Network Intrusion Detection System (NIDS) Based on Pseudo-Siamese Stacked Autoencoders in Fog Computing

Shanshan Tu, *Member, IEEE,* Muhammad Waqas, *Senior Member, IEEE,*
Akhtar Badshah, Mingxi Yin, Ghulam Abbas, *Senior Member, IEEE*

***Abstract*—The proliferation of Internet of Things (IoT) devices in the 5G era has resulted in increased security vulnerabilities and zero-day attacks, underscoring the importance of network intrusion detection systems (NIDS). However, existing NIDS have limitations in terms of accuracy, recall rates, false alarm rates, and generalization capabilities, and they cannot meet the IoT's requirements for low latency and limited computing resources. To overcome these challenges, we propose a NIDS based on a pseudo-siamese stacked autoencoder (PSSAE), deployed in the fog computing layer. Our system uses unsupervised training of stacked autoencoders (SAEs) to extract deep semantic features of normal and abnormal traffic, followed by supervised learning with labels to improve characterization and classification capabilities. The results show that our proposed method's accuracy and detection rate (DR) is 2% to 15% and 1%-14% higher than the existing techniques using the KDDTest+ dataset, respectively. Our proposed method outperformed the existing methods by 1% to 4% using the KDDTest+ dataset. The F1-Score is higher by 3% - 11.55% using the KDDTest+ dataset. On the other hand, using the KDDTest-21 dataset, the accuracy of our proposed method also outperformed the existing technique by 6.09% - 13.81%. The DR and F1-Score are higher by 7.02% and 5.57%, respectively, using the KDDTest+ dataset. This is due to the fact that each layer of the network trained by SAEs is more capable of extracting the semantic features of the data than the DNN-trained network directly.**

***Index Terms*—Intrusion detection, IoT, fog computing, autoencoder, pseudo-siamese neural network.**

I. Introduction

Internet of Things (IoT) devices usually collect and process spatiotemporal information about specific events to complete various tasks. Therefore, the IoT plays a pivotal role in various industrial fields, such as logistics tracking, energy distribution, smart cities, and healthcare [1]. However, the rapid commercialization of the IoT has caused its security issues to be seldom paid attention to due to the diversity of devices, communication protocols, interfaces, and services. As a result, the IoT has become one of the weakest links in cyber security, and smart devices are connected to the Internet. Therefore, the security attack can affect the security of the IoT ecosystem and threaten the entire Internet ecosystem. For example, in 2016, service provider DYN was attacked by up to 620 Gbps traffic, which caused hundreds of websites, such as Twitter, Netflix, Reddit, and GitHub, to shut down for several hours [2]. Therefore, the analysis and detection of attacks should be taken seriously.

The IoT also has the characteristics of low latency, resource constraints, distribution, scalability, and mobility [3]. In addition, IoT devices face the challenges of low computing power, smaller bandwidth, power, and storage. Therefore, fog computing emerges as a new type of distributed computing paradigm that extends the cloud to the edge of the network [4]. It can provide adequate data access, computing, networking, and storage for IoT devices and support mobility, location awareness, heterogeneity, and low latency [5]. The linkage between IoT devices, fog computing, and cloud architecture is shown in Fig. 1. We can deploy the intrusion detection system (IDS) to the fog layer and provide security services to IoT devices, effectively reducing the computing, storage, power consumption, and cloud server load of IoT devices. Considering the low latency of the IoT and the limited computing resources, if the IDS is deployed in the fog computing layer, the IDS requires less computing power and minimizes hardware costs.

According to the classification of the deployed platform, the IDS can be deployed on a single computer, known as host intrusion detection (HIDS) [6]. On the other side, the IDS deployed on large networks is called network intrusion detection system (NIDS) [7]. According to the classification of detection methods, IDS can be divided into anomaly-based intrusion detection (ANIDS) and signature-based intrusion detection (SNIDS) [8]. The former mainly calculates the deviation between normal and abnormal behavior at a given threshold. In comparison, an abnormal alarm is raised if the deviation exceeds a given threshold. The latter relies on matching the behavior model of the traffic with the known attack patterns in the database to identify the attack quickly. ANIDS can detect zero-day attacks. There is no specific signature pattern for zero-day attacks. Therefore, SNIDS cannot effectively detect new attacks. ANIDS estimation of the deviation be-

S. Tu and M. Yin are with the Engineering Research Center of Intelligent Perception and Autonomous Control, Faculty of Information Technology, Beijing University of Technology, Beijing, China (e-mail: yinmx.3@gmail.com, sstu@bjut.edu.cn).

M. Waqas is with the Computer Engineering Department, College of Information Technology, University of Bahrain, 32038, Bahrain and School of Engineering, Edith Cowan University, Perth WA 6027, Australia and also with the Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi 23460, Pakistan (e-mail: engr.waqas2079@gmail.com).

A. Badshah is with the Department of Software Engineering, University of Malakand, Dir Lower, Pakistan. (e-mail: akhtarbadshah@uom.edu.pk).

G. Abbas is with the Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi 23460, Pakistan (e-mail: abbasg@giki.edu.pk).

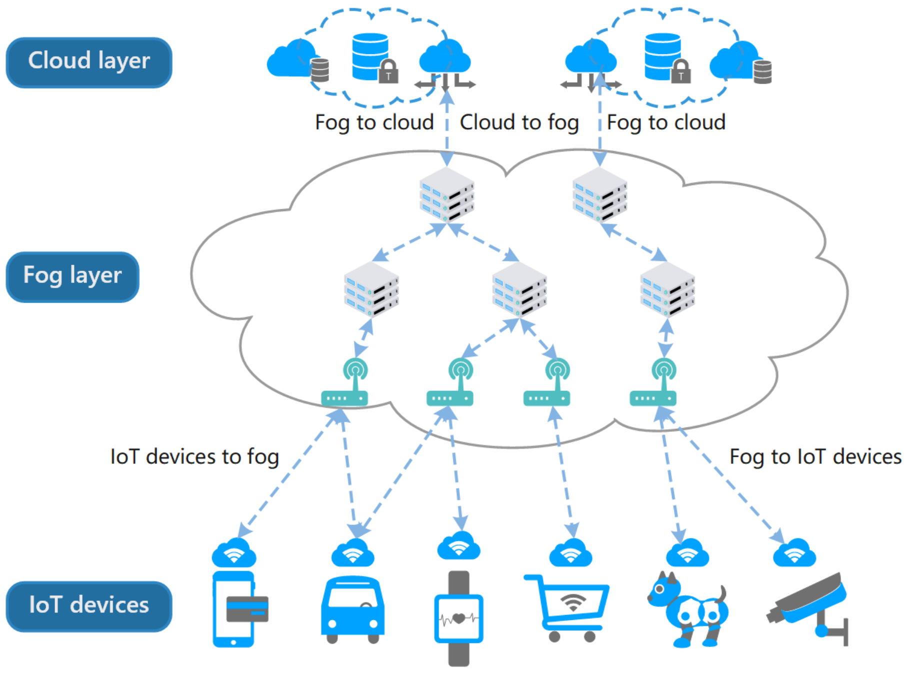

Fig. 1. Fog computing architecture

tween abnormal behavior and normal behavior determines the detection performance, leading to the possibility that ANIDS may recognize normal traffic as abnormal traffic. Therefore, for ANIDS, in addition to the accuracy rate, the true positive and false positive rates are also important evaluation indicators [9]. At present, many deep learning methods have been applied to ANIDS. Anomaly detection needs to learn the deviation between normal and abnormal samples based on large data. Deep learning has good performance in feature extraction; hence, it is very suitable to be applied to anomaly detection [10].

