Hindawi
Security and Communication Networks
Volume 2023, Article ID 4374385, 16 pages
https://doi.org/10.1155/2023/4374385

Research Article

Network Traffic Classification Based on SD Sampling and Hierarchical Ensemble Learning

Jian Qin,[1,2] Xueying Han,[1,2] Chonghua Wang,[3] Qing Hu,[4] Bo Jiang,[1,2] Chen Zhang,[1,2] and Zhigang Lu[1,2]

[1]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China*
[3]*China Industrial Control Systems Cyber Emergency Response Team, Beijing 100040, China*
[4]*Information Center of China North Industries Group Corporation, Beijing, China*

Correspondence should be addressed to Chonghua Wang; chonghuaw@live.com and Chen Zhang; zchen@iie.ac.cn

Received 8 August 2022; Revised 2 November 2022; Accepted 12 November 2022; Published 17 May 2023

Academic Editor: Zhiyuan Tan

With the increase in cyber threats in recent years, there have been more forms of demand for network security protection measures. Network traffic classification technology is used to adapt to the dynamic threat environment. However, network traffic has a natural unbalanced class distribution problem, and the single model leads to the low accuracy and high false-positive rate of the traditional detection model. Given the above two problems, this paper proposes a new dataset balancing method named SD sampling based on the SMOTE algorithm. Different from the SMOTE algorithm, this method divides the sample into two types that are easy and difficult to classify and only balances the difficult-to-classify sample, which not only overcomes the SMOTE's overgeneralization but also combines the idea of oversampling and undersampling. In addition, a two-layer structure combined with XGBoost and the random forest is proposed for multiclassification of anomalous traffic, since using a hierarchical structure can better classify minority abnormal traffic. This paper conducts experiments on the CICIDS2017 dataset. The results show that the classification accuracy of the proposed model is more than 99.70% and that the false-positive rate is less than 0.34%, indicating that the proposed model is better than traditional models.

1. Introduction

1.1. Background. In recent years, with the rapid popularization of computer network applications in various fields, network threats have become increasingly serious. Many mechanisms, such as firewalls, antivirus, antimalware, and spam filters, are used as tools to protect network security. Network traffic classification is also an effective and powerful network security technique. However, today's cyberattacks are systematic and long-term. In addition, the traffic data in the network are so large and complex, making them difficult to analyze and detect. Machine learning (ML) [1] methods are widely used for network traffic detection.

Machine learning can identify abnormal traffic by learning features in a large amount of data. It can be divided into supervised learning and unsupervised learning. Supervised learning refers to learning labeled training data to discover relationships between input and output data for prediction and classification, including deep neural network (DNN) [2], decision tree (DT) [3], support vector machine (SVM) [4], K nearest neighbor (KNN) [5], and Gaussian naive Bayes (Gaussian NB) [6]. Unsupervised learning refers to learning and summarizing patterns and structures of unlabeled training data for prediction and classification, including principal component analysis (PCA) [7] and K-means clustering [8].

1.2. Problem Statement. Network traffic has a natural unbalanced class distribution problem. For example, there is far more normal traffic than abnormal traffic in the network. To solve the problem of the unbalanced dataset, the traditional method is to sample the dataset. SMOTE [9] is the widely used method of sampling data based on the spatial distribution of samples, and its process is simple but has many disadvantages, such as overgeneralization. To deal with these disadvantages of the SMOTE algorithm, this paper proposes a new sampling algorithm named SD sampling, which can obtain a dataset that is easier to classify.

Traditional network intrusion detection based on machine learning usually uses a single algorithm to classify traffic, which leads to many problems. First of all, single machine learning algorithms have some limitations, such as being easy to overfit or underfit and difficult to deal with multiclassification problems, which lead to a low detection rate. In addition, some single algorithms can only guarantee a high detection rate for the dataset with specific data distribution, resulting in poor generalization ability. Therefore, this paper proposes a new detection structure, which contains two ensemble models, which can ensure a high detection rate and improve the generalization ability of the model.

1.3. Key Contributions and Paper Organization. In summary, the paper's main contributions are as follows:

(1) We propose a new sampling algorithm named SD sampling. The SD sampling algorithm combines oversampling and undersampling methods and considers the spatial distribution of samples during sampling, which overcomes the overgeneralization problem of the SMOTE algorithm to some extent.
(2) We propose a two-layer structure combined with XGBoost [10] and the random forest [11] to realize multiclassification of traffic, which improves the detection rate and generalization ability of the model.
(3) We evaluate the performance of the SD sampling algorithm and the proposed classification model using the CICIDS2017 dataset [12]. Compared with other sampling modes and classification models, we verify the advantages of the proposed method.

The remainder of the paper is organized as follows: Section 2 introduces the research of ML-based intrusion detection and unbalanced datasets. Section 3 introduces the framework of the proposed model and details each module, including the workflow of the SD sampling algorithm and the two-layer structure combined with XGBoost and the random forest. In Section 4, we evaluate the performance of the SD sampling algorithm and the proposed detection model using the CICIDS2017 dataset. Finally, Section 5 summarizes and discusses future directions.

2. Related Work

2.1. Improved Machine Learning Algorithms for Network Traffic Classification. Machine learning is widely used in network intrusion detection. However, traditional machine learning models have the problem of low accuracy, which can be solved by the following methods: ensemble learning, model optimization, and sample optimization. For the reader's convenience, we provide the explanation of acronyms, as shown in Table 1.

In terms of ensemble learning, Gao et al. [13] proposed a new method of ensemble voting based on classifier resolution. For each base classifier, a classification prediction is made with a base classifier, and the probability of correctly classifying samples into each class is calculated, which is recorded as the classification weights of the base classifier. When voting, the weights of the corresponding classes of base classifiers with the same classification result are added, and then, the sum of the weights of each class is compared. The class with higher weights is taken as the final result. Xia and Sun [14] proposed an ensemble learning scheme using isolated forest (IForest) [15], local outlier factor (LOF), and K-means clustering methods. The base classifiers are IForest and LOF, making them complementary in detecting global outliers and local outliers. As for the selection of the k-means initial clustering center, the normal points detected by the IForest and LOF can be selected as the initial clustering center to solve the problem of poor clustering effects if the initial clustering center contains outliers. The experimental results show that the accuracy of the model has improved significantly. Ling et al. [16] proposed a multiclassifier ensemble algorithm based on probability weighted voting to improve model accuracy. Xu et al. [17] proposed a weighted majority algorithm based on the random forest to improve the performance of the random forest, and the model is trained on nontraffic datasets, so it has the ability to detect unknown traffic types. Ren et al. [18] proposed category detection and a partition technique to improve the detection accuracy of minority attacks (Probe, U2L, R2L) on the random forest. Data and Aritsugi [19] proposed an incremental learning framework, which can avoid the problem of conceptual drift in traffic. Aceto et al. [20] proposed an encrypted traffic classification framework based on hard/soft combinators, which takes the existing high performance traffic classification model as a base classifier and considers the training requirements and learning philosophy for improving classification performance. Possebon et al. [21] investigated and evaluated a wide range of metalearning techniques, including voting, stacking, bagging, and boosting.

In terms of model optimization, Yang and Zhang [22] proposed the use of a multigranularity cascade algorithm based on the traditional isolated forest model. The traditional isolated forest has some problems, such as undetectable local outliers parallel to the axis and a lack of sensitivity and stability to high-dimensional data outliers. To solve these problems, an isolation mechanism based on a stochastic hyperplane is proposed. The stochastic hyperplane simplifies the isolation boundary of the data model by using the linear combination of multiple dimensions, and the isolation boundary of the stochastic linear classifier can detect more complex data patterns. Experiments show that the improved isolated forest algorithm has better robustness to complex anomaly data patterns. Qiu et al. [23] used an LSTM model with a sliding window to avoid the problem of concept drift in streaming data. Giuseppe et al. [24] proposed a novel multimodal data allocation framework

Table 1: List of the acronyms used in the manuscript.

Acronym	Definition
SD sampling	Difficult set sampling
SMOTE	Synthetic minority oversampling technique
XGBoost	Extreme gradient boosting
ML	Machine learning
DNN	Deep neural network
DT	Decision tree
SVM	Support vector machine
KNN	*K* nearest neighbor
GNB	Gaussian naive Bayes
PCA	Principal component analysis
IForest	Isolated forest
LOF	Local outlier factor
U2L	Unix to Linux
R2L	Remote to local
LSTM	Long short-term memory
MIMETIC	Mobile encrypted traffic classification using multimodal deep learning
wGAN-GP	Gradient penalty Wasserstein generative adversarial networks
AUC	Area under the curve
TGAN	Tabular gan
HM-loss	A cost-sensitive method for loss calculation

Table 2: Related works of improved machine learning algorithms.

Detailed method	Literature	Description	Dataset	Best accuracy (%)
Ensemble learning	[13]	Ensemble voting based on classifier resolution and a multitree ensemble model	KDDTest+	85.20
	[14]	An AdaBoost model combining IForest, LOF, *K*-means algorithms	Datasets in UCI machine learning library	96.29
	[16]	A multiclassifier ensemble algorithm based on probability weighted voting	NSL-KDD	95.70
	[17]	A weighted majority algorithm based on the random forest	NSL-KDD	90.48
	[18]	A multilayer random forest model based on category detection and a partition technique	KDD Cup 1999	94.36
	[19]	An incremental learning framework	CICIDS2017	86.70
	[20]	A framework based on hard/soft combinators	Real traffic data	79.20
	[21]	Investigate and evaluate voting, stacking, bagging, boosting ensemble frameworks	Datasets in UCI machine learning library	99.97
Model optimization	[22]	A multidimensional stochastic hyperplane isolation method	Personal real data	AUC = 100.00
	[23]	The stacked LSTM model combines the idea of sliding windows	Personal real data	AUC = 91.55
	[24]	A multimodal data allocation framework MIMETIC	Real traffic data	96.74
Sample optimization	[25]	Naive Bayes feature embedding method	UNSW-NB15/CICIDS2017	93.75/98.92
	[18]	An outlier detection algorithm based on KNN	KDD Cup 1999	94.36

MIMETIC for encrypted traffic classification, which can fully exploit the heterogeneity of traffic data by learning intramodal and intermodal dependencies and overcome the performance limitations of single-modal data.

In terms of sample optimization, Gu and Lu [25] proposed the method of feature transformation of data by using naive Bayes feature embedding. The dataset and kernel density estimates are calculated, and then, the marginal density ratios of each feature of the sample are calculated using the naive Bayes' principle. Taking the marginal density ratio of each feature as a new feature, which makes the dataset easier to classify, Ren et al. [18] proposed an outlier detection algorithm based on KNN, which removes some outliers to help the model classify traffic more easily.

We summarize all of the above work of improved machine learning algorithms, as shown in Table 2.

2.2. Balanced Dataset. There are two approaches to solving the problem of the unbalanced dataset, the first is from the perspective of the data and the second is from the perspective of the algorithm.

In terms of data, Zhang et al. [26] proposed a method to generate samples based on the variational autoencoder generation model to balance the dataset. The core idea is to expand only boundary samples, which are most likely to cause confusion to machine learning when expanding minority samples. Liu et al. [27] proposed the method of using wGAN-GP, an improved method of the generative adversity network, to generate a small number of samples and balance the dataset. Yan and Han [28] improved the SMOTE algorithm and put forward three-point domains that are divided according to the number of majority samples around samples to generate samples. Seo and Kim [29] proposed a support vector machine model to predict the optimal sampling rate of the SMOTE algorithm and then get an optimal sampling dataset. Wang and Sun [30] put forward a new sampling method, which takes into account the problems of class overlap and data distribution lacking in the traditional oversampling method. Compared with the traditional SMOTE algorithm, the AUC for four datasets increases by 1.6% on average. Liu et al. [31] proposed a technique for sampling samples based on the difficulty of sample classification. Park and Hyunhee [32] proposed a method combining TGAN and slow start to generate samples and prevent overfitting caused by oversampling. Wang et al. [33] proposed an encrypