Dynamic Network Anomaly Detection System by Using Deep Learning Techniques

Peng Lin[1,2], Kejiang Ye[1(✉)], and Cheng-Zhong Xu[3]

[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
{peng.lin,kj.ye}@siat.ac.cn
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Faculty of Science and Technology, University of Macau, Taipa, Macao, Special Administrative Region of China
czxu@um.edu.mo

Abstract. The Internet and computer networks are currently suffering from serious security threats. Those threats often keep changing and will evolve to new unknown variants. In order to maintain the security of network, we design and implement a dynamic network anomaly detection system using deep learning methods. We use Long Short Term Memory (LSTM) to build a deep neural network model and add an Attention Mechanism (AM) to enhance the performance of the model. The SMOTE algorithm and an improved loss function are used to handle the class-imbalance problem in the CSE-CIC-IDS2018 dataset. The experimental results show that the classification accuracy of our model reaches 96.2%, which is higher than other machine learning algorithms. In addition, the class-imbalance problem is alleviated to a certain extent, making our method have great practicality.

Keywords: Network anomaly detection · Deep learning · Attention · SMOTE

1 Introduction

Nowadays, due to the rapid development of Internet and cloud computing techniques, the number of global networked devices has become very large [1]. However, under such a large-scale network infrastructure, faults or attacks occur very frequently which bring a very bad experience to users and cause serious economic losses. In order to prevent network attacks, people often use firewalls as the first line of defense to ensure that the network works properly and use Intrusion Detection System (IDS) as the second line of defense to further improve system security.

IDS is a kind of network security device that monitors network traffics in real time and will alert or take proactive measures when an anomaly is detected. Abnormal network traffics refer to the network traffics that adversely affect the network, which deviate greatly from normal network traffics in pattern. The cause of abnormal network traffics can be the unreasonable network operation or external network attacks [2].

There are mainly three steps in IDSs. Firstly, IDS needs to track and collect the network flow data. Secondly, IDS needs to clean the raw data and convert them to the

D. Da Silva et al. (Eds.): CLOUD 2019, LNCS 11513, pp. 161–176, 2019.
https://doi.org/10.1007/978-3-030-23502-4_12

input-format needed for the next step. Finally, a classification engine is needed to identify the network traffics as normal or abnormal.

Among the above three steps, the most important one is the classification operation, which determines the detection performance of an IDS. The classification engine can be implemented by signature-based methods and anomaly-based methods. The former method implements the classification by comparing the network traffics with the signatures of the abnormal traffics that have been already defined, while the latter one generally learns the characteristics of abnormal traffics through some machine learning (ML) algorithms and then uses the trained ML model to make a judgment. Although the signature-based methods can achieve high accuracy and have a fast detection speed, it is powerless for identifying unknown network traffics. In contrast, the anomaly-based approaches are more flexible as well as having better generalization, and they perform well even in the face of the classification tasks on unknown network traffics [3]. Nowadays, with new network attacks emerging, an excellent network anomaly detection system should have the ability to discover unknown anomalies. The systems discussed above are refer to as *dynamic network anomaly detection systems*, which are usually implement by anomaly-based approaches.

In recent years, with the improvement of computing power and the outbreak of data volume, deep neural networks (or deep learning) have attracted people's attention again. The strong nonlinear fitting ability of deep learning techniques make them exhibit excellent performance in many fields [4]. Compared to traditional machine learning algorithms, deep learning techniques have a faster processing speed when dealing with big data and can learn the deep hidden representation of features with higher accuracy.

Some researchers have used deep learning approaches to detect network anomaly. Aksu et al. [5] compared the classification results of SVM and deep learning, and the results show that the deep learning method performed better. But they only studied the classification research on PortScan and normal network traffic. In the actual network environment, the network traffic's types are much more than two, which increases the difficulty of detection. Zhu et al. [6] used Convolutional Neural Network (CNN) to study the network traffics classification issue, but the accuracy obtained by the experiment is not high. And there are also some researches [7, 8] that use the outdated datasets such as KDD CUP99 [9] to do the experiments, which can no longer reflect the characteristics of today's network traffics.