To improve the security of IoT devices without increasing the amount of computing for IoT devices, this paper proposes the IDS is deployed in the fog layer and SAE based algorithm runs on each fog node in a distributed manner. Through experiments, we observed that ordinary neural network models are challenging to learn the deep semantic features of normal and abnormal traffic and are insensitive to abnormal traffic. This leads to lower accuracy and detection, poor generalization ability, and a higher false-positive rate of NIDS based on traditional deep learning methods. Therefore, we propose NIDS based on pseudo-siamese stacked autoencoder (PSSAE). First, using positive and negative samples for unsupervised training, two stacked autoencoders (SAEs) that are pseudo-siamese structures can extract the deep semantic feature space of traffic and amplify the difference between positive and negative samples after feature reconstruction. Secondly, the ability of PSSAE classification can be improved through supervised training with labels. Finally, using logic operations to combine two pseudo-siamese SAEs makes the final detection result optimal. We built PSSAE using TensorFlow and verified the model with the NSL-KDD dataset. The results show that our proposed model is effective. In addition, our proposed model's hierarchical data format-5 (HDF5) file is only 500kb, which has low computational power requirements for fog nodes and will not add too much hardware cost. Our work has the following contributions.

- We propose NIDS based on pseudo-siamese stacked autoencoder. Using normal and abnormal data sets to train PSSAE unsupervised separately can effectively extract the deep semantic features of normal and abnormal traffic. It also amplifies the difference between normal and abnormal flow by reconstructing features.
- Supervised classification training of PSSAE based on feedforward neural network and backward propagation (BP) algorithm to fine-tune parameters. Hence, it further improves the ability of PSSAE to extract deep semantic features of data and maximize the deviation between normal traffic and abnormal traffic.
- Use logical operations to combine the detection results of two SAEs that are pseudo-siamese structures with each other to obtain the optimal detection result.
- The HDF5 file of our proposed model is only about 500kb, and the total number of parameters is about 60000, which requires less computing power for hardware. Therefore, building an IDS in the fog computing layer will not increase hardware costs.

The paper is structured as follows: After the introduction in Section I, we review related work on the subject in Section II. In Section III, we present our proposed IDS. In Section IV, we provide experimental verification and evaluation. Finally, we conclude the paper in Section V.

II. Related Work

Machine learning and deep learning have made significant achievements in the fields of computer vision [11], speech processing [12], and recommendation systems [13]. Due to the increasing diversity and complexity of network attacks, machine learning and deep learning are gradually being applied to NIDS [14]. For instance, the authors studied NIDS based on deep neural networks (DNN) in [15]. The DNN comprises an input layer, five hidden layers, and an output layer. The detection accuracy of the two classifications in the benchmark data set NSL-KDD is 78.9%. The authors of [16] studied an IDS based on RNN. The accuracy of the two classifications on KDDtest+ in the NSL-KDD data set was 83.28%, and the accuracy on KDDTes-21 was 68.55%. In addition, the authors of [17] studied offline IDS based on multi-layer perceptron (MLP) in the IoT environment, which is more inclined to detect denial of service (DoS) and distributed denial of service (DDoS) attacks. The accuracy rate of the IDS evaluated was 99.4%, and the false alarm rate was 0.6%. The authors of [18] proposed distributed lightweight IDS deployed in the fog layer based on an artificial immune system (AIS). The IDS has an accuracy rate of 98.35% and a false alarm rate of 3.51% on the data set composed of KDD-Cup99 and ISCX. Using the KDD-Cup99 dataset, the authors of [19] proposed intrusion detection based on a deep learning approach using autoencoders and isolation forests in fog computing. The method targeted the binary classification of the incoming packets to the fog nodes by differentiating the attacks from normal packets. Compared to [19], we used a pseudo-siamese stacked autoencoder to extract the data's deep semantic features, which helped us differentiate between positive and negative samples and normal and abnormal records.

In [20], the authors proposed an IDS that consists of two-dimensionality reduction layers and two classification layers for the backbone of the IoT. The proposed work mainly targets user-to-root and remote-to-local attacks. The accuracy of this IDS in NSL-KDD is 84.86%. In [21], the authors studied an unsupervised NIDS for the IoT environment based on conditional variational autoencoder (CVAE). CVAE can retrieve and reconstruct missing features from incomplete data sets. The accuracy of the proposed IDS on NSL-KDD (KDDTest+) is 85.97%. In [22], the authors studied distributed intrusion detection in the IoT environment based on fog computing and realized distribution by sharing model training parameters by adjacent fog nodes. The authors claimed that although training the model takes longer, it can make the detection faster and reduce the data transmitted to the cloud server. However, in the distributed training model, sharing parameters between adjacent nodes may increase the network delay when transmitting parameters. In [23], the authors proposed the fog computing distributed IDS. The IDS is based on the online sequential extreme learning machine (OS-ELM). The authors used NSL-KDD as a benchmark data set to evaluate the model. The results show that the IDS has higher accuracy and lower response time. Compared with IDS deployed in cloud servers, the detection speed of IDS deployed in fog nodes is increased by 25%. At the same time, the model can learn new data patterns online.

In [24], the authors studied the ELM-based semi-supervised fuzzy C-means (ESFCM) distributed NIDS deployed in the fog layer. The distributed IDS can solve the resource constraints and delays of the IoT through the fog computing problem. The authors used NSL-KDD as a benchmark data set to evaluate the IDS. The results showed that the IDS has a faster detection rate and higher accuracy than the centralized cloud IDS and traditional machine learning methods. The authors of [25] studied the IDS based on the asymmetrical structure of SAEs [26]. In such IDS, the authors constructed an asymmetric structure of SAE to extract deep semantic features and used random forest (RF) as the classifier. The accuracy rates of this IDS on NSL-KDD and KDD CUP 99 are 85.42% and 97.85%, respectively, and the false alarm rates are 14.58 and 2.15%, respectively. In [27], the authors studied deep CNN based on channel boosted and residual learning for NIDS. The authors used SAE to reconstruct the original features and proposed channel boosted to process the original features further. Finally, based on multi-path residual learning, CNN is used to learn features of different levels of granularity. The accuracy of the work on NSL-KDD is 87.28%. In [28], the authors deployed the NIDS in the fog layer, combining neural networks and KNN algorithms. The model includes two steps. Firstly, DNN is used for pre-detection, and secondly, KNN is used to detect the traffic whose predicted value is less than the threshold in the first step. The authors claimed that the hybrid IDS has higher detection accuracy than traditional machine learning methods.

III. Proposed NIDS

Through experimental analysis, we found that the accuracy and recall rate of IDS based on traditional neural networks is low, and the false alarm rate is high. This is because conventional neural networks, such as DNN and RNN, are challenging to extract higher-order text data features, making the model insensitive to normal and abnormal records during classification [29]. If the model extracts deep semantic features that better characterize the original data, the reconstruction feature can amplify the deviation between normal and abnormal records. Hence, the model accurately distinguishes between normal and abnormal records. We use SAEs to detect abnormal traffic more efficiently by increasing the deviation between abnormal data and normal records. The input and output dimensions of the SAEs are the same. Deep semantic features can be extracted from the original features by reconstructing the same output as the original input features. The reconstructed features can amplify the deviation between normal and abnormal records, and SAEs can be regarded as the deviation amplifier of normal and abnormal records. Therefore, we propose NIDS based on pseudo-twin SAEs to solve the above problems.

