Hindawi
Security and Communication Networks
Volume 2021, Article ID 6674325, 11 pages
https://doi.org/10.1155/2021/6674325

Research Article

Cost-Sensitive Approach to Improve the HTTP Traffic Detection Performance on Imbalanced Data

Wenmin Li, Sanqi Sun, Shuo Zhang, Hua Zhang, and Yijie Shi

The State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Shuo Zhang; shuozhang@bupt.edu.cn

Received 18 December 2020; Revised 28 April 2021; Accepted 13 May 2021; Published 24 May 2021

Academic Editor: Weizhi Meng

Aim. The purpose of this study is how to better detect attack traffic in imbalance datasets. The deep learning technology has played an important role in detecting malicious network traffic in recent years. However, it suffers serious imbalance distribution of data if the traffic model skews towards the modeling in the benign direction, because only a small portion of traffic is malicious, while most network traffic is benign. That is the reason why the authors wrote this manuscript. *Methods*. We propose a cost-sensitive approach to improve the HTTP traffic detection performance with imbalanced data and also present a character-level abstract feature extraction approach that can provide features with clear decision boundaries in addition. Finally, we design a spark-based HTTP traffic detection system based on these two approaches. *Results*. The methods proposed in this paper work well in imbalanced datasets. Compared to other methods, the experiment results indicate that our system has F1-score in a high precision. *Conclusion*. For imbalanced HTTP traffic detection, we confirmed that the method of feature extraction and the cost function is very effective. In the future, we may focus on how to use the cost function to further improve detection performance.

1. Introduction

1.1. Background. In the past few years, cybersecurity incidents have occurred frequently. In the first half of 2018, 360 Internet Security Center intercepted 140 million malicious programs in all, nearly 795,000 ones per day on average [1]. Moreover, around 8% of Hypertext Transfer Protocol (HTTP) messages in 2017 were reported to be malicious [2].

Deep learning, as one of the most currently remarkable machine learning techniques, has achieved great success in many applications such as image analysis, speech recognition, and text understanding [3]. In the field of objection detection, Girshick et al. [4] greatly improved the accuracy of objection detection through the deep learning technology. Wu et al. [5] used weakly supervised learning to classify and annotate images. Rattani et al. [6] applied deep learning technology to the field of selfie biometrics and has made good progress. In medical image segmentation, U-NET [7] is undoubtedly one of the most successful methods, which was proposed at the MICCAI conference in 2015. In the field of HTTP traffic detection, the deep learning technology is prominent way to detect malicious network traffic. However, it suffers serious imbalanced distribution of data. For example, the traffic detection tasks usually focus on reducing malicious traffic such as web attack, but not the data of web browsing accounts for the majority. The contribution of the majority class to the cost function far exceeds that of the minority class. Therefore, it is difficult to identify the small amount of traffic, which brings serious challenges to network traffic classification [8].

1.2. Related Work. The detection technologies of imbalanced data can be classified into three types: data-level methods, feature extraction, and cost-sensitive learning. Oversampling, undersampling, and random sampling are the most commonly used in data level. Jin et al. [9] and Lim et al. [10] applied the data-level methods to rebalance traffic data and improve the performance of imbalanced dataset detection. Oversampling improves classification performance by increasing the number of the minority class samples.

However, due to the large number of copies of the minority class samples, the classification algorithm is difficult to avoid overfitting. Undersampling improves classification performance by reducing the number of the majority class samples. However, in the field of HTTP traffic detection, the majority class samples are far more than the minority class samples, and the quantity difference may be hundreds of times, so the downsampling method may not be suitable. Random sampling randomly abandons the minority class samples, which may remove potentially useful information from the minority class samples. Park et al. [11] proposed an anomaly detection technique for imbalanced HTTP traffic utilizing convolutional autoencoders (CAE), which belongs to the type of feature extraction. However, converting HTTP massage into an image via one-hot encoding will lose some original information. And we will improve the feature extraction method mentioned in the paper. Another common method is cost-sensitive learning, which uses a cost function to train the classifier. The cost-sensitive method in [12] is based on decision tree. However, due to the complexity of the problem, the performance of the algorithm based on neural network is usually better than the algorithm based on decision tree in the field of HTTP traffic detection. Chen et al. [13] introduced a novel imbalanced classification model, named simplex imbalanced data gravitation classification (S-IDGC). This model uses Euclidean distance to calculate gravity, but fails to consider the data distribution characteristics and the results were in a low precision. Recently, the focal-loss cost function proposed by Lin et al. [14] has been proved to be effective in the field of image segmentation. This method performs well in image segmentation. Tong [15] proposed a traffic classification method based on convolutional neural network which consists of two main traffic classification stages and combines the flow and packet-based features to predict the services based on quick UDP internet connection. Aceto et al. [16] used multimodal deep learning to study mobile encrypted traffic classification which has a good result about TSL traffic detection. Lotfollahi et al. [17] combined port-based, payload inspection and statistical machine learning to analyze encrypted traffic classification. Bovenzi [18] imposed a hierarchical hybrid intrusion detection approach, which has been proved to be very effective in the Internet of things scenario. Aceto [19] firstly investigated and experimentally evaluated the adoption of DL-based network traffic classification strategies as supported by BD frameworks. The recent schemes focused on the mobile or light equipment and they analyzed encrypted traffic classification. However, in the field of HTTP traffic detection, the contribution of the minority class samples to the loss function will be reduced according to the predicted value in the training model, which is not conducive to the detection of the minority class samples. The characteristics of recent related works are shown in Table 1.