To overcome the above challenges, this paper proposes a deep learning method to implement the dynamic IDS. The main contributions are as follows:

- We study the issue of multi-classification, which is more challenging and practical.
- An up to date dataset CSE-CIC-IDS2018 [10] is used in our experiment, which can reflect the characteristics of the latest network traffics.
- We use LSTM to establish our model, which has good performance in processing time-correlated sequences such as network traffics.
- We use the SMOTE, an over-sampling algorithm to get more samples and then optimize the loss function, which make some progress on the class-imbalance issue.
- Experimental results show that our method achieves an overall accuracy of 96.2%, which is higher than other machine learning algorithms used in the experiment.

The rest of this paper is organized as follows. We introduce the proposed methods in Sect. 2 and give the implementation details in Sect. 3. In Sect. 4, we conduct the network traffic classification experiments and analyze the experimental results. Section 5 introduces the related work and Sect. 6 concludes the whole paper.

2 The Method

2.1 Long Short Term Memory (LSTM)

LSTM is a special recurrent neural network structure, which is proposed to solve the problem of long-term dependence [11]. It adds the forget gate, input gate, and output gate to the standard Recurrent Neural Network (RNN). The forget gate lets the neural network forget the useless information, the input gate adds new content to the neural network and the output gate determines the final output of current node. Figure 1 shows the structure of a single LSTM cell.

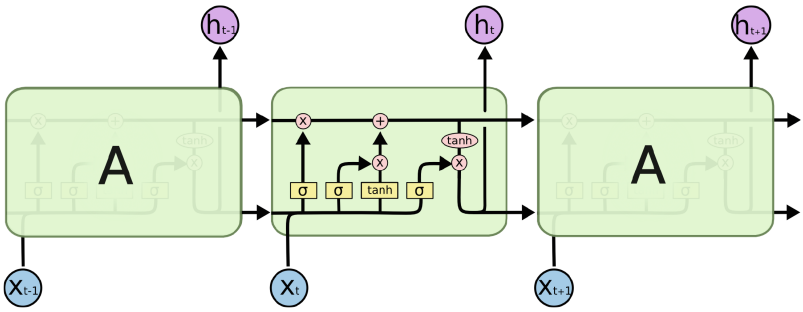

Fig. 1. Structure diagram of a single LSTM cell

The process of forward propagation of LSTM can be described by the following equation, where $h^{(t)}$ and $C^{(t)}$ are the two hidden states of the LSTM model, σ represents the sigmoid function, i, f and o are respectively the input gate, forget gate and output gate, W are weight matrices for different peephole connections.

Update the output of the forget gate:

$$f^{(t)} = \sigma\left(W_f * \left[h_{(t-1)}, x_t\right] + b_f\right)$$

Update the output of the input gate:

$$i^{(t)} = \sigma\left(W_i * \left[h_{(t-1)}, x_t\right] + b_i\right)$$

$$\widetilde{C}(t) = tanh\left(W_c * \left[h_{(t-1)}, x_t\right] + b_C\right)$$

Update cell's state:

$$C^{(t)} = f^{(t)} * C^{(t-1)} + i^{(t)} * \tilde{C}(t)$$

Update the output of the output gate:

$$o^{(t)} = \sigma\left(W_o * \left[h_{(t-1)}, x_t\right] + b_o\right)$$

$$h^{(t)} = o^{(t)} * tanh\left(C^{(t)}\right)$$

Classification engine is the most important part of the system, and we used LSTM to implement it. LSTM can not only learn the current network traffics, but also can remember previous network traffics' characteristics. When it comes to the network attacks, generally the attackers will carry out a series of continuous operations. So the current network traffic is normal or not strongly related to the previous network traffics.

2.2 Attention Mechanism

The Attention Mechanism (AM) [13] in deep learning is actually imitating the attention mechanism of the human brain. When reading a piece of text, we usually focus on some keywords so that we can quickly summarize the main content of the text. If deep neural network techniques have the ability to focus on different aspects of information, it is beneficial for the extraction and representation of important information. It is the inspiration for introducing attention mechanisms in neural networks. The core idea of AM is to extract and represent the part of the information that is most relevant to the target.

