Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine

Wenjuan Wang, Xuehui Du, Dibin Shan, Ruoxi Qin, and Na Wang

Abstract—Security issues have resulted in severe damage to the cloud computing environment, adversely affecting the healthy and sustainable development of cloud computing. Intrusion detection is one of the technologies for protecting the cloud computing environment from malicious attacks. However, network traffic in the cloud computing environment is characterized by large scale, high dimensionality, and high redundancy, these characteristics pose serious challenges to the development of cloud intrusion detection systems. Deep learning technology has shown considerable potential for intrusion detection. Therefore, this study aims to use deep learning to extract essential feature representations automatically and realize high detection performance efficiently. An effective stacked contractive autoencoder (SCAE) method is presented for unsupervised feature extraction. By using the SCAE method, better and robust low-dimensional features can be automatically learned from raw network traffic. A novel cloud intrusion detection system is designed on the basis of the SCAE and support vector machine (SVM) classification algorithm. The SCAE+SVM approach combines both deep and shallow learning techniques, and it fully exploits their advantages to significantly reduce the analytical overhead. Experiments show that the proposed SCAE+SVM method achieves higher detection performance compared to three other state-of-the-art methods on two well-known intrusion detection evaluation datasets, namely KDD Cup 99 and NSL-KDD.

Index Terms—Cloud computing, intrusion detection system (IDS), feature extraction, deep learning, contractive auto-encoder, support vector machine

1 Introduction

Cloud computing [1] is an emerging Internet-based computing model that provides tenants with seemingly "unlimited" IT services, thereby freeing them from complex underlying hardware, software, and protocol stacks. Although "open for all service" is the essence of cloud computing, it does not necessarily comprise useless information. Tenants can use cloud services for efficient computing. However, they can also abuse the cloud environment and attack the network. For example, a malicious tenant may reside in a virtual machine (VM), successfully intrude into other VMs in the cloud, and use the puppet machines to spread malicious software, or launch distributed denial of service (DDoS) attack, and so on. In fact, tenant behavior will generate massive network traffic in the cloud environment, mainly including "north-south" and "east-west" traffic. The "north-south" traffic mainly refers to the traffic of tenants accessing cloud services from the external network, and the "east-west" traffic refers to the traffic between VMs in the virtual network [2]. Cisco's cloud industry research report predicts that the global cloud network traffic will account for 95% of the total network traffic by 2021. In particular, the "east-west" traffic between VMs in the cloud environment will account for 85% [2]. Network traffic will continue to increase dramatically and will inevitably encounter malicious attacks. Network attacks not only result in severe damage to the cloud environment but also cause tenants to lose confidence in cloud computing itself, which will adversely affect the healthy and sustainable development of cloud computing. Intrusion detection is one of the technologies for protecting cloud computing from malicious attacks. Therefore, we study cloud intrusion detection systems (CIDSs) to detect and analyze network traffic, particularly "east-west" traffic, identify malicious attack behaviors, and prevent damages, thus guaranteeing the safety and reliability of cloud computing.

However, cloud infrastructure is constructed with the technology of virtualization, which renders the virtual network flow between VMs invisible and uncontrollable by the traditional intrusion detection system (IDS). A software-defined network (SDN) has characteristics such as programmability, centralized control, and global view, so it is widely used in cloud computing. Previous studies [3], [4], [5], [6] proposed the use of SDN technology to redirect network traffic to Snort IDS for detecting malicious attacks. Snort, a signature-based detection system, cannot detect unknown attacks and adapt to large-scale traffic. On the contrary, anomaly detection, developed as classifiers to differentiate anomalous traffic from normal traffic, is well suited for the detection of unknown attacks, but it has a high false alarm rate.

Many types of shallow discriminative machine learning techniques have been extensively applied to IDS, such as the neural network (NN), random forest (RF), decision tree (DT), and support vector machine (SVM). However, these

• *The authors are with PLA Information Engineering University, Zhengzhou, Henan 450000, China. E-mail: wwjhhx@solu.com, dxh37139@sina.com, {officeshan, 18530023930}@163.com, twftina_w@126.com.*

Manuscript received 9 September 2019; revised 21 February 2020; accepted 4 June 2020. Date of publication 9 June 2020; date of current version 6 September 2022.
(Corresponding author: W. Wang.)
Recommended for acceptance by V. Piuri.
Digital Object Identifier no. 10.1109/TCC.2020.3001017

approaches provide unsatisfactory classification or detection accuracy. The intrusion detection result depends not only on the performance of the classifier but also on the quality of the input data. Network traffic data usually involves high dimensionality and redundancy of features, which can easily cause a feature dimensionality disaster. Therefore, feature dimensionality reduction is particularly important for effectively improving the performance of the above-mentioned supervised classifiers [7]. It includes two types of techniques: feature subset selection and feature extraction [8]. Feature subset selection works by removing relevant or redundant features; the subset of features selected will give the best performance according to some objective function. Many studies [9], [10], [11], [12], [13] have demonstrated that feature selection methods can overcome the "dimensionality curse" and achieve high detection performance in CIDS. Feature extraction maps the original high-dimensional features into low-dimensional features and generates new linear or nonlinear combinations of the original features [8]. Recently, various researchers have demonstrated that deep learning technology has considerable potential for IDS, especially in feature extraction. This study aims to use the deep learning technique to automatically extract essential features from raw network data and input them into a shallow classifier for effective identification of attacks.

The remainder of this paper is organized as follows. Section 2 reviews exiting studies. Section 3 presents relevant background information. Section 4 describes the design and training process of the proposed stacked contractive autoencoder (SCAE)–SVM model in detail. Section 5 discusses the design of the CIDS framework. Section 6 presents and analyzes the experimental results. Finally, Section 7 states the conclusions and explores directions for future work.

2 Existing Studies

Deep learning approaches are mainly categorized into supervised learning and unsupervised learning. The difference between these two approaches lies in the use of labeled training data. Specifically, convolutional neural network (CNN) [14] that use labeled data fall under supervised learning, which employs a special architecture suitable for image recognition. Unsupervised learning methods include deep belief network (DBN) [15], recurrent neural network (RNN) [16], autoencoder (AE) [17], and their variants. Next, we describe recent studies related to our work; these studies are mainly based on KDD Cup 99 or NSL-KDD datasets.

Studies on intrusion detection using the NSL-KDD dataset have been reported [18], [19], [20], [21], [22]. Tang *et al.* [18] used deep neural networks (DNNs) to build an anomaly detection model in the software-defined network (SDN) environment. They trained their model by using 6 basic features taken from the 41 features of the NSL-KDD dataset. Salama *et al.* [19] used DBN to extract features for intrusion detection and SVM to classify the data after dimensionality reduction. Experimental results showed that their hybrid DBN+SVM method improves the detection performance compared with using SVM or DBN as standalone classifiers. Aygun *et al.* [20] proposed two deep learning-based anomaly detection models using AE and denoising AE to detect zero-day attacks with high accuracy. And they used a stochastic approach to determine the threshold value that directly affects the accuracy of the proposed models. Niyaz *et al.* [21] developed an effective and flexible NIDS referred as self-taught learning (STL), which combines a sparse AE used for unsupervised dimensionality reduction with softmax regression used to train the classifier. This method achieved satisfactory classification accuracy in 2-class, 5-class, and 23-class classification tasks. Subsequently, Niyaz *et al.* [22] developed a DDoS intrusion detection system and applied it to the SDN environment. They used the stacked auto-encoder (SAE) for feature reduction and evaluated the detection performance of the SAE-SVM model on network traffic collected from real and private network test beds.

Studies on intrusion detection using the KDD Cup 99 dataset have been reported [23], [24], [25]. Kim *et al.* [23] specifically targeted advanced persistent threats and proposed a deep neural network (DNN) using 100 hidden units, combined with the rectified linear unit activation function and the ADAM optimizer. Their approach was implemented on a GPU using TensorFlow. Papamartzivanos *et al.* [24] proposed a novel method that combines the benefits of a sparse AE and the MAPE-K framework to deliver a scalable, self-adaptive and autonomous misuse IDS. They merged the datasets provided by KDD Cup 99 and NSL-KDD to create a single voluminous dataset. Shone *et al.* [25] designed a new non-symmetric deep autoencoder (NDAE) model, which unlike typical AEs, provides non-symmetric data dimensionality reduction. This model was combined with an RF classification algorithm to construct a classifier. This method achieved satisfactory results on the KDD Cup 99 and NSL-KDD datasets.