Contributions: the main motivation of this paper is to detect attacks from serious imbalanced network traffic. To achieve this goal, we address it from two aspects: feature extraction and cost function. The main contributions of this paper can be summarized as follows:

(i) In terms of feature extraction, we present character-level abstract feature extraction approach which can provide features with clear decision boundaries.

(ii) In terms of cost function, we present the HM-loss cost function to improve the http traffic detection performance on imbalanced data. The cost-sensitive approach can reduce the contribution of the majority class in the cost function.

(iii) Finally, we design and implement spark-based HTTP traffic detection system and apply the cost-sensitive approach and the feature extraction approach into this detection system. The experiment results show that proposed scheme has higher precisions, recall, and F1-score.

The rest of this paper is organized as follows. Section 2 is a detailed description on character-level abstract feature extraction approach whereas Section 3 describes cost-sensitive approach. The experiment is given in Section 4. Conclusion and future directions are given at the end of the paper.

2. Methods

2.1. The Character-Level Abstract Feature Extraction Approach. We present the character-level abstract feature extraction approach in this section which combines character-level features and abstract features. Our main work is to extract character-level features based on spark [20] clusters, design a one-dimensional convolutional autoencoder, and then extract abstract features.

For the feature extraction of http traffic, n-gram feature [21] and character-level feature [22] are the most popular methods for converting HTTP messages into input vectors fed into neural networks. However, n-gram features can cause a large amount of information loss and have higher feature dimensions, and character-level features have no clean decision boundaries on imbalanced data because it contains a lot of noise information. Considering that the abstract features generated by CAE have clean decision boundaries, but CAE cannot directly obtain input vectors from http traffic, we present the character-level abstract feature extraction approach based on character-level features. The workflow of the abstract feature extraction approach is shown in Figure 1.

2.2. Character-Level Feature Extraction Method Based on Spark. Zhang et al. [22] have done a lot of research on character-level features. However, Zhang's experiment is implemented on a single computer. In an actual production environment, it will encounter a calculation bottleneck. Therefore, we extend the character-level feature extraction method on spark and combine it with abstract features to extract character-level abstract features used in the paper.

We perform preprocessing steps and extract character-level features on the spark cluster. First, we install and configure the Hadoop cluster [23] on the ubuntu server, and our spark mode is spark on yarn. Second, we allocate

appropriate computing resources for spark tasks based on the amount of task data and expected completion time. For example, the gateway produces 15 GB http traffic stored on Hadoop Distributed File System (HDFS) every 5 minutes. We set the size of HDFS default block to 128 MB, so the traffic can be split into 120 (15 GB/128 MB) tasks. Then, we assume that 120 tasks need to run 2 to 3 times in the cluster (according to experience, this configuration can maximize resource utilization). Therefore, we can assign 50 executor instances to the cluster, each instance assigning 2 to 3 CPU. Third, we write spark programs with the Jupyter notebook tool. The specific steps of data preprocessing and feature extraction are same as in the paper [22]. The pseudocode of the character-level feature extraction algorithm is shown in Algorithm 1. In pseudocode, we will merge the URL and post fields into feature string and then filter out non-ASCII characters. In the end, we need to generate a string of fixed length L, and if the length is greater than L, truncate it; if the length is less than L, repeat filling until the length reaches L. After extracting the character-level features, we transfer the feature data into Kafka [24] system for using in subsequent steps.

For malicious http traffic, the URL and body fields are more likely to contain sensitive attack information. Therefore, this paper chooses these two fields as the main detection target. Part of the training set containing only the URL field and the body field is shown in Table 2.

2.3. Abstract Feature Extraction by Autoencoder. The section mainly presents the workflow of extracting the abstract feature generated by the one-dimensional CAE. The main motivation of the abstract features generated by CAE is to generate a clean decision boundary. Therefore, we use the one-dimensional CAE to generate abstract features by learning the character-level feature. The results show that this method can effectively reduce the impact of imbalanced data distribution on the malicious traffic detection.