Attention mechanism can be seen as an automatic weighting scheme. In the scenario of anomaly detection, the role of AM is to calculate the impacts of each network traffic on the last network traffic. We can use the following formula to calculate the attention value of each flow:

$$\alpha_t = \frac{exp\left(u_t^T * u_w\right)}{\sum_t exp\left(u_t^T * u_w\right)}$$

Where u_w is the weight matrix and u_t represents the implicit representation of the LSTM hidden state (h_t) at time t, and u_t can be calculated by the following formula:

$$u_t = tanh(W_w h_t + b_w)$$

where W_w is the weight matrix and b_w is the bias. After obtaining the attention probability distribution value at each moment, the feature vector v that contains the network traffic information is calculated as follows:

$$v = \sum_t \alpha_t * h_t$$

Finally, we can use the *softmax* function to get the predicted label y:

$$y = softmax(W_v * v + b_v)$$

2.3 Smote

We have used the CICIDS2017 dataset to conduct an experiment on network traffics classification [14], but there was a serious class-imbalance problem in their experimental results. In their results, four of the eight categories have the precisions rate below 40%, and even three of them are close to 0. This is because in the IDS2017 dataset, the amounts of some categories are very small, the neural network cannot learn the characteristics of these categories well. In this paper, we experimented with the CSE-CIC-IDS2018 dataset and used the SMOTE [12] over-sampling algorithm to synthesize new samples for the small size classes. The principle of the SMOTE oversampling algorithm is as follows:

Let the size of a small size class be T, considering a sample i of the class, and its feature vector is $x_i, i \in \{1, \ldots, T\}$:

a. Find k neighbors of the sample x_i from all T samples of this small size class (For example, using Euclidean Distance), and denoted it as $x_{i(near)}, near \in \{1, \ldots, k\}$;
b. A sample $x_{i(nn)}$ is randomly selected from the k neighbors, and a random number ζ_1 between 0 and 1 is generated to synthesize a new sample x_{i1} as the following Equation: $x_{i1} = x_i + \zeta_1 \cdot \left(x_{i(nn)} - x_i\right)$;
c. Repeat step b. N times to synthesize N new samples: x_{inew}, $new \in \{1, \ldots, N\}$

2.4 Loss Function

In this paper, Adam gradient descent method is used to further optimize the model. In order to improve the efficiency, mini-batch algorithm is used for training. By calculating the gradient of the loss function, Adam can update the parameters of the model step by step, and finally reach convergence. The loss function we use is the cross-entropy function, which is defined as follows: $L = -\sum_i y_i' * log(y_i)$, where y_i' is the

actual label of the sample while y_i is the label predicted by the deep neural network. We make some changes to the function, which enhances the accuracy of the classification on small size classes:

$$L^{'} = -\sum_{i} w_i * y_i^{'} * log(y_i)$$

We set different weights to each class. The weights of large size classes are setting smaller and the weights of small size classes are setting larger. If the samples of small size classes are classified incorrectly, the loss value of the system will increase rapidly so that the updating parameters of the neural network will be closer to the direction of small size classes. Note that the weights of small size classes cannot be the very large values, otherwise the system will tend to classify most of the samples into these classes, resulting in a very low overall accuracy.

3 Implementation

3.1 Dataset

We used CSE-CIC-IDS2018 as the experimental dataset, which was created by The Canadian Institute for Cyber-security (CIC) and Communications Security Establishment (CSE). The dataset includes seven different attack scenarios such as DDoS attack, Botnet attack, Infiltration attack, BruteForce attack, DoS attack, Web attack, and Heartleech (a type of DoS attack). By using the tool CICFlowMeter-V3, we can extract more than 80 features of the raw network data and save them as several csv files. Some of the features are listed in Table 1.

Table 1. Some features in CSE-CIC-IDS2018 dataset

Feature name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	Total size of packet in forward direction
fw_pkt_l_max	Maximum size of packet in forward direction
fw_pkt_l_min	Minimum size of packet in forward direction
fw_pkt_l_avg	Average size of packet in forward direction
fw_pkt_l_std	Standard deviation size of packet in forward direction

We compared the differences in sample sizes between CICIDS2017 and CSE-CIC-IDS2018, and the results are shown in Table 2. It can be seen that the sample sizes of the CSE-CIC-IDS2018 dataset have been comprehensively improved compared with the CICIDS2017 dataset, especially in the Botnet attack and Infiltration attack, which have increased by 143 times and 4497 times respectively. But the amount of samples for Web Attack is very small, only 928 samples are provided.

