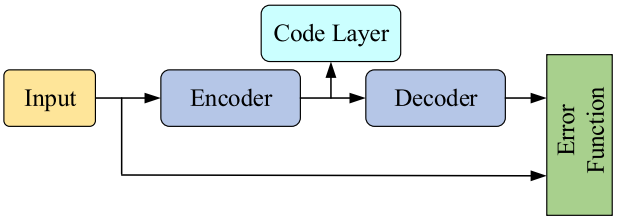


FIGURE 2. Autoencoder representation.

automatic encoder can be divided into two stages: encoding and decoding and these two steps can be defined as:

The encoding process from the input layer to the hidden layer:

$$H = g_{\theta_1}(X) = \sigma(W_{ij}X + \varphi_1) \tag{1}$$

The decoding process from the hidden layer to the reconstruction layer:

$$Y = g_{\theta_2}(H) = \sigma(W_{jk}H + \varphi_2) \tag{2}$$

In the above formulas, $X = (x_1, x_2, \ldots, x_n)$ is the input data vector, $Y = (y_1, y_2, \ldots, y_n)$ is the reconstruction vector of the input data and $H = (h_1, h_2, \ldots, h_m)$ is the low-dimensional vector output from the hidden layer, $X \in R^n$, $Y \in R^n$, $H \in R^m$(n is the dimension of the input vector and m is the number of hidden units). $W_{ij} \in R^{m\times n}$ is the weight connection matrix between input layer and hidden layer. $W_{jk} \in R^{n\times m}$ is the weight connection matrix between hidden layer and output layer. In order to reconstruct the input data as accurately as possible while reducing the resource consumption during model training, $W_{jk} = W_{ij}^T$ usually exists in the experiment. $\varphi_1 \in R^{n\times 1}$ and $\varphi_2 \in R^{m\times 1}$ are the bias vectors of the input layer and hidden layer respectively. $g_{\theta_1}(\cdot)$ and $g_{\theta_2}(\cdot)$ are the activation function of the hidden layer neurons and output layer neurons respectively, which roles are to map the network summation result to [0,1]. We use sigmoid function as activation function in this paper:

$$g_{\theta_1}(\cdot) = g_{\theta_2}(\cdot) = \frac{1}{1 + e^{-x}} \tag{3}$$

By adjusting the parameters of the encoder and the decoder, the error between the output reconstructed data and the original data can be minimized, which means that AE reconstructs the original data through training. We believe that the data output by the hidden layer units at this time is the optimal low-dimensional representation of the original data and includes all the information that exists in the original data. The reconstruction error function $J_E(W, \varphi)$ between H and Y uses the mean squared-error function as shown in formula 4, where N is the number of input samples.

$$J_E(W, \varphi) = \frac{1}{2N}\sum_{r=1}^{N}\left\|Y^{(r)} - X^{(r)}\right\|^2 \tag{4}$$

B. SPARSE AUTOENCODER

The idea of sparse coding was originally proposed by Olshausen [18] to simulate the computational learning of the receptive fields of simple cells in mammalian primary visual cortex. Due to the unavoidable problem of the autoencoder, for example, the input data is transmitted to the output layer by simple copying. Although the original input data can be recovered perfectly, the autoencoder does not extract any meaningful features in this case. Therefore, Ng *et al.* [19] used the idea of sparse coding to introduce a sparse penalty term in the hidden layer of autoencoder, so that the autoencoder can obtain more concise and efficient low-dimensional data features under sparse constraints to better express the input data. Suppose that the average activation of the neurons in the hidden layer is $\hat{\rho}_j$, $\hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N}\left[n_j(x_i)\right]$. We hope that the average activation $\hat{\rho}_j$ approaches a constant ρ which is close to zero.

Therefore, the Kullback–Leibler (KL) divergence is added as a regularization term to the error function of the autoencoder to achieve the above purpose:

$$KL(\rho\|\hat{\rho}_j) = \rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j} \tag{5}$$

At this point, the error function of the sparse autoencoder consists of two parts: the first term is the mean square error term, and the second term is the regularization term. It is shown in formula 6:

$$J_{sparse}(W, b) = J(W, b) + \mu\sum_{j=1}^{m} KL\left(\rho\left\|\hat{\rho}_j\right.\right) \tag{6}$$

where m is the number of the hidden units and μ is a weighting factor that controls the strength of the sparse item.

Furthermore, in order to prevent overfitting, we also added the weight attenuation items to the error function as shown in formula (7), λ is the attenuation coefficient of the weight.

$$J_{sparse}(W, b) = J_E(W, b) + \mu\sum_{j=1}^{m} KL\left(\rho\left\|\hat{\rho}_j\right.\right) + \frac{\lambda}{2}\sum_{r=1}^{3}\sum_{i=1}^{m}\sum_{j=1}^{m+1}\left(w_{ij}^r\right)^2 \tag{7}$$

C. STACKED SPARSE AUTOENCODER

Stacked sparse autoencoder (SSAE) neural network is a neural network composed of multiple sparse auto-encoders connected end to end, the structure shown in Figure 3. The output of the previous layer of sparse self-encoder is used as the input of the next layer of self-encoder, so that higher-level feature representations of the input data can be obtained.

