

Autoencoder ensembles for network intrusion detection

Chun LONG^{ab}, JianPing XIAO^{ab}, Jinxia WEI^a, Jing ZHAO^{ab}, Wei WAN^{ab}, Guanyao DU^{ab}

^a Computer Network Information Center, CAS (Chinese Academy of Sciences), China

^b University of Chinese Academy of Sciences, China

anquanip@cnic.cn, xiaojianping@cnic.cn, weijinxia@cnic.cn, jingzhao@cnic.cn, wanwei@cnic.cn, duguanyao@cnic.cn

Abstract— Machine learning methods have been widely used in the field of intrusion detection. However, most methods require labeled data sets, and the overhead is very high. Network data is often high-dimensional and has the problem of data imbalance, which makes many techniques unable to adapt to the real network environment. In this paper, we propose a network intrusion detection model based on autoencoder ensembles. This model uses a recursive feature addition algorithm to select the optimal subset of features, which can significantly reduce the training time of classifiers, and improve the performance of intrusion detection system. After feature selection, the feature subset is grouped, and then each group is mapped to an autoencoder. Multiple such autoencoders ensembles form the detection model. Only normal samples are used for training. The detection model is unsupervised, which improves the efficiency of detecting known and unknown attacks. The experimental results show that feature selection can effectively reduce training and detection time. Our model has high detection accuracy and strong adaptability.

Keywords— Autoencoder, Feature Selection, Machine Learning, Network Intrusion Detection System, Recursive Feature Addition

I. INTRODUCTION

The rapid development of the Internet has brought great convenience to people's lives, but people are also facing numerous security threats, and the security situation of the Internet is not optimistic. Therefore, network security is gradually becoming the focus of attention. In recent years, new types of attacks are emerging one after another, and network attack brings an outbreak of vicious abnormal activities. The development of network communication techniques is increasingly exerting rigorous and scalable requirements on network security awareness. Detecting network anomaly is of great significance and value. Understanding the difference between normal network events and anomalous activities is essential for monitoring network security, thereby suggesting the behavior of potential security risks. In this way, detecting and reasoning of network anomaly can ensure damage mitigation and future prevention of new network attack or network failure. In the era of big data, how to identify network attacks, especially unknown attacks, and maintain high detection rate is an important technical issue that must be solved.

Network Intrusion Detection systems (NIDS) can effectively detect network attacks. With the development of

machine learning, there are many methods used to detect attacks, such as SVM, random forest [1], decision tree [2], etc. NIDS based on traditional machine learning methods have become increasingly mature and have high detection accuracy for known attacks. However, most current solutions often fail to accurately detect unknown attacks and have a high false alarm rate. In addition, the existing intrusion detection technologies require a lot of time overhead and training data, which are difficult to be applied in the actual dynamic network environment.

After the deep learning theory [3] was proposed, its related technologies have been rapidly developed in the application field of machine learning. In recent years, deep learning has promoted the vigorous development of artificial intelligence technologies and related industries. Deep learning methods have good application prospects in the field of intrusion detection. These technologies have good performance and excellent adaptability, and can overcome some of the problems faced by machine learning methods [4].

In intrusion detection, the main problem of machine learning is the computational complexity with the increase of the dimension and size of the dataset. Feature selection can reduce the dimension and size by selecting a subset of features. With the continuous changes of network environment, it is necessary to improve the adaptability of intrusion detection systems. The detection model needs to have self-learning ability. In this paper, we propose an intrusion detection system based on autoencoder and feature selection. The model uses recursive features addition algorithm to select features, reduces the dimensionality of data, and obtains the best features. Feature subsets can improve the performance of the detector. Then we use the autoencoder to reconstruct the new representation of the input data in the detection module, calculate the root mean square error (RMSE) of the original data and reconstructed data, and classify network traffic based on RMSE.

The main contributions of this paper are as follows:

(1) We propose a feature selection method, which is recursive feature addition based on Random Forest (RF). It can select the optimal subset of features and find interdependent features, which improves the results of neural networks and traditional machine learning algorithms. We use One-Class SVM as the detection model, and compare the

detection accuracy before and after feature selection on the NSL-KDD.

(2) Autoencoders are used for anomaly detection, and only normal samples are used for training. Anomaly detection module has an ensemble of simple autoencoders, which are trained to reconstruct network traffic data. The autoencoder has self-learning ability. Ensemble structure reduces the resources required for model training compared to deep neural networks. The detection model is unsupervised, which can reduce the dependence on labeled data and adapt to the environment of unbalanced data.

(3) We build a lightweight intrusion detection model based on feature selection and autoencoders. Feature selection can reduce the complexity of the problem, simplify the structure of the detection model. The model uses autoencoders in the anomaly detection, which improves the detection accuracy. It can effectively detect unknown attacks.

The rest of this paper is organized as follows. Section 2 discusses on some related work. Section 3 outlines the background technologies. Section 4 describes the network intrusion detection method proposed in this paper. Section 5 presents datasets and experimental results. Finally, in Section 6 we conclude the paper with future direction.

II. RELATED WORK

In previous research work, many models based on machine learning methods have been constructed and applied to network intrusion detection. These methods include Hidden Markov Models (HMM), K-Nearest Neighbor, Support Vector Machine (SVM) and Artificial Neural Network.

