Journal of Computers Vol. 32 No. 4, 2021, pp. 123-136
doi:10.53106/199115992021083204010

A Spark-based Method for Identifying Large-scale Network Burst Traffic

Yu-Lu Sun, Ben-Sheng Yun*, Ya-Guan Qian, Jun Feng

School of Big Data Science, Zhejiang University of Science and Technology, Hangzhou 310023, China
1160778787@qq.com, yunbsh@zust.edu.cn

Received 5 January 2021; Revised 20 April 2021; Accepted 22 April 2021

Abstract. The identification of network traffic plays a vital role in ensuring the safe, stable, and efficient operation of the network. To identify large-scale network burst traffic efficiently, a distributed convolutional neural network method based on Spark is proposed. The 'Raw' data in the TCP protocol is extracted as inputs, and CLR-Distributed-CNN (a distributed convolutional neural network with a cycle learning rate) is used to distinguish network traffic. The accuracy rate of this method reaches 90.417%. Finally, Distributed-RF (a distributed random forest) and EXP-Distributed-CNN (a distributed convolutional neural network with exponential decay of learning rate) are designed to compare with the new method. The accuracy of EXP-Distributed-CNN is 88.167% and that of Distributed-RF is 81.433%. Therefore, the experimental results demonstrate its feasibility and validity.

Keywords: Distributed Convolutional Neural Network, Spark, Traffic Recognition

1 Introduction

The rapid development of the Internet and the enrichment of business network applications brought great convenience to people's lives, but also brought a huge challenge for network management. There are uneven distribution and low utilization rates in the current network resources [1], thus identifying the network traffic can improve the utilization rate of network resources to a certain extent and enhance the controllability of the network. At the same time, network traffic identification has great potential in solving capacity planning, traffic engineering, fault diagnosis, application performance, anomaly detection, and network trend analysis [2]. Network operators can also dynamically deploy QoS (Quality of Service) based on the real-time network traffic identification results and improve the network architecture based on the analysis results, then avoid network congestion and improve network utilization [3]. A complete process of network flow identification can be shown in Fig. 1.

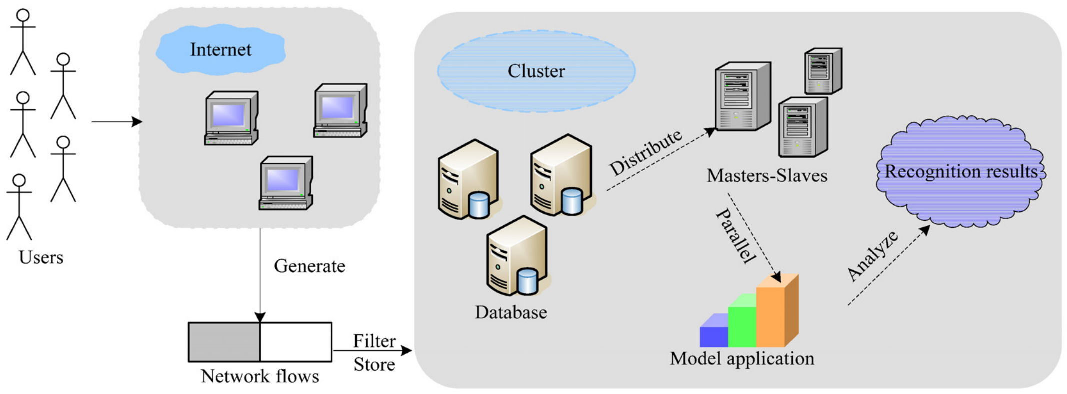


Fig. 1. The process of network flow identification

* Corresponding Author

In the traditional network environment, the service port of the transport layer was relatively fixed, and all kinds of protocols and network applications followed these rules. It is easy and effective to classify network traffic by using port matching technology. When obfuscation techniques are used in network applications, the network traffic generated by them cannot be identified by a port-based approach [1]. Later, the method based on deep packet inspection (DPI) appeared, which is a relatively mature technology, but it cannot distinguish and identify encrypted communication. Meanwhile, it also involves the scanning of users' privacy, which has certain security problems [1].

Recently, it is quite common that a large number of scholars begin to use machine learning methods to identify and distinguish network traffic. These methods do not take port and load into consideration when selecting features [2], and they can build models flexibly and automatically without involving user privacy. It effectively overcomes the defects of port-based and DPI technology. Chen combined RBF neural network and PSO algorithm in their research and compared this method with DT algorithm, NBK algorithm, and BP algorithm. The results show that the new method can obtain relatively good classification results when the network flow features are less [4]. Cai considered that network anonymous communication technology will bring huge challenges to network supervision. For this reason, they designed a new hybrid feature selection algorithm isAnon, which is a combination of improved interactive information and random forest. It can effectively and quickly filter irrelevant and redundant features in network traffic. The experimental results show that this method has good effectiveness in traffic identification [5]. Shafiq conducted comparative experiments on four machine learning classification models (SVM, C4.5 decision tree, naive Bayes, Bayesian network) and found that the effect of C4.5 decision tree is better than other classifiers [6]. Similarly, Dong also selected 4 classic classification methods (C4.5, Naive Bayes, SVM, SVM-RBMS). Through the KDD-99 dataset, it is concluded that SVM-RBMS has the best classification effect [7].

Compared with classic machine learning models, CNNs have higher abstraction capabilities and can train data on the original high-dimensional space. A convolutional neural network has an excellent performance in image classification, handwritten digit recognition, speech recognition, and it is also used in other fields [8]. Since 2017, Wang W. used convolutional neural networks to process and analyze the 'Raw' data of network traffic first time, and this method achieved good results in detecting malicious traffic [9]. From then on, there were more and more scholars improve the convolutional neural network and apply it to the classification and recognition of network traffic. Rahul used convolutional neural networks to learn the data signatures of network traffic and obtained good classification results [10]. Lotfollahi used SAE and CNN for application identification and traffic classification. When identifying whether network traffic is based on VPN, this method is better than other classification methods [11]. Liu proposed two classification methods: the payload classification method based on a convolutional neural network (PL-CNN) and the pay-load classification method based on a recurrent neural network (PL-RNN). Their experiments proved that these two methods are effective and practical [12]. Wang P. conducted comparative experiments on the three deep learning methods (multilayers perceptrons, stacked autoencoders (SAE), and convolutional neural networks), and the results showed that deep learning could identify network traffic well. Finally, they proposed to deploy the three methods into the Distributed Application Awareness Framework (SDN-HGW) in the future [13].