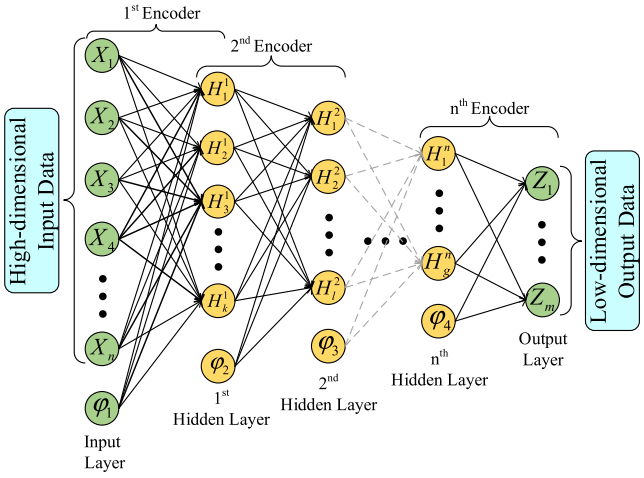


FIGURE 3. The structure of stacked autoencoder model.

The greedy layer-wise pre-training method [20] is used to sequentially train each layer of SSAE to get access to the optimized connection weights and bias values of the entire stacked sparse auto-encoded network. Then the error back propagation method is used to fine tune the SSAE until the result of the error function between the input data and the output data satisfies the expected requirements, so as to acquire the optimal parameter model.

For error function $J_{sparse}(W, b)$defined in III.B:

$$\frac{\partial}{\partial w_{ij}^r} J_{sparse}(W, b) = \frac{1}{2n_r} \sum_{r=1}^{n_r} \frac{\partial}{\partial w_{ij}^r} J_{sparse}(W, b, X(n), Y(n)) + \lambda w_{ij}^r \quad (8)$$

$$\frac{\partial}{\partial b^r} J_{sparse}(W, b) = \frac{1}{2n_r} \sum_{r=1}^{n_r} \frac{\partial}{\partial b^r} J_{sparse}(W, b, X(n), Y(n)) \quad (9)$$

Therefore, the update process of weight and bias is as follows:

$$w_{ij}^k = w_{ij}^k - \eta \frac{\partial}{\partial w_{ij}^k} J(W, b) \quad (10)$$

$$b^r = b^r - \eta \frac{\partial}{\partial b^r} J(W, b) \quad (11)$$

Where $X(n)$ and $Y(n)$ are respectively represented as the n*th* original vector and its corresponding reconstruction vector. η indicates the update learning rate.

Considering that there are sparse constraints in the SSAE network, we want to use different learning rate for different parameters, such as reducing the frequency of updates for infrequent features. However, most of the traditional popular gradient descent algorithms include stochastic gradient descent and mini-batch gradient descent, which use the same learning rate for all network parameters that need to be updated, making it difficult to choose a suitable learning rate and easily reach a local minimum [21]. Therefore, in order to train a better SSAE network model, we use the adaptive moment estimation (Adam) gradient descent algorithm proposed by Kingma and Ba [22] to achieve dynamic adaptive adjustment of different parameters.

The Adam algorithm implements dynamic adjustment of different parameters by calculating the gradient first-order moment estimate m_t and second-order moment estimate v_t, as shown in formula (12-14), where β_1 and β_2 respectively represent the first-order exponential damping decrement and the second-order exponential damping decrement. g_t represents the gradient of the parameters at timestep $t$in the loss function $J_{sparse}(W, b)$.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t \quad (12)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad (13)$$

$$g_t \leftarrow \nabla_\theta J_t(\theta_{t-1}) \quad (14)$$

Computer bias-corrected for m_t and v_t:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (15)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (16)$$

Update parameters:

$$\theta_{t+1} = \theta_t - \frac{\gamma}{\sqrt{\hat{v}_t} + \xi} \cdot \hat{m}_t \quad (17)$$

γ is the update stepsize, ξ takes a small constant to prevent the denominator to be zero.

IV. EXPERIMENTAL SETUP

A. THE FRAMEWORK

The framework of the SSAE-based intrusion detection system is shown in Figure 4. First, the original input dataset is preprocessed so that the data can be used for training and testing of the SSAE network. Then, the processed dataset is divided into two parts: a training set and a testing set. The training set is used for the pre-training and fine-tuning of the model, while the testing set is input with the optimal model to obtain the low-dimensional representation dataset. Finally, the classifiers are trained by using low-dimensional dataset to test the effect of low-dimensional data on the performance of the classifier, thereby validating the effectiveness of the SSAE model.