Studies using private or other public datasets for intrusion detection have also been documented [26], [27], [28]. Loukas *et al.* [26] used RNN-based deep learning enhanced by long short term memory (LSTM) to considerably increase intrusion detection accuracy for a robotic vehicle. They demonstrated that their approach achieves high accuracy with considerably more consistency than with standard machine learning techniques. Yu *et al.* [27] proposed a network intrusion detection model based on TCP, UDP, and ICMP sessions, which combines the stacked denoising auto-encoder (SDAE) and softmax classifier. Comparative experiments showed that the performance of this model is better than that of DBN and SAE in 2-class and 8-class classification tasks. This method was also validated on other public datasets, such as the UNB ISCX IDS 2012 and CTU-13 datasets. Later, Yu *et al.* [28] proposed a stacked dilated convolutional auto-encoder (DCAE) method, which can automatically learn key features from a large amount of raw network traffic. DCAE has fewer parameters than fully connected neural networks such as SAE. However, the limitation of the DCAE model lies in the relatively long training process, and the authors planned to adopt GPU parallelization technology to overcome this problem in the future. Yu *et al.* trained and tested their model with a private dataset, so the model cannot be directly compared with other schemes.

From the above-mentioned findings, we can conclude that deep learning has been successfully applied to network intrusion detection. However, it remains in its infancy. Most researchers are still combining it with various algorithms

through numerous experiments (structure, training, optimization, etc.) to explore the most effective solution. Hence, we believe that our study can make a significant contribution to cloud intrusion detection research.

The contributions of this study can be summarized as follows:

- We propose a novel stacked contractive autoencoder (SCAE)-based deep learning method for network intrusion detection. The SCAE method can automatically learn essential and robust low-dimensional features from raw network traffic and input them into a shallow SVM classifier. Leveraging on the respective strengths of deep and shallow learning techniques, their combination can effectively improve detection performance.
- We design a cloud intrusion detection system (CIDS) that uses an SDN frame for collecting virtual network traffic from the Xen cloud platform and apply the SCAE+SVM model for feature extraction and classification detection. The proposed system attempts to detect attacks on the data plane and is implemented on the application plane.
- We evaluate our SCAE+SVM IDS model by applying it on two well-known datasets frequently used to evaluate the detection performance of IDS. The experimental results show the effect of the network structure depth and extracted feature number on the detection performance. In addition, they demonstrate that compared with the results of existing similar models, our model achieves better or similar results.

3 Autoencoder and Its Variants

3.1 Autoencoder (AE)

An autoencoder (AE) is an unsupervised feature dimensionality reduction technique, with its structure consisting of an encoder and a decoder, including an input layer, a hidden layer, and an output layer. The encoder is used for dimensionality reduction and the decoder is used for reconstruction, which is regarded as the reverse process of the encoder.

Let a training dataset D have n samples; then $D = \{x^{(i)}, y^{(i)} | x^{(i)} \in \Re^{d_x}, y^{(i)} \in \Re\}_{i=1}^{n}$, where each sample $\boldsymbol{x}$ is a d_x-dimensional feature vector and y is the class label. The encoder function f maps input $\boldsymbol{x}$ into a hidden representation $h \in \Re^{d_h}$, and the decoder function g maps hidden representation $\boldsymbol{h}$ back to a reconstruction $z \in \Re^{d_x}$. When the number of hidden layer neurons are less than the number of input layer and output layer neurons, that is, $d_h < d_x$, we obtain the compressed vector of the input, and thus realize dimensionality reduction. The encoding and decoding processes are defined as follows:

$$h = f(wx + b) \tag{1}$$

$$z = g(w'h + b'), \tag{2}$$

where f and g are non-linear activation functions (typically, they are *sigmoid* or *hyperbolic tangent* functions), $\boldsymbol{w}$ and $\boldsymbol{w}'$ are the weight matrices, where $\boldsymbol{w}$ is a $d_x \times d_h$ matrix and $w' = w^T$, and $\boldsymbol{b}$ and $\boldsymbol{b}'$ are the bias values with d_h and d_x dimensions, respectively.

Here, we use $\theta = \{w, b\}$ and $\theta' = \{w', b'\}$, where θ and θ' represent the parameters of the encoder and decoder, respectively. The goal of learning is to minimize the reconstruction error between the input $\boldsymbol{x}$ and the output $\boldsymbol{z}$ by adjusting these parameters. The minimization objective function is defined as

$$J_{AE} = \sum_{x \in D_n} L(x, z) = \sum_{i=1}^{n} L(x^{(i)}, g_{\theta'}(f_\theta(x^{(i)}))), \tag{3}$$

where L is the loss function or reconstruction error function, which is typically the *mean squared error* (4) or the *cross-entropy loss* (5).

$$L(x, z) = \frac{1}{n} \sum_{i=1}^{n} \left(x^{(i)} - z^{(i)}\right)^2 \tag{4}$$

$$L(x, z) = -\frac{1}{n} \sum_{i=1}^{n} [x^{(i)} \log z^{(i)} + (1 - x^{(i)}) \log (1 - z^{(i)})]. \tag{5}$$

In general, the smaller the reconstruction error, the closer output $\boldsymbol{z}$ is to input $\boldsymbol{x}$, which implies that $\boldsymbol{h}$ is an effective low-dimensional feature representation. However, the reconstruction criterion alone will lead to the problem of the output being identical to the input; therefore, the AE cannot effectively extract features. To address this problem, we can adopt strategies such as adding constraint representation or corrupting the input by adding noise.

3.2 Denoising Autoencoder (DAE)

Unlike the conventional AE, the denoising autoencoder (DAE) [29] aims to learn a more effective and robust feature representation from the corrupted input data $\hat{x}$. The DAE first corrupts the input $\boldsymbol{x}$ and then sends the corrupted data $\hat{x}$ into the auto-encoder for denoising. Finally, it reconstructs the clean version $\boldsymbol{x}$. This process yields the following objective function:

$$J_{DAE} = \sum_{i=1}^{n} E_{\hat{x}^{(i)} \sim q(\hat{x}^{(i)} | x^{(i)})} L(x^{(i)}, g_{\theta'}(f_\theta(\hat{x}^{(i)}))), \tag{6}$$

where the expectation is over the corrupted versions $\hat{x}$ of samples $\boldsymbol{x}$ obtained from a corruption process $q_{(\hat{x}|x)}$. Common corruption approaches include additive isotropic Gaussian noise (GS), i.e., $\hat{x}|x \sim \mathrm{N}(0, \sigma^2 I)$, and binary masking noise (MN), which randomly sets a fraction of input features to 0.

3.3 Contractive Autoencoder (CAE)

Rifai *et al.* followed up on DAE and proposed the contractive autoencoder (CAE) [30]. The aim of CAE is to learn robust feature representation. Although DAE and CAE have the same purpose, they adopt two distinct methods. DAE learns robust feature representation from a relatively intuitive perspective by randomly adding noise to the input. CAE learns robust feature representation from the perspective of analysis by regularization. The minimization objective function is given by

$$J_{CAE} = \sum_{i=1}^{n} (L(x^{(i)}, g_{\theta'}(f_\theta(x^{(i)}))) + \lambda ||J_f(x^{(i)})||_F^2), \tag{7}$$

where $J_f(x)$ is the Jacobian matrix representing the partial derivative of the hidden features $\boldsymbol{h}$ with respect to the weight $\boldsymbol{w}$. λ represents the penalty coefficient used to adjust the proportion of the penalty term in the objective function. Further, $||J_f(x)||_F^2$ is the square of the Frobenius norm (F norm) of the Jacobian matrix, as shown in (8). When the activation functions f and g are a *sigmoid* function, that is, $sigmoid(x) = \frac{1}{1+e^{-x}}$, the contractive penalty term can easily be computed as (9).

$$||J_f(x)||_F^2 = \sum_{tj} \left(\frac{\partial h_j(x)}{\partial x_t} \right)^2 \tag{8}$$

$$||J_f(x)||_F^2 = \sum_{j=1}^{d_h} (h_j(1-h_j))^2 \sum_{t=1}^{d_x} ||w_{tj}||^2 \tag{9}$$

where $h_j \in h$ represents the element of the hidden representation, d_x and d_h represent the dimensions of input $\boldsymbol{x}$ and hidden representation $\boldsymbol{h}$, respectively. w_{tj} is the element of a $d_h \times d_x$ matrix that represents the weight of connection between input $\boldsymbol{x}$ and hidden representation $\boldsymbol{h}$. The overall computational complexity is O($d_x \times d_h$).

As can be seen, the final objective function consists of two parts. First, the reconstruction error function is used to obtain as much effective information as possible from the input data. Second, the newly added penalty term is used to suppress minor perturbations in the input data, and by introducing the F norm of the Jacobian matrix as the constraint term, the learned feature is made locally invariant.

3.4 Contrastive Analysis

The three technologies described in Sections 3.1 to 3.3 can achieve feature dimensionality reduction. However, the CAE has some advantages compared with the AE and DAE. In general, there are two criteria for good feature representation: (1) good reconstruction of input data, and (2) excellent robustness when the input data is disturbed to a certain extent. The AE satisfies only the first criterion, while the DAE and CAE satisfy both criteria. In some classification tasks, the second criterion is more important. Therefore, the DAE and CAE are more suitable for the task of classification detection.

In addition, there are at least three differences between the DAE and the CAE. First, the sensitivity of the *features* is penalized directly rather than indirectly. Thus, DAE is a robust reconstruction of $g(f(x))$, and the robustness of $f(x)$ is partial or indirect. CAE will punish $f(x)$ instead of $g(f(x))$, and the encoder part $f(x)$ is used for classification, robustness of the extracted features appears more important than robustness of the reconstruction inputs. Second, DAE improves the robustness of feature extraction by adding random noise to the input data, while the robustness of CAE against perturbations is achieved by calculation. The robustness is analytic rather than stochastic. Third, CAE can finely control the trade-off between the reconstruction input and the punishment by setting the hyper-parameter λ. Thus, CAE is superior to DAE, we believe that CAE will be a better choice than DAE for learning useful features and achieving higher classification accuracy.

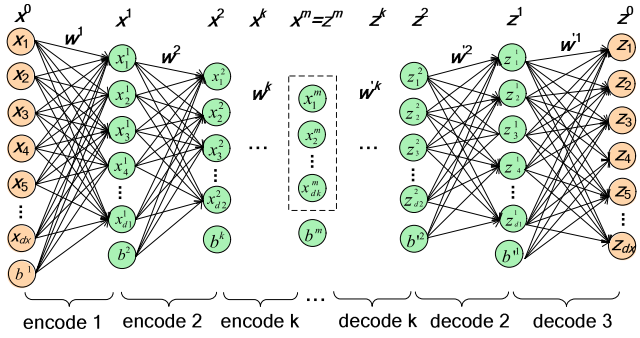


Fig. 1. Structure of stacked contractive autoencoder.

4 Proposed Methodology

In this section, we first describe the SCAE used for feature learning. Subsequently, we describe the training process of the SCAE model, and the SVM classifier used for multiclass anomaly detection.

4.1 Designing SCAE Model for Feature Learning

In general, deep feedforward networks have many advantages, which are also applicable to the AE and its variants. The SCAE consists of several hidden layers for encoding, and a set of symmetrical layers for decoding in which the output of each layer is fed as the input of the subsequent layer. The detailed structure of the SCAE is presented in Fig. 1. Here, the superscript numbers refer to the hidden layer identity, and the subscript numbers signify the dimension for the layer.

In the encoding phase, the k-th hidden layer of the SCAE can learn k-order features from the output of the (k-1)-th layer. That is, the first hidden layer learns 1-order features from raw input. The second hidden layer learns 2-order features in the appearance of the 1-order features. Subsequent higher layers learn higher-order features. Conversely, in the decoding phase, the (k-1)-th layer is reconstructed from (k-1)-th-order features from the output of the k-th layer, and so on, until the input is reconstructed.

Thus, the encoding and decoding processes of the SCAE network can be expressed as

$$x^k = f(w^k x^{k-1} + b^k) k = 1, \ldots, m \tag{10}$$

$$z^{k-1} = g(w'^k z^k + b'^k) \tag{11}$$

Here x^0 represents the input data, x^k represents the k-th-order features learned by the k-th hidden layer, and x^m denotes the low-dimensional m-th-order features that will be transported to the classifier, where m is the number of hidden layers or the depth of the network. Here, we let $x^m = z^m$, and allow layer-wise mapping of the m-th-order features back to reconstruction z^0. We use $\theta^k = \{w^k, b^k\}$ and $\theta'^k = \{w'^k, b'^k\}$, which represent the parameters corresponding to the k-th encoder and decoder, respectively. Thus, the minimization objective function of the SCAE is expressed as follows:

$$J_{SCAE} = \sum_{k=1}^{m} \sum_{i=1}^{n} (L(x^{(i,k-1)}, g_{\theta'^k}(f_{\theta^k}(x^{(i,k-1)}))) + \lambda^k ||J_f(x^{(i,k-1)})||_F^2) \tag{12}$$

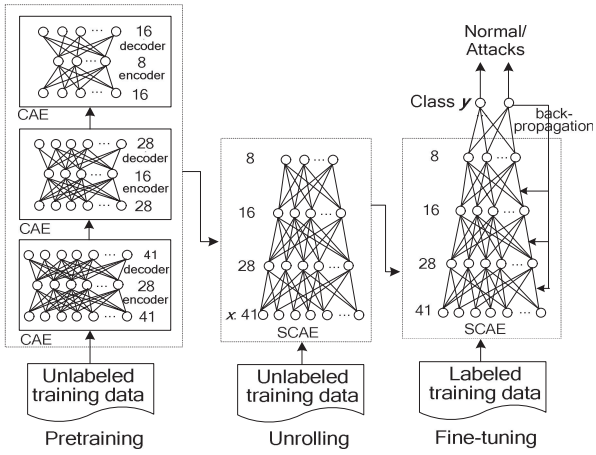


Fig. 2. Training process of SCAE model.

From (12), we can see that the SCAE has some practical problems. The first problem is that the objective function is difficult to calculate in the case of a deep network with multiple layers compared with the case of a single hidden layer. Thus, we use the greedy layer-wise strategy in which the learning problem is broken down into simpler steps. That is, some basic CAEs are trained separately, and then stacked into a deep neural network (detailed in the next section). Since each layer is trained to be contractive locally, the SCAE is also naturally contractive. Another problem is that the penalty term may be considerably larger than the reconstruction error; hence, we can either set a small λ value or make changes to the penalty term, such as by calculating its average value. Here, we use the latter method; thus, the minimization objective function of each CAE network is as follows:

$$J_{CAE}^{k} = \frac{1}{n}\sum_{i=1}^{n}\left(x^{(i,k-1)} - z^{(i,k-1)}\right)^2 + \lambda^k \sum_{i=1}^{n}\left(\frac{1}{d_k}\sum_{j=1}^{d_k}\left(x_j^{(i,k)}(1-x_j^{(i,k)})\right)^2 \times \frac{1}{d_k d_{k-1}}\sum_{t=1}^{d_{k-1}}||w_{t_j}^{k}||^2\right) \tag{13}$$

where $d_{k\text{-}1}$ and d_k represent the feature dimensions of the (k-1)-th layer and the k-th layer, respectively. λ^k represents the penalty coefficient of the k-th CAE network. λ^k is set such that the penalty value is less than the reconstruction error but is of the same order of magnitude as the reconstruction error, and can achieve higher classification performance.

4.2 Training Process of SCAE

Fundamentally, the exact structure of our deep learning model will be obtained through experiments and training on a large number of structural combinations (i.e., number of hidden layers, number of neurons). Next, we introduce the training process of the SCAE–SVM model in detail, as shown in Fig. 2. The training process can be divided into three stages: unsupervised greedy layer-wise pretraining, unrolling, and supervised fine-tuning.