There are many types of network traffic, and the amount is huge. If it is placed in a single machine, it will inevitably encounter problems such as memory overflow and slow calculation. Therefore, building it on a distributed platform can effectively alleviate these problems. Scholars build machine learning on a distributed platform to identify network traffic. Mustapha built 4 commonly used classification algorithms (SVM, Naive Bayes, Decision Tree, Random Forest) on Spark, then analyzed and compared them on the public dataset UNSW-NB15, and found that the distributed random forest performs better than the other three distributed classifiers [14]. Manish also built the commonly used classification algorithm on Spark to classify the dataset KDD-99, and found that the distributed random forest has the best effect [15]. Kong used non-distributed SVM and spark-based distributed SVM to detect abnormal traffic on network traffic. Experimental results proved that using distributed SVM is very effective in reducing training and prediction time [16].

For the above technologies, we summarize their advantages and limitations, as shown in the following Table 1:

Table 1. The advantages and limitations of each technique

Techniques	Advantages	Limitations
Port-based approach	Simple and easy to implement.	Unable to identify network traffic using obfuscation techniques.
Deep packet inspection	High accuracy, mature technology.	1. the feature library needs to be maintained manually. 2. involve privacy issues. 3. cannot identify the encrypted content.
Machine-learning methods	1. do not involve user privacy. 2. filter irrelevant features quickly and effectively. 3. have strong generalization ability.	Take a lot of time and resources to train classification models.
Deep-learning methods	1. automatic feature extraction. 2. save labor costs. 3. have strong generalization ability.	1. the model training period is longer. 2. run out of memory on a single machine.
Distributed machine-learning methods	1. include some aspects of machine learning. 2. can parallel processing at the same time. 3. save training time. 4. high efficiency.	Require a lot of servers.

Convolutional neural networks can learn more subtle and complex structures from pixel values [17], and it does not require manual extraction of features, which greatly reduces labor and time costs. Similarly, in the field of network traffic identification and differentiation, convolutional neural networks can also show their excellent performance. The complex structural characteristics only can be learned from the 'Raw' data of the network traffic application layer. However, in real life, when a networking event breaks out, its propagation speed is rapid, which will produce a large amount of sudden traffic and lead to network congestion and paralysis. To efficiently, quickly, and accurately identify and distinguish network traffic can solve the problems above to a certain extent. At present, in the distributed environment of big data, almost all scholars deploy machine learning algorithms on the distributed platform [14-16] without considering the deep learning algorithms. In this paper, the main contribution is to design a new Spark-based Distributed Convolutional Neural Network (CLR-Distributed-CNN) method to identify and distinguish normal and abnormal traffic. We improve Alexnet and build it on a distributed platform. Our experimental results show that this method can discriminate the network traffic effectively. At the same time, distributed processing can alleviate the problems such as memory overflow, time-consuming, and laboriousness caused by the huge network traffic when traffic identification is carried out in a single computer environment.

The remainder of this paper is organized as follows: Section 2 introduces the related theories and the improved model. Section 3 presents the experiments and performance analyses and the conclusion is drawn in Section 4.

2 Related Theories and Analysis Methods

A convolutional neural network is essentially a multilayer perceptron, which is specially used to deal with the variability of two-dimensional shapes, and its performance is better than other technologies [18]. It can guarantee the invariance of displacement, scale, and distortion to a certain extent. Convolutional neural networks are generally composed of convolutional layers, pooling layers, fully connected layers, and output layers. Among them, a convolutional layer will have multiple feature maps, and their weight vectors are different, so multiple features can be extracted at each position. In this way, it can be considered that the convolutional neural network can automatically extract features. Since Spark is based on in-memory operations, this is very friendly for situations that require a large number of iterations [19]. Therefore, it is feasible and meaningful to build a convolutional neural network on Spark.

2.1 The Cycle Learning Rate

When training a deep neural network, the learning rate plays a very important role in the model. A small difference in learning rates can lead to a great difference in model effectiveness. If the learning rate is too small, the training algorithm will converge slowly, while a big learning rate will cause the training algorithm to diverge. The cycle learning rate (CLR) was proposed by Leslie in 2017. It is not to reduce the learning rate monotonically but to give a suitable range and let the learning rate show periodic changes within this range. In this way, the classification accuracy can be improved without tuning, and the number of iterations can be effectively reduced [20].

The CLR method adopted in this paper is triangular [20]:

$$\begin{cases} stepsize = \dfrac{numbers\ of\ sample}{batchsize} * 2, \\ cycle = np.floor\left(1 + \dfrac{clrinterations}{2 * stepsize}\right), \\ x = np.abs\left(\dfrac{clrinterations}{stepsize - 2 * cycle + 1}\right), \\ lr = baselr + (maxlr - baselr) * np.maximum(0, (1 - x)), \end{cases} \tag{1}$$

where, *clrinterations* represents the number of training epochs, *stepsize* usually means the number of iterations in one cycle or half a cycle, but here it means the number of iterations in two cycles, *baselr* represents the lower bound of the learning rate, and *maxlr* represents the upper bound of the learning rate [20].

2.2 He Initialization

Weight initialization plays an important role in training models. Zero initialization, constant initialization, and too large or too small initialization values will all affect the final effect of the model. A good initialization needs to have a certain degree of randomness, that is, the weight expectation is 0. At the same time, the consistency of its variance needs to be considered. In this way, initializing of the weights can be transformed into random sampling from a certain probability distribution.

He Initialization [21] was proposed by He Kaiming in 2015 and has two forms, namely formulas (2) and (3):

- When the activation function is *ReLu* :

$$\begin{cases} W \sim N\left(0, \dfrac{2}{fan\_in}\right), Forward\ Propagation\ Case, \\ W \sim N\left(0, \dfrac{2}{fan\_out}\right), Backward\ Propagation\ Case. \end{cases} \tag{2}$$

- When the activation function is *PReLu* :

$$\begin{cases} W \sim N\left(0, \dfrac{2}{(1+a^2)fan\_in}\right), Forward\ Propagation\ Case, \\ W \sim N\left(0, \dfrac{2}{(1+a^2)fan\_out}\right), Backward\ Propagation\ Case. \end{cases} \tag{3}$$

In this paper, the *ReLu* is chosen as the activation function, so the probability distribution is in the form of the formula (2).