A. Background

To detect zero-day attacks more accurately, learning the deep semantic features of existing attacks is necessary. Therefore, we use a pseudo-siamese stacked autoencoder (PSSAE) to learn the deep semantic features of traffic.

1) Autoencoders (AEs): The AEs are based on a feedforward neural network, as shown in Fig. 2. Its input and output layers are usually the same, so the network can reconstruct the output to make it as close to the input as possible. Since labels are not required for training, this is an unsupervised nonlinear feature learning algorithm.

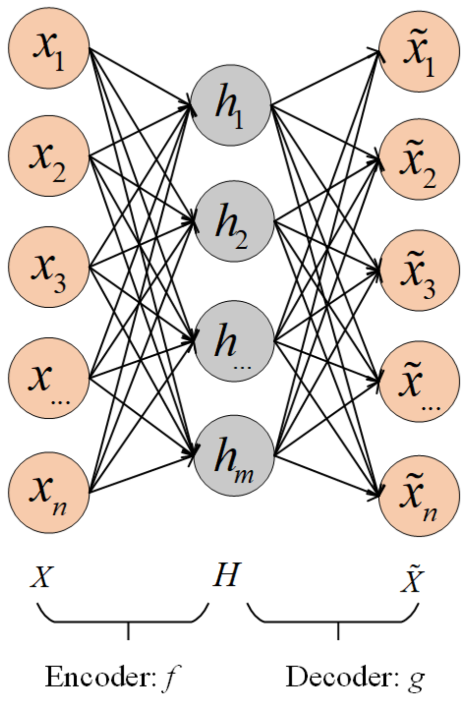

Fig. 2. General structure of autoencoder

As shown in Fig. 2, X is the input layer of AEs. H is the hidden layer of AEs. $\tilde{X}$ is the output layer of AEs. f is the encoder and g is the decoder. The encoding and decoding processes are shown in (1), (2), respectively.

$$f : x \rightarrow h : h_j = \sigma(W_{ei} x_i + b_{ei}), \tag{1}$$

$$g : h \rightarrow \tilde{x} : \tilde{x}_i = \sigma(W_{dj} h_j + b_{dj}), \tag{2}$$

where x_i is the original feature. x_i is converted to h_j after being encoded by the encoder. h_j is converted to $\tilde{x}_i$ after being decoded by the decoder g. W_e and b_e are the weight and bias of the encoder of f, respectively. W_d and b_d are the weight and bias of the decoder g, respectively. σ is the activation function. σ can make the original features nonlinear, thereby improving the model's ability. This study illustrates that the model can learn the deep semantic features of normal and abnormal traffic. Therefore, we can make the dimension of H smaller than X. When the AEs reconstruct the output feature $\tilde{x}$, which is as same as the input feature x, the hidden layer can capture the most prominent feature h among the original features at sparse AEs. The learning process of the AEs can be described as minimizing the loss function. Normally, the loss function is the mean square error as follows.

$$\begin{aligned} MSE &= \frac{1}{2n} \sum_{i=1}^{n} (x_i - g(f(x_i)))^2, \\ &= \frac{1}{2n} \sum_{i=1}^{n} (x_i - \tilde{x}_i)^2. \end{aligned} \tag{3}$$

2) Stacked autoencoder: As mentioned earlier, the hidden layer of AEs can extract the most significant feature of the original features. If multiple AEs are stacked, the feature extracted by the previous AE hidden layer is used as the input of the next AE. By analogy, the higher-order and abstract features can be gradually extracted. Hence, it is more suitable for complex classification tasks. Fig. 3 shows stacking two AEs into SAE. As shown in Fig. 3, we can see that by unsupervised training, AE1, first-order features $h^{(1)}$ can be extracted from the original features x. Then, $h^{(1)}$ is used as the input of AE2 to unsupervised training AE2, and second-order features $h^{(2)}$ can be extracted from the first-order features. Finally, the input layer X of AE1, the hidden layer $H^{(1)}$ of AE1, and the hidden layer $H^{(2)}$ of AE2 are stacked to form SAE. We can see that SAE only has an encoding process, not a decoding process.

B. Pseudo-siamese stacked autoencoder (PSSAE)

Our proposed PSSAE is shown in Fig. 4. According to the labels, we divide the preprocessed training set into two subsets, the normal dataset with all normal records and the abnormal dataset with all abnormal records. The normal and abnormal datasets are used to train two sets of AEs with the same number and structure layer by layer, respectively, and then two SAEs with pseudo-siamese structures are obtained. The two SAEs have learned the normal and abnormal states, respectively, and can initially extract the deep semantic features of the data. This step is called greedy layer-wise pre-training. Next, we use the complete training set to perform supervised classification training on the pseudo-twin SAE based on the feedforward neural network and the backward propagation algorithm to improve the model's ability and extract deep semantic features. Therefore, it can maximize the difference between normal data and abnormal data. This step is called fine-tuning. Since two SAEs with pseudo-siamese structures are different in sensitivity to positive and negative samples, we set a threshold to perform logical operations on the detection results of the two SAEs to obtain the optimal detection result.

1) Greedy layer-wise pre-training: We divide the train set into normal and abnormal datasets according to the label and use 6 AEs to construct a pseudo-siamese SAE. Three AEs are stacked into SAE_N based on the normal subset, and the other three AEs are stacked into SAE_A. The process of constructing SAE_N and SAE_A is called greedy layer-wise pre-training. Taking the construction of SAE_N as an example, for AE(i)_N, N indicates that the AE is trained based on a normal data set, and i is the sequence number of AE_N. X, $H^{(i)}$ and $\tilde{X}$ are the input layer, hidden layer, and output layer of AE(i)_N, respectively. $f^{(i)} : X \rightarrow H^{(i)}$ is the encoder for AE(i)_N. $g^{(i)} : H^{(i)} \rightarrow \tilde{X}$ is the decoder for AE(i)_N. X is encoded as $H^{(i)}$ by the encoder $f^{(i)}$, and then decoded as $\tilde{X}$ by the decoder $g^{(i)}$. The encoder and the decoder are not linear, so X and $\tilde{X}$ will not be the same. To make the output and the input as similar as possible, the backward propagation algorithm is used to minimize the loss function as in (3). Hence, the hidden layer $H^{(i)}$ of AE(i)_N can be obtained.