Table 2. Differences in samples of two datasets

	Normal	DDoS	PortScan	BOT	Inf	Web attack	BF	DoS
CICIDS-2017	1743179	128027	158930	1966	36	2180	13835	252661
CSE-CIC-IDS2018	6112151	687742	–	286191	161934	928	380949	654301

3.2 Pre-processing

In the original dataset, there are some features have little impacts on whether the traffic is abnormal or not, such as timestamps and IP addresses. The timestamp records the time when the anomalous network traffic occurred, which are of little help in training our neural network, so we removed this feature. In addition, as an anomaly detection system, we hope it can classify the network traffics according to their behavioral characteristics, and should not be biased against the IP address, so we also deleted the column of feature.

After completing the above works, we divide the dataset into training set, test set and validation set, which are 90%, 9% and 1% of the original data respectively. The training set is used for training, the validation set is used for rapid evaluation of the model during training, and the test set is used for final evaluation of the model. In addition, we noticed that there are too many normal network traffic samples in the dataset, which can easily affect the classification preference of the model. So we under-sampled the normal traffics and only took 2 million records randomly. Furthermore, we over-sampled the samples of Web attack and Infiltration attack by using SMOTE algorithm. Oversampling is only implemented in training set. After dividing the dataset, we shuffle the training set to ensure the loss value change smoothly during training.

3.3 Metrics

Three metrics are used to evaluate the performance of our experiment: Accuracy, Precision and Recall rate. Accuracy represents the proportion of correctly classified samples, and its formula is as follows:

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

In all samples classified as Category-A, the proportion of those really belong to Category-A is defined as precision. Generally, the higher the Precision, the lower the False Alarm Rate (FAR) of the system will be.

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

Recall rate represents the proportion of all samples in Category-A that are eventually classified as A. Recall rate reflects the system's ability to detect anomalies. The higher it is, the more anomalous traffics are detected correctly.

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

TP, FP, TN, FN represent True Positive, False Positive, True Negative and False Negative respectively.

3.4 Experimental Setup

Tensorflow [15] that runs on the Ubuntu 16.04 OS is used to build the deep neural network architecture. The server's CPU is Intel Xeon E5-2650 v4 with 48 cores and 128 GB of memory. In addition, 4 Nvidia Titan XP GPUs are used as the accelerator. The architecture of the deep neural network used in the experiment is shown in Fig. 2. We use two LSTM layers and three full connected dense layers to build our model, and add the attention mechanism to the LSTM.

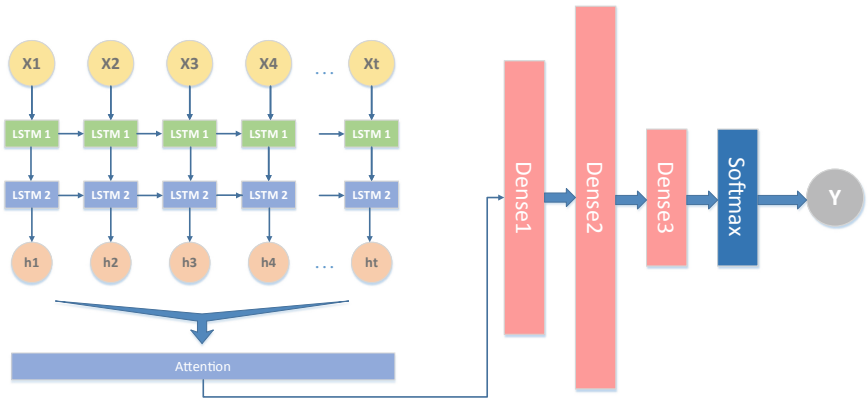

Fig. 2. Architecture of our model

4 Experiment

4.1 Performance

In this experiment, the hyperparameters that we need to optimize are: LSTM hidden nodes, flow length, batch size, learning rate and activation function. We carried out a lot of experiments, and found a set of optimal hyperparameters, which are as follows (Table 3).

Table 3. Best hyperparameters of DNN

Name	Value
LSTM hidden nodes	256
Flow length	10
Batch size	128
Learning rate	0.00005
Activation function	Relu

Under this hyperparameters setting, the best performance of the deep neural network is show in Table 4.

Table 4. Best performance of DNN

Class	Precision	Recall
0	0.93	0.99
1	1.00	1.00
2	0.99	1.00
3	0.93	0.17
4	0.95	0.98
5	0.98	0.99
6	0.30	0.98

And the confusion matrix of results is shown in Table 5.

Table 5. Confusion matrix

	Normal	DDoS	BOT	Inf	BF	DoS	Web attack
Normal	190500 *99.07%*	15	234	111	370	348	207
DDos	196	67721 *99.71%*	0	0	0	1	0
BOT	26	0	28238 *99.91%*	0	0	0	0
Inf	11425	0	0	2563 *17.12%*	904	76	0
BF	361	0	0	4	36895 *98.10%*	352	0
Dos	393	1	0	0	377	63842 *98.80%*	0
Web Attack	2	0	0	0	0	0	90 *97.82%*

As can be seen from the above results, the overall performance of the classifier is very good. The average Precision and Recall rate are as high as 96%, reaching a practical level. Six of the seven categories have a Precision that more than 93%, and similarly there are six categories with a recall rate of over 98%.

In terms of Precision, the values for all categories have reached more than 93% except the web attack samples. Precision of web attack is only 27%, but the reason is obvious. Because the sample size of web attack is very small, the TP (True Positive) is limited to a very small value, therefore, even if a small amount of network traffics that don't belong to web attack category are classified into this category, the denominator of Precision's formula will increase rapidly, making it difficult to achieve a high Precision.

In terms of Recall rate, the classifier also performs well. There are six of the seven categories with a recall rate over 98%, indicating that most of the network traffics are correctly classified to the category that they belong. In other words, the system can detect most of the abnormal traffics. In addition, the classification performance of web attack greatly exceeded our expectations. After using the SMOTE algorithm and improved loss function, the Recall rate of web attack samples actually reached 98%, while it was 0 before the optimization. But we also found that the Recall rate of Infiltration samples which are processed by the same method with web attack was 17%, and it was only 6% higher than before. For this phenomenon, we guess that the pattern between web attack network traffics are similar. The new samples synthesized by SMOTE algorithm can well reflect the characteristics of this kind of traffics, so the neural network can fit them well. However, Infiltration is relatively rich in diversity. The new data synthesized by SMOTE algorithm cannot reflect the characteristic distribution of Infiltration well, so the effect is not greatly improved. In addition, we also find that most of the Infiltration samples are classified into the normal categories, which indicates that they are similar in patterns, thus it is difficult for neural networks to distinguish them.

Figure 3 shows the changes of Infiltration and Web attack before and after optimization on Recall rate.

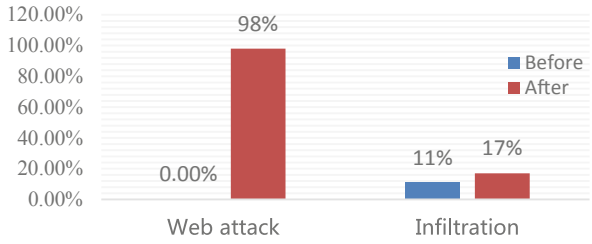

Fig. 3. Changes of recall rate before and after optimization

4.2 Influence of Hyperparameters

In the above experiments, we find that different hyperparameter settings have a great impact on the results of the model. Now let's explore the impacts of different hyperparameter settings, including LSTM hidden nodes, learning rate, flow length, and mini-batch size. We introduce F1-Score to evaluate the whole system, defined as follows:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Hidden nodes of LSTM. We changed the values of LSTM hidden nodes from 64 to 128, 256, 384 and 512 respectively and fixed the other hyperparameters. Each experiment was done three times and then calculated the average value. Accuracy, Precision and Recall rate were recorded when the model converged. The experimental results are shown in Fig. 4. It can be seen that when LSTM hidden nodes are too few, the neural network cannot learn the network traffics' features very well, so its performance is not very good. With the increase of hidden nodes, the classification performance of model goes up. But when it reaches 256, the number of hidden nodes have little influence on the classification effect. Continuing to increase hidden nodes will not only prolong the training time, but also bring the risk of over-fitting. Thus, the best hidden nodes is 256.

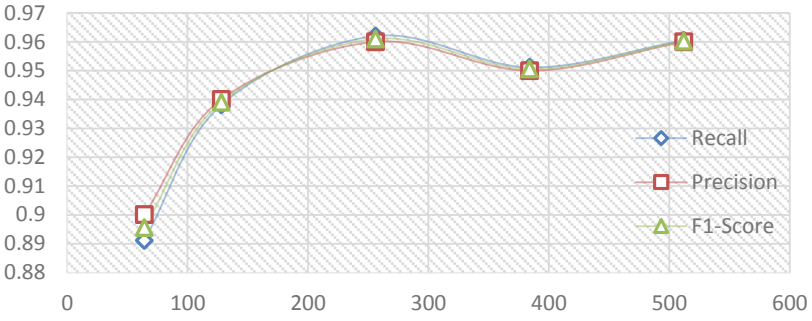

Fig. 4. Influence of LSTM hidden nodes

Learning Rate. Learning rate determines the speed of gradient descent so it plays a vital role in the training. We fix the values of other hyperparameters and then change the learning rates with logarithmic scales to 0.1, 0.01, 0.001, 0.0001 and 0.00001, respectively. We find that the best interval of learning is [0.0001, 0.00001], so we changed the learning rates again to 0.00001, 0.00003, 0.00005, 0.00007 and 0.00009 and repeat the experiments. The results are shown in Fig. 5. It can be seen that when the learning rate is 0.0005, the performance of the model is optimal.

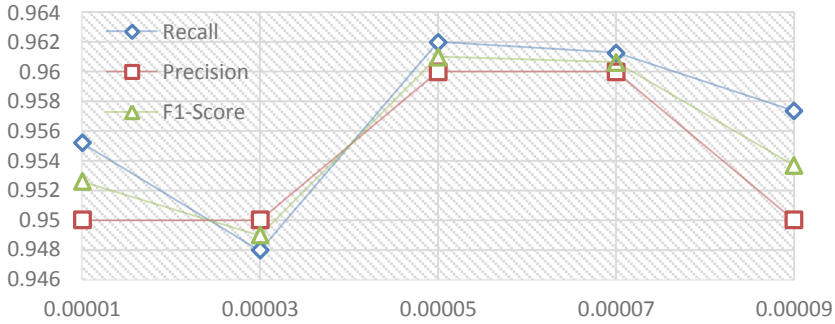

Fig. 5. Influence of learning rate

Flow Length. It is also important to choose the appropriate size of network traffics to train. Let the flow length be n, change the values of n to 6, 8, 10, 12, 14 respectively, and then do the experiments separately. The experimental results show the growth of n has no significant impact on the performance of the system, as shown in Fig. 6. When n is greater than 10, the classification performance can hardly be improved, so we set the flow's length to 10.

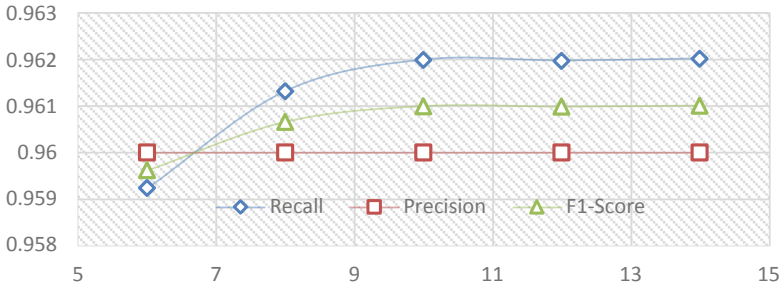

Fig. 6. Influence of flow's length

Batch Size. We also changed the batch size to 64, 128, 256, and 512 respectively and found when the batch size equals to 256, the classification performance is the best.