2.3 Alexnet based on Cycle Learning Rate

Alexnet was proposed by Alex in 2012 [22]. It is usually used for three-channel color image processing and has high performance in the field of computer vision. The original structure of this convolutional neural network is shown in Fig. 2.

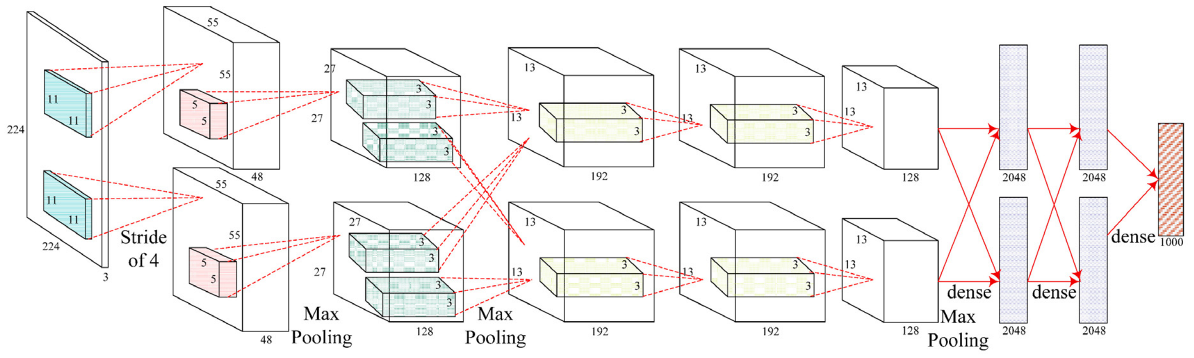


Fig. 2. Alexnet structure

The goal here is to identify large-scale burst traffic, which can be transformed into identifying malicious traffic and normal traffic. Obviously, it is a binary classification problem. In response to the research goal, we improved Alexnet by changing it to a single-channel mode and removing a full connection layer. The input is in the form of 32*32; in the Conv1 layer, 96 5*5 convolution kernels are used, and the step size is set to 1; in Conv2, 192 5*5 convolution kernels are used, and the step size is also set to 1; in the FC layer, 1024 nodes are used; the output layer is 2 nodes; except that the output layer activation function is *softmax*, all the rest is *ReLu*; the weight is initialized by *He initialization*; CLR is used to adjust the learning rate, the average cross-entropy with penalty terms is used as the loss function, the *adam optimizer* is adopted as the optimizer. To prevent over-fitting, a dropout layer is added to the neural network. To avoid losing edge information too quickly, we use all 0 paddings in the Convolution and MaxPooling process. The improved Alexnet network structure is shown in Fig. 3.

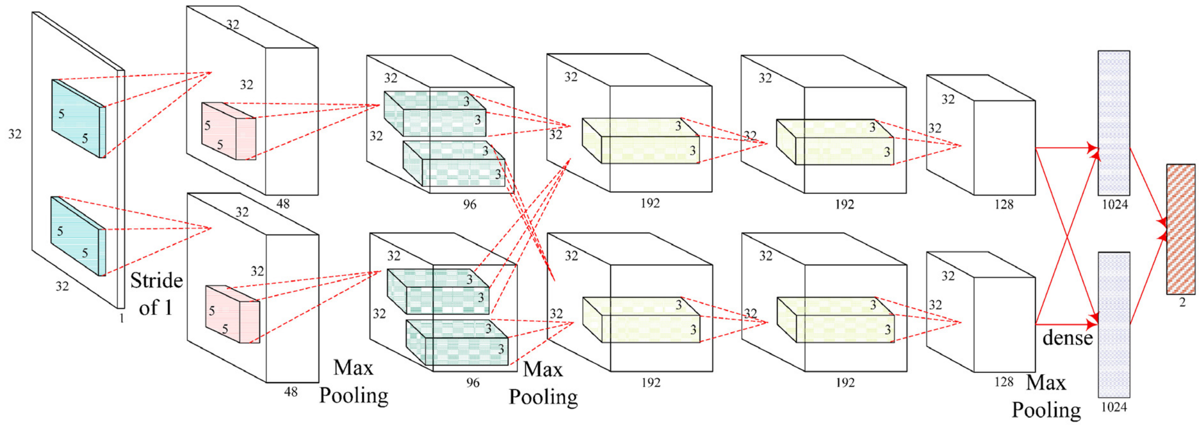


Fig. 3. Improved-Alexnet network structure

2.4 Spark Parallelization

Based on TensorFlowOnSpark framework, the improved Alexnet is built on Spark. In this method, the training of the model is deployed to each node in the Spark cluster, so that each node can train the model at the same time, parallelization namely, and it is greatly increasing training speed. After the model training of each node is completed, the cluster will merge the models according to the results of the nodes, and then obtain the final model. The specific process is shown in Fig. 4.

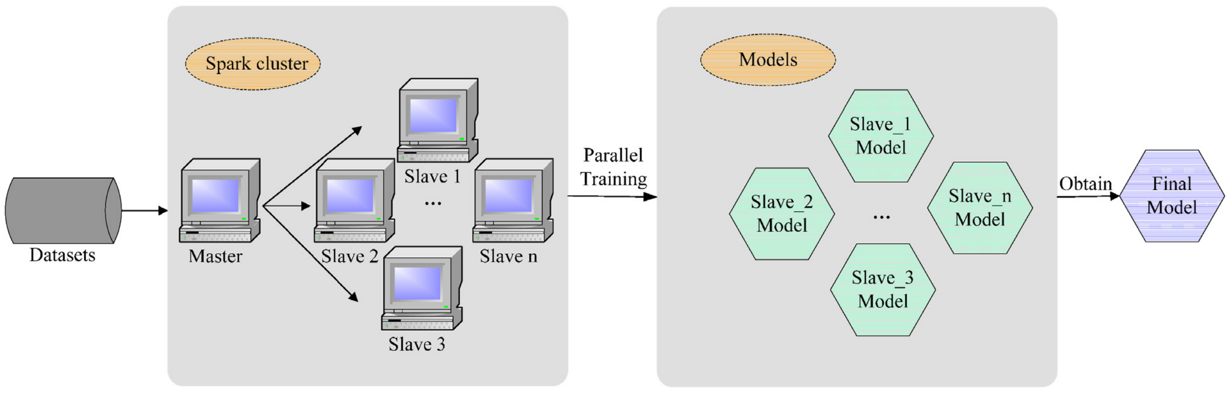


Fig. 4. Spark parallelization flowchart

3 Experiment and Analysis

To show the advantages of distributed convolutional neural network with cycle learning rate (CLR-Distributed-CNN), we compare it with distributed random forest (Distributed-RF) and distributed convolutional neural net-work with exponential learning rate decay (EXP-Distributed-CNN) respectively and observing the *accuracy*, *precision*, *recall*, and *F1 values* of the three methods, then drawing their *Roc curves*.