As the input layer of AE(i+1)_N, $H^{(i)}$ continues to repeat, the above calculation steps until the training of AE(i=3)_N

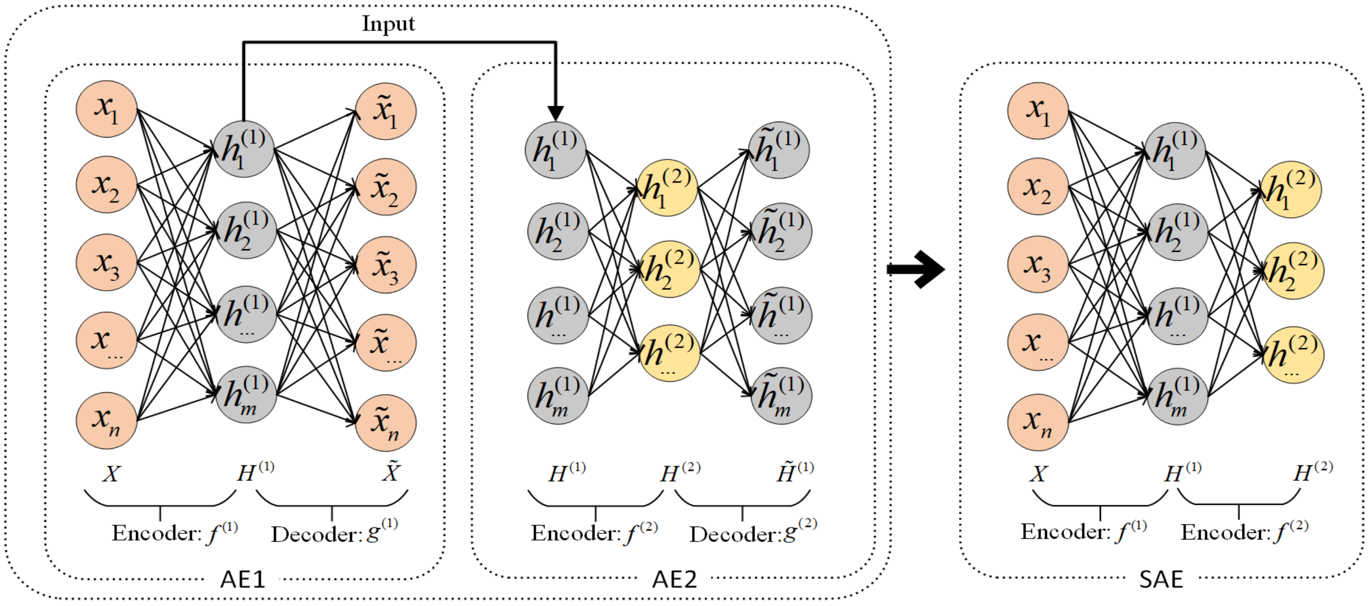

Fig. 3. General structure of SAE

is completed and combines the hidden layers of the three AEs into SAE_N. Similarly, the same process is used for constructing SAE_A. SAEs can extract deep semantic features of data and magnify the difference between positive and negative samples by reconstructing features. SAE is composed of three AEs for two reasons. The first reason is to consider the hardware cost of the fog computing layer and the characteristics of low delay. In addition, the intrusion detection model must be simple, and the consumption of hardware computing power should be low. The second reason is the fitting and gradient explosion occurred after AEs increased in the performance analysis. Through extensive simulations, we found that the effect of SAE composed of three AEs outperforms.

2) Fine-tune: We use the entire training set to perform supervised classification training on the pre-trained SAE_A and SAE_N. The backward propagation algorithm is also used to fine-tune the network hyperparameters and minimize the loss function. Hence, it improves the model's discrimination between normal and abnormal records and maximizes the difference between positive and negative samples. Hence, the loss function is

$$L = \frac{1}{N}\sum_{i}^{N} - \left[y_i \log(\widehat{y}_i) + (1 - y_i)\log(1 - \widehat{y}_i)\right], \tag{4}$$

y_i represents the actual label of the sample, and $\tilde{y}_i$ indicates the probability that the model predicts that the sample is a positive label.

3) Logic operation: Since SAE_N and SAE_A are pre-trained based on the normal and abnormal datasets, respectively. As a result, SAE_N and SAE_A have different sensitivity to positive and negative samples. To prevent the model from shifting to the pre-training data set and combine the advantages of the two pseudo-twin SAEs, we propose to use logical operations to calculate the prediction results of SAE_A and SAE_N on the same record so that the final result is optimal. We stipulate that the attack label is positive and the normal label is negative. The logical operation is as follows.

$$p = \begin{cases} \frac{1}{2}(p_a + p_n), & p_a \geq 0.5 \wedge p_n \geq 0.5 & \text{(5a)} \\ p_n, & p_a < 0.5 \wedge p_n \geq 0.5 & \text{(5b)} \\ p_a, & 0.5 \leq p_a < 0.9 \wedge p_n < 0.5 & \text{(5c)} \\ p_n, & p_a \geq 0.9 \wedge p_n < 0.5 & \text{(5d)} \\ \frac{1}{2}(p_a + p_n), & p_a < 0.5 \wedge p_n < 0.5 & \text{(5e)} \end{cases}$$

For a certain sample S, p_a represents the probability that SAE_A predicts S is positive. p_n means the probability that SAE_N predicts S is positive, and p is the probability after using logical operation to combine p_a and p_n. When two SAEs with pseudo-siamese structures each have the same predicted label for S, i.e., $p_a \geq 0.5$ and $p_n \geq 0.5$, we think the prediction is credible, so $p \geq 0.5$. When two SAEs with pseudo-siamese structures have a contradictory prediction, considering the network's security, S is more likely to be positive. We noticed that when $p_a \geq 0.9$ and $p_n < 0.5$, $p = p_n$, i.e., S is negative, and rest is positive.

IV. Performance Evaluation

A. Performance Indicators

In our experiments, we evaluated the performance of our proposed method using several metrics, including accuracy rate, detection rate, precision rate, F1-Score, false alarm rate, receiver operating curve (ROC), area under ROC (AU-ROC), precision-recall curve (PRC), and area under PRC (AU-PRC). These metrics were calculated using the values obtained from the confusion matrix, which is shown in Table I. In the confusion matrix, TP represents true positive, FN represents