4.3 Comparison

In order to show the benefits of our method, we compared with some traditional machine learning algorithms, including: DecisionTree, GaussianNB, RandomForest, KNN, SVM. The experimental results are shown in Table 6.

Table 6. Comparison between ML methods

	Decision tree	Gaussian NB	Random forest	KNN	SVM	**Our method**
Precision	93%	66%	94%	94%	86%	**96%**
Recall	93%	55%	94%	94%	75%	**96%**
Accuracy	92.8%	55.4%	94.2%	94.2%	74.7%	**96.2%**

According to the results, we can know that the proposed method of this paper achieves both the highest Precision and Recall rate. The performance of traditional machine learning algorithms are also not bad. The Precision and Recall rate of Decision Tree, KNN and RandomForest algorithms both achieve more than 93%, but the classification effect of GaussianNB and SVM is poor, which have big gaps with the our method. In addition, we find that the training time of traditional machine learning algorithms is much longer than that of deep learning algorithm. For large volume data, the processing speed of traditional machine learning methods will become very slow. While the deep learning technique can quickly see the convergence of training results because of the mini-batch algorithm.

Based on the above experimental results, it can be concluded that the LSTM+AM model proposed in this paper achieves the best results. To further demonstrate the effectiveness of our model, we compared with other two deep learning algorithms: (1) using classical Multi-Layer Perception (MLP); (2) using LSTM without AM. The results are shown in Table 7.

Table 7. Comparison between other DL methods

MLP				LSTM			Our method		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
Normal	0.88	0.99	0.93	0.94	0.93	0.93	0.93	0.99	0.93
DDos	1.00	1.00	1.00	0.85	1.00	0.92	1.00	1.00	1.00
BOT	1.00	1.00	1.00	0.99	1.00	1.00	0.99	1.00	0.99
Inf	0.93	0.19	0.31	0.74	0.23	0.35	0.93	0.17	0.93
BF	0.83	0.94	0.88	0.98	0.98	0.98	0.95	0.98	0.95
Dos	0.90	0.66	0.76	0.98	0.99	0.98	0.98	0.99	0.98
WebAttack	0.81	0.98	0.84	0.66	0.98	0.79	0.30	0.98	0.30
Average	0.91	0.90	0.89	0.93	0.93	0.93	0.96	0.96	0.93
Accuracy	0.904880			0.933332			0.961995		

From Table 7, we know that our method achieves the highest accuracy of 96.2%. The LSTM method is followed by an accuracy of 93.3%, and the accuracy of MLP is only 90.5%. The results show that: (1) LSTM method can indeed learn the previous network traffic information, and can effectively combine the characteristics of historical traffics to make classification. It can achieve better results than the classical multi-layer neural network; (2) AM can focus on those more valuable network traffics, which can help LSTM achieve better classification results.

5 Related Work

We summarized the related work of network anomaly detection into four parts [3].

- **Statistical:** Kruegel et al. [16] introduced a statistical intrusion detection scheme based on Bayesian network, which significantly reduces false alarm rate. Wang et al. [17] presented a payload-based anomaly detector called PAYL for intrusion detection. PAYL can model the normal application payload of network traffic in a fully automated, unsupervised and very efficient manner.
- **Rule-based:** Snort [18] is an open source network anomaly detection system (NIDS), which can analyze and record network data packets in real time. Users can discover various network attacks by performing protocol analysis, content search and matching. Scheirer et al. [19] reported a scheme that consider both syntax and semantics based approaches for dynamic network intrusion detection.
- **Machine Learning:** Boosting Trees (BT) has evolved from the application of boosting methods to Regression Trees, and has been successfully used in many IDS [20, 21]. An intrusion detection system using support vector machine (SVM) and feature selection method is proposed in [22].
- **Deep Learing:** Aksu et al. [5] compared the classification results of SVM and deep learning, and the results show that the deep learning method performs better. Zhu et al. used Convolutional Neural Network to study the network traffics classification issue, but the accuracy obtained by the experiment is not high so it lacks practicality.