3.1 Experimental Environment

We build a Hadoop-Spark cluster on three Ubuntu, and configuring the corresponding Tensor Flow On Spark frame, as detailed in Table 2, Table 3:

Table 2. Configuration information of each node

Node	Configuration Information
Sparkmaster	Ubuntu 18.04.4 LTS RAM 15.5GiB CPU Intel® Xeon(R) E-2224G CPU @ 3.50GHz × 4 Graphics AMD® Radeon pro wx2100 GNOME 3.28.2 Operating system type 64-bit Disk 245.7G
Sparkslave1	Ubuntu 19.04 LTS RAM 15.5GiB CPU Intel® Core™ i7-9700 CPU @ 3.00GHZ × 8 Graphics Quadro P620/PCle/SSE2 GNOME 3.28.1 Operating system type 64-bit Disk 2.0TB
Sparkslave2	Ubuntu 19.04 LTS RAM 15.5GiB CPU Intel® Core™ i7-9700 CPU @ 3.00GHZ × 8 Graphics Quadro P620/PCle/SSE2 GNOME 3.28.1 Operating system type 64-bit Disk 2.0TB

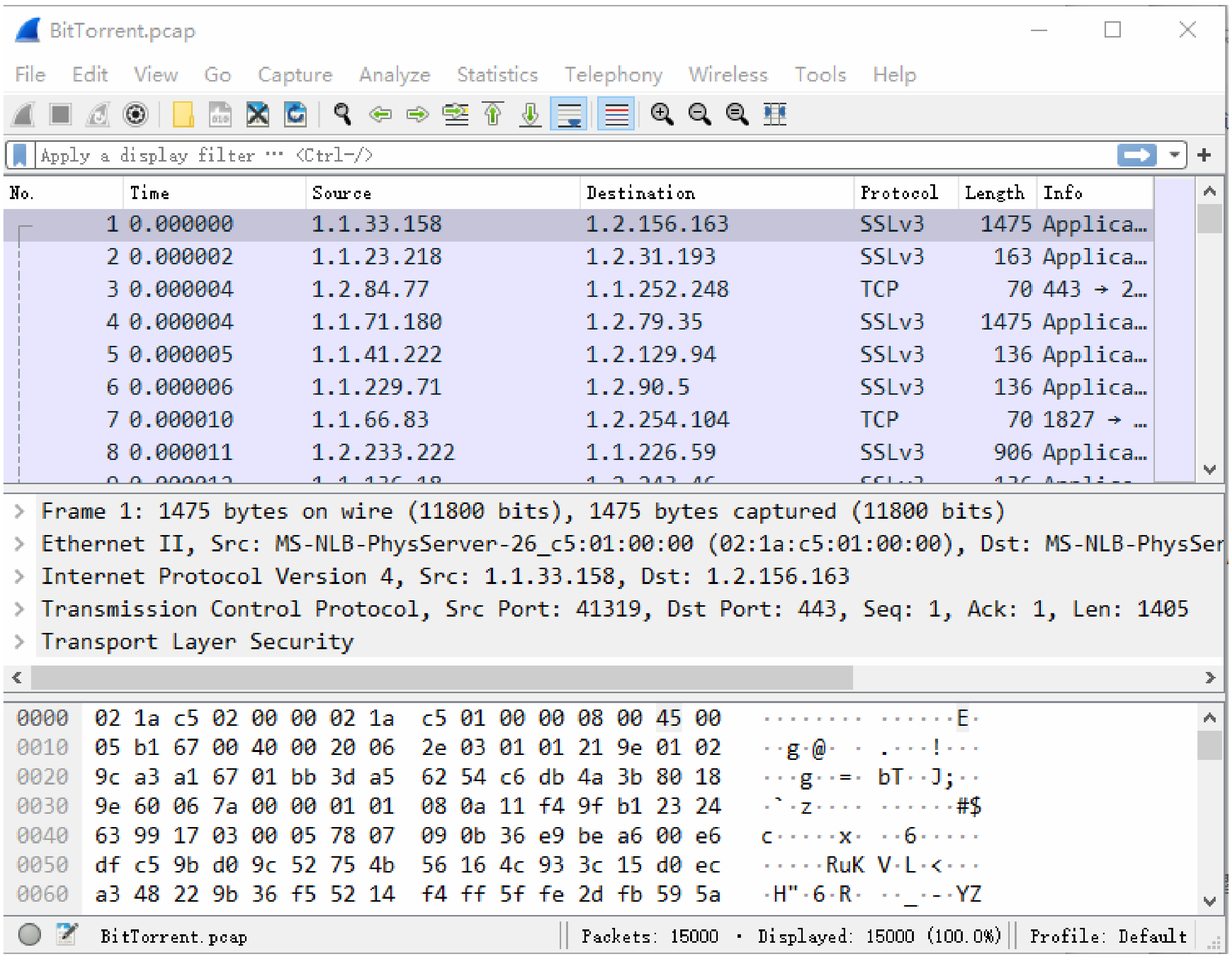
Table 3. Plug-in version information

Plug-in Name	Version
Hadoop	2.7.1
Spark	3.0.0
Java	1.8.0_251
Pydoop	2.0.0
Tensorflow	2.3.0
TensorFlowOnSpark	2.2.1

3.2 Data Preprocessing

The public dataset USTC-TFC2016 [9] is adopted for the experiment, which contains 10 types of malicious traffic and 10 types of normal traffic.