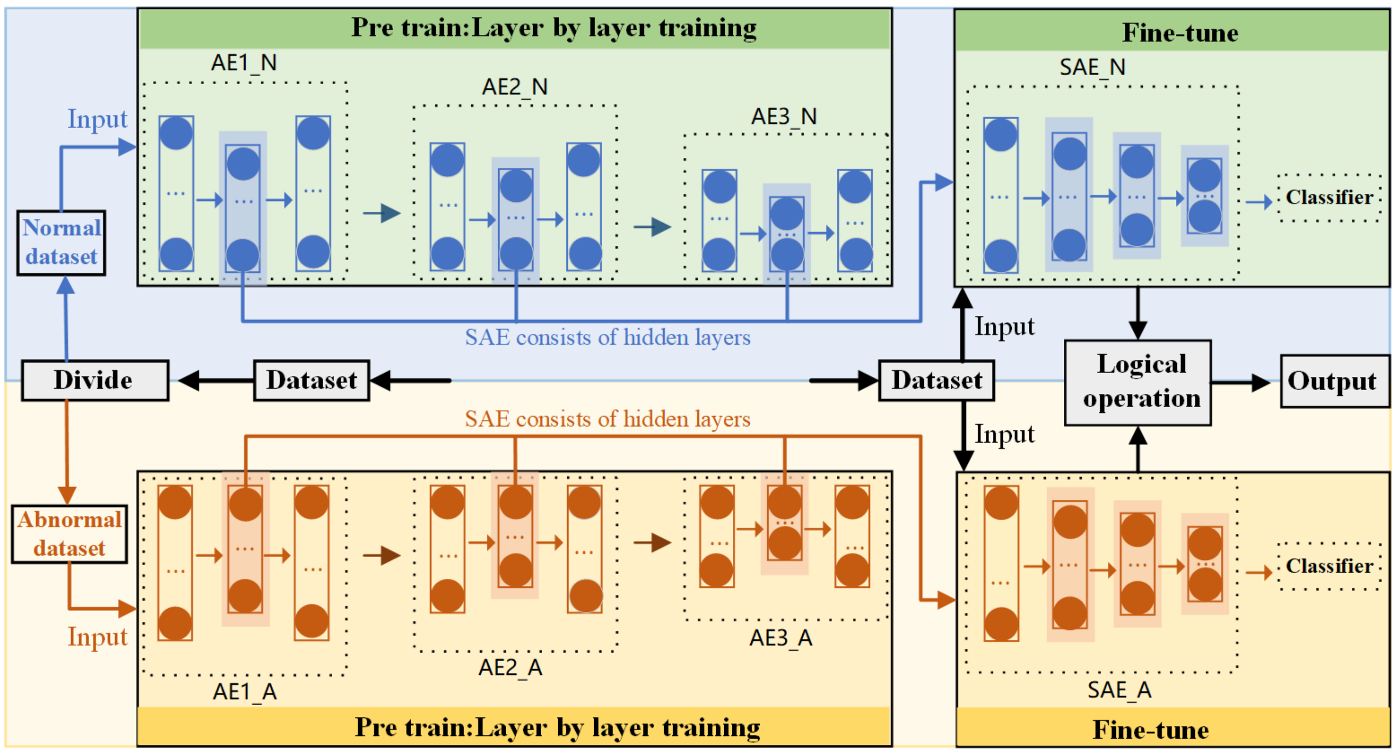

Fig. 4. Structure of proposed PSSAE

false negative, FP represents false positive, and TN represents true negative [30].

TABLE I
CONFUSION MATRIX

	Predicted Attack	Predicted Normal
Actual Attack	TP	FN
Actual Normal	FP	TN

1) Accuracy: Accuracy is a metric that represents the proportion of correctly classified instances to the total number of instances evaluated. In other words, it measures the degree to which a classifier correctly identifies both positive and negative samples, i.e.,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

2) Detect Rate (DR): The detection rate (DR), also referred to as recall, sensitivity, and true positive rate, measures the proportion of relevant instances that are correctly identified by the classifier out of the total number of actual relevant instances. In other words, it calculates the ratio of correctly retrieved instances to the total number of related instances, i.e.,

$$DR = \frac{TP}{TP + FN} \tag{7}$$

3) Precision: Precision, also known as positive predictive value, is a measure of how accurate a model is in predicting true positives. Specifically, precision is calculated by dividing the number of relevant instances correctly retrieved by the total number of retrieved instances. A high precision score indicates that the model is making fewer false positive predictions, which can be an important consideration in applications where false positives are costly or have serious consequences.

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

4) F1-score: The F1 score is a metric used to evaluate the performance of a classification model. It is computed as the harmonic mean of the model's precision and recall, and provides a single value that combines both measures. By integrating precision and recall, the F1 score can provide a more comprehensive evaluation of a model's effectiveness in accurately identifying true positives while minimizing false positives and false negatives.

$$F1 - Score = 2 \times \frac{Precision \times DR}{Precision + DR} \tag{9}$$

5) False Alarm Rate: The false alarm rate, also known as the false positive rate, is a measure of how often an algorithm mistakenly identifies an unrelated instance as being relevant. Specifically, the false alarm rate is calculated by dividing the number of unrelated instances that are incorrectly classified as relevant by the total number of unrelated instances. A lower false alarm rate is generally desirable, as it indicates that the algorithm is less likely to produce irrelevant results.

$$FAR = \frac{FP}{FP + TN} \tag{10}$$

6) ROC and AU-ROC: The ROC plots the false positive rate (FPR) on the X-axis and the true positive rate (TPR) on the Y-axis. Each point on the curve corresponds to a specific FPR and TPR value obtained by varying the classification threshold.

The AU-ROC measures the model's overall ability to correctly classify positive and negative samples. A higher AU-ROC value indicates that the model is better at distinguishing between positive and negative samples. Additionally, since the AU-ROC is not influenced by class imbalance, it can be used to evaluate the performance of a classifier when the number of positive and negative samples is relatively balanced. A smoother and more convex ROC curve indicates better classifier performance.

7) PRC and AU-PRC: The PRC is a graphical representation of the trade-off between precision and recall at different classification thresholds. Recall is plotted on the X-axis, and precision is plotted on the Y-axis. The curve shows how the precision and recall values change as the threshold is varied. A smoother, more convex curve indicates better classifier performance. The AU-PRC is used to evaluate the overall performance of the classifier. A higher AU-PRC value indicates better classification ability. Unlike the ROC curve, the PRC curve is more effective in reflecting the quality of a classifier when the ratio of positive and negative samples is unbalanced, making it a useful tool for evaluating classifier performance in real-world scenarios.

B. Dataset and data preprocessing

1) NSL-KDD: NSL-KDD is an improvement of the KDD Cup99 dataset by Tavallaee *et al.* [31]. KDD Cup99 is based on the DARPA 1998 TCP/IP data set created for the 1999 KDD Cup Challenge. After a comprehensive analysis of the KDD Cup99 data set, the Tavallaee *et al.* [31] found that the data set has many redundant records, fuzzy attack, and partial overlap of the training and testing sets. Therefore, they created the NSL-KDD data set based on KDD Cup99. The authors removed the redundant part, abnormal data, and the overlap between the training and testing sets in the original data. Next, the authors record the times each sample is correctly classified through multiple experiments and divide the data set into five subsets according to the number of correct classifications. Finally, a new data set named NSL-KDD is constructed by selecting records from the subsets inversely proportion to the total number of records in the subsets. There is no overlap between the training and testing sets of NSL-KDD. Some attack types in the test set are not in the training set, so the data set can effectively evaluate the model's generalization ability. NSL-KDD has become one of the most widely used data sets. Although this data set still has some problems, most NIDS still use this data set as a benchmark data set. Therefore, we also use NSL-KDD to evaluate our method to facilitate comparison with other NIDS.

NSL-KDD contains four types of attacks and normal records. The four attacks, i.e., DoS, Probe, U2R, and R2L, are divided into 39 specific attacks. We use KDDTrain+, KDDTest+ and KDDTest+21 in NSL-KDD to perform the experiments. KDDTrain+ is a train set that contains 22 attack records. Both KDDTest+ and KDDTest+21 are test sets with 37 attack records, of which 17 attacks did not appear in KDDTrain+. The difference from KDDTest+ is that KDDTest-21 does not contain records. Therefore, we assume that the records in KDDTest+21 are challenging to detect correctly. The data set composition and distribution are shown in Table II.

TABLE II
DATA SET COMPOSITION AND DISTRIBUTION.

Class	KDDTrain+	KDDTest+	KDDTest-21
Attacks	58630	12833	9698
Normal	67343	9711	2152
Total	125973	22544	11850

2) Data preprocessing: Each record in NSL-KDD has 41 characteristics. After analysis, we found that the value of the feature named *num_outbound_cmds*® in each record is 0. Hence, we deleted this feature from each record. Among the remaining 40 features, "protocol_type", "service", and "flag" are symbol features with a total number of symbols more significant than 2. To make the same distance between each value in the symbol feature, we convert these three symbol features to one-hot code. After the processing, the number of features in each record is converted from 41 to 121. It is also indicated that the value range of different continuous features in the data set is very diverse. To eliminate the influence of dimensions between features and results and make the gradient descent algorithm converge faster, we have performed feature scaling so that each feature is in the same order of magnitude. We use the Z-score feature scaling method to scale the feature value to near 0 without changing the data distribution. The feature scaling is as follows.

$$x' = \frac{x - mean(x)}{\sigma} \tag{11}$$

x is the original feature. $mean(x)$ is the mean value of feature, and σ is the standard deviation.