In Figure 2, the classic CAE's input is two-dimensional images. But the URL and post fields for HTTP traffic are one-dimensional, unlike images with two-dimensional spatial information; the paper processes the input text data into one-dimensional.

Figure 3 shows the structure of the one-dimensional CAE designed by us. Each layer of the encoder consists of multiple nodes, and the last hidden layer of the encoder generates the abstract feature. The decoder also has a multi-layered structure that is symmetrical with the corresponding layer in the encoder, and the last layer of decoder is the output layer. The cost function calculates the error based on the output layer and the input layer. And, to reduce overfitting, the dropout ratio between the encoder and the decoder is set to 0.1.

CAE is unsupervised learning, so manual labels are not required and the CAE's input is $300 * 1$ character, and after convolution steps, an abstract feature of size $25 * 8$ is generated.

3. The Cost-Sensitive Approach

We present the HM-loss cost function in this section. Our main work consists of two parts. First, we describe the disadvantages of the CE-loss when dealing with imbalanced http traffic. Second, we design the HM-loss cost function which is cost-sensitive. In this approach, we design a coefficient for the loss function and when the classification algorithm predicts the minority class samples, the weight coefficient factor can dynamically adjust the contribution of the majority class samples to the loss function. When the minority class samples are predicted, the contribution of the sample of the loss function is kept unchanged.

3.1. The Disadvantages of the CE-Loss on Imbalanced HTTP Traffic. The CE-loss function used as the cost function for deep learning techniques is very popular in the classification task. However, it suffers from a low F1-score value when dealing with severely imbalanced HTTP traffic in an actual production environment. Because the contribution of the majority class to the cost function far exceeds that of the minority class, the model's decision tends to support the majority class and ignore the minority exception class [14].

Figure 4 shows the loss value of the CE-loss varies with the prediction probability. As shown in the picture, the predicted value of a single normal sample tends to be large, close to 1, but the contribution to the loss function is small. Conversely, the predicted value of a single malicious sample tends to be small (less than the normal sample), but contributes a lot to the loss function.

Now, we assume that the predicted probability of a normal sample is 0.97, and we can calculate that the sum of contribution to the cost function of 500,000 normal samples is 15,229. At the same time, assuming that the predicted probability of a malicious sample is 0.88, the sum of contribution to the cost function of 705 malicious samples is 39.14. The loss value of the benign sample was about 389 (15,229/39.14) times that of the malicious sample. Therefore, in the backpropagation of the neural network, the loss of normal samples dominates the decline of the gradient, and the algorithm focuses on the majority class.

3.2. The Definition of HM-Loss Cost Function. In this part, we design the HM-loss cost function and give the definition of HM-loss cost function. First, we present the idea of HM-loss, then give the definition of HM-loss, and finally summarize the advantages and characteristics of HM-loss.

The main idea of the HM-loss cost function is to dynamically adjust the weight of sample's contribution to the loss value [14]. When the true label belongs to the majority category (negative), the weight of the contribution to the loss decreases, and the degree of decrease varies according to the predicted probability value, and usually, when the prediction is correct, the contribution to the loss decreases greatly. In addition, our cost function has another property. When the

true label belongs to the minority class (positive), the weight of contribution to the loss remains. Therefore, we can adjust the algorithm to focus on the minority class samples by giving the majority class samples less attention [25]. We focus on the minority class samples but not the majority class samples.

The idea of the HM-loss cost function is present in the previous paragraph. Now, we give the specific definition. The definition of HM-loss is shown in formula (1). "$ytrue$" represents the real label, and there are only two values of 0 and 1, where 0 represents the positive class and 1 represents the negative class. "$ypred$" represents the prediction probability value, which ranges from 0 to 1.

$$\text{loss} = -\left(ytrue * \cos(\alpha * ypred)^{\gamma} + (1 - ytrue)\right) * \log(ypred), \quad ytrue\,\varepsilon\{1, 0\} \text{ and } \alpha\varepsilon\left(0, \frac{\pi}{2}\right). \tag{1}$$

The HM-loss cost function derived from the CE-loss consists of two parts. The first part is "$ytrue * \cos(\alpha * ypred)^{\gamma}$" which controls the weight that varies from ypred value. The two hyperparameters contained in this part are to adjust the degree of weight reduction. The second part is "$(1 - ytrue)$" which controls the weight of the minority class's contribution to the loss function and it remains unchanged. Figure 5 shows the loss value of the HM-loss cost function under different hyperparameters.