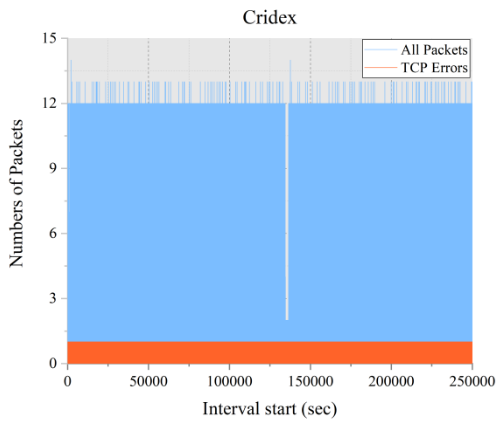
Firstly, we use Wireshark to observe the Pcap file. The specific format of the Pcap file is shown in Fig. 5. The file is divided into three parts. The first part is multiple data packets (including data packet serial number, capture time, source protocol address, destination protocol address, transmission protocol, etc.), The second part is the specific field content of each data packet, and the third part is the hexadecimal information of each data packet, namely ‘Raw’ byte information [23].


BitTorrent.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

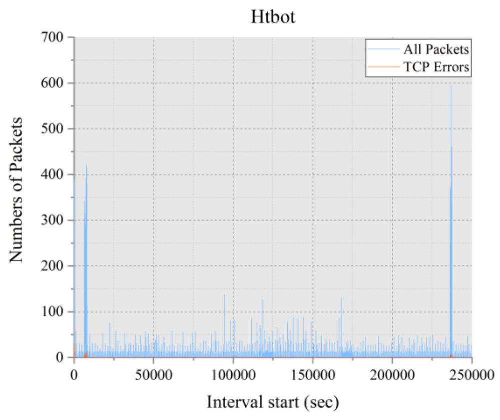
```
No.   Time       Source         Destination     Protocol  Length  Info
    1 0.000000   1.1.33.158     1.2.156.163     SSLv3     1475 Applica…
    2 0.000002   1.1.23.218     1.2.31.193      SSLv3      163 Applica…
    3 0.000004   1.2.84.77      1.1.252.248     TCP         70 443 → 2…
    4 0.000004   1.1.71.180     1.2.79.35       SSLv3     1475 Applica…
    5 0.000005   1.1.41.222     1.2.129.94      SSLv3      136 Applica…
    6 0.000006   1.1.229.71     1.2.90.5        SSLv3      136 Applica…
    7 0.000010   1.1.66.83      1.2.254.104     TCP         70 1827 → …
    8 0.000011   1.2.233.222    1.1.226.59      SSLv3      906 Applica…

> Frame 1: 1475 bytes on wire (11800 bits), 1475 bytes captured (11800 bits)
> Ethernet II, Src: MS-NLB-PhysServer-26_c5:01:00:00 (02:1a:c5:01:00:00), Dst: MS-NLB-PhysSer
> Internet Protocol Version 4, Src: 1.1.33.158, Dst: 1.2.156.163
> Transmission Control Protocol, Src Port: 41319, Dst Port: 443, Seq: 1, Ack: 1, Len: 1405
> Transport Layer Security

0000  02 1a c5 02 00 00 02 1a  c5 01 00 00 08 00 45 00   ········ ······E·
0010  05 b1 67 00 40 00 20 06  2e 03 01 01 21 9e 01 02   ··g·@· · .···!···
0020  9c a3 a1 67 01 bb 3d a5  62 54 c6 db 4a 3b 80 18   ···g··=· bT··J;··
0030  9e 60 06 7a 00 00 01 01  08 0a 11 f4 9f b1 23 24   ·`·z···· ······#$
0040  63 99 17 03 00 05 78 07  09 0b 36 e9 be a6 00 e6   c·····x· ··6·····
0050  df c5 9b d0 9c 52 75 4b  56 16 4c 93 3c 15 d0 ec   ·····RuK V·L·<···
0060  a3 48 22 9b 36 f5 52 14  f4 ff 5f fe 2d fb 59 5a   ·H"·6·R· ··_·-·YZ

BitTorrent.pcap   Packets: 15000 · Displayed: 15000 (100.0%)   Profile: Default
```

Fig. 5. Pcap file format

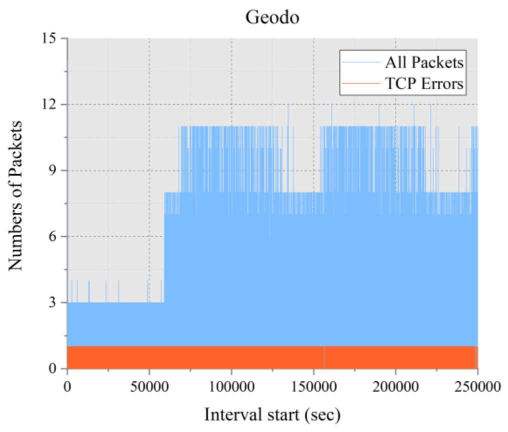
The flow diagram of abnormal flow is given below, as shown in Fig. 6. It can be seen that when packets are sent too frequently or a large number of TCP protocol errors occur within a certain period, there are good reasons to believe that the traffic is abnormal.