6 Conclusion

This paper proposes a dynamic network anomaly detection system using deep learning method. We use LSTM to build the neural network model and incorporate attention mechanism to deal with time-correlated network traffic's classification issues. In order to solve the class-imbalance problem, we used the up to date dataset CSE-CIC-2018 to conduct our experiments, and used the SMOTE algorithm as well as the improved loss function to optimize the training process. The experimental results show that our optimization plays a very significant role. The final trained model achieved a very good result in traffic classification. The overall Accuracy of the system reached 96.2%, and the Recall rate of 6 categories reached 98%. We also compared our method with traditional machine learning methods and other deep learning approaches, and our model achieved the best results.

In the future, we are planning to use the raw data of network traffics so that deep neural networks can automatically learn their features instead of using the artificially extracted features, which can stimulate the maximum potential of neural networks.

Acknowledgment. This work is supported by China National Basic Research Program (973 Program, No. 2015CB352400), National Natural Science Foundation of China (No. 61702492), Equipment Pre-Research Foundation (No. 61400020403), Shenzhen Basic Research Program (No. JCYJ20170818153016513, JCYJ20170307164747920), and Shenzhen Discipline Construction Project for Urban Computing and Data Intelligence.

References

1. Ngu, A.H., et al.: IoT middleware: a survey on issues and enabling technologies. IEEE Internet of Things J. **4**(1), 1–20 (2017)
2. Gill, P., Jain, N., Nagappan, N.: Understanding network failures in data centers: measurement, analysis, and implications. ACM SIGCOMM Comput. Commun. Rev. **41** (4), 350–361 (2011)
3. Karatas, G., Demir, O., Sahingoz, O.K.: Deep learning in intrusion detection systems. In: 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), pp. 113–116 (2018)
4. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
5. Aksu, D., Aydin, M.A.: Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms. In: 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), pp. 77–80 (2018)
6. Zhu, M., Ye, K., Xu, C.-Z.: Network anomaly detection and identification based on deep learning methods. In: Luo, M., Zhang, L.-J. (eds.) CLOUD 2018. LNCS, vol. 10967, pp. 219–234. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94295-7_15
7. Javaid, A., et al.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies, pp. 21–26 (2016)
8. Dong, B., Wang, X.: Comparison deep learning method to traditional methods using for network intrusion detection. In: 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), pp. 581–585 (2016)
9. KDD Cup 1999 (1999). http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
10. CSE-CIC-IDS2018. https://www.unb.ca/cic/datasets/ids-2018.html (2018)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Chawla, N.V., et al.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
13. Chorowski, J.K., et al. Attention-based models for speech recognition. In: Advances in Neural Information Processing Systems, pp. 577–585 (2015)
14. Zhu, M., Ye, K., Wang, Y., Xu, C.-Z.: A deep learning approach for network anomaly detection based on AMF-LSTM. In: Zhang, F., Zhai, J., Snir, M., Jin, H., Kasahara, H., Valero, M. (eds.) NPC 2018. LNCS, vol. 11276, pp. 137–141. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05677-3_13
15. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016), pp. 265–283 (2016)
16. Kruegel, C., et al.: Bayesian event classification for intrusion detection. In: Proceedings of the 19th Annual Computer Security Applications Conference. IEEE (2003)
17. Wang, K., Stolfo, S.J.: Anomalous payload-based network intrusion detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30143-1_11
18. Roesch, M.: Snort: lightweight intrusion detection for networks. Lisa **99**(1), 229–238 (1999)
19. Scheirer, W., Chuah, M.C.: Syntax vs. semantics: competing approaches to dynamic network intrusion detection. Int. J. Secur. Networks **3**(1), 24–35 (2008)
20. Pfahringer, B.: Winning the kdd99 classification cup: bagged boosting. ACM SIGKDD Explor. Newsl. **1**(2), 65–66 (2000)

21. Levin, I.: Kdd-99 classifier learning contest: Llsoft's results overview. SIGKDD Explor. **1** (2), 67–75 (2000)
22. Li, Y., Xia, J., Zhang, S., Yan, J., Ai, X., Dai, K.: An efficient intrusion detection system based on support vector machines and gradually feature removal method. Expert Syst. Appl. **39**(1), 424–430 (2012)