B. DATASET

There are few public data sets used in intrusion detection, mainly based on KDD99 [23] data set, NSL-KDD [24] data

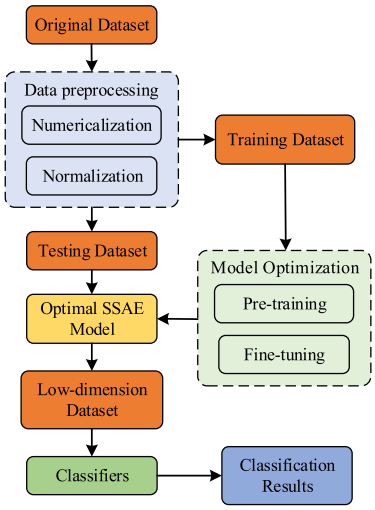


FIGURE 4. Flowchart of the proposed method.

set and Kyoto2006 [25] data set. Considering the advantages and disadvantages of the existing datasets comprehensively, and in order to make an effective and fair comparison with the existing intrusion detection models and methods, we have selected the NSL-KDD dataset to evaluate the performance of our SSAE model in intrusion detection.

TABLE 1. Classes and numbers of attacks of NSL-KDD dataset.

Data type		Sample Size	
		Training set	Testing set
Normal		67343	9691
Attacks	DOS	45927	7457
	Probe	11656	2421
	R2L	995	2754
	U2R	52	220
Total		125973	22543

The NSL-KDD dataset was improved on the KDD 99 dataset, eliminating the redundant data contained in the KDD 99 dataset. It contained a total of 125973 training samples and 22543 test samples, including four attack sample types: Denial of service attacks (DOS), Probing attacks(Probe), Remote to Local attacks(R2L), User to root attacks(U2R), and the specific distribution is shown in Table 1. Besides, in order to simplify the experimental process and ease the computational pressure, we randomly sample the original data in NSL-KDD and reassemble the sampled samples into several independent datasets, including training dataset, validation dataset and testing dataset, as shown in Table 2. The main purpose of the training datasets are to train the model. The validation datasets are used to fine-tune the model and adjust the parameters of the neural network and classifier. The purpose of the test datasets

TABLE 2. Number of training, validation and testing samples in new datasets.

No.	Training dataset		Validation dataset		Testing set	
	Normal	Attacks	Normal	Attacks	Normal	Attacks
1	13463	2947	4917	3789	5232	4390
2	9945	2629	5103	3910	4987	3876
3	11564	3016	4601	3165	5503	4103
4	12678	3108	4385	2954	5019	3952
5	12326	2811	5084	3375	5299	4078

is to test the detection ability of the trained model. In the experimental process, each dataset of No.1-5 is repeated several times independently, and cross-validation is performed between each dataset. The final experimental results are averaged for each experiment to ensure that the results are unbiased.

C. DATA PREPROCESSING

The NSL-KDD dataset contains 41 classification features, which are divided into symbolic features, 0-1-type features and percentage-type features. Among them, the feature value of Num_outbound_cmds is all 0, which has no effect on the classification process, so this feature is removed. Besides, since the input of the SSAE network is a numeric matrix, we need to transform the symbolic features into numerical features. In addition, in order to facilitate the comparison, the original feature values are subjected to a maximum-minimum normalization process so that the feature values are in the same order of magnitude.

1) NUMERALIZATION

we use the one-hot encoding to perform the numeralization. The symbolic features of NSL-KDD dataset include "Protocol_type", "Service" and "Flag", where "Protocol_type" includes three different symbolic feature values, "Service" includes 70 different symbolic feature values and "Flag" includes 11 different symbolic feature values. Therefore, after the completion of the numeric processing, the dimensions of features in NSL-KDD dataset is extended to 121-dimensions.

2) NORMALIZATION

To facilitate the comparison of the results, the maximal-minimum normalization method shown in formula 18 is used to normalize the feature values in the NSL-KDD dataset, where x_{max} and x_{min} represent the maximum and minimum values of the original feature values, respectively. x denotes the original feature value and x_{norm} denotes the normalized feature value.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{18}$$

D. EVALUATION

We use the metrics based on the confusion matrix to measure the experimental results of this paper. The definition of the confusion matrix is shown in Table 3. In the table, TP (True Positive) indicates the number of correctly identified normal records, TN (True Negative) indicates the number of correctly identified attack records, FP (False Positive) indicates the number of incorrectly identified normal records and FN (False Negative) indicates the number of erroneously identified attack records.

TABLE 3. Confusion matrix.

Sample Class		Predicted	
		normal	attack
Real	normal	TP	FP
	attack	FN	TN

The metrics used in this paper mainly include ACC (accuracy), DR (detection rate), and FAR (false alarm rate), and the calculation method is shown in formula 19-21. In addition, model training and testing time are represented by T_{train} and T_{test} respectively.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{19}$$

$$\mathrm{DR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{20}$$

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{21}$$

E. MODEL PARAMETERS

We use the TensorFlow, one of the most popular machine learning frameworks, to conduct the experimental simulation in this paper, and Python is selected as the programming language. The hardware experimental environment is a desktop with an Inter Core i7-7700 quad-core processor, 16G RAM, 256G SSD, and 64-bit Windows 10 operating system. At the same time, we use the GeForce GTX1060 graphics processing unit to speed up the training and testing of the model.

According to Section IV-C, after the samples in the NSL-KDD dataset are preprocessed, the features are extended from 41-dimension to 121-dimension. Therefore, we select the number of input layer neurons of the SSAE as 121. In addition, it is proved by experiments that the hidden structure of the four-layer SSAE network is the optimal experimental model (Section V-A), and the best sparse parameter is selected as 0.04(Section V-B). The parameters of the Adam algorithm adopt the default values recommended by the author. Table 4 shows the specific experimental model parameters.

V. EXPERIMENTAL RESULTS AND DISCUSSION

Here, we evaluate the performance of the proposed SSAE model by performing a variety of experiments on the NSL-KDD dataset, including five-category classification (Normal, DoS, Probe, R2L, U2R) and binary-category classification (Normal, Anomaly). More specifically, the experiments in this paper aim to achieve the following:

a. Evaluate the impact of network structure on the performance of the SSAE.

b. Evaluate the impact of sparse parameter on the performance of the SSAE.

c. Evaluate the impact of the low-dimensional features extracted by the SSAE on classifiers.

d. Compare SSAE with other state-of-the-art feature selection and feature compression methods.

e. Compare SSAE with other state-of-the-art shallow learning and deep learning models.

TABLE 4. The experimental parameters of SSAE and Adam.

Algorithm	Parameter	Value
SSAE	Number of nodes in input layer	121
	Number of neurons in 1st hidden layer	100
	Number of neurons in 2nd hidden layer	85
	Number of neurons in 3rd hidden layer	55
	Number of neurons in 4th hidden layer	30
	Number of neurons in output layer	5
	Sparse parameter	0.04
	SAE pre-training learning rate	0.01
	Batch size	500
Adam	Stepsize	0.001
	First-order exponential damping decrement	0.9
	Second-order exponential damping decrement	0.999
	Non-zero constant	10^{-8}

A. IMPACT OF THE DIFFERENT NETWORK STRUCTURE

In deep neural network, there is no fixed method to determine the number of hidden layers and the number of neurons in each layer. We need to set different network structures according to different experimental backgrounds. If the number of hidden layers is small and the number of neurons in each layer is insufficient, it may lead to the model cannot effectively match the distribution of data, and for SSAE network, that is, the high-dimensional features cannot be effectively compressed. Conversely, if the number of hidden layers is too large and the number of neurons in each layer is excessive, it may lead to the extremely complex training process of the model, which greatly increases the training time and the consumption of computing resources, and at the same time, may cause the model to be overfitting.

Table 5 shows the effect on the performance of the classifier when SSAE takes different number of hidden layers on validation datasets, including mean and standard deviation for each indicator. In order to achieve the purpose of

TABLE 5. The ACC, DR, FAR and training time of SSAE with different hidden layer structure.

Hidden Layer Structure	ACC(%)	DR(%)	FAR(%)	T_{train} (s)
[110,95,70,55,30,15]	99.32 ± 0.096	99.27 ± 0.089	0.096 ± 0.052	23.04
[105,90.70,55,30]	99.17 ± 0.103	99.02 ± 0.114	0.109 ± 0.054	11.03
[100,80,60,40,20]	99.01 ± 0.098	98.92 ± 0.121	0.117 ± 0.032	10.32
[100,80,50,30]	**98.63 ± 0.101**	**98.32 ± 0.120**	**0.131 ± 0.021**	**5.03**
[105,80,45,20]	**98.54 ± 0.109**	**98.15 ± 0.125**	**0.136 ± 0.024**	**5.26**
[90,60,30]	93.39 ± 0.259	93.12 ± 0.364	0.158 ± 0.019	3.19
[85,50,20]	92.78 ± 0.234	93.01 ± 0.318	0.164 ± 0.038	2.98
[75,30]	87.97 ± 0.198	87.32 ± 0.256	0.305 ± 0.029	2.18
[80,40]	88.74 ± 0.218	88.04 ± 0.187	0.289 ± 0.259	2.25
[65]	84.92 ± 0.356	83.78 ± 0.321	0.351 ± 0.298	1.45
[60]	86.12 ± 0.328	85.93 ± 0.315	0.348 ± 0.384	1.37

controlling variables, we connect the same Softmax classifier to the output layer of SSAE with different structures, and the remaining algorithm parameters in the experiment are consistent. It could be found that with above settings, as the number of hidden layers of increases, the classification results of the classifier become more and more satisfactory. However, a major drawback of the multilayer network structure is that it requires more time to train, which causes the infeasibility in real network environment. For example, in our experiments, although the detection accuracy of the five-layer hidden structure is better than that of the four-layer hidden structure, the training time of five-layer has almost stiffened to double that of four-layer. Further, when the model is in a large data environment, the training time will increase explosively. Therefore, after a comprehensive comparison, the four-layer hidden structure is the best network structure suitable for our experiment.

TABLE 6. The ACC, DR and FAR of SSAE with different hidden layers neurons when the number of hidden layers is the same.

Network Structure	ACC(%)	DR(%)	FAR(%)
[150,150,150,150]	83.17 ± 0.259	82.98 ± 0.219	0.252 ± 0.038
[60,60,60,60]	90.24 ± 0.217	90.11 ± 0.221	0.167 ± 0.058
[100,85,55,30]	**98.81** ± 0.108	**98.74** ± 0.112	**0.121** ± 0.021
[30,55,85,100]	86.34 ± 0.231	87.36 ± 0.215	0.219 ± 0.019
[200,150,100,50]	91.51 ± 0.198	91.73 ± 0.168	0.201 ± 0.047
[50,100,150,200]	88.51 ± 0.152	88.08 ± 0.165	0.244 ± 0.036

Table 6 shows the effect of different number of hidden neurons on the classification results when the number of hidden layers is the same. It can be concluded that the performance of the SSAE with the same number of neurons in each layer is worse than that when the number of neurons in each layer is different. It can be also observed that better classification results can be obtained when the number of neurons in the hidden layer decreases by layer. Further, the experimental results show that the number of neurons in the hidden layer should be less than that in the input layer, which not only reduces the training time of the model, but also is more conducive to the compression of the original features.

Another important study in this paper is to choose the smallest feature dimension without losing the information between original samples. Since the compressed feature dimension is equal to the number of output layer neurons, therefore, we measure the number of neurons in the output layer of the SSAE network to observe how they affect the performance of intrusion detection. The [100, 85, 55, 30] four-hidden layers' model with optimal results in Table 6 is used to perform experiments, while the other parameters remain unchanged, and the number of output layer neurons changes from 1 to 10. Figure 5 shows the experimental results, from which, we can see that those metrics yield the best results when the number of output layer neurons is 5.

B. IMPACT OF THE DIFFERENT SPARSE PARAMETER

Bengio [26] conducted a large number of experiments and concluded that a good experimental effect can be achieved when the sparse parameter is chosen to be 0.05 in the deep architectures and many existing literatures use this parameter to perform their experiments. However, we think this parameter may not apply to our model because sparse parameter is relevant to the number of bottleneck code dimension. Figure 6 indicates that the classification results of SSAE, with four-hidden layers' model of [100], [85], [55], [30], are affected by the sparse parameter from 0.01 to 0.1. It can be intuitively seen that when the value of the sparse parameter is 0.04, the comprehensive performance of the SSAE is the best. Less than this value, it leads to a deficit in the number of

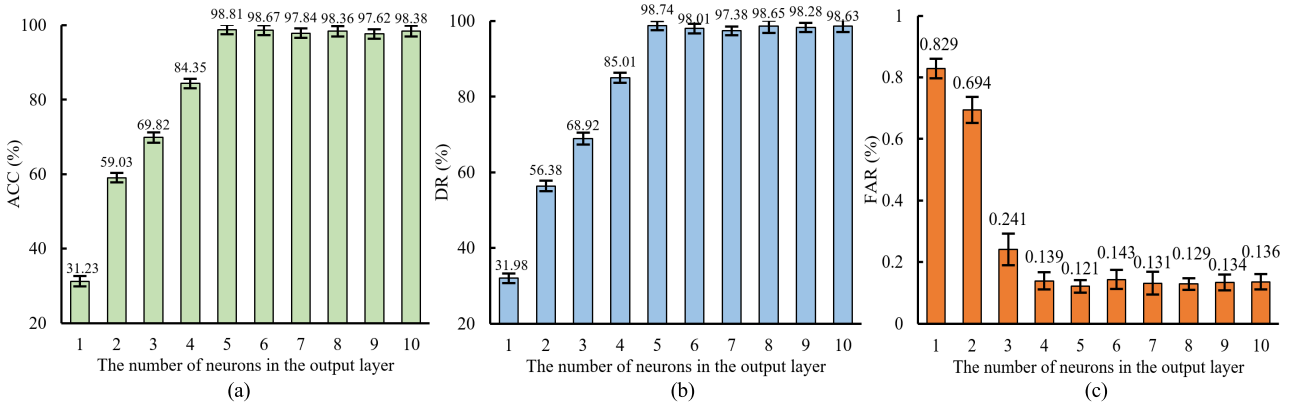


FIGURE 5. Experimental results with the number of output layer neurons changes from 1 to 10. (a) ACC. (B) DR. (C) FAR.

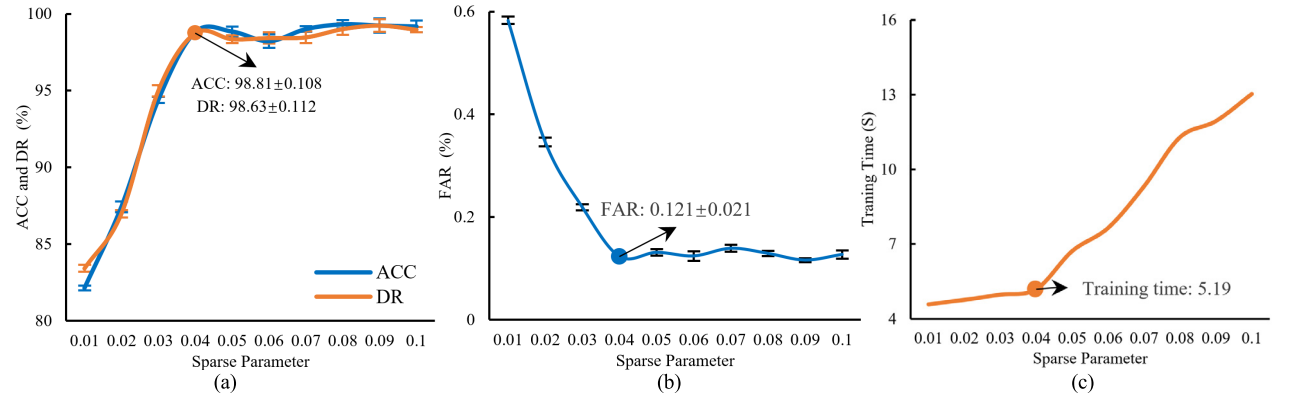


FIGURE 6. Experimental results with the sparse parameter changes from 0.01 to 0.1. (a) ACC and DR. (B) FAR. (C) Training Time.

TABLE 7. Effect of low-dimensional features extracted by the SSAE on the performance of the base classifiers for overall classification.

Method	ACC(%)	DR(%)	FAR(%)	Training Time(s)	Testing Time(s)
SVM	99.02 ± 0.135	98.97 ± 0.168	0.156 ± 0.054	73.96	59.20
KNN	98.93 ± 0.214	98.32 ± 0.237	0.171 ± 0.066	78.54	63.85
RF	99.13 ± 0.169	98.94 ± 0.187	0.153 ± 0.049	68.30	51.39
SSAE+SVM	**99.35 ± 0.127**	**99.01 ± 0.134**	**0.130 ± 0.051**	8.32	**3.29**
SSAE+KNN	98.87 ± 0.146	98.69 ± 0.153	0.152 ± 0.062	9.19	4.83
SSAE+RF	99.21 ± 0.138	98.53 ± 0.210	0.148 ± 0.044	**8.25**	3.49

neurons used for training, so there is a downward trending for the detection performance, and larger than the value, the input features are not effectively compressed, then the training time of the model sustains over long periods of time.

C. IMPACT OF THE LOW-DIMENSIONAL FEATURES ON CLASSIFIERS

To verify the effectiveness of the low-dimensional features extracted by the SSAE, we used three base classifiers, including support vector machine (SVM), K-nearest neighbor (KNN), and random forests (RF), to conduct comparative experiments. The parameters of SSAE use the optimal parameters measured in the preceding experiments. Table 7 and Table 8 summarizes the experimental results for the overall classification and five-category classification, respectively. It can be seen that the low-dimensional features compressed by the SSAE not only has no negative effect on the performance of the classifiers, but also significantly reduce the training time and testing time of the classifiers. The results also clearly demonstrate that the SSAE can almost retain all the amount of information contained in the original data while learning the high-level representation of features.

TABLE 8. Effect of low-dimensional features extracted by the SSAE on the performance of the base classifiers for five-category classification.

Method	Normal	DOS		Probe		R2L		U2R	
	DR(%)	DR(%)	FAR(%)	DR(%)	FAR(%)	DR(%)	FAR(%)	DR(%)	FAR(%)
SVM	99.20±0.215	99.33±0.198	0.023±0.0067	98.35±0.254	0.031±0.0092	81.32±0.342	0.035±0.0058	67.53±0.318	0.067±0.0069
KNN	99.04±0.220	99.14±0.211	0.027±0.0059	98.57±0.235	0.028±0.0069	76.32±0.361	0.043±0.0062	65.95±0.298	0.073±0.0058
RF	99.12±0.198	99.28±0.206	0.027±0.0066	98.23±0.215	0.025±0.0091	80.44±0.196	0.039±0.0059	**68.34**±0.317	0.062±0.0046
SSAE+SVM	**99.43**±0.176	**99.35**±0.199	**0.021**±0.0052	**99.03**±0.196	**0.019**±0.0061	**83.43**±0.315	**0.031**±0.0094	67.94±0.296	**0.059**±0.0051
SSAE+KNN	99.19±0.189	99.08±0.215	0.030±0.0064	98.21±0.214	0.021±0.0053	79.30±0.365	0.040±0.0089	66.43±0.398	0.061±0.0058
SSAE+RF	99.36±0.188	99.27±0.168	0.029±0.0064	98.48±0.222	0.023±0.0068	81.98±0.197	0.038±0.0052	66.96±0.269	0.058±0.0039

TABLE 9. Performance comparison obtained by the proposed and other approaches for overall classification.

Method		ACC(%)	DR(%)	FAR(%)	Datasets	Training Dataset Size
Feature Selection Method	FMIFS[13]	99.94	98.93	0.28	NSL-KDD	125,973
	TDTC[27]	N/A	84.86	4.86	NSL-KDD	125,973
	SVM-ELM[28]	95.75	95.17	1.87	KDD 99	311.029
	DNEDRON [29]	97.55	95.97	1.08	NSL-KDD	125,973
Shallow Learning Model	TVCPSO[6]	97.84	97.03	0.87	NSL-KDD	125,973
	NBC-A[8]	98.50	94.21	0.32	KDD 99	497,021
	CANN[9]	99.46	99.28	2.95	KDD 99	494,009
	TLHA[11]	93.29	91.86	0.78	KDD 99	98,456
Deep Learning Model	HAST-IDS[15]	99.68	97.78	0.07	KDD 99	2,466,929
	LSTM[16]	93.82	77.12	0.09	NSL-KDD	125,973
	DBN^4+LR[17]	97.90	97.51	0.51	KDD 99	145,584
	RNN-IDS[30]	99.81	96.92	5.09	NSL-KDD	49,402
Proposed method	SSAE+SVM	99.35	99.01	0.13	NSL-KDD	74,487

D. COMPARISON OF DIFFERENT ALGORITHMS

To evaluate the performance of the proposed method, the optimal SSAE+SVM model is selected to perform comparative experiments against the conventional and topical algorithms, including the following three categories: features selection methods, shallow learning and deep learning models. The overall ACC, DR, FAR and the size of training dataset are depicted in Table 9, while Table 10 shows the detection rate for the five categories. According to the experimental results shown in these tables, it can be seen that the performance of SSAE+SVM is very close to or more than other state-of-the-art approaches in terms of overall accuracy and detection rate. Moreover, the false alarm rate is only lower than that of HAST-IDS[15] and LSTM[16], and the gap remains within 0.05%. This indicates that the method proposed in this paper has reached or exceeded the average detection level of other state-of-the-art methods and models. The strength of our method is that we can achieve above experimental results with only 74,487 training samples, far less than the number of training samples required for other experimental methods except the RNN-IDS model. However, the performance of the RNN-IDS is inferior to our method in the metrics of FAR and DR. Obviously, less consumption of training samples means that under the same experimental conditions, our method will train the model in a faster way to meet the requirements of the system's real-time performance.

On the other hand, in the multi-classification experiments, for the detection rate of two types of low frequency attack samples, R2L and U2R, the methods we proposed have not achieved satisfactory results, only 84.43% and 67.94%, respectively. The two methods of DBN^4 + LR [17] and NBC-A [8] get the best performance in the detection of R2L and U2R respectively. Furthermore, the detection rate of NBC-A for all five types of samples exceeds 97.5%.

TABLE 10. Comparisons of DR obtained by the proposed and other approaches for five-category classification.

Method		Normal	DOS	Probe	R2L	U2R
Feature Selection Method	FMIFS[13]	98.98	98.76	86.08	88.38	22.11
	TDTC[27]	94.43	88.20	87.32	42.00	70.15
	SVM-ELM[28]	98.13	99.54	87.22	31.39	21.93
	DNEDRON [29]	98.92	95.94	97.17	83.75	76.92
Shallow Learning Model	TVCPSO[6]	99.13	98.84	89.29	75.08	59.62
	NBC-A[8]	98.21	99.11	97.67	98.30	99.19
	CANN[9]	95.98	96.59	82.85	78.95	61.54
	TLHA[11]	99.26	97.37	98.61	79.39	14.02
Deep Learning Model	HAST-IDS[15]	N/A	99.10	83.35	74.19	64.25
	LSTM[16]	99.50	99.30	87.00	30.40	75.10
	DBN^4+LR[17]	94.51	98.74	86.66	100.00	38.46
	RNN-IDS[30]	N/A	83.49	83.40	24.69	11.50
Proposed method	SSAE+SVM	99.43	99.35	99.03	83.43	67.94

Through further detailed research on existing literature, the main reason for this result is that the number of R2L and U2R attack samples contain in the training set used in our experiments are scarce and the classification features are insufficient, which result in the SVM classifier failing to learn the sample features effectively. Moreover, conventional machine learning classifiers tend to be more biased towards the majority samples, because they do not consider the distribution of the data in the optimization of the loss function. Worse still, in this case, low frequency attack samples may be ignored as noise points or outliers of the majority class [31]. Therefore, we believe that all of the above reasons lead to a low detection rate of R2L and U2R in our experiments.

VI. CONCLUSION AND PROSPECT

Conventional shallow machine learning methods cannot effectively deal with the high-dimensional feature classification task under the condition of large data volume, which lead to the intrusion detection systems based on machine learning methods, cannot meet the real-time requirements. Therefore, this paper proposes a novel feature extraction method based on deep learning theory, which uses stacking sparse autoencoder to realize the nonlinear mapping of high-dimensional features to low-dimensional features. Then, the new dataset containing the optimal low-dimensional features is used to train the classifiers and test for performance. The experimental results on the NSL-KDD dataset show that the SSAE with the optimal structure can compress the original features to 5 dimensions without losing the amount of information existing between the original data. In addition, the results compared with the methods proposed in the existing literatures show that SSAE can reach or even exceed the average detection level of conventional machine learning classifiers and other state-of-the-art approaches in terms of overall detection performance with relatively less resource consumption.

The disadvantage of the proposed method is that it cannot effectively detect R2L and U2R low-frequency attack samples, that is, it cannot overcome the adverse effects caused by imbalanced data distribution. In future research, how to use the existing methods or propose new methods to handle the problem of imbalanced data in the feature extraction process deserves further attention. Besides, it will be very interesting to find out the patterns of features learned by SSAE which can help with manual feature engineering.

REFERENCES

[1] G. Kumar, K. Kumar, and M. Sachdeva "The use of artificial intelligence based techniques for intrusion detection: A review," *Artif. Intell. Rev.*, vol. 34, no. 4, pp. 369–387, 2010.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[3] R. F. Erbacher, K. L. Walker, and D. A. Frincke, "Intrusion and misuse detection in large-scale systems," *IEEE Comput. Graph. Appl.*, vol. 22, no. 1, pp. 38–47, Jan. 2002.

[4] W. Lee and D. Xiang, "Information-theoretic measures for anomaly detection," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 2001, pp. 130–143.

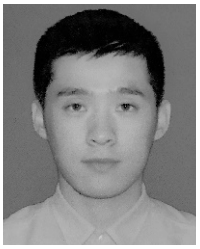
[5] G. Wang, J. Hao, J. Ma, and L. Huang, "A new approach to intrusion detection using artificial neural networks and fuzzy clustering," *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6225–6232, 2010.

[6] S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, "An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization," *Neurocomputing*, vol. 199, pp. 90–102, Jul. 2016.

[7] D. Moon, M. Im, I. Kim, and J. H. Park, "DTB-IDS: An intrusion detection system based on decision tree using behavior analysis for preventing APT attacks," *J. Supercomputing*, vol. 73, no. 7, pp. 2881–2895, 2017.

[8] Y. Wang *et al.*, "A novel intrusion detection system based on advanced naive Bayesian classification," in *Proc. Int. Conf. 5G Fut. Wireless Netw. (5GWN)*, Beijing, China, Dec. 2017, pp. 581–588.

[9] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowl.-Based Syst.*, vol. 78, pp. 13–21, Apr. 2015.
[10] B. M. Aslahi-Shahri *et al.*, "A hybrid method consisting of GA and SVM for intrusion detection system," *Neural Comput. Appl.*, vol. 27, no. 6, pp. 1669–1676, Aug. 2016.
[11] C. Guo, Y. Ping, N. Liu, and S.-S. Luo, "A two-level hybrid approach for intrusion detection," *Neurocomputing*, vol. 214, pp. 391–400, Nov. 2016.
[12] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, Jan. 2016.
[13] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016.
[14] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," *Eur. J. Wireless Commun. Netw.*, vol. 2016, pp. 130–140, Dec. 2016.
[15] W. Wang *et al.*, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, no. 1, pp. 1792–1806, Dec. 2017.
[16] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South Afr. Comput. J.*, vol. 56, pp. 136–154, Jul. 2015.
[17] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Anaheim, CA, USA, Dec. 2016, pp. 195–200.
[18] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.
[19] A. Ng, "Sparse autoencoder," *CS294A Lect. Notes*, vol. 72, pp. 1–19, 2011.
[20] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. 21st Int. Conf. Neural Inform. Process. Syst. (NIPS)*, Vancouver, BC, Canada, Jan. 2007, pp. 153–160.
[21] T. T. H. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," in *Proc. IEEE Int. Conf. Plat. Technol. Service (PlatCon)*, Busan, South Korea, Feb. 2017, pp. 1–6.
[22] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, Dec. 2015, pp. 1–13.
[23] S. Rosset and A. Inger, "KDD-CUP 99: Knowledge discovery in a charitable organization's donor database," *ACM SIGKDD Explor. Newslett.*, vol. 1, no. 2, pp. 85–90, 2000.
[24] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA)*, Ottawa, ON, Canada, Dec. 2009, pp. 1–6.
[25] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," in *Proc. 1st Workshop Building Anal. Datasets Gathering Exper. Returns Secur.*, Salzburg, Austria, Apr. 2011, pp. 29–36.
[26] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2012, pp. 437–478.
[27] H. H. Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks," *IEEE Trans. Emerg. Topics Comput.*, Nov. 2016, doi: 10.1109/TETC.2016.2633228.
[28] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system," *Expert Syst. Appl.*, vol. 67, pp. 296–303, Jan. 2017.
[29] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "*Dendron*: Genetic trees driven rule induction for network intrusion detection systems," *Future Gener. Comput. Syst.*, vol. 79, pp. 558–574, Feb. 2018.
[30] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, no. 99, pp. 21954–21961, Oct. 2017.
[31] T. Fawcett. (Aug. 2016). *Learning From Imbalanced Classes*. [Online]. Available: https://www.svds.com/learning-imbalanced-classes



BINGHAO YAN was born in 1993. He received the B.S. degree from the University of Electronic Science and Technology of China, in 2015, and the M.S. degree from PLA Information Engineering University, Zhengzhou, China, in 2017. He is currently pursuing the Ph.D. degree with the National Digital Switching System Engineering & Technology Research Center, Zhengzhou. His research areas are intrusion detection, information security, machine learning, and deep learning.

GUODONG HAN was born in 1964. He received the B.S. and M.S. degrees from PLA Information Engineering University, Zhengzhou, China, in 1986 and 1990, respectively, and the Ph.D. degree from the National Digital Switching System Engineering & Technology Research Center, Zhengzhou. He is currently an Associate Professor with the Provincial Key Laboratory of System on a Chip, National Digital Switching System Engineering & Technology Research Center. He has authored over 40 journal and conference publications. His research areas are network security, signal processing, broadband network, and deep learning.