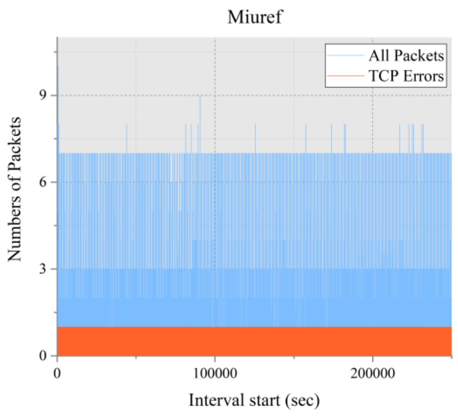
(a) The flow diagram of Cridex



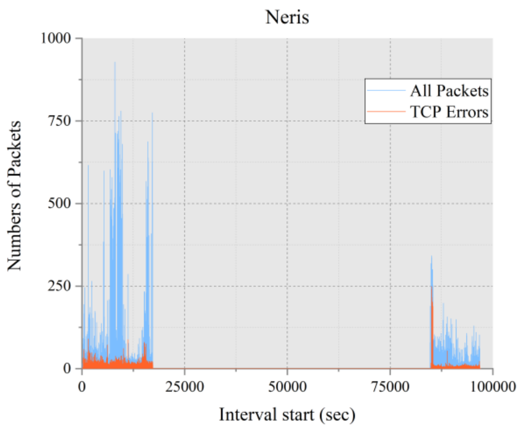
(b) The flow diagram of Htbot



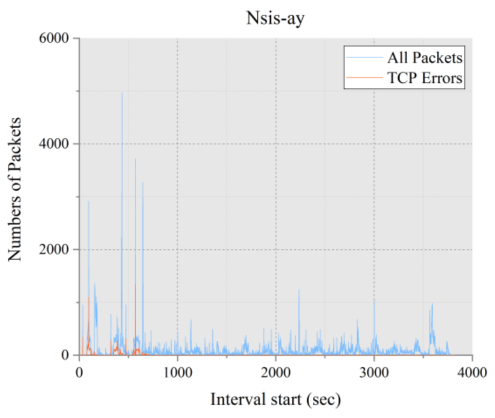
(c) The flow diagram of Geodo



(d) The flow diagram of Miuref

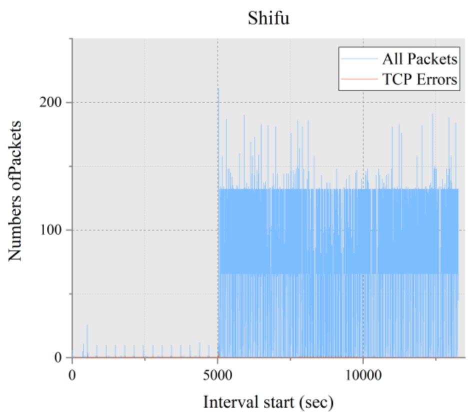


(e) The flow diagram of Neris

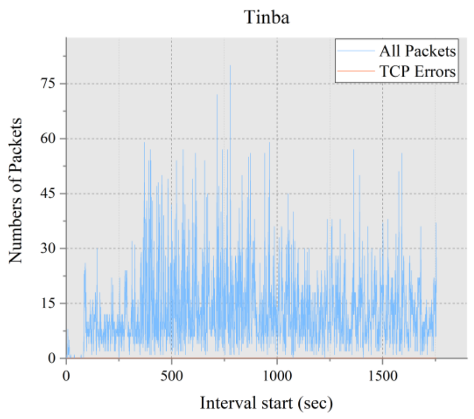


(f) The flow diagram of Nsis-ay

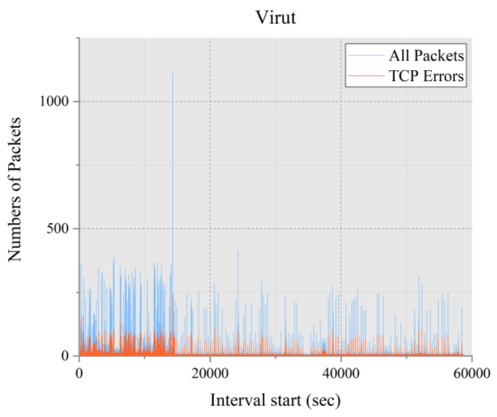
Fig. 6. Schematic diagram of abnormal network traffic



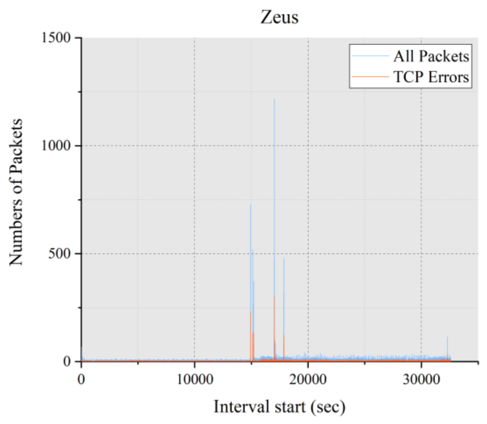
(g) The flow diagram of Shifu



(h) The flow diagram of Tinba



(i) The flow diagram of Virut



(j) The flow diagram of Zeus

Fig. 6. Schematic diagram of abnormal network traffic (continue)

The experiment mainly extracts information from “Raw” data as traffic characteristics. Therefore, using Python’s “scapy” library to extract and parse the Pcap file to obtain the “Raw” information of each packet in the file. Among them, because the MAC address and IP address may occur deviation byte information [24], we only select the “Raw” field and convert the hexadecimal byte information into the corresponding decimal system. Since the length of bytes contained in the “Raw” field between different data packets may be inconsistent, we perform descriptive statistics on the “Raw” field of each data packet, and respectively count the number of non-zero bytes of traffic packets at each byte of normal traffic and abnormal traffic, and subtract the two to find out the absolute value. At the same time, draw the difference map after subtraction, as shown in Fig. 7.

It can be found from Fig. 7 that the difference in the first 1500 bytes is large, almost all exceeding 4000, while the difference in bytes after 1500 is generally less than 4000, so we have reason to choose 1500 as a demarcation point. Considering that the power of 2 is generally adopted as the input size in convolutional neural networks, so we choose the first 1024 bytes as the input. And we set the normal traffic label as “0” and the abnormal traffic label as “1”. After screening, the dataset contains 120,000 samples (60,000 for normal traffic and 60,000 for abnormal traffic), of which 90% are used for training and 10% are used for testing. Specific information can be seen in Table 4.

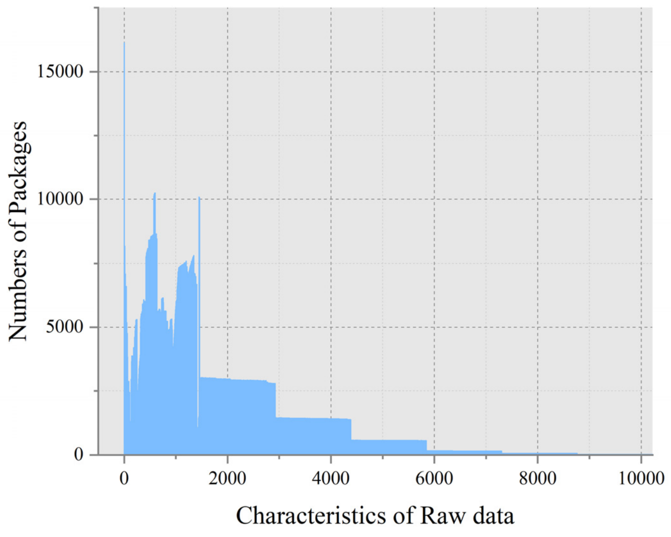


Fig. 7. The difference in byte information between normal traffic and abnormal traffic

Table 4. USTC-TFC2016 processed datasets

Normal traffic	Quantity	Abnormal traffic	Quantity
BitTorrent	6000	Cridex	6000
Facetime	6000	Geodo	6000
FTP	6000	Htbot	6000
Gmail	6000	Miuref	6000
MySQL	6000	Neris	6000
Outlook	6000	Nsis-ay	6000
Skype	6000	Shifu	6000
SMB	6000	Tinba	6000
Weibo	6000	Virut	6000
WordOfWarcraft	6000	Zeus	6000

The overall flow of the above preprocessing can be seen in Fig. 8.