C. Simulation environment and implementation

1) Simulation environment: The simulation environment is shown in Table III.

TABLE III
EXPERIMENTAL ENVIRONMENT CONFIGURATION

Software	Hardware
Ubuntu 16.04	Intel Xeon E5-2683 v3
Python 3.6	NVIDIA GeForce GTX Titan X
Keras 2.3.1	128GB RAM
Tensorflow 2.0	512GB ROM
Visual Studio Code 1.51.1	

2) Implementation details: The specific structure of the model described in Section III is shown in Table IV, where we set batch_size = 40 for all AEs during layer-by-layer pre-training. In the fine-tuning stage, batch_size = 64 and dropout rate = 0.3. In the two stages, the optimizer is Adam®, and the learning rate is initialized to 0.001. The Batch Normalization between any two layers of all models is also added. The HDF5 file size of our model is 500 KB, and the total number of parameters is about 60000.

TABLE IV
DETAILS OF PROPOSED MODEL

	Module	Layers	Loss function	Activation	Train epoch
Pre-train	AE1_N&AE1_A	121×80×121	MSE	ReLU	100
	AE2_N&AE2_A	80×60×80	MSE	ReLU	100
	AE3_N& AE3_A	60×40×60	MSE	ReLU	100
Fine tune	SAE_N& SAE_A	121×80×60×40×1	Binary cros-sentropy	121 → 80: ReLU 80 → 60: ReLU 60 → 40: ReLU 40 → 1: Sigmoid	500

D. Ablation studies

We conduct an ablation experiment to evaluate our model by comparing our results with several baselines on NSL-KDD Dataset. First, we describe our baseline models. Then, we present our results on the NSL-KDD Dataset.

1) DNN: The model's structure and training details are the same as those in the fine-tuning, as shown in Table IV. For example, KDDTrain+ is the train set. KDDTest+ and KDDTest-21 are the test sets. In the experiment, we referred to this model as DNN3 for short.

2) Single SAE: The SAE comprises a stack of 3 AEs with the same structure as AE1_N, AE2_N, and AE3_N, as shown in Table IV. The configuration of layer-by-layer pre-training and fine-tuning is the same as in Table IV. The model is based on KDDTrain+ for unsupervised layer-by-layer pre-training and supervised fine-tuning. KDDTest+ and KDDTest-21 are test sets. In the experiment, we referred to this model as SAE3 for short.

3) Single SAE pre-trained based on normal data: This model is composed of three stacks of AEs with the same structure as AE1_N, AE2_N, and AE3_N, as depicted in Table IV. The configuration of layer-by-layer pre-training and the fine-tuning is the same as in Table IV. We divide KDDTrain+ into the normal record data set named KDDTrain+_normal and the abnormal record data set named KDDTrain+_abnormal. The pre-train is based on KDDTrain+_normal, and the fine-tuning is based on KDDTrain+. KDDTest+ and KDDTest-21 are test sets. In the experiment, we referred to SAE_N for short.

4) Single SAE pre-trained based on abnormal data: The model consists of three stacks of AEs with the same structure as AE1_A, AE2_A, and AE3_A, as illustrated in Table IV. The configuration of layer-by-layer pre-training and fine-tuning is the same as the configuration in Table IV. It is based on KDDTrain+_abnormal during pre-training and based on KDDTrain+ during fine-tuning. KDDTest+ and KDDTest-21 are test sets. In the experiment, we referred to this model as SAE_A for short.

5) Pseudo-siamese SAE: The model combines the third baseline and the fourth as a pseudo-twin SAE and combines its classification results through logical operations to optimize the final result.

E. Ablation studies on the NSL-KDD

In this section, we present the results of ablation studies on NSL-KDD. Since the attacks contained in NSL-KDD are relatively comprehensive, the train set KDDTrain+ is quite different from the test sets KDDTest+ and KDDTest-21. Therefore, the data sets can comprehensively evaluate the model attack detection and generalization abilities. In Table V, the classification results on KDDTest+ of our proposed architecture are compared with the baselines.

We used the fully connected layer network, B1-DNN3, for IDS experiments. The experiment showed that the DR is low, FAR is high, and the overall accuracy is slightly lower. This is because ordinary neural networks find it difficult to learn the data's deep semantic features, making them insensitive to the normal and abnormal state of the network. Therefore, based on B1, we also use B2-SAE for IDS experiments. The results show that the DR of this model has increased by 1.52%, and the FAR has decreased by 3.46%, indicating that SAE can learn the deep semantic features of the data. To further prove that SAE can extract the deep semantic subspace of original features, we use B3-SAEN and B4-SAEA to conduct experiments. The experimental results show that B3 pre-trained based on normal data has a 0.92% increase in DR and a decrease in FAR by 1.3% compared to B2. Compared to B2, the DR of B4 pre-trained based on abnormal data increased by 13.84%, but the FAR also increased by 5.6%. It shows that B3 and B4 are more sensitive to normal or abnormal records through pre-training of different data types, but it is also possible that the model is shifted to the train set. B5-PSSAE is our final method. It combines the advantages of B3 and B4 while making up for the deficiencies of the two models. As shown in the tables and figures, our proposed PSSAE model significantly outperforms the other baseline models. Compared to B1, the Accuracy of PSSAE increased significantly by about 10%, the DR increased significantly by about 16%, and the FAR decreased significantly by about 3%. Experiments prove that PSSAE has strong detection and generalization abilities.

In Table VI, the classification results on KDDTest-21 of our proposed architecture are compared with the baselines. We mentioned in the previous article that the attack in KDDTest-21 is more challenging to detect. As shown in Table VI, for the KDDTest-21 data set, our method also has good results. Compared with other baseline methods, the accuracy, DR, and F1-score of PSSAE have increased significantly, but the FAR

TABLE V
COMPARISON OF OUR FINAL METHOD WITH OTHER BASELINE METHODS ON THE NSL-KDD(KDDTEST+)

Method	Accuracy(%)	DR(%)	Preston(%)	F1-Score(%)	FAR(%)	AU-ROC	AU-PR
B1-DNN3	80.46	71.42	92.55	80.62	7.60	0.9374	0.9502
B2-SAE3	82.86	73.00	95.89	82.89	4.14	0.9355	0.9479
B3-SAEN	83.67	73.92	96.60	83.75	3.44	0.9224	0.9517
B4-SAEA	88.31	86.84	92.18	89.43	9.74	0.9404	0.9479
B5-PSSAE	90.05	87.06	95.86	91.25	4.96	0.9507	0.9605

TABLE VI
COMPARISON OF OUR FINAL METHOD WITH OTHER BASELINE METHODS ON THE NSL-KDD(KDDTEST-21)

Method	Accuracy(%)	DR(%)	Presion(%)	F1-Score(%)	FAR(%)	AU-ROC	AU-PR
B1-DNN3	58.00	51.52	94.78	66.75	12.78	0.7799	0.9318
B2-SAE3	59.99	54.10	94.76	68.88	13.48	0.8220	0.9403
B3-SAEN	69.12	65.49	95.32	77.64	14.51	0.7844	0.9444
B4-SAEA	78.41	82.75	90.07	86.25	41.12	0.7636	0.9234
B5-PSSAE	82.36	82.88	94.92	88.49	19.98	0.8390	0.9477

has increased compared to B1- B3. The abnormal deviation of the model may cause this. However, because the positive and negative samples of KDDTest-21 are seriously imbalanced, we have observed that the AU-PR is rising, indicating that the classification ability of the model is rising. We mentioned earlier that compared with kddtest +, kddtest-21 does not contain records that the producer can correctly classify through multiple tests. Therefore, the detection of attacks in kddtest-21 is difficult. Typically, its accuracy is lower than kddtest+.

In Fig. 5, the ROC of our method is compared with the other baseline methods based on the KDDTest+ dataset. The AU-ROC can be interpreted as randomly selecting sample A from all positive examples and then selecting sample B from all negative examples. The AU-ROC can be interpreted as follows.

- Randomly selects a sample A from all positive examples,
- Randomly selects a sample B from all negative examples,
- The possibility that the classifier has a higher probability of judging A as a positive example than judging B as a positive example.

At this point, TPR is always greater than FPR, which means that the probability of positive cases is greater than the negative cases. This is because the ROC is not affected by changes in the ratio of positive and negative samples. Hence, it has strong robustness. In Fig. 5, we can observe that the ROC of PSSAE is smoother than the ROC of other baseline methods, and the AU-ROC is larger than the AU-ROC of other baseline methods. It shows that the classification ability of PSSAE is better than other baseline methods.

The PRC of our proposed method on the KDDTest+ dataset is shown in Fig. 6 and compared with baseline methods. Similar to ROC, PRC also measures the effectiveness of the classifier by using TPR (Recall) and AUC. However, PRC focuses on Precision instead of FPR, making it more suitable for imbalanced datasets. Thus, both indicators of the PRC are more focused on positive examples. In Fig. 6, we can observe that the PRC of PSSAE is smoother than the PRC of other baseline methods, and the AU-PRC of PSSAE is larger than

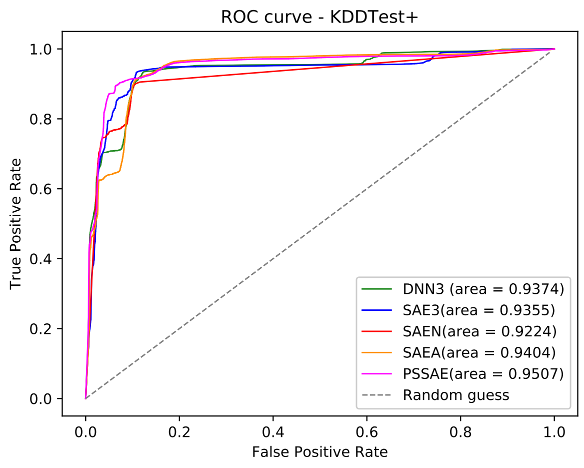

Fig. 5. ROC of our method and other baseline methods on KDDTest+

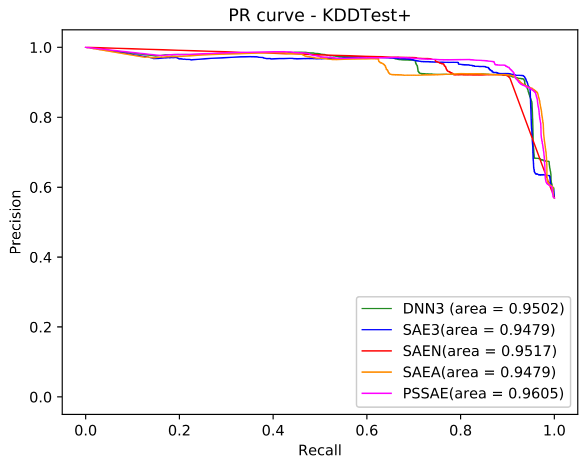

Fig. 6. PRC of our method and other baseline methods on KDDTest+

TABLE VII
COMPARISON WITH STATE-OF-THE-ART METHODS.

Dataset	Method	Accuracy(%)	DR(%)	FAR(%)	F1-Score(%)
KDDTest+	5-layers DNN [15]	78.9	-	-	79.7
	RNN [16]	83.28	72.95	3.06	-
	ICVAE-DNN [21]	85.97	77.43	2.74	86.27
	S-NDAE [25]	85.42	85.42	14.58	87.37
	CBR-CNN [27]	87.28	-	-	-
	Feature learning [32]	84.12	-	-	-
	PIO [33]	88.3	86.6	8.8	88.2
	Proposed PSSAE	90.05	87.06	4.96	91.25
KDDTest-21	RNN [16]	68.55	-	-	-
	ICVAE-DNN [21]	75.43	72.86	12.96	82.92
	CBR-CNN [27]	76.27	-	-	-
	Proposed PSSAE	82.36	82.88	19.98	88.49

that of other models, which indicates that PSSAE's ability to detect attacks is higher than other baseline methods.

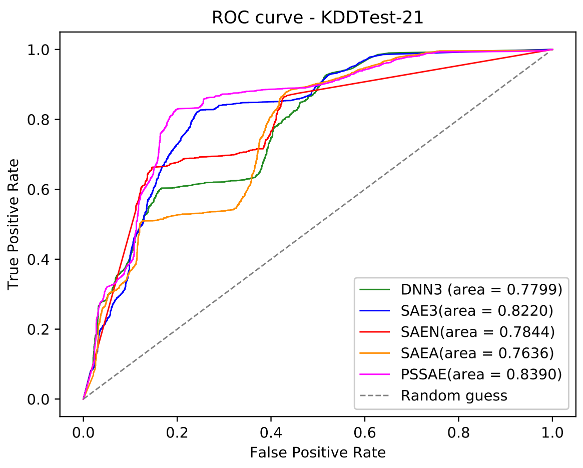

Fig. 7. ROC of our method and other baseline methods on KDDTest-21

As shown in Fig. 7, the ROC of our method is compared with the other baseline methods on the KDDTest-21 dataset. We can see that the ROC of PSSAE is smoother than other baseline methods. The AU-ROC is also larger than the AU-ROC of other baseline methods, indicating that PSSAE has better classification capabilities on KDDTest-21.

In Fig. 8, the PRC of our method is compared with the other baseline methods on the KDDTest-21 dataset. We mentioned in the previous sections that the ratio of positive and negative samples in KDDTest-21 is unbalanced. The PRC is widely considered better than the ROC in the category of imbalance problem because it is mainly concerned with positive examples. From Fig. 8, we can see that the PRC of PSSAE is smoother than that of other baseline methods, and the AU-PRC is also larger than that of other baseline methods. Therefore, PSSAE has a better ability to detect attacks.

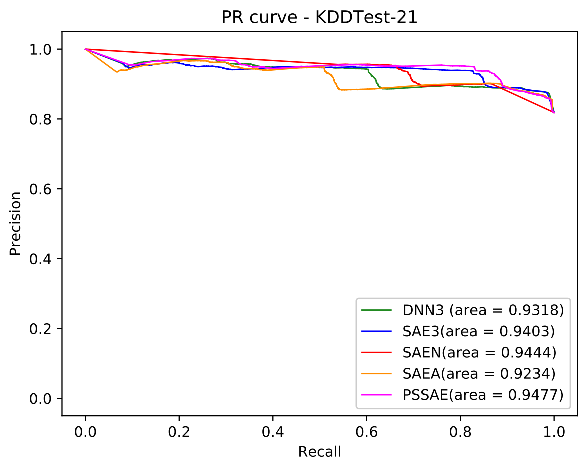

Fig. 8. ROC of final method and other baseline methods on KDDTest-21

F. Overall performance

To further prove the detection performance of PSSAE, we compare PSSAE with some intrusion detection algorithms. As shown in Table VII, on the KDDTest+ and KDDTest-21 datasets, general neural network structures, such as DNN [15] and RNN [16] have low discrimination and sensitivity to normal records and abnormal records. Therefore, the accuracy of the model and DR is low. On the other hand, the accuracy and feature learning [32] of S-NDAE [25], ICVAE-DNN [21], CBR-CNN [27], and the DR based on the AE method are higher than those of DNN [15], and RNN [16]. This shows that SAE can effectively extract deep semantic data features, amplify the difference between positive and negative samples, and make the model more discriminating. Our proposed PSSAE has higher accuracy, DR and F1-Score, and lower FAR than other methods. This shows that our model meets our initial expectations, i.e., a higher degree of discrimination between normal and abnormal records to improve detection and reduce the false alarm rate. Herefore, PSSAE is a better model for intrusion detection performance based on the comparison results.

V. Conclusion

Considering IoT's low latency and limited computing resources, we propose an intrusion detection model that can be deployed in fog computing. We analyzed the existing IDS methods and found that most models have low detection rates and high FAR. To address these issues, we believe that SAEs can extract deep semantic features more effectively than ordinary neural networks, enabling the model to learn the normal and abnormal states of the network, amplify the differences between positive and negative samples, and improve the distinction between normal and abnormal records. Therefore, we proposed a NIDS based on PSSAE and compared its performance with existing NIDS models. The results of the ablation experiment demonstrated that PSSAE achieved higher accuracy, DR, F1-Score, AU-ROC, and AU-PRC, while maintaining a lower FAR compared to other baseline methods. These results validate the effectiveness of PSSAE in extracting deep semantic features, exhibiting a high degree of discrimination between normal and abnormal records, and demonstrating strong generalization ability. Furthermore, the compact size of the PSSAE's HDF5 file (only 500kb) requires low computing power and does not increase hardware costs, making it easily deployable in the fog computing layer. As part of our future work, we plan to enhance the evaluation of our model's performance through extensive testing on a diverse array of datasets. By doing so, we aim to gauge its efficacy and versatility in various intrusion detection scenarios. Additionally, we plan to explore the combination of SAEs with traditional machine learning algorithms in the study of NIDS. This approach aims to further enhance the model's accuracy and reduce the false-positive rate, ultimately improving its overall performance.

Acknowledgement

This work was supported in parts by the China National Key Research & Development Program under Grant 2018YFB0803600, Natural Science Foundation of China under Grant 61801008, the Scientific Research Common Program of Beijing Municipal Commission of Education under Grant KM201910005025, and the Chinese Postdoctoral Science Foundation under Grant 2020M670074.

References

[1] M. Haus, M. Waqas, A. Y. Ding, Y. Li, S. Tarkoma, and J. Ott, "Security and privacy in device-to-device (D2D) communication: A review," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 1054–1079, 2017.

[2] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[3] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[4] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the Internet of things: Security and privacy issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[5] M. Waqas, Y. Niu, M. Ahmed, Y. Li, D. Jin, and Z. Han, "Mobility-aware fog computing in dynamic environments: Understandings and implementation," *IEEE Access*, vol. 7, pp. 38 867–38 879, 2019.

[6] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 255–264.

[7] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Network*, vol. 8, no. 3, pp. 26–41, 1994.

[8] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Citeseer, Tech. Rep., 2000.

[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 303–336, 2013.

[10] M. Waqas, S. Tu, Z. Halim, S. Rehman, G. Abbas, Z. H. Abbas, "The Role of Artificial Intelligence and Machine Learning in Wireless Networks Security: Principle, Practice and Challenges", *Artificial Intelligence Review*, vol 2022, no. 55, pp. 5215-5261, Feb. 2022.

[11] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[12] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1702–1726, 2018.

[13] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[14] S. Tu, M. Waqas, S. U. Rehman, M. Aamir, O. U. Rehman, Z. Jianbiao, and C.-C. Chang, "Security in fog computing: A novel technique to tackle an impersonation attack," *IEEE Access*, vol. 6, pp. 74 993–75 001, 2018.

[15] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[16] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[17] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of IoT networks using artificial neural network intrusion detection system," in *2016 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2016, pp. 1–6.

[18] F. Hosseinpour, P. Vahdani Amoli, J. Plosila, T. Hämäläinen, and H. Tenhunen, "An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach," *International Journal of Digital Content Technology and its Applications*, vol. 10, 2016.

[19] K. Sadaf and J. Sultana, "Intrusion Detection Based on Autoencoder and Isolation Forest in Fog Computing," *IEEE Access*, vol. 8, pp. 167059-167068, Sept. 2020.

[20] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, 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 Transactions on Emerging Topics in Computing*, vol. 7, no. 2, pp. 314–323, 2016.

[21] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors*, vol. 19, no. 11, p. 2528, 2019.

[22] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[23] S. Prabavathy, K. Sundarakantham, and S. M. Shalinie, "Design of cognitive fog computing for intrusion detection in Internet of Things," *Journal of Communications and Networks*, vol. 20, no. 3, pp. 291–298, 2018.

[24] S. Rathore and J. H. Park, "Semi-supervised learning based distributed attack detection framework for IoT," *Applied Soft Computing*, vol. 72, pp. 79–89, 2018.

[25] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[26] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, 2014, pp. 4–11.

[27] N. Chouhan, A. Khan *et al.*, "Network anomaly detection using channel boosted and residual learning based deep convolutional neural network," *Applied Soft Computing*, vol. 83, p. 105612, 2019.

[28] C. A. de Souza, C. B. Westphall, R. B. Machado, J. B. M. Sobral, and G. dos, Santos Vieira, "Hybrid approach to intrusion detection in fog-based IoT environments," *Computer Networks*, vol. 180, p. 107417, 2020.

[29] S. Tu, M. Waqas, Y. Meng, S. U. Rehman, I. Ahmad, A. Koubaa, Z. Halim, M. Hanif, C.-C. Chang, and C. Shi, “Mobile fog computing security: A user-oriented smart attack defense strategy based on DQL,” *Computer Communications*, vol. 160, pp. 790–798, 2020.

[30] F. Huang, M. Waqas, S. Tu, G. Abbas, and Z. H. Abbas, “A revocable and outsourced multi-authority attribute-based encryption scheme in fog computing,” *Computer Networks*, p. 108196, 2021.

[31] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–6.

[32] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, “Autoencoder-based feature learning for cyber security applications,” in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 3854–3861.

[33] H. Alazzam, A. Sharieh, and K. E. Sabri, “A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer,” *Expert systems with applications*, vol. 148, p. 113249, 2020.