We explain the definition of the HM-loss cost function in the previous section, and we present the advantages of it as follows. The first advantage of the HM-loss cost function is that the contribution of the majority class samples to the loss function can be dynamically reduced according to the predicted value. The second is that, regardless of whether the minority classes samples are predicted correctly or not, its contribution to the loss function does not change. The third is that only when the majority class samples are correctly predicted and the probability value is close to 1, the weight value decreases faster.

Now, we take a simple example of what the HM-loss cost function does. Figure 6 shows the loss of 500,000 normal samples under different loss functions, and to show the data better, we select the data with the probability value between 0.85 and 1. We assume that the probability of the majority class sample is 0.97, and we can calculate that the sum of 500,000 samples loss value under the CE-loss cost function is 15,229. Similarly, we can calculate that the loss using the HM-loss cost function under different hyperparameters is 4642, 431, and 40, respectively. It can be seen that the HM-loss cost function is very effective in this case.

4. Results

4.1. Experiment. In this section, we explain the details of the datasets and the performance metrics used in the experiments. Using these metrics, we compare the performance of the proposed scheme with related schemes including data-level methods and Park's method [11]. The experiment consists of three parts. Firstly, we introduce the preparation stage of the experiment, including the dataset and the experimental evaluation index. Then, we show the structure of convolutional neural network used to detect malicious samples. The third part shows the experimental results.

4.2. Experiment Setup. We use real traffic data accumulated over time to validate our approach. And we collect around 701,000 HTTP messages from the gateway of a university in 2019 for this experiment. The collected data is highly sensitive because it contains most of the network activities during work hours of teachers and students. For these data, we perform manual verification and tagging. The types and quantities of malicious samples are shown in Table 3. The numbers of normal and anomalous HTTP messages are around 700,000 and 1,000, respectively. We divide it into training datasets and test datasets according to a certain proportion.

Existing studies have shown that AUC has certain limitations in performance evaluation, especially when the numbers of normal and anomalous messages are significantly different [26, 27]. Therefore, we use F-score [28]. Specifically, the F-score directly related to the recall and the precision is the harmonic average of the precision and recall, and the specific definition is shown in the following formula.

$$F = \frac{2}{(1/\text{recall}) + (1/\text{precision})}. \tag{2}$$

4.3. The Neural Network Model Structure. After obtaining character-level abstract features, we can train the CNN network to classify samples. Our model is based on one-dimensional convolutional neural network which can acquire more local feature. The reason for using one-dimensional vector is that HTTP traffic has no two-dimensional space attributes. The structure of the model used by this experiment mainly includes the input layer, the hidden layer, and the output layer. The input layer first converts the input data into the input tensor fed into one-dimensional convolutional neural network. After the convolution, pooling, ReLU, and flattening steps, the softmax function produces the prediction value. The model architecture is shown in Figure 7.

Table 1: Comparison of related works.

Scheme	Method	Encrypted or not	Friendly to lightweight devices	Friendly to a few samples
Jin et al. [9]	Data-level method	No	No	No
Lim et al. [10]	Data-level method	No	No	No
Park et al. [11]	Convolutional autoencoders	No	No	No
Ting [12]	Decision tree	No	No	No
Chen et al. [13]	Implex imbalanced data gravitation classification	No	No	No
Lin et al. [14]	The focal-loss cost function	No	No	No
Tong [15]	Convolutional neural network	No	No	No
Aceto et al. [16]	Multimodal deep learning	Yes	Yes	No
Lotfollahi [17]	Port-based, payload inspection and statistical machine learning	Yes	No	No
Bovenzi [18]	Hierarchical hybrid intrusion detection	Yes	Yes	No
Aceto [19]	Big data-enabled DL framework for mobile TC	Yes	Yes	No
Ours	Spark-based HTTP traffic detection	No	Yes	Yes

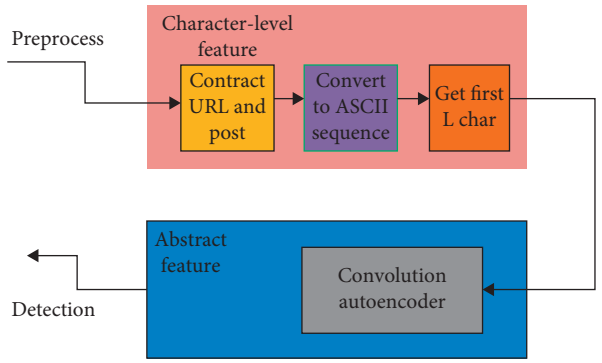

Figure 1: Feature extraction flow chart.

```
Input: HTTP traffic path
(1) Configure the resources occupied by the spark task
(2) Init spark session
(3) Initialize: Truncated fixed length: L, result: res
(4) feat-contract URL and post
(5) Filter non-ASCII characters of feat
(6) if(the length of feat ≥ L){
(7)     feat = the first L character of the feat
(8) }
(9) else{
(10)        do{
(11)            feat = merge two feat strings
(12)        }
(13)        While(getLength( feat) > L)
(14) }
(15) if(the length of feat ≥ L){
(16)        feat = the first L character of the feat
(17) }
(18) Return feat; //return the string of fixed length
```

Algorithm 1: Character-level feature (HTTP traffic path).

Table 2: URL and post features.

Label	Post	URL
1	−7 = @eval(get_magic_quotes_gpc()?stripslashes($...	http://weki.php/admin/login.action
1	sqzr = @eval(get_magic_quotes_gpc()?stripslashes($...	http://plus/sdfg.php//plusmytag_js.php?aid=8080
1	C = /var/www/vhosts/13/133103/webspace/httpdocs/...	http://wp-includes/js/crop/data.php
1	z2 = ???php+\r\n**titleline = file('key.txt')**;\r\n...	http://upload/2015/09/07/1441610150952000.jsp
1	action = editfile&fname = d:\wwwroot\www.jsjyjxx.c...	http://com4.yichen.asp?action2=post
1	sqzr = response.write("------>/");var err:except...	http://news/pics/20151017/201510171445088639847.php
0	----------a1fa340557\0 × 0d\0...	http://upload.php?hid=sgpy-windows-generic-device-id...
0	api_token = f04362c49325b5cf1ef9d373e4fb89dc16d7...	http://kancolle/proxykancolleapi?h=125.6.189.247&p=/...
0	----------cqdems00sd0wevzsr...	http://cloudquery.php
0	ejx9k8ty4yaqrb/gs1mkeahpmys81vmfruyhmxojoqg5cb...	http://restapi.php

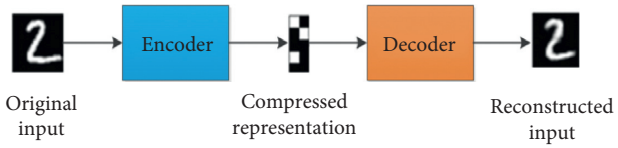

Figure 2: Autoencoder structure.

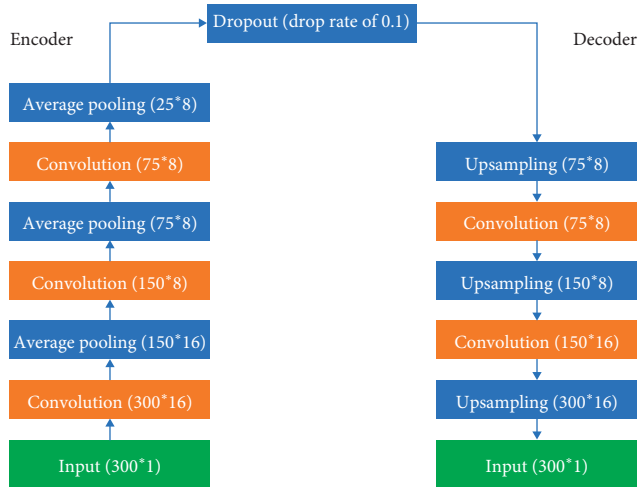

Figure 3: One-dimensional convolution autoencoder architecture.

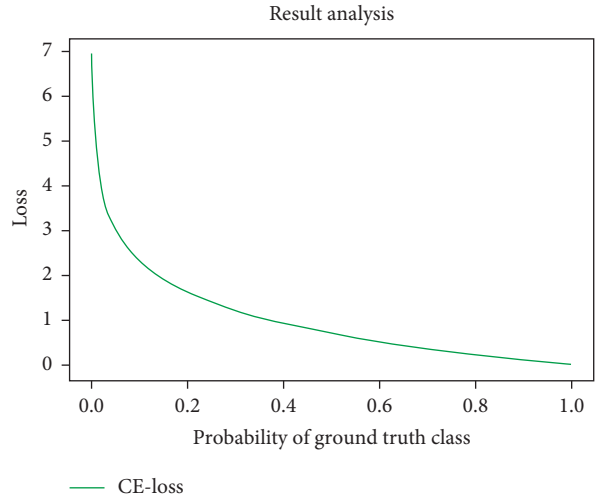

Figure 4: The loss value of CE-loss varies with the prediction probability.

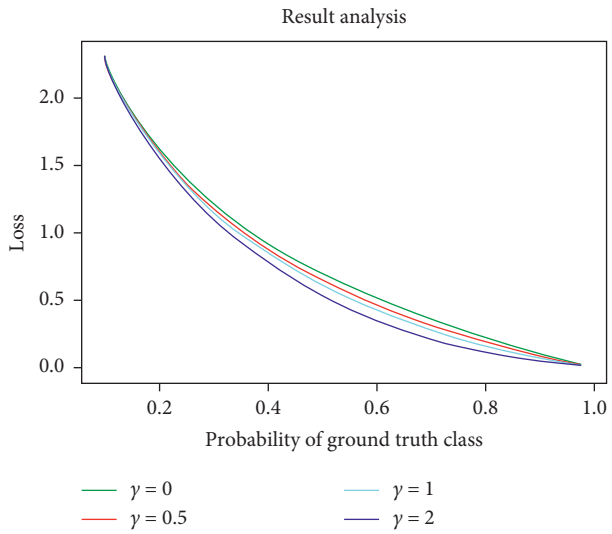

Figure 5: $-\cos\ (\alpha * ypred)^{\gamma} * \log(ypred)$ with different values in γ.

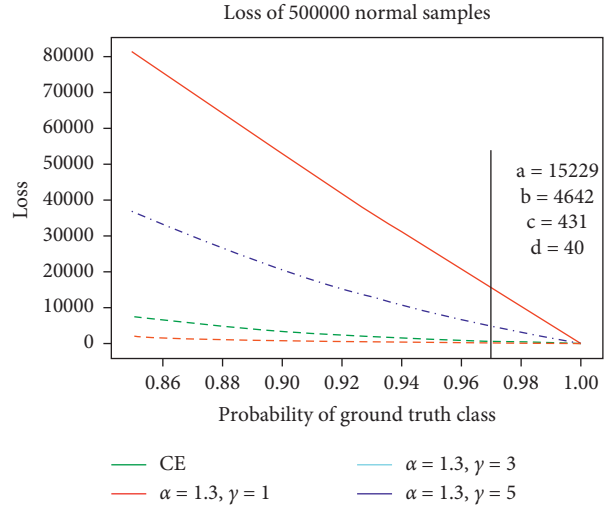

Figure 6: Contribution of the 500,000 normal sample to the loss function.

4.4. Experiment Result

4.4.1. HM-Loss Cost Function with Different Hyperparameters. The purpose of this experiment is to find the best two hyperparameters of HM-loss, the experimental data used in this paper.

Table 4 shows that when the hyperparameter alpha and gamma take 1.3 and 3, respectively, the HM-loss performs best on the dataset. And precision recall and F1-score value can reach 0.90, 0.84, and 0.87, respectively.

4.4.2. Comprehensive Experimental Results. The following experimental results are divided into two parts. First, we verify the effectiveness of the character-level feature extraction method. Second, we compare our method with other methods.

(i). We compare Park's feature and character-level abstract features. In this process, we apply different feature extraction methods, but the same algorithm. As shown in Figure 8, our feature has a higher F-score value. Therefore, through this comparative experiment, we can conclude that our feature can work.

(ii). We compare our method with others' methods. The ordinary method which does not adopt any strategy. The oversampling method focuses on the data level. Park's method mentioned in paper [11] focuses on feature extraction. And Lin et al.'s method mentioned in the paper [14] mainly focuses on cost function, which is effective in the field of computer vision. Our method focuses on both feature extraction and cost functions.

Table 3: Dataset.

	Normal	Malicious	Total
Training set	500000	705	500705
Test set	250000	352	250352
total	700000	1057	701057

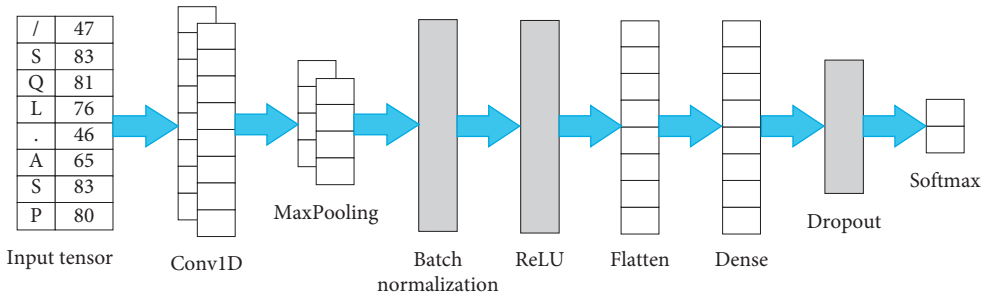

Figure 7: Convolutional neural network structure.

Table 4: The performance metrics when the HM-loss's hyperparameters are assigned different values.