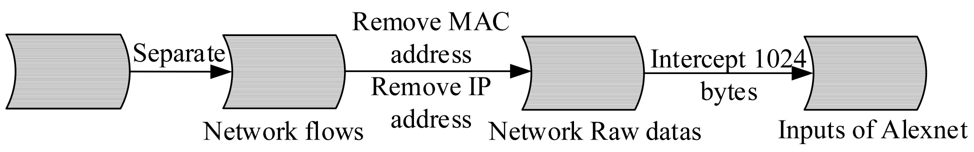


Fig. 8. Pretreatment process

3.3 Evaluation Index

Our experiment uses *accuracy*, *precision*, *recall*, and *F1 value* as evaluation indicators. Table 5 is a confusion matrix, where TP means that the actual is a positive example, and the predicted result is also a positive example. FN means that the actual is a positive example, but the predicted result is a negative example. FP means that the actual is a negative example, but the predicted result is a positive example. TN means that the actual is a negative example, and the predicted result is a negative example. The malicious traffic is taken as a positive example, and normal traffic is taken as a negative example.

Table 5. Confusion matrix

Prediction / Truth	Positive example	Negative example
Positive example	TP	FN
Negative example	FP	TN

Accuracy. For a given test dataset, the ratio of the number of samples correctly classified by the classifier to the total number of samples [25], the formula is:

$$auccracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4}$$

Precision. The ratio of correctly classified positive samples to predicted positive samples. The formula is:

$$precision = \frac{TP}{TP + FP}. \tag{5}$$

Recall. The ratio of correctly classified positive samples to true positive samples. The formula is:

$$recall = \frac{TP}{TP + FN}. \tag{6}$$

F1 value. The harmonic average of precision and recall. The formula is:

$$\frac{1}{F1} = \frac{1}{2}\left(\frac{1}{precision} + \frac{1}{recall}\right). \tag{7}$$

3.4 Performance Analysis and Comparison of the Proposed Method with Other Methods

In this section, we analyze the experimental results of the three methods CLR-Distributed-CNN, EXP-Distributed-CNN, and Distributed-RF.

After a lot of experiments, we finally determined the parameters with the best performance. The learning rate is dynamically adjusted by the *cycle learning rate* (CLR), where the initial learning rate is set to 0.0001 and set the maximum learning rate as 0.001. In this way, the learning rate can dynamically change periodically within this range until the best learning rate is found. For the initialization of the convolution kernel, *He initialization* is selected, and the weights and bias initialization of the fully connected layer use t*runcated normal distribution initialization* and *constant initialization*, respectively. The *Batchsize* is set to 50, the *epoch* is 2, and the fully connected layer adds L2 regularization with a parameter of 0.05. The EXP-Distributed-CNN used for comparison experiments has the same parameter settings as the CLR-Distributed-CNN except for the learning rate. In the Distributed-RF, we chose a forest size of 100 trees and a depth of 5. The specific parameter settings are shown in Table 6. Our experiment combines Spark and Hadoop to build a model on the TensoflowOnSpark framework.

Table 6. Parameter settings

Methods	Learning Rate	Dropout	L2 Regularization	Forest Size	Depth
CLR-Distributed-CNN	Min=0.0001 Max=0.001	0.5	0.05		
EXP-Distributed-CNN	0.0001	0.5	0.05		
Distributed-RF				100	5

Fig. 9 shows the Roc curves of three methods when distinguishing between normal and abnormal traffic. The blue line represents the CLR-Distributed-CNN, the orange line is the EXP-Distributed-CNN, and the red line is the Distributed-RF. Since the higher the TPR (true positive rate) and the lower the FPR (false positive rate), the better the classification effect of the model. As can be seen from this figure, compared with the Distributed-RF, the Roc curve of the distributed convolutional neural network is closer to the upper left corner of the coordinate, so it can be judged that the performance of the distributed convolutional neural network is much better than that of the Distributed-RF, and the CLR-Distributed-CNN has the best performance.

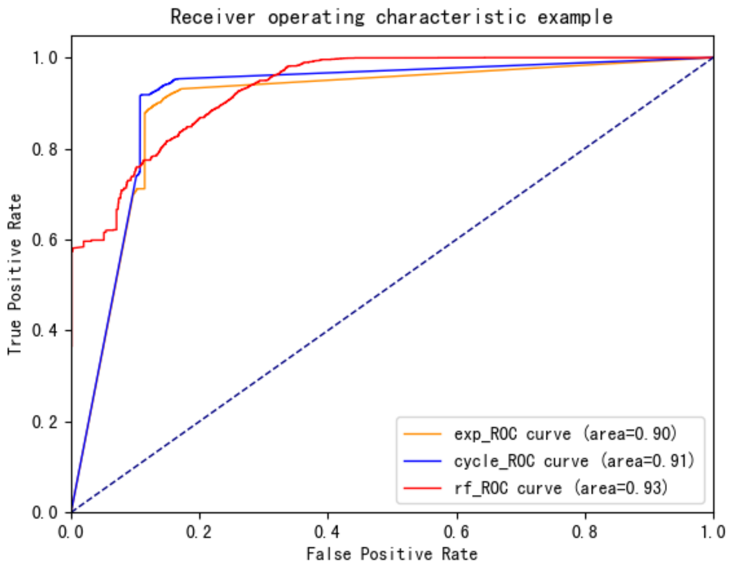


Fig. 9. ROC curves of the three methods

The other performance indicators of the three methods are shown in Table 7. Consistent with the ROC curve, the CLR-Distributed-CNN has the best effect of distinguishing normal flow from abnormal flow among the three methods. Except that its precision is slightly lower than that of Distributed-RF, other indicators are the highest among the three methods, with an accuracy of 90.417%, Recall of 91.617%, and F1 of 90.539%. The EXP-Distributed-CNN is also much higher than the Distributed-RF except for its precision. We can have the result: when identifying network traffic, distributed deep learning can automatically learn the underlying information of network traffic without manually selecting features, and its effect is far better than distributed machine learning.

Table 7. Performance of the three methods

Methods	Accuracy	Precision	Recall	F1
CLR-Distributed-CNN	90.417%	89.469%	91.617%	90.53%
EXP-Distributed-CNN	88.167%	88.359%	87.917%	88.137%
Distributed-RF	81.433%	89.655%	71.067%	79.286%

4 Conclusion

In this article, we propose a method for recognizing normal network traffic and abnormal network traffic: CLR-Distributed-CNN. The ‘Raw’ data of network traffic is used as the input. Our experimental results show that the distributed convolutional neural network does not require manual feature screening of network traffic, but the underlying information is automatically extracted directly from the ‘Raw’ data byte and can achieve good classification performance. Compared with the other two existing methods (EXP-distributed-CNN and Distributed-RF), the experimental results demonstrate that this method can discriminate the network burst traffic effectively. Deploying CLR-Distributed-CNN into the actual network operation environment can strengthen network management and maintain network security.

However, the CLR-Distributed-CNN has some limitations also. First, we simply identified the traffic as normal traffic and abnormal traffic in the process of the experiment, without considering the specific applications of malicious traffic. Second, our method only considers the spatial characteristics of traffic without considering its temporal characteristics. Therefore, in the subsequent progress of the work, we will modify the model, take into account the time characteristics of traffic and identify which application these malicious traffics belong to. Then, to deal with these applications in time and ensure the normal operation of other applications and improve the security of the network.

Acknowledgements

This research is supported by the Scientific Project of Zhejiang Provincial Science and Technology Department under grant agreement No. LGG19F030001, and by the National Natural Science Foundation of China under grant agreement No. 61972357.

References

[1] H.-J. Jiang, J. Xie, X.-F. Guo, H.-Q. Qiu, Q. Qiang, A survey of network traffic classification technology based on SDN, Information Technology and Network Security 37(2)(2018) 40-45,50.

[2] B. Yamansavascilar, M.-A. Guvensan, A.-G. Yavuz, M.-E. Karsligil, Application identification via network traffic classification, in: Proc. 2017 International Conference on Computing, Networking and Communications (ICNC), 2017.

[3] X.-M. Li, H. Ren, J.-Y. Yan, Research on Network Traffic Classification Algorithm Based on Machine Learning, Journal of Communication University of China Science and Technology 24(2)(2017) 9-14.

[4] Y. Chen, H.-Q. Ji, H.-L. Liu, L.-Z. Sun, A traffic identification based on PSO-RBF neural network in peer-to-peer network, International Journal of Computational Science and Engineering 13(2)(2016) 158-164.

[5] Z. Cai, B. Jiang, Z. Lu, J. Liu, P. Ma, isAnon: Flow-Based Anonymity Network Traffic Identification Using Extreme Gradient Boosting, in: Proc. 2019 International Joint Conference on Neural Networks (IJCNN), 2019.

[6] M. Shafiq, X. Yu, A.-A. Laghari, L. Yao, N.-K. Karn, F. Abdessamia, Network traffic classification techniques and comparative analysis using machine learning algorithms, in: Proc. 2016 2nd IEEE International Conference on Computer and Communications (ICCC), 2016.

[7] B. Dong, X. Wang, Comparison deep learning method to traditional methods using for network intrusion detection, in: Proc. 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), 2016.

[8] S. Tabik, D. Peralta, A. Herrera-Poyatos, F. Herrera, A snapshot of image pre-processing for convolutional neural networks: case study of MNIST, International Journal of Computational Intelligence Systems 10(1)(2017) 555-568.

[9] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware traffic classification using convolutional neural network for representation learning, in: Proc. 2017 International Conference on Information Networking (ICOIN), 2017.

[10] R.-K. Rahul, T. Anjali, V.-K. Menon, K.-P. Soman, Deep learning for network flow analysis and malware classification, in: Proc. International Symposium on Security in Computing and Communication, 2017.

[11] M. Lotfollahi, M.-J. Siavoshani, R.-S.-H. Zade, M. Saberian, Deep packet: A novel approach for encrypted traffic classification using deep learning, Soft Computing 24(3)(2020) 1999-2012.

[12] H. Liu, B. Lang, M. Liu, H. Yan, CNN and RNN based payload classification methods for attack detection, Knowledge-Based Systems 163(2019) 332-341.

[13] P. Wang, F. Ye, X. Chen, Y. Qian, Datanet: Deep learning based encrypted network traffic classification in sdn home gateway, IEEE Access 6(2018) 55380-55391.

[14] M. Belouch, S. El Hadaj, M. Idhammad, Performance evaluation of intrusion detection based on machine learning using Apache Spark, Procedia Computer Science 127(2018) 1-6.

[15] M. Kulariya, P. Saraf, R. Ranjan, G.-P. Gupta, Performance analysis of network intrusion detection schemes using Apache Spark, in: Proc. 2016 International Conference on Communication and Signal Processing (ICCSP), 2016.

[16] L. Kong, G. Huang, Y. Zhou, J. Ye, Fast Abnormal Identification for Large Scale Internet Traffic, in: Proc. Proceedings of the 8th International Conference on Communication and Network Security, 2018.

[17] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps. <https://arxiv.org/pdf/1312.6034.pdf>, 2014 (accessed 19.04.14).

[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11) (1998) 2278-2324.

[19] J. Xu, S. Ma, Image classification model based on spark and CNN, in: Proc. In MATEC Web of Conferences, 2018.

[20] L.-N. Smith, Cyclical learning rates for training neural networks, in: Proc. 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), 2017.

[21] K.-M. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proc. Proceedings of the IEEE international conference on computer vision, 2015.

[22] A. Krizhevsky, I. Sutskever, G.-E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60(6) (2017) 84-90.

[23] L. Liu, Research on DPI-based Network Service Flow Identification Technology, [dissertation] Qufu Normal University, 2017.

[24] G. Marín, P. Casas, G. Capdehourat, DeepMAL--Deep Learning Models for Malware Traffic Detection and Classification. <https://arxiv.org/pdf/2003.04079.pdf>, 2020 (accessed 10.03.20).

[25] H. Li, Statistical learning methods, second ed., Qing hua da xue chu ban she, Beijing, 2019 (Chapter 1).