In the pretraining stage, the greedy layer-wise strategy is used to train a series of basic CAEs separately, and the output of each CAE hidden layer is fed as the input of the next CAE network. Specifically, a CAE network is first trained by minimizing the objective function and the network parameters such as weight and bias are recorded. The second CAE network is then trained by taking the hidden-layer output of the first CAE as input, and so on. Each CAE network is trained and all parameters can be initialized.

After the pretraining of each basic CAE network, the hidden layer of each CAE network is unrolled and stacked into a deep CAE network (i.e., SCAE); in other words, only the encoder is retained while discarding the decoder along with its parameters.

Fine-tuning is the process of further adjusting the initial parameters to obtain an optimal model. For this purpose, we mainly use the multiclass cross-entropy loss function, which is the difference between the measurement of predicted target probability distribution and the actual probability distribution, as shown below.

$$L(y, y') = -\frac{1}{n}\sum_{i=1}^{n}[y^{(i)}\log y'^{(i)} + (1-y^{(i)})\log(1-y'^{(i)})] \tag{14}$$

where $\boldsymbol{y}$ and $\boldsymbol{y}'$ denote the actual and predicted values of a class, respectively. Here, back-propagation (BP) and the ADAM optimizer are used to determine the minimum loss value in order to deduce the corresponding learning parameters and achieve an optimized model.

After completing the three stages mentioned above, the exact structure of the SCAE can be definitely determined. For the case of the NSL-KDD dataset, the structure of the SCAE includes three hidden layers consisting of 28, 16, and 8 neurons, where each neuron represents a feature. These learned low-dimensional features can be used to train the classifier.

However, the classification power of SCAE with multiclass cross-entropy is relatively weak compared to other discriminative models such as the NN, DT, and SVM. SVM shows excellent performance in handling classification problems and has gradually emerged as a popular solution in the field of anomaly detection. Hence, we can combine the deep learning power of SCAE with the shallow learning technique of SVM.

4.3 Output Layer: SVM Classifier

SVM is essentially a binary classification model, but attack types in the cloud computing environment are diverse. Hence, more than one classifier should be employed. SVM can solve multi-class (m-class) classification problems, and it involves two methods: "one-versus-one" (OVO) and "one-versus-all" (OVA) [31]. OVO takes the i and j class samples from training dataset and labels them as positive and negative classes, respectively. Further, it constructs $m(m\text{-}1)/2$ binary classifiers. By contrast, OVA takes the j class samples from the training dataset labels them as a positive class; the remaining samples are labeled as a negative class. Further, it constructs m binary classifiers.

Obviously, the OVA approach requires fewer binary classifiers. Rifkin and Klautau [32] demonstrated that OVA is preferred to OVO. Hence, we employ the SVM using the OVA approach to construct our classifier.

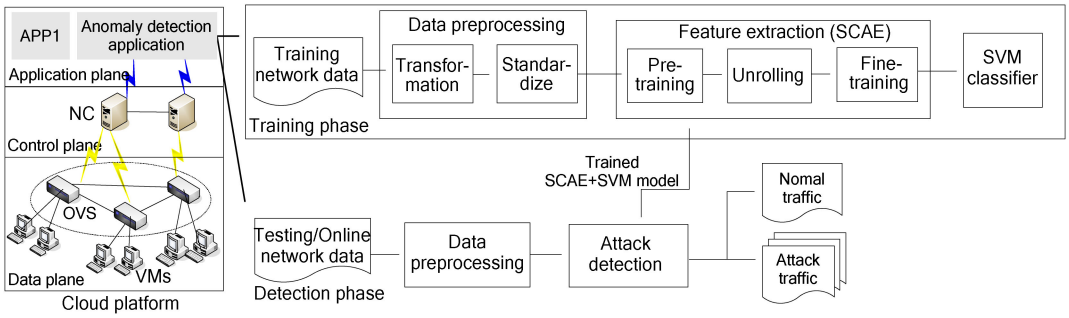


Fig. 3. Framework of cloud intrusion detection system.

5 Cloud Intrusion Detection System Based on SCAE and SVM

Here, we use SDN technology to build our CIDS, which decouples the traditional network structure into data plane, control plane, and application plane. The CIDS framework is shown in Fig. 3. An openflow virtual switch (OVS) is used to forward the virtual network flow; this represents the data plane. A network controller (NC) is used to install the flow table and routing control as well as to collect network traffic; this represents the control plane. NC configures and manages the OVS according to the openflow protocol. The application plane includes various applications to implement different functions, such as the anomaly detection application.

The anomaly detection application is used to achieve three main functions: (1) data preprocessing, where the network traffic is transformed and standardized, (2) classifier training, where the SCAE+SVM model used for feature extraction and classification detection is trained from the preprocessed network traffic, and (3) attack recognition, where the trained classifier is used to detect intrusion on the testing dataset or online network traffic.

5.1 Data Collection

Usually, data collection is the first and a critical step in intrusion detection. In our study, we use the OVS and NC to collect virtual network traffic data from the Xen cloud platform, as shown in Fig. 4. Each physical machine (PM) consists of a privileged domain named dom0 and a non-privileged domain named domU. PMs are connected by a traditional switch and VMs are connected by the OVS, which is deployed in dom0. A VM can communicate with another VM in the same or a different PM through the OVS. The OVS is used to forward virtual network flow, while the NC is responsible for routing control and network flow collection. The network traffic obtained by the NC is handed over to the anomaly detection application for the intrusion detection.

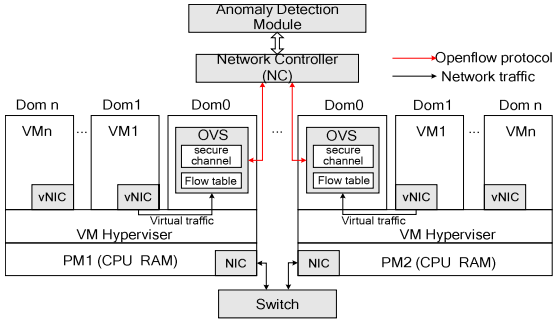


Fig. 4. Data collection from Xen cloud environment.

5.2 Data Preprocessing

Data preprocessing mainly includes data transformation and standardization. Data transformation is used to convert the nominal features into numeric values. For example, in the NSL-KDD dataset, there are three nominal features: protocol type, service type, and TCP status flag. The attack type is also nominal. We transfer the attack type into numeric values. For example, 0, 1, 2, 3, and 4 represent Normal, DOS, Probe, R2L, and U2R, respectively.

In addition, to eliminate the bias in favor of features with greater values as well as the large number of sparse features whose values are 0, we use standardization to scale each feature value into a well-proportioned range. Here, we use the Z-score method for standardization.

5.3 SCAE+SVM Classifier

When building classifiers or other predictors, combining feature learning methods can lead to dimensionality reduction and high detection performance. Here, we use the SCAE deep learning algorithm to extract essential features from raw network traffic. Note that the SCAE is pretrained in an unsupervised mode and fine-tuned by employing a supervised back-propagation algorithm. Once the essential features are extracted, they will be used to train the SVM classifier. Here, the SVM classifier exploits the OVA approach to distinguish between normal and abnormal data. We consider SCAE+SVM as a whole or a black-box, and the learned features are not visible.

5.4 Attack Detection

After the SCAE+SVM classifier has been trained, we use the trained and saved classifier to detect the testing data or online traffic. When the network traffic is transported to the SCAE+SVM classifier, an output is generated, which indicates whether the data is normal or an attack (i.e., DOS, Probe, R2L, U2R). For example, if the classifier considers records as normal, then the records will be labeled as Normal, and others will be labeled as non-Normal. By contrast, if the classifier considered records as DOS, then the records will be labeled as DOS, and others will be labeled as non-DOS (including Normal, Probe, R2L, and U2R) and so on.