Alpha	Gamma	Precision	Recall	$f1$-score
1.1	1	0.89	0.71	0.79
1.1	3	0.89	0.67	0.76
1.1	5	0.91	0.76	0.83
1.3	1	0.91	0.75	0.82
1.3	3	0.90	0.84	0.87
1.3	5	0.90	0.8	0.85
1.4	1	0.87	0.81	0.84
1.4	3	0.84	0.83	0.83
1.4	5	0.88	0.8	0.84
Pi/2	1	0.86	0.82	0.84
Pi/2	3	0.82	0.79	0.8
Pi/2	5	0.86	0.83	0.84

As shown in Table 5, through the comparative experiment of ordinary method and our method, we can conclude that our method can work. Comparing our method with other method, we can find that our method has higher accuracy and F-score than oversampling technique, Park's method, and Tsung-Yi Lin's method. From the ROC curve comparison results of the three methods in Figure 9, under the same FPR, the HM loss method proposed by us can obtain higher TPR than the method proposed by Tsung-Yi Lin's and Park's, which is better than the other two methods.

According to the experimental results, we can conclude that our method works and performs better than the above related methods when dealing with the imbalanced traffic dataset.

5. Discussion

In this paper, we propose a cost-sensitive approach to improve the HTTP traffic detection performance with imbalanced data. In this approach, we design a coefficient for the loss function and when the classification algorithm predicts the minority class samples, the weight coefficient factor can dynamically adjust the contribution of the majority class samples to the loss function. When the minority class samples are predicted, the contribution of the sample to the loss function is kept unchanged. The experimental results show that this approach is more effective than others. In addition, we also present a character-level abstract feature extraction approach that can provide features with clear decision boundaries in addition. In conclusion, the methods proposed in this paper work well in imbalanced datasets. Compared to other methods, the experiment results indicate that our system has F1-score in a high precision. For imbalanced HTTP traffic detection, we confirmed that the method of feature extraction and cost function is very effective.

In our future work, we will analyze the influence of different types of autoencoders in character-level abstract feature extraction and examine their capabilities and characteristics of improving the performance, whether streamlined autoencoders can keep precision and increase computational efficiency. The theoretical causes of the results require more rigorous regulation. In addition, we will explore more about the performance of stacked autoencoder

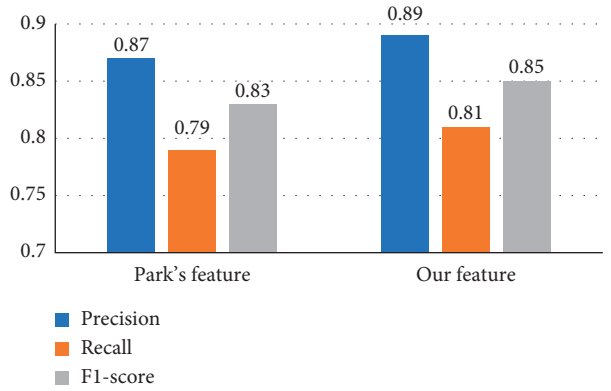

Figure 8: The performance metrics on the ordinary method and our method.

Table 5: The performance metrics on different methods.

	Precision	Recall	*F*1-score
Ordinary method	0.94	0.35	0.51
Oversampling	0.79	0.82	0.81
Park's method	0.87	0.79	0.83
Tsung-Yi Lin's method	0.85	0.83	0.84
Our method	0.9	0.84	0.87

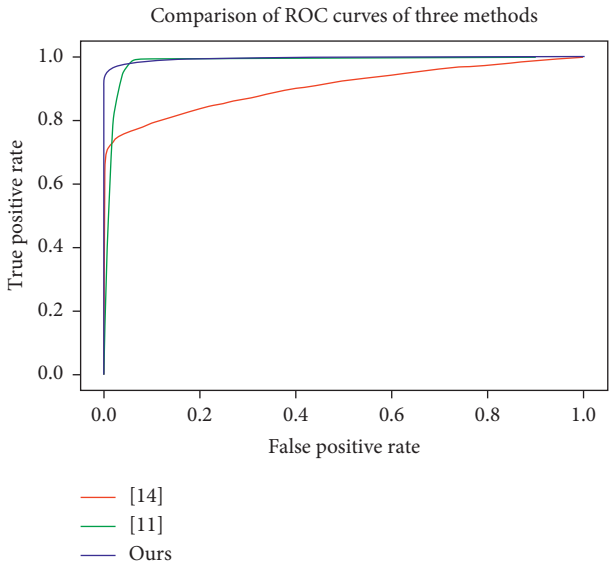

Figure 9: Comparison of ROC curves of three methods.

when extracting HTTP traffic feature, make the focal loss suitable for HTTP traffic feature, and verify that our method is feasible in other mobile communication protocols.