Considering the high-dimensional character of network traffic, data consists of many features. These features may contain many irrelevant features which can affect the classifier performance, or can lead to overfitting. Feature selection is applied to the field of anomaly detection. Xiao et al. [5] applied HMM to network intrusion detection based on abnormal traffic, took the traffic characteristics extracted based on principal component analysis (PCA) as the input value of HMM, and judged the type of traffic based on the probability of output. Aljawarneh et al. [6] adopted information gain as an evaluation index to select 8 features from 41 features of the NSL-KDD dataset as the final features subset. The defect of this method is that it overestimates the importance of some features with more different values so that to select redundant and irrelevant features. The classifier dependent method compensates for this defect, which adopts machine learning algorithms to evaluate the importance of features [7]. Chaouki et al. [8] utilized logistic regression as the evaluation model, and genetic algorithm as the search strategy to select the most relevant subset with the smallest size. However, these feature selection methods are still difficult to solve the overfitting problem, leading to poor results, i.e., not better than without feature selection.

In order to improve the ability to detect unknown attacks, many machine learning and deep learning algorithms are used for intrusion detection. Chapaneri et al. [9] used Gaussian mixture model method to learn the statistical characteristics of

each traffic category, and applied interquartile-based adaptive threshold technology to identify outliers. The evaluation results on the CICIDS2017 dataset showed that this method can effectively detect unknown attacks. Deep learning technologies can learn complex concepts, as well as the concepts from the domain of network communication. The autoencoder is an artificial neural network, which can be used for anomaly detection. The authors propose to use autoencoders to extract features from datasets in order to improve the detection of network threats. However, the autoencoders themselves were not used for anomaly detection. Finally, the authors use machine learning classifiers to detect the attacks [10, 11]. Therefore, their solution requires an expert to label instances, while our solution is unsupervised.

Li et al. [12] proposed an autoencoder intrusion detection system based on the random forest algorithm, which used shallow autoencoder neural network to reduce the computational complexity, shorten the detection time, and effectively improve the detection accuracy. Yisroel Mirsky et al. [13] proposed a online, unsupervised and efficient network intrusion detection system, which is Kitsune based on ANN. Kitsune has an ensemble of small neural networks (autoencoders), which are trained to mimic (reconstruct) network traffic patterns, and whose performance incrementally improves overtime. Kitsune used unsupervised learning to detect attacks online. Kitsune's core algorithm is KitNET, which is an unsupervised machine learning algorithm. In Kitsune, the features are mapped to the visible neurons of an ensemble of autoencoders, which is KitNET, Kitsune's core algorithm. Then, each autoencoder attempts to reconstruct the instance's features, and computes the reconstruction error in terms of root mean squared errors (RMSE). Finally, the RMSEs are forwarded to an output autoencoder, which acts as a non-linear voting mechanism for the ensemble. The evaluation showed that Kitsune could detect a variety of attacks with performance comparable to offline anomaly detectors. In this paper, we use autoencoder as the anomaly detection module, which is inspired by Kitsune.

III. BACKGROUND

A. AUTOENCODERS

The autoencoder is the core part of the proposed model. Figure 1 illustrates the basic architecture of an autoencoder.

For a given input X , an autoencoder is an artificial neural network which is trained to reconstruct its input [13], that is, to minimize the reconstruction error, as shown in the following formula:

$$\begin{aligned} \phi: X &\rightarrow F \\ \psi: F &\rightarrow X \\ \phi, \psi &= \underset{\phi, \psi}{\operatorname{argmin}} \|X - (\phi \circ \psi)X\|^2 \end{aligned} \quad (1)$$

Here, ϕ and ψ represent the encoding and decoding functions, respectively. are shown in Table 1.

The encoding process can be expressed by the following formula:

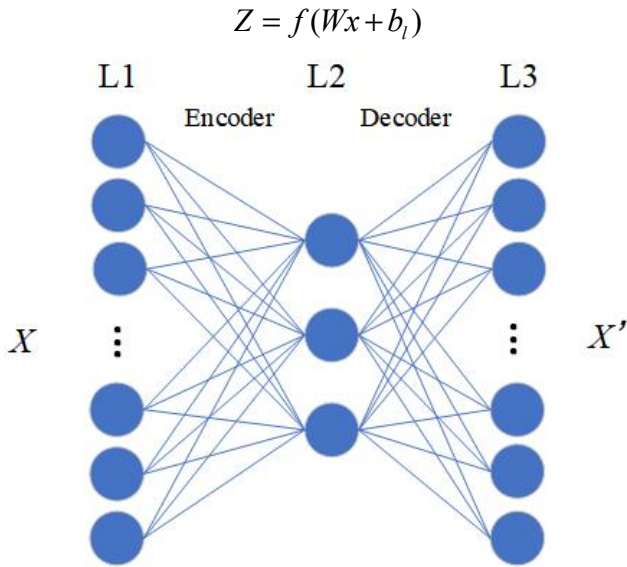


Figure 1. The Architecture of the Autoencoder

The output of the decoding layer is expressed as follows:

$$O = g(Z) = g(W^T y + b_z) \quad (3)$$

f and g are what's known as the neuron's activation functions. We use the sigmoid function, which is defined as:

$$f(z) = \frac{1}{1 + \exp(z)} \quad (4)$$

The reconstruction value of an input X is:

$$x' = g(f(x)) \quad (5)$$

The reconstruction error of the input X can be computed by Root Mean Square Error (RMSE). The RMSE between x_i and x_i' is defined in Equation 6.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (x_i - x_i')^2}{n}} \quad (6)$$

When an autoencoder is used for anomaly detection, we use normal samples to train the autoencoder. By optimizing the parameters, the RMSE value is close to zero. When an abnormal sample is input, its RMSE will be greater than zero, so the RMSE can be used to judge whether the network flow is abnormal.

B. One-Class SVM

The SVM is a classic supervised machine learning method. The SVM is a kind of binary classification model. Its basic model is defined as a linear classifier with the largest interval in the feature space. Its learning strategy is to maximize the interval, which can be transformed into a convex quadratic programming problem that best separates the classes [14]. Unlike the supervised SVM, the One-Class SVM is unsupervised. It is a novelty detection method [15]. The goal of One-Class SVM is to fit a hyperplane as a boundary, which can best include all training data and exclude any other data points. One-Class SVM is considered to be one of the most

- (2) mature outlier-based machine learning technologies. After One-Class SVM being trained, a spherical boundary will be formed [16].

IV. PROPOSED MODEL

In this section, we will introduce the proposed model. The overall framework of the model presented in Figure 2 consists of three steps.

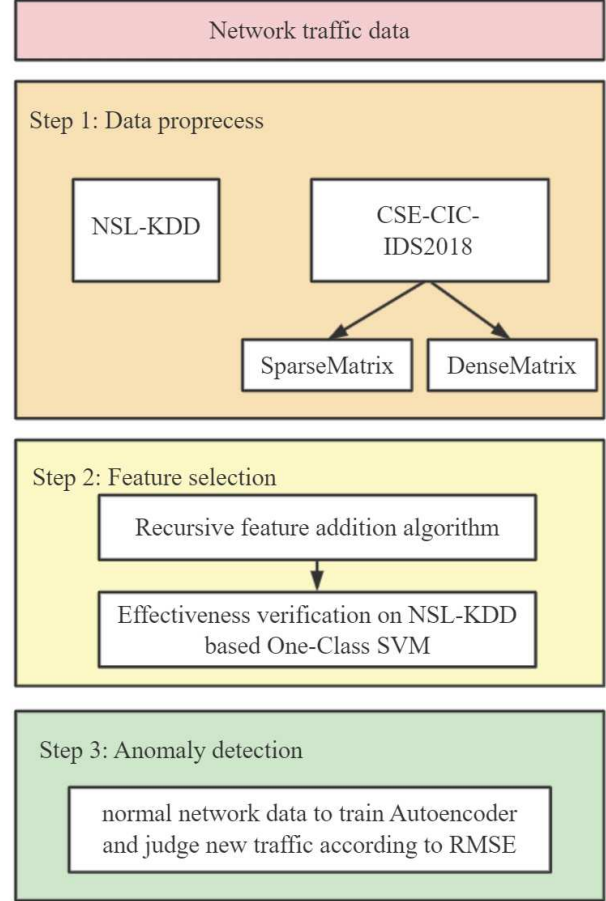


Figure 2. The overall framework of the proposed anomaly detection model

STEP1: Data preprocessing. The two benchmark data sets NSL-KDD and CSE-CIC-IDS2018 used in the experiment need to be preprocessed, and the specific processing process is described in section V-B.

STEP2: Feature selection. Recursive feature addition algorithm is applied to select the optimal subset of features.

STEP3: Anomaly detection. The anomaly detection part is composed of autoencoder and k-means. After feature selection, AP clustering algorithm [17-19] is used to group feature subsets. Each features group corresponds to an autoencoder. These autoencoders ensembles compose the core of the anomaly detection module. The normal network data is used to train the detection model. After the model is deployed, the autoencoder processes the new incoming traffic and calculates the RMSE. The RMSE value is used to determine whether the network traffic is normal. K-means algorithm clusters the results according to the RMSE value.

A. Recursive feature addition algorithm

Recursive feature addition algorithm is a kind of the embedded feature selection algorithm [20], which combines with classification model and evaluation indexes. Recursive feature addition is a greedy search strategy for feature selection. In this method, an empty feature set is initialized and new features are added to it step by step. If one feature can improve the evaluation effect, it will be selected. Otherwise, it will not be selected into the feature subset. In this paper, the random forest is applied as a classifier, and roc_auc is as the desired metric to optimize the performance of the classifier. When the roc_auc index does not increase, it indicates that the selected feature subset has the optimal classification effect, and this feature subset contains informative features that is the final selected feature subset. Algorithm 1 shows the detailed process of recursive feature addition algorithm.

ALGORITHM 1: Recursive Feature Addition(RFA)

Input: Data set D, feature set F

Output: The selected features set SF

```

1: Set an empty selected features set F'
2: Set an roc_auc empty set R
3: Train a random forest model using all the features F
4: Rank the features according to their importance, derived
   from the model, and get the ranked feature set F1
5: Train a random forest model with the most important
   feature f1 and determine its performance roc_auc1
6: R.add(roc_auc1), base_roc_auc= roc_auc1
7: F'.add(f1)
8: For i=2:N do
9:   Add a feature in F1 and train a new random forest
   model
10:  Determine its performance roc_auci
11:  if roc_auci> base_roc_auc
12:    F'.add(fi)
13:    base_roc_auc= roc_auci
14: End for

```

B. One-Class SVM model

We use One-Class SVM to verify the effectiveness of the recursive feature addition algorithm. One-Class SVM is trained using the benign instances. In order to train the One-Class SVM, a 'v' value was specified. The v default value is 0.5, which includes 50% of the training sample in the hyperplane. In this experiment, multiple v values were chosen (0.15, 0.20, 0.25).

Algorithm 2 shows the process of training the One-Class SVM model. In our experiment, 75% of the normal samples are used to fit the One-Class SVM model. A One-Class SVM trained model outputs a binary value, '0' and '1'. The output

represents whether an instance belongs to the class to which the SVM is fit. Hence, each attack is evaluated based on how many instances are predicted with a '0' SVM output.

ALGORITHM 2: One-Class SVM Model

Input: normal_data, nu_value

Output: Trained SVM

```

1: training=75% i ∈ normal_data in train_data
2: testing=other 20% normal_data and all test_data
3: oneclasssvm One-Class SVM (nu_value,'rbf')
4: oneclasssvm.fit(training)
5: return oneclasssvm

```

V. EVALUATION & RESULTS

In this section, we describe the evaluation experiment process and results. The experimental environment is Intel (R) Core (TM) i7-7700 CPU on the Windows 10 platform, the main frequency is 3.60GHz, and the memory is 16 GB RAM.

A. The description of the benchmark datasets

We select NSL-KDD [21] and CSE-CIC-IDS-2018 [22, 23] as the evaluation datasets, and NSL-KDD is only used to evaluate the effect of the recursive feature addition algorithm. The NSL-KDD data set is an improved version of the KDD'99 [24] data set to solve the problems existing in the KDD'99 data set. In NSL-KDD, redundant records in the training and test sets in KDD'99 have been deleted. There are 39 types of attacks, of which 22 are in the training data, and the other 17 are in the test data. The data set includes three parts: KDDTrain+, KDDTest+ and KDDTest-21. Generally, KDDTrain+ is used as the train set, and KDDTest+ and KDDTest-21 are used as the test set. This article only uses KDDTrain+ and KDDTest+. Table 1 shows the distribution of normal records and four types of attack records in each part of the data.

TABLE 1. Different classifications in the NSL-KDD dataset

	Total	Normal	DoS	Probe	R2L	U2R
KDDTrain+	125973	67343	45927	11656	995	52
KDDTest+	22544	9711	7458	2421	2754	200

Each record in the NSL-KDD dataset is a connection vector containing 41-dimensional features and a label. These features can be divided into three categories: basic features, content features, and traffic features, as shown in Table 2. The attacks can be divided into four categories: Probe (Probing attacks), R2L (Root to Local attacks), U2R (User to Root attacks) and DoS (Denial of Service attacks) attacks.

CSE-CIC-IDS2018 is a cooperative project between the Communications Security Agency (CSE) and the Canadian Cyber Security Institute (CIC). The dataset includes seven different attack scenarios: brute force, Heartbleed, botnet, DoS, DDoS, Web attacks and penetration of the network from

within. In the feature extraction part, a bidirectional flow is generated through CICFlowMeter, and 83 statistical features such as duration, number of data packets, number of bytes, and data packet length are calculated in the forward and reverse directions.

TABLE 2. FEATURES OF NSL-KDD DATASET

No.	Features	No.	Features
1	duration	22	is_guest_login
2	protocol_type	23	count
3	service	24	srv_count
4	flag	25	error_rate
5	src_bytes	26	srv_error_rate
6	dst_bytes	27	error_rate
7	land	28	srv_error_rate
8	wrong_fragment	29	same_srv_rate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_error_rate
18	num_shells	39	dst_host_srv_error_rate
19	num_access_files	40	dst_host_error_rate
20	num_outbound_cmds	41	dst_host_srv_error_rate
21	is_host_login		

B. Data preprocessing

There are 38 numerical features and 3 symbolic features in the NSL-KDD. In order to ensure the uniformity of the overall data format, the symbolic features need to be converted into numerical representations, including 'protocol_type', 'service' and 'flag'. By analyzing the dataset, we can see that there are 3 types of 'protocol_type', 70 types of 'service', and 11 types of 'flag'. Therefore, 41-dimensional features are converted to 122-dimensional features through one-hot encoding. In addition, it is necessary to standardize the features. For some features, the range of values varies greatly, which will have an impact on the training of the classification model. In this paper, MinMax standardization is used to map the value of each feature to [0,1]. The standardization formula is as follows:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (7)$$

Some features in the CSE-CIC-IDS2018 have NaN or Infinite values, so these data need to be replaced. In our experiment, NaN and Infinity are replaced with 0. In the CSE-CIC-IDS2018 dataset, there are many records with feature value of 0. We divide the original dataset into sparse matrix and dense matrix according to the values of feature 'TotLen Fwd Pkts' and 'TotLen Bwd Pkts'. Since most data sets extracted from network traffic can be divided into two categories, namely sparse matrix and dense matrix. A sparse matrix means that the number of elements with a value of 0 is

much greater than the number of non-zero elements, and the distribution of non-zero elements is irregular. On the contrary, dense matrix means that non-zero elements occupy most of the matrix. Table 3 shows the number of samples, normal samples and attack samples after dividing each sub-dataset into a sparse matrix (dense matrix).

After dividing the data set into a sparse matrix or a dense matrix, in the training phase of the model, we use 85% of the normal samples for training. The remaining 15% normal samples and all abnormal samples are used for testing.

TABLE 3. SPARSEMATRIX AND DENSEMATRIX ON CSE-CIC-IDS2018

Datasets		SparseMatrix	DenseMatrix
Brute Force - Web	Total	317701	730670
	Normal	317591	730418
	Abnormal	110	252
Brute Force - XSS	Total	317664	730496
	Normal	317591	730418
	Abnormal	73	78
Botnet	Total	338900	709675
	Normal	194892	567492
	Abnormal	144008	142183
SQL Injection	Total	388624	659438
	Normal	388598	659411
	Abnormal	26	27
DoS attacks- Hulk	Total	440737	467947
	Normal	4373	442399
	Abnormal	436364	25548
DoS attacks- Slow HTTPTest	Total	144263	442399
	Normal	4373	442399
	Abnormal	139890	0
DDoS attack- HOIC	Total	524458	522387
	Normal	2196	358637
	Abnormal	522262	163750
DDoS attack- LOIC-UDP	Total	275	362288
	Normal	275	360558
	Abnormal	0	1730
FTP-BruteForce	Total	412410	448576
	Normal	219050	448576
	Abnormal	193360	0
SSH-Bruteforce	Total	312821	542394
	Normal	219050	448576
	Abnormal	93771	93818
Infiltration	Total	171660	441411
	Normal	152467	391733
	Abnormal	19193	49678

C. Model Evaluation Criteria

In order to evaluate the effect of the model, the confusion matrix is introduced in this paper, which is shown in Table 4.

TABLE 4. CONFUSION MATRIX

Predicted Class \ Actual Class	attack	normal
attack	TP	FN
normal	FP	TN

The following evaluation indicators are used in this paper:

- (1) Accuracy: the proportion of samples with correct predictions to all samples:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

- (2) Recall: the proportion of correctly detected attack samples to all attack samples:

$$Recall = \frac{TP}{TP + FN} \quad (9)$$

- (3) AUC: The ROC curve is based on a confusion matrix. The abscissa of the curve is the False Positive Rate (FPR), and the ordinate is the True Positive Rate (TPR), which can reflect the sensitivity of the model when selecting different thresholds. (Sensitivity, FPR) and its accuracy (Specificity, TPR) trend. When the distribution of positive and negative samples changes, the shape of the ROC curve can basically remain unchanged, so this evaluation index can reduce the interference caused by different test sets, and more objectively measure the performance of the model itself. The area under the ROC curve is the value of the AUC index, and the maximum value of AUC is 1.

D. Experimental results and analysis

In this section, we describe the evaluation experiment process and results. Firstly, the impact and validity of the recursive feature addition algorithm is proved. In the NSL-KDD dataset, feature selection is performed by this algorithm. One-Class SVM is a simple anomaly detection algorithm, and it is convenient to train and effective. We use One-Class SVM model to compare the detection accuracy of each category before and after feature selection. Then, we verify the effectiveness of the proposed method based on the recursive feature addition algorithm and autoencoder ensembles, and compare with other methods to further illustrate the advantage of our method. In this part of the experiment, the recall rate, AUC, and detection time are compared.

1) Analysis of the impact of the recursive feature addition algorithm on the performance of the detection model

This experiment evaluates the effect of the recursive feature addition algorithm. The KDDTrain+ is selected for the experiment during feature selection, and it is divided into a train set and a test set, and ROC_AUC is used as the evaluation index. The goal is to make the performance of the classifier using the selected optimal feature subset equal to the

performance of the classifier using all the features. The importance ranking of 41-dimensional features is shown in Figure 3. In the process of feature selection, the addition of each dimension feature will cause the change of model detection performance.

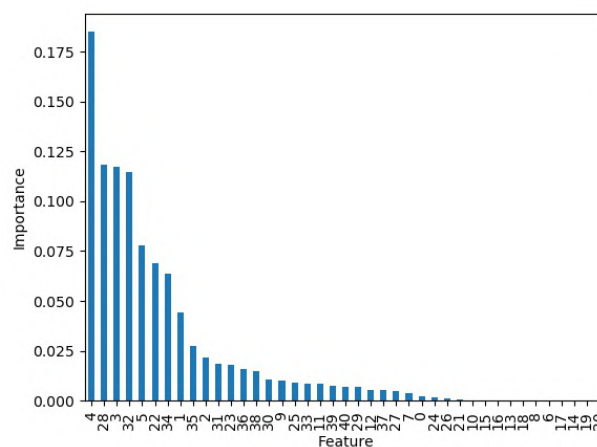


Figure 3. NSL-KDD feature importance ranking(Subscribe 0 represents the first dimension feature, and so on)

The result of the recursive feature addition algorithm is shown in Table 5. After feature selection, the number of features in the optimal subset of features is 8. Then the One-Class SVM is trained according to the selected features. The results of One-Class SVM tested on NSL-KDD with all features and the optimal subset of features are summarized in Table 6.

TABLE 5. FEATURES SELECTED BY RECURSIVE FEATURE ADDITION (RFA) ALGORITHM

No.	Features	No.	Features
5	src_bytes	33	dst_host_srv_count
6	dst_bytes	34	dst_host_same_srv_rate
4	flag	23	count
29	same_srv_rate	3	service

It can be seen that the detection accuracy of DoS and Probe does not change significantly, and the detection performance of R2L and U2R is greatly improved after feature selection. Specifically, on the KDDTrain+ dataset, when the v value is 0.15 and 0.2, the detection accuracy of R2L and U2R are significantly improved after feature selection; on the KDDTest+ dataset, the detection accuracy of R2L is significantly improved under three different v values after feature selection. When the v value is 0.15, the detection accuracy of U2R has also increased by more than 5%. In addition, the change of v value has little effect on the detection accuracy for some categories, which is mainly due to the limited number of attacks and changes on the NSL-KDD dataset.

TABLE 6. COMPARISON OF DETECTION RESULTS BEFORE AND AFTER FEATURE SELECTION ON NSL-KDD

Class	Model	Accuracy		
v		0.15	0.2	0.25
KDDTrain+ .csv				
Normal (Validation)	1	85.14%	80.54%	75.41%
	2	86.15%	80.35%	75.41%
DoS	1	98.14%	98.14%	98.19%
	2	98.13%	98.13%	98.13%
Probe	1	98.77%	99.52%	99.81%
	2	99.48%	99.48%	99.99%
R2L	1	52.26%	81.71%	95.58%
	2	69.75%	93.07%	95.58%
U2R	1	80.77%	82.69%	100.00%
	2	88.46%	100.00%	100.00%
KDDTest+ csv				
Normal	1	86.02%	84.72%	83.51%
	2	86.67%	85.26%	81.72%
DoS	1	94.67%	94.69%	94.79%
	2	94.67%	94.67%	94.67%
Probe	1	99.91%	100.00%	100.00%
	2	99.82%	99.91%	100.00%
R2L	1	82.22%	90.31%	95.59%
	2	90.86%	95.50%	98.77%
U2R	1	78.38%	83.78%	86.49%
	2	83.78%	83.78%	86.49%

(* 1: One-Class SVM 2: RFA-One-Class SVM)

Next, we compare the training and detection time of the One-Class SVM model before and after feature selection, as shown in Table 7. Therefore, according to the detection results of the NSL-KDD dataset, the recursive feature addition algorithm is effective.

TABLE 7. DETECTION TIME BEFORE AND AFTER FEATURE SELECTION

model\v	Detection Time(s)		
	0.15	0.2	0.25
1	332.68	447.96	552.21
2	236.77	331.14	401.37

(*1: One-Class SVM 2: RFA-One-Class SVM)

2) Performance evaluation of the proposed model on CSE-CIC-IDS2018

By comparing the sparse matrix and the dense matrix, it is confirmed that dividing the data set can reduce training time and improve detection accuracy. Since the autoencoder model

in this paper is inspired by KitNet [13], we also compare our model with KitNet to prove its advantage.

We compared the experimental results on the sparse and dense matrix. Table 8 is the results of the confusion matrix. The results show that the use of sparse matrix and dense matrix for detection can significantly improve the accuracy and efficiency of detection. For most of the sub-datasets such as Botnet, DoS attack-SlowHTTPTest, FTP-BruteForce, DDoS attack-HOIC, SSH-Bruteforce, we can get relatively good detection results.

After analyzing the experimental results, we can get the following conclusions: when the features of the abnormal sample data are quite different from the normal data, the detection result is good. When the features of the abnormal sample data are almost the same as the normal sample data, the detection result will be relatively poor.

We hope our model can achieve a high recall rate. Table 9 shows the performance of our model and KitNet in terms of recall rate. On the same dataset, the experimental results show that the overall recall rate of our model is better than KitNet. It is noted that the distribution of normal data and abnormal data in some sub-datasets is unbalanced. Overall, it is clear that our model can maintain good recall rate and adapt well to the environment with unbalanced data distribution.

We calculated the AUC value of the experiment on different sub-datasets. Figure 4-7 shows the ROC curve of some experiments. Our model can distinguish abnormal samples in most experiments.

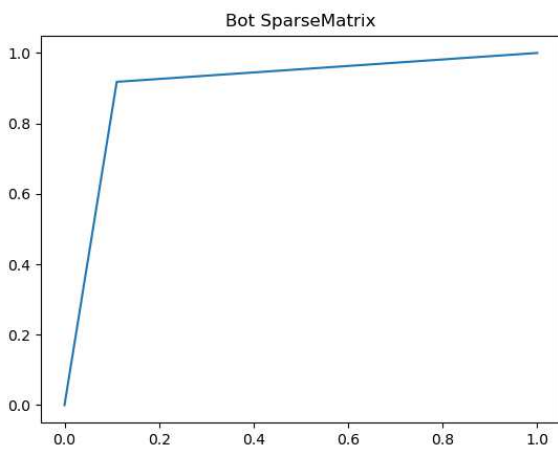
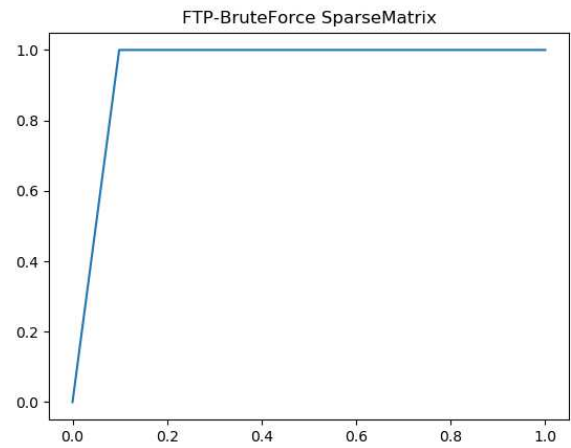
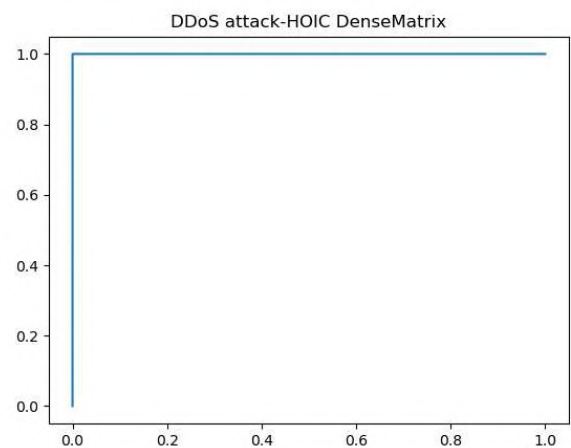
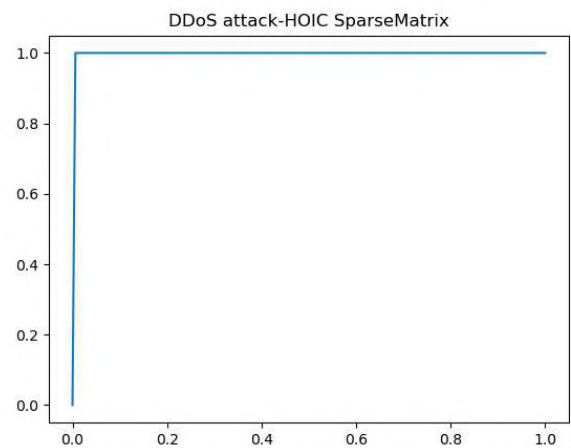
Table 8. THE CONFUSION MATRIX OF THE PROPOSED MODEL

	SparseMatrix		Predicted		DenseMatrix		Predicted	
Brute Force -Web			0	1			0	1
	Actual	0	39619	8019	Actual	0	109552	11
		1	48	62		1	192	60
Brute Force -XSS			0	1			0	1
	Actual	0	45783	1855	Actual	0	102484	7079
		1	59	14		1	6	72
Botnet			0	1			0	1
	Actual	0	26004	3230	Actual	0	84538	585
		1	11815	132193		1	4236	137947
SQL Injection			0	1			0	1
	Actual	0	58169	120	Actual	0	97996	916
		1	25	1		1	27	0
DoS attacks-Hulk			0	1			0	1
	Actual	0	663	0	Actual	0	66353	0
		1	0	436364		1	15693	9855
DoS attacks-SlowHTTPTest			0	1			0	1
	Actual	0	656	0	Actual	0	61275	5078
		1	0	139890		1	0	0
DDoS attack-HOIC			0	1			0	1
	Actual	0	328	2	Actual	0	53790	5
		1	0	522262		1	0	163750
DDoS attack-LOIC-UDP			0	1			0	1
	Actual	0	29	12	Actual	0	54082	2
		1	0	0		1	0	1730
FTP-BruteForce			0	1			0	1
	Actual	0	29614	3243	Actual	0	53618	13668
		1	0	193360		1	0	0
SSH-Bruteforce			0	1			0	1
	Actual	0	32791	66	Actual	0	54082	2
		1	0	93771		1	0	0
Infiltration			0	1			0	1
	Actual	0	16190	6680	Actual	0	176	58584
		1	11335	7858		1	194	49484

(0:Normal Sample; 1:Abnormal Sample)

Table 9. THE RECALL RATE OF OUR MODEL AND KITNET

Dataset	Our Model		KitNet
	SparseMatrix	DenseMatrix	
Brute Force -Web	56.36%	23.81%	35.36%
Brute Force -XSS	19.18%	92.31%	3.31%
Botnet	91.80%	97.02%	0.05%
SQL Injection	3.85%	/	3.77%
DoS attacks-Hulk	100%	38.57%	0.27%
DoS attacks-SlowHTTPTest	100%	/	100%
DDoS attack-HOIC	100%	100%	100%
DDoS attack-LOIC-UDP	/	100%	98.44%
FTP-BruteForce	100%	/	100%
SSH-Bruteforce	100%	100%	100%
Infiltration	40.94%	99.61%	/

**Figure 4.** The RoC of Botnet SparseMatrix**Figure 5.** The RoC of FTP-BruteForce SparseMatrix**Figure 6.** The RoC of DDoS attack-HOIC DenseMatrix**Figure 7.** The RoC of DDoS attack-HOIC SparseMatrix

We compared the detection time by our model and KitNet on the same dataset, as shown in Table 10. After splitting the original dataset into the sparse matrix and the dense matrix, the amount of data is significantly reduced, so the detection time will be less. For the dense matrix of DoS attacks-SlowHTTPTest and FTP-BruteForce, only normal samples are provided. Therefore, feature selection cannot be performed on them. Compared with the dataset for feature selection, we can see that feature selection can significantly reduce the detection time.

From Table 10, we can see that using KitNet for detection takes a long time, because it does not divide the datasets, and uses all the features for training. However, the recursive feature addition algorithm of our model also consumes a lot of time, which still needs to be improved.

TABLE 10. DETECTION TIME OF OUR MODEL AND KITNET

Dataset	Our Model		KitNet(s)
	SparseMatrix(s)	DenseMatrix(s)	
Brute Force -Web	41.75	118.6	3420.34
Brute Force -XSS	39.76	119.09	3799.07
Botnet	47.54	84.54	3495.81
SQL Injection	47.04	156.31	2246.02
DoS attacks-Hulk	42.72	75.22	2495.78
DoS attacks-SlowHTTPTest	17.63	998.89	1902.74
DDoS attack-HOIC	22.73	55.09	1412.46
DDoS attack-LOIC-UDP	0.93	43.33	1497.70
FTP-BruteForce	38.65	1009.53	2271.77
SSH-Bruteforce	33.38	120.28	2277.37
Infiltration	20.41	92.85	1739.20

VI. CONCLUSION AND FUTURE WORK

Our model RFA-AE is a lightweight intrusion detection model based on feature selection and autoencoder. It consists of the following three parts: data preprocessing, feature selection and anomaly detection. In the feature selection module, we use a recursive feature addition algorithm, which can select the optimal subset of features and relieve the overfitting problem. We use One-Class SVM on the NSL-KDD to verify its effectiveness. In the anomaly detection module, we use the autoencoder, which can adapt to the problem of sample imbalance, so it is suitable for deployment in real network scenarios. Experimental results show that our model can speed up training and detection time, and accurately detect attacks. In the future research, we attempt to use real network traffic data to further evaluate our model.

VII. ACKNOWLEDGMENT

The authors would like to thank the editorial board and reviewers. This work was supported by the "14th Five-Year Plan" Special Preliminary Construction Project of Internet

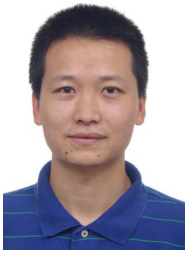
Information of Chinese Academy of Sciences (WX145XQ10, WX145XQ11).

REFERENCES

- [1] I. Ahmad, M. Basher, M. J. Iqbal, and A. Rahim, "Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection," *IEEE Access*, vol. 6, pp. 33789-33795, 2018.
- [2] B. Dong, and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection." pp. 581-585.
- [3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-44, May 28, 2015.
- [4] 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)*, pp. 1-6, 2016.
- [5] L. Xiao, and H. Wang, "Network Intrusion Detection Based on Hidden Markov Model and Conditional Entropy," in *International Conference on Smart City and Informatization*, pp. 509-519, 2019.
- [6] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152-160, Mar. 2018.
- [7] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system," *Expert Systems with Applications*, vol. 67, pp. 296-303, Jan. 2017.
- [8] C. Khammassi, and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Computers & Security*, vol. 70, pp. 255-277, Sep. 2017.
- [9] R. Chapaneri, and S. Shah, "Multi-level Gaussian mixture modeling for detection of malicious network traffic," *The Journal of Supercomputing*, vol. 77, no. 5, pp. 4618-4638, 2021.
- [10] 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)*, pp. 3854-3861, 2017.
- [11] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," *Eai Endorsed Transactions on Security and Safety*, vol. 3, no. 9, pp. e2, 2016.
- [12] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Computers & Security*, vol. 95, pp. 101851, 2020.
- [13] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.
- [14] C. Cortes, and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, Sep. 1995.
- [15] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," *NIPS*, vol. 12, pp. 582-588, 1999.
- [16] D. M. J. Tax, and R. P. W. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45-66, Jan. 2004.
- [17] W. Hang, F.-I. Chung, and S. Wang, "Transfer affinity propagation-based clustering," *Information Sciences*, vol. 348, pp. 337-356, 2016.
- [18] G. Gan, and M. K.-P. Ng, "Subspace clustering using affinity propagation," *Pattern Recognition*, vol. 48, no. 4, pp. 1455-1464, 2015.
- [19] B. J. Frey, and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972-976, 2007.
- [20] T. Hamed, R. Dara, and S. C. Kremer, "Network intrusion detection system based on recursive feature addition and bigram technique," *Computers & Security*, vol. 73, pp. 137-155, Mar. 2018.
- [21] M. Tavallaei, 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*, pp. 1-6, 2009.
- [22] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *ICISSP: Proceedings of the 4th International*

Conference on Information Systems Security and Privacy, vol. 1, pp. 108-116, 2018.

- [23] University of New Brunswick. CSE-CIC-IDS2018 on AWS dataset. [Online]. Available: <https://www.unb.ca/cic/datasets/ids-2018.html>.
- [24] (1999) KDDCup. KDD Cup Dataset. [Online]. Available: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.



Chun LONG is currently a full engineer in Computer Network Information Center, Chinese Academy of Sciences, China. He received Ph.D. degree in Computer Software and Theory from University of Chinese Academy of Sciences. He developed Compound Attack Prediction Method based on the Attack Graph and Multi-source Security Event Fusion method based on EA-DS Evidence Theory. His research interests include Information Security, Cyber Security, Cyber Risks and Web Vulnerabilities.



Jianping XIAO is currently a postgraduate student in Computer Network Information Center, Chinese Academy of Sciences, China. He received B.S. degree in Network Engineering from Beijing University of Posts and Telecommunications, and he is currently studying for M.S. degree in University of Chinese Academy of Sciences, majoring in Cyberspace Security. His research interest is Cyber Security.



Jinxia WEI is currently a full engineer in Computer Network Information Center, Chinese Academy of Sciences, China. She received Ph.D. degree in Cryptography from Beijing University of Posts and Telecommunications. Her current research interests include network security situation awareness, security big data analysis, and cloud computing security.



Jing ZHAO is currently a full engineer in Computer Network Information Center, Chinese Academy of Sciences, China. She received M.S. degree from Beijing University of Posts and Telecommunications, and she is currently studying for Ph.D. in University of Chinese Academy of Sciences, majoring in Computer Software and Theory. Her current research interests include network security situation awareness, security big data analysis, and cloud computing security.



Wei WAN is currently a full engineer in Computer Network Information Center, Chinese Academy of Sciences, China. He received Ph.D. degree in Computer Software and Theory from University of Chinese Academy of Sciences. He has presided over and participated in many national projects. Moreover, he developed Investigation of state division in botnet detection model, Botnet detecting method based on activity similarity and so on. His research interests include Information Security, Cyber Risks and Web Vulnerabilities.



Guanyao DU is currently a full engineer in Computer Network Information Center, Chinese Academy of Sciences, China. She received Ph.D. degree from Beijing Jiaotong University. Her current research interests include network security situation awareness, security big data analysis, and cloud computing security.