TABLE 1
Sample Distribution of KDD Cup 99 and NSL-KDD Datasets

Class	Attack type	KDD Cup 99		NSL-KDD	
		Training	Testing	Training	Testing
Dos	Back	2203	1098	956	359
	Land	21	9	18	7
	Neptune	107201	58001	41214	4657
	Pod	264	87	201	41
	Smurf	280790	164091	2646	665
	Teardrop	979	12	892	12
Probe	Ipsweep	1247	306	3599	141
	Nmap	231	84	1493	73
	portsweep	1040	354	2931	157
	Satan	1589	1633	3633	735
R2L	Ftp_write	8	3	8	3
	Guess_passwd	53	4367	53	1231
	Imap	12	1	11	1
	Multihop	7	18	7	18
	Phf	4	2	4	2
	Spy	2	0	2	0
	Warezclient	1020	0	890	0
	Warezmaster	20	1602	20	944
U2R	Loadmodule	9	2	9	2
	Buffer_overfolw	30	22	30	20
	Rootkit	10	13	10	13
	Perl	3	2	3	2
Normal		97278	60593	67343	9711
Total		494021	292300	125973	18794

6 Experimental Results and Analysis

To verify the effectiveness of the proposed SCAE+SVM model, we conduct some experiments. First, we introduce the NSL-KDD [33] and KDD Cup 99 [34] datasets used in this paper. Second, we describe the experimental design and environment, including classification performance metrics and model parameters. Finally, we present some experimental results and a comparative analysis.

6.1 Dataset

In our experiments, we use two well-known intrusion detection evaluation datasets that are widely used to validate CIDS. The KDD Cup 99 dataset contains around five million training data records and two million testing data records. Here, we use around 10% of the records, i.e., 494,021 training data records and 311,029 testing data records, to evaluate the SVM classifier. By removing the records that appear only in the testing data but not in the training data, we are left with 292,300 testing records. Each record consists of 41 different features and is labeled as either normal or an attack. These attacks can be classified into four types: DOS, Probe, R2L, and U2R.

As a revised version of the KDD Cup 99 dataset, the NSL-KDD dataset proposed by Tavallaee *et al*. [35] contains 125,973 training data records and 22,544 testing data records. Similarly, by removing the records that appear only in the testing data but not in the training data, we are left with 18,794 testing records. Each record in the NSL-KDD dataset is also composed of 41 different features. Table 1 summarizes the exact distribution of the KDD Cup 99 and NSL-KDD datasets.

Although these two datasets have some limitations and 41 is a relatively small number of dimensional features, they are used widely in similar studies. Therefore, we validate the performance of the SCAE+SVM model using these two datasets.

TABLE 2
Models Considered in This Work for Comparison With SCAE+SVM Model

Acronyms	Description
SVM	Support Vector Machine
SAE+SVM	Stacked Auto-Encoder plus Support Vector Machine
SDAE+SVM	Stacked Denoising Auto-Encoder plus Support Vector Machine
STL [21]	Sparse Auto-Encoder plus Softmax
S-NDAE [25]	Stacked Non-symmetric Auto-Encoder plus Softmax
SCAE+SVM	Stacked Contractive Auto-Encoder plus Support Vector Machine

6.2 Experimental Design and Environment

Here, we design two groups of experiments to verify the effectiveness of the proposed SCAE+SVM model. These two experiments answer two questions: (1) whether SCAE can extract effective features and achieve the objective of dimensionality reduction, (2) whether the proposed SCAE+SVM model can effectively improve the detection performance of the CIDS.

Therefore, we will prove the ability of the SCAE to generate low-dimensional features as well as the effect of parameters in the SCAE+SVM model on the intrusion detection efficiency. Then, we will compare the results of the SCAE+SVM method with those of three state-of-the-art approaches and those obtained by two similar methods (see Table 2). The structures of these three approaches are similar to that of the SCAE+SVM (here, SDAE employs 0.3 times Gaussian noise). We perform three classification tasks, i.e., 2-class, 5-class, and 13-class classification, on the NSL-KDD dataset. And executing two classification tasks, i.e., 2-class and 5-class classification, on the KDD Cup 99 dataset. Specifically, 2-class classification involves normal and attack data, 5-class classification involves normal data and four types of attack traffic data (i.e., DOS, Probe, U2R, and R2L), and 13-class classification involves more than the minimum 20 entries of the training data in Table 1.

In general, six metrics are used for evaluating the detection performance: accuracy rate (*ACC*), precision rate (*P*), recall rate (*R*), *f*-measure (*F*), confusion matrix (*M*), and receiver operating characteristic (*ROC*) [18]. They are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

$$F\text{-}measure = 2 \times \frac{precision \times recall}{precision + recall}, \tag{18}$$

where the accuracy rate is the proportion of records that are all correctly identified. The precision rate is the proportion

TABLE 3
Comparison of Different SCAE+SVM Structures

Model	ACC	P	R	F
$SCAE^1$+SVM	75.97	73.49	75.97	71.15
$SCAE^2$+SVM	85.45	86.73	85.45	81.70
$SCAE^3$+SVM	85.46	84.44	85.46	80.96
$SCAE^4$+SVM	**87.33**	**87.96**	**87.33**	**85.01**
$SCAE^5$+SVM	83.57	82.70	83.57	79.26

of correctly identified records among all identified attack records. The recall rate represents the percentage of records that are correctly identified for the original type of attack. The *f*-measure better evaluates the performance because it is the harmonic mean of the precision rate and recall rate. These four metrics can be obtained by a confusion matrix M, and the leading diagonal elements in the confusion matrix are the number of records correctly predicted. For example, the element $M_{i,j}$ denotes the number of records that are actually from the class i but incorrectly identified to be from the class j. The ROC curve illustrates the classification performance through the true positive rate (TP) and false positive rate (FP). The larger the area under the ROC curve (AUC), the higher is the TP and the lower is the FP.

Similar to most existing deep learning models, the proposed SCAE+SVM model was implemented using TensorFlow running on a 64-bit Ubuntu 16.04 machine with the configuration of Intel Core 2 Duo processor (2.7 GHz) with 8 GB RAM. The hyperparameters of the SCAE+SVM model are set as follows.

In the pretraining stage, each CAE network structure is trained: 41-28-41, 28-16-28, and 16-8-16. The number of training epochs is 20, the learning rates are 0.03, 0.01, 0.01 respectively. And the penalty coefficient λ of each CAE network is set to 100, 40, and 10 respectively.

In the fine-tuning stage, we cascade all the CAE networks to form a 41-28-16-8-5 structure, and optimize all the parameters. The number of epochs is 50, and the learning rate is 0.001.

The SVM classifier uses the Gaussian radial basis function (RBF) kernel. The best values of the penalty coefficient c and the kernel parameter *gamma* obtained through grid search in the closed interval [2], [3], [4], [24] are 1 and 0.125, respectively.

TABLE 4
Three Types of Classification Tasks on NSL-KDD Dataset

(a) detection performance of 2-class classification task

Algorithm	ACC	P	R	F
SVM	77.71	82.99	77.71	76.59
SAE+SVM	85.09	85.78	85.09	84.83
SDAE+SVM	85.61	87.14	85.61	85.40
STL [21]	88.39	85.44	**95.95**	**90.4**
SCAE+SVM	**88.73**	**89.87**	88.73	88.04

(b) detection performance of 5-class classification task

Algorithm	ACC	P	R	F
SVM	75.97	73.49	75.97	71.15
SAE+SVM	81.71	82.05	81.71	78.12
SDAE+SVM	84.04	84.54	84.04	80.88
STL [21]	79.10	83.00	69.00	75.76
S-NDAE [25]	85.42	**100**	85.42	**87.37**
SCAE+SVM	**87.33**	87.96	**87.33**	85.01

(c) detection performance of 13-class classification task

Algorithm	ACC	P	R	F
SVM	80.18	77.15	80.18	76.54
SAE+SVM	86.60	81.82	86.60	83.75
SDAE+SVM	87.75	88.61	87.75	84.24
S-NDAE [25]	89.22	**92.97**	89.22	**90.76**
SCAE+SVM	**89.93**	91.04	**89.93**	86.94

TABLE 5
Two Types of Classification Tasks on KDD Cup 99 Dataset

(a) detection performance of 2-class classification task

Algorithm	ACC	P	R	F
SVM	87.39	92.13	87.39	88.31
SAE+SVM	97.64	97.76	97.64	97.67
SDAE+SVM	97.82	97.91	97.82	97.85
SCAE+SVM	**98.11**	**98.21**	**98.11**	**98.13**

(b) detection performance of 5-class classification task

Algorithm	ACC	P	R	F
SVM	76.94	87.69	76.94	78.13
SAE+SVM	93.37	96.48	93.37	94.19
SDAE+SVM	93.30	96.86	93.30	94.54
S-NDAE [25]	97.85	**99.99**	97.85	**98.15**
SCAE+SVM	**97.87**	98.13	**97.87**	97.96

6.3 Experimental Results and Comparative Analysis

6.3.1 Ability of SCAE to Extract Essential Features

In the previously published article [36], we demonstrated that the classification accuracy does not improve as the number of features increases. We can conclude that the accuracy of a classifier that employs a smaller number of feature sets is similar to or better than that of a classifier that employs all the features. Therefore, we need to extract essential or important features from the original features and build a more effective classifier. Feature learning can achieve this goal by extracting a smaller number of features that best represent the original data.

The exact structure of the SCAE model can be determined through experiments with numerous structural combinations (i.e., number of hidden layers and neurons). Next, we discuss how to select an appropriate SCAE model experimentally. The depth of the SCAE network has a significant influence on the effects of feature learning and classification. As the number of network layers increases, better feature representation can be extracted and the classification performance can be improved. However, some researchers [37] have shown that the training time increases considerably

TABLE 6
Precision, Recall, and F-Measure of Various Learning Methods on NSL-KDD Dataset

Attack class	SAE+SVM			SDAE+SVM			SCAE+SVM		
	P	R	F	P	R	F	P	R	F
Normal	0.78	0.93	0.85	0.81	0.92	0.87	**0.84**	**0.96**	**0.89**
DOS	0.91	0.93	0.92	0.91	**0.99**	0.95	**0.96**	**0.99**	**0.98**
Probe	0.68	0.75	0.71	0.70	0.74	0.72	**0.76**	**0.81**	**0.78**
R2L	0.81	0.09	0.17	0.90	0.15	0.25	**0.93**	**0.24**	**0.38**
U2R	**1.00**	**0.05**	**0.10**	0.00	0.00	0.00	0.25	**0.05**	0.09
Total	0.82	0.82	0.78	0.84	0.86	0.81	**0.88**	**0.87**	**0.85**

TABLE 7
Confusion Matrix of 5-Class Classification Task

	Normal	DOS	Probe	R2L	U2R
Normal	**9325**	123	217	40	6
DOS	14	**5667**	60	0	0
Probe	131	79	**896**	0	0
R2L	1664	4	9	**522**	0
U2R	29	4	0	2	**2**

with the number of network layers, and additional layers will easily lead to over-fitting. In our experiment, five different types of SCAE+SVM models were set up on the NSL-KDD dataset. The comparative analysis results in terms of the accuracy rate are shown in Table 3.

$SCAE^1$+SVM is set as a shallow network, i.e., a 41-5 structure. Similarly, $SCAE^2$+SVM, $SCAE^3$+SVM, $SCAE^4$+SVM, and $SCAE^5$+SVM are set as 41-20-5, 41-28-16-5, 41-28-16-8-5, and 41-28-20-12-6-5 structures, respectively. From Table 3, the $SCAE^4$+SVM model shows the best classification performance. Thus, SCAE can extract optimal features and achieve the objective of dimensionality reduction. The performance of deep structures is better than that of shallow structures because multi-layer mapping units can extract important structural information.

6.3.2 Classification Performance of the SCAE+SVM

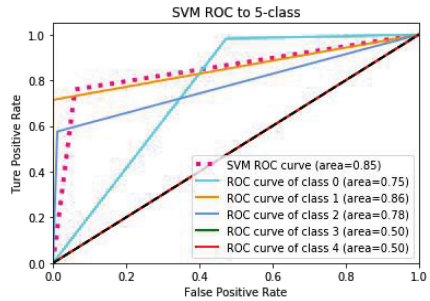
To further explore the validity of the proposed SCAE+SVM model, we evaluate the detection performance of the model and perform comparative analysis with some other methods as mentioned previously. We evaluate our proposed model on three types of classification tasks: 2-class, 5-class, and 13-class classification. The numbers of hidden layers and neurons of SAE and SDAE are the same as those of SCAE. Tables 4 (a)–(c) show the detection performance of 2-class, 5-class, and 13-class classification tasks on the NSL-KDD dataset. Tables 5 (a)–(b) show the detection performance of 2-class and 5-class classification tasks on the KDD Cup 99 dataset.

From the experimental results, we can observe the following:

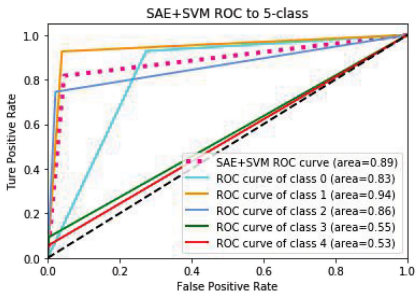
1) The SVM classifiers combined with different deep learning methods are superior to the standalone shallow SVM. The SAE, SDAE, and SCAE deep learning algorithms all achieve the goal of dimensionality reduction. They can not only capture the essential features but also improve the classification accuracy. We demonstrate that combining deep and shallow learning techniques, can play to their respective strengths and achieve better detection performance.
2) In the 2-class classification task, for the NSL-KDD dataset, the SCAE+SVM model has the highest accuracy rate and precision rate. However, the recall rate and *f*-measure of the SCAE+SVM model are higher than those of the SAE+SVM and SDAE+SVM methods and lower than those of the STL method.
3) In the 2-class classification task, for the KDD Cup 99 dataset, the accuracy rate, precision rate, recall rate, and *f*-measure of the SCAE+SVM model are all the highest, albeit only slightly better than those of the other three methods.
4) In the 5-class classification task, for the NSL-KDD dataset, the SCAE+SVM model achieves the highest accuracy rate and recall rate among all the models. However, the precision rate and *f*-measure of the SCAE+SVM model are higher than those of all models except the S-NDAE. The AUC value of the S-NDAE method illustrated later is lower than that of our SCAE+SVM model.

TABLE 8
Classification Performance of 13-Class Classification Task on NSL-KDD Dataset

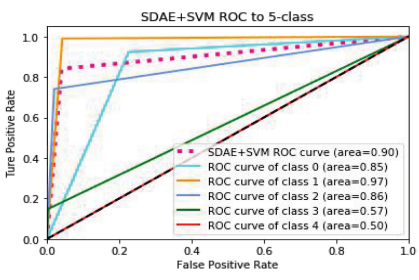
Attack class	Training	Testing	SAE+SVM			SDAE+SVM			SCAE+SVM		
			P	R	F	P	R	F	P	R	F
back	956	359	**0.99**	0.59	0.74	0.74	0.06	0.10	0.98	**0.97**	**0.97**
buffer_overflow	30	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
guess_passwd	53	1231	0.00	0.00	0.00	0.92	0.01	**0.02**	**1.00**	**0.01**	**0.02**
ipsweep	3599	141	**0.95**	**0.97**	**0.96**	0.79	0.97	0.87	0.82	**0.97**	0.89
neptune	41214	4657	**1.00**	**0.99**	**0.99**	0.99	**0.99**	**0.99**	0.99	**0.99**	**0.99**
nmap	1493	73	0.46	0.99	0.63	**0.81**	**1.00**	**0.90**	0.58	0.85	0.69
pod	201	41	0.62	0.73	0.67	**0.64**	**0.88**	**0.74**	0.58	0.27	0.37
portsweep	2931	157	0.51	**0.96**	0.67	0.48	0.94	0.64	**0.59**	0.89	**0.71**
satan	3633	735	**0.81**	0.52	0.63	0.75	0.80	0.77	0.80	**0.86**	**0.83**
smurf	2646	665	0.79	0.71	0.75	**0.97**	0.98	0.97	0.96	**1.00**	**0.98**
teardrop	892	12	0.18	0.83	0.29	0.24	**1.00**	**0.39**	0.22	**1.00**	0.36
warezclient	890	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Normal	67343	9711	0.85	0.93	0.89	0.85	**0.96**	0.91	**0.88**	**0.96**	**0.92**
Total	125881	17802	0.82	0.87	0.84	0.89	0.88	0.84	**0.91**	**0.90**	**0.87**



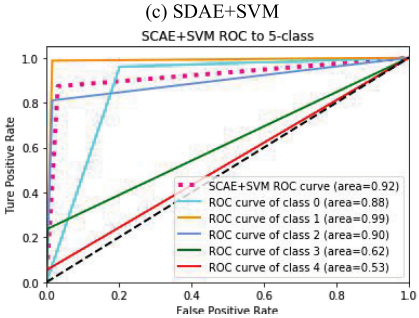
(a) SVM



(b) SAE+SVM

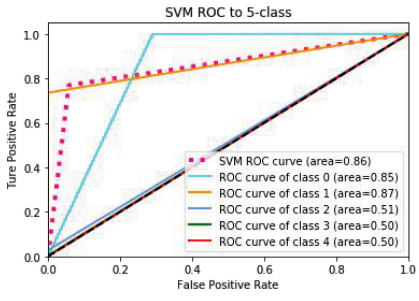


(c) SDAE+SVM

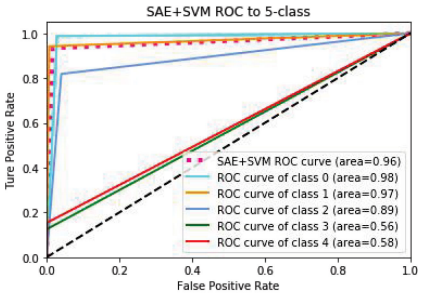


(d) SCAE+SVM

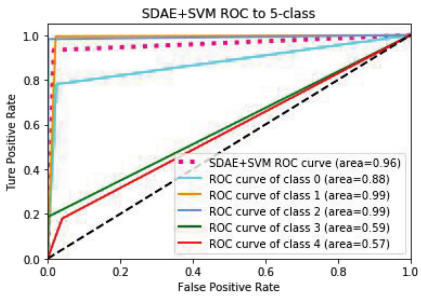
Fig. 5. ROC curves for various methods in 5-class classification tasks on NSL-KDD dataset.



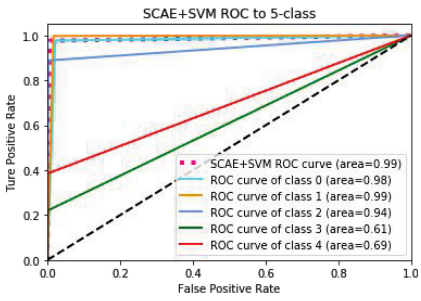
(a) SVM



(b) SAE+SVM



(c) SDAE+SVM



(d) SCAE+SVM

Fig. 6. ROC curves for various methods in 5-class classification tasks on KDD Cup 99 dataset.

5) In the 5-class classification task, for the KDD Cup 99 dataset, the accuracy rate and the recall rate of the SCAE+SVM model are higher than those of the SAE+SVM and SDAE+SVM methods and slightly higher than those of the S-NDAE models. However, the precision rate and *f*-measure are lower than those of the S-NDAE model.
6) In the 13-class classification task, for the NSL-KDD dataset, the SCAE+SVM model achieves the highest accuracy rate and recall rate among all the models. However, the precision rate and *f*-measure of the SCAE+SVM model are lower than those of the S-NDAE model.

In summary, the SCAE+SVM method achieves better detection performance than the three state-of-the-art approaches, and the results are better or at least match those of the similar studies compared herein.

Table 6 lists the precision rate, recall rate, and *f*-measure of every class of the 5-class classification task on NSL-KDD dataset. We can see that our model achieves superior performance in terms of most of the measures except the U2R class. The results re-emphasize that our model does not handle smaller classes such as "R2L" and "U2R", which show lower performance because the data size affects the classification results to some degree.

The confusion matrix of the 5-class classification obtained using our SCAE+SVM model is presented in Table 7. The values of the leading diagonal denote the number of correctly classified records of the testing dataset. "R2L" and "U2R" have more samples that are identified incorrectly.

TABLE 9
AUC Value of Four Methods

(a) AUC value of four methods in 5-class classification tasks

Method	SVM	SAE+SVM	SDAE+SVM	SCAE+SVM
NSL-KDD	0.85	0.89	0.90	**0.92**
KDD CUP 99	0.86	0.96	0.96	**0.99**

(b) AUC value of four methods in 2-class classification tasks

Method	SVM	SAE+SVM	SDAE+SVM	SCAE+SVM
NSL-KDD	0.77	0.85	0.85	**0.86**
KDD CUP 99	0.92	**0.98**	**0.98**	**0.98**

TABLE 10
AUC Value of Two Methods in 5-Class Classification Tasks on NSL-KDD Dataset

Method	Normal	DOS	Probe	R2L	U2R
S-NDAE [25]	0.82	0.95	**0.99**	0.45	0.44
SCAE+SVM	**0.88**	**0.99**	0.90	**0.53**	**0.62**

Next, we evaluate the detection performance of the proposed SCAE+SVM model on the 13-class classification task. These 13 classes are those with more than the minimum 20 entries listed in Table 1. The purpose of the experiment is to determine whether the proposed deep learning model can identify each type of attack with fine granularity and maintain stable detection performance when the number of attack categories increases. The corresponding performance analysis is presented in Table 8. The experimental results show the following:

1) The total detection performance of the SCAE+SVM model is satisfactory and stable. It achieves the highest level compared to other methods.
2) All the classes of our model achieve better detection performance, except for three classes: buffer_overflow, nmap, and pod. Because the number of testing data of the warezclient class is 0, so the detection performance of this attack is 0.

Figs. 5a, 5b, 5c, 5d and 6a, 6b, 6c, 6d show the ROC curves of four different methods in 5-class classification tasks on the NSL-KDD and the KDD Cup 99 datasets. The dotted lines represent the ROC curve of the total classification performance of the method. The other lines represent the five attack types. A larger area under the ROC curve implies a high true positive rate and a low false positive rate. As can be seen, the area under the ROC curve of the SCAE+SVM model is the largest; thus, it can be considered to have the best performance.

The AUC values of four methods are listed in Tables 9 (a)–(b). In the 5-class classification tasks, the AUC values of the SCAE+SVM model on the NSL-KDD and the KDD Cup 99 datasets are 0.92 and 0.98, respectively. This implies that the proposed SCAE+SVM method achieves higher detection performance than the other methods. In addition, in the 2-class classification tasks, the AUC values of the SCAE+SVM on the NSL-KDD dataset is 0.86, which implies higher detection performance than the other method. On the KDD Cup 99 dataset, the AUC values of the SAE+SVM, SDAE+SVM, and SCAE+SVM methods are all 0.98, that is, all achieve high detection performance. Thus, our method performs well in the 2-class classification task.

TABLE 11
AUC Value of Four Methods in 13-Class Classification Tasks on NSL-KDD Dataset

Method	SVM	SAE+SVM	SDAE+SVM	SCAE+SVM
NSL-KDD	0.89	0.93	0.93	**0.95**

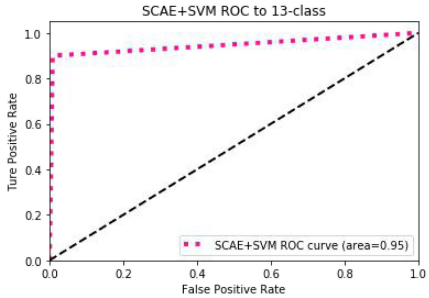


Fig. 7. ROC curve for SCAE+SVM method in 13-class classification task on NSL-KDD dataset.

Table 10 lists the AUC values of the 5-class classification task in two methods—the S-NDAE and SCAE+SVM methods—on the NSL-KDD dataset. These results indicate that our proposed method has four AUC values superior to the S-NDAE.

In the 13-class classification tasks, the AUC values of four methods are listed in Table 11, SCAE+SVM achieves an AUC value of 0.95 on the NSL-KDD dataset; thus, it achieves higher detection performance than the other methods. The ROC curve of our method in the 13-class classification task is shown in Fig. 7.

From the above-mentioned results, it is nearly certain that our proposed SCAE+SVM method has a high true positive rate and a low false positive rate. Thus, our method performs well in the 2-class, 5-class, and 13-class classification tasks.

7 Conclusion

Security concerns not only lead to severe losses in the cloud computing environment but also cause users to lose confidence in cloud computing itself, which will inevitably have a serious impact on the healthy and sustainable development of cloud computing. Building a cloud intrusion detection system is one of the solutions for protecting cloud computing from malicious attacks. Recently, researchers have demonstrated that an efficient and effective CIDS can be built by combining a deep learning algorithm for feature extraction with a classifier. In this study, we designed a hybrid system that uses a stacked contractive auto-encoder (SCAE) for feature reduction and the SVM classification algorithm for the detection of malicious attacks. Using the NSL-KDD and KDD Cup 99 intrusion detection datasets, we experimentally demonstrated that the proposed SCAE+SVM-IDS model achieves promising classification performance in terms of six metrics compared with three state-of-the-art methods.

Although the proposed SCAE+SVM-IDS approach has shown encouraging performance, it can be improved by further optimizing the classifier. The SVM classifier cannot effectively recognize some new attacks existing in the testing dataset. Therefore, designing an optimal classifier requires careful consideration in future studies. As a long-term objective for future work, on the one side, we aim to reduce the controller's bottleneck and implement an CIDS that can detect different kinds of network attacks. On the other side, we plan to implement our solution in a real cloud environment to evaluate its performance.

Acknowledgments

The authors were very grateful to the anonymous reviewers and editor for their helpful and constructive comments and suggestions. This work was supported by the Natural National Key Basic Research Program of China under Grant 2016YFB050190104 and the National Natural Science Foundation of China (61802436, 61702550).

References

[1] P. M. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, 2011. [Online]. Available: http://dx.doi.org/10.6028/nist.sp.800-145

[2] C. G. C. Index, "Forecast and methodology, 2016–2021 White Paper," 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[3] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," in *Proc. IEEE Int. Conf. Netw. Protoc.*, 2012, pp. 1–6.

[4] C J Chung, P Khatkar, T Xing, J. Lee, and D. Huang,"NICE: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Trans. Depend. Secure Comput.*, vol. 10 no. 4, pp. 198–211, Jul./Aug. 2013.

[5] T. Xing, Z. Xiong, D. Huang, and D. Medhi, "SDNIPS: Enabling software-defined networking based intrusion prevention system in clouds," in *Proc. Int. Conf. Netw. Serv. Manage. Workshop*, 2014, pp. 308–311.

[6] J. S. Cui, C. Guo, L. Chen, Y. N. Zhang, and D. Huang, "Establishing process-level defense-in-depth framework for software defined networks," *J. Softw.*, vol. 25, no. 10, pp. 2251–2265, Oct. 2014, doi: 10.13328/j.cnki.jos.004682.

[7] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006, doi: 10.1126/science.1127647.

[8] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Adv. Bioinf.*, vol. 2015, pp. 1–13, 2015, doi: 10.1155/2015/198363.

[9] A. Kannan, G. Q. Maguire Jr, A. Sharma, and P. Schoo, "Genetic algorithm based feature selection algorithm for effective intrusion detection in cloud networks," in *Proc. IEEE 12th Int. Conf. Data Mining Workshops*, 2012, pp. 416–423. [Online]. Available: http://dx.doi.org/10.1109/icdmw.2012.56

[10] A. Kannan *et al.*, "A novel cloud intrusion detection system using feature selection and classification," *Int. J. Int. Inf. Technol.*, vol. 11, no. 4, pp. 1–15, 2015, doi: 10.4018/ijiit.2015100101.

[11] O. Osanaiye *et al.*, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," *EURASIP J. Wireless Commun. Netw.*, vol. 2016, no. 1, 2016, Art. no. 130, doi: 10.1186/s13638-016-0623-3.

[12] A. Javadpour, S. Kazemi Abharian, and G. Wang, "Feature selection and intrusion detection in cloud environment based on machine learning algorithms," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. IEEE Int. Conf. Ubiquitous Comput. Commun.*, 2017, pp. 1417–1421. [Online]. Available: http://dx.doi.org/10.1109/ispa/iucc.2017.00215

[13] N. M. Ibrahim and A. Zainal, "A feature selection technique for cloud IDS using ant colony optimization and decision tree," *Adv. Sci. Letts.*, vol. 23, no. 9, pp. 9163–9169, 2017, doi: 10.1166/asl.2017.10045.

[14] Y. L. Cun *et al.*, "Handwritten digit recognition: Applications of neural network chips and automatic learning," *IEEE Commun. Mag.*, vol. 27, no. 11, pp. 41–46, Nov. 1989, doi: 10.1109/35.41400.

[15] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006, doi: 10.1162/neco.2006.18.7.1527.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[17] G Alain and Y Bengio. "What regularized auto-encoders learn from the data generating distribution," *J. Mach. Learn. Res.*, vol. 15 no. 1, pp. 3563–3593, 2014.

[18] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun.*, 2016, pp. 258–263. [Online]. Available: http://dx.doi.org/10.1109/wincom.2016.7777224

[19] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme," in *Advances in Intelligent and Soft Computing*, K. Janusz, Ed., Berlin Germany: Springer, 2011, pp. 293–303.

[20] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *Proc. IEEE 4th Int. Conf. Cyber Secur. Cloud Comput.*, 2017, pp. 193–198.

[21] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol.*, 2016, pp. 21–26. [Online]. Available: http://dx.doi.org/10.4108/eai.3-12-2015.2262516

[22] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," *ICST Trans. Secur. Safety*, vol. 4, no. 12, 2017, Art. no. 153515, doi: 10.4108/eai.28-12-2017.153515.

[23] J Kim *et al.*, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput.*, 2017, pp. 313–316.

[24] D Papamartzivanos, F. G. Marmol, and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," *IEEE Access*, vol. 2019, pp. 13546–13560, 2019.

[25] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/tetci.2017.2772792.

[26] G Loukas *et al.*, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6 no. 1, pp. 3491–3508, 2017.

[27] Y. Yu, J. Long, and Z. Cai, "Session-based network intrusion detection using a deep learning architecture," in *Modeling Decisions for Artificial Intelligence*, Berlin, Germany: Springer, 2017, pp. 144–155.

[28] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Secur. Commun. Netw.*, vol. 2017, pp. 1–10, 2017, doi: 10.1155/2017/4184196.

[29] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103. [Online]. Available: http://dx.doi.org/10.1145/1390156.1390294

[30] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: explicit invariance during feature extraction," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 833–840.

[31] J. Weston and C. Watkins, "Multi-class support vector machines," Support Vector Machines for Pattern Classification. London, U.K.: Springer, 2005.

[32] R. Rifkin and K. Aldebaro, "In defense of one-vs-all classification," *Learn. Res.*, vol. 5, no. 1, pp. 101–141, 2004.

[33] *NSL_KDD dataset*. 2009. [Online]. Available: http://www.unb.ca/cic/datasets/nsl.html

[34] *KDD CUP 1999 dataset*. 1999. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data

[35] 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.*, 2009, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/cisda.2009.5356528

[36] W. Wang, X. Du, and N. Wang, "Building a cloud IDS using an efficient feature selection method and SVM," *IEEE Access*, vol. 7, pp. 1345–1354, 2019, doi: 10.1109/access.2018.2883142.

[37] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.*, vol. 1, no. 10, pp. 1–40, 2009.

Wenjuan Wang received the master's degree from Zhengzhou University, China, in 2007. Currently, she is working toward the PhD degree in the PLA Strategic Support Force Information Engineering University. From 2016 she was working as an associate professor with the PLA Information Engineering University. Her primary research interests include intrusion detection and security of cloud computing.

Xuehui Du received the PhD degree from the PLA Information Engineering University, in 2011. From 2010 she is working as a professor with the PLA Strategic Support Force Information Engineering University. Her current research works include network security and security of big data. She has more than 23 years of research and teaching experience.

Dibin Shan received the master's degree from the PLA Information Engineering University, China, in 2008. From 2011 he is working as a lecturer with the PLA Strategic Support Force Information Engineering University. His research interests include network security and security of big data.

Ruoxi Qin received the BS degree in information engineering from Electronic Engineering Institute, Hefei, China, in 2017, currently he is working toward the master's degree in the PLA Strategic Support Force Information Engineering University. His research interests include image processing, deep learning, and artificial intelligence.

Na Wang received the PhD degree from the PLA Information Engineering University, China, in 2008. From 2011 she is working as an associate professor with the PLA Strategic Support Force Information Engineering University. Her research interest includes network security.