Data Availability

The authors cannot share their data because the data are confidential.

Conflicts of Interest

All authors declare that there are no conflicts of interest.

Authors' Contributions

Wenmin Li and Sanqi Sun made substantial contributions to the drafting of the manuscript. Wenmin Li and Shuo Zhang made contributions to the outlining and editing of the manuscript. Hua Zhang and Yijie Shi made substantial contribution in the revision process. Wenmin Li and Shuo Zhang made substantial contribution in giving final approval of the submitted version and the revised version to be submitted.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) (grant nos. 62072051, 61671082, 61672110, 61976024, 61972048, and 61802025).

References

[1] China's National Bureau of Statistics. Report on the 70th Anniversary of the Founding of the People's Republic of China[EB/OL]. http://www.stats.gov.cn/tjsj/zxfb/201908/t20190813_1690833.html,2019-08-13.

[2] 360 Internet Security Center. China Internet security report for the first half of 2018 [EB/OL]. http://zt.360.cn/1101061855.php?dtid=1101062360&did=491357630,2018-07-30.

[3] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, June 2014.

[5] J. Wu, Y. Yu, C. Huang, and K. Yu, "Deep multiple instance learning for image classification and auto-annotation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015.

[6] A. Rattani, R. Derakhshani, and A. Ross, *Selfie Biometrics: Advances and Challenges*, Springer International Publishing, Berlin, Germany, 2019.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Springer, Cham, Switzerland, October 2015.

[8] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," in *Proceedings of ACM SIGCOMM'07*, pp. 7–16, Kyoto, Japan, August 2007.

[9] Y. Jin, N. Duffield, J. Erman, P. Haffner, and S. Sen, "A modular machine learning system for flow-level traffic classification in large networks," *The ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, p. 4, 2012.

[10] Y. Lim, H. Kim, J. Jeong, C. K. Kim, and T. T. Kwon, "Internet traffic classification demystified: on the sources of the discriminative power," in *Proceedings of the 2010 ACM International Conference on Emerging Networking Experiments and Technologies*, p. 9, Philadelphia, PA, USA, November 2010.

[11] S. Park, M. Kim, and S. Lee, "Anomaly detection for http using convolutional autoencoders," *IEEE Access*, vol. 6, pp. 70884–70901, 2018.

[12] K. M. Ting, "An instance-weighting method to induce cost-sensitive trees," *IEEE Trans. Knowl. Data Eng.*vol. 14, no. 3, pp. 659–665, 2002.

[13] Z. Chen, Q. Yan, H. Han et al., "Machine learning based mobile malware detection using highly imbalanced network traffic," *Information Sciences*, vol. 433-434, no. 3, 2017.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," arXiv preprint arXiv: 1708.02002, 2017.

[15] V. Tong, "A novel QUIC traffic classifier based on convolutional neural networks," in *Proceedings of 2018 IEEE Global Communications Conference IEEE*, Abu Dhabi, UAE, December 2018.

[16] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, pp. 106944.1–106944.12, 2019.

[17] L. Mohammad, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[18] G. Bovenzi, "A hierarchical hybrid intrusion detection approach in IoT scenarios," in *Proceedings of IEEE Globecom 2020*, IEEE, Waikoloa, HI, USA, December 2020.

[19] G. Aceto, "Know your big data trade-offs when classifying encrypted mobile traffic with deep learning," in *Proceedings of 2019 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, Paris, France, June 2019.

[20] M. Zaharia, M. Chowdhury, T. Das et al., "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association*, Berkeley, CA, USA, April 2012.

[21] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces," *NSDI*, vol. 10, p. 14, 2010.

[22] H. Zhang, H. Guan, H. Yan et al., "Webshell traffic detection with character-level features based on deep learning," *IEEE Access*, vol. 6, pp. 75268–75277, 2018.

[23] The Apache Software Foundation, Apache Hadoop[EB/OL]. http://hadoop.apache.org, 2019.

[24] J. Kreps, N. Narkhede, and J. Rao, "Kafka: a distributed messaging system for log processing," in *Proceedings of the 6th International Workshop on Networking Meets Databases*, Athens, Greece, June, 2011.

[25] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," *CVPR*, vol. 2, p. 5, 2016.

[26] Q. Song, Y. Guo, and M. Shepperd, "A comprehensive investigation of the role of imbalanced learning for software defect prediction," *IEEE Transactions on Software Engineering*, vol. 99, 2018.

[27] A. J. Vickers and E. B. Elkin, "Decision curve analysis: a novel method for evaluating prediction models," *Medical Decision Making*, vol. 26, no. 6, pp. 565–574, 2006.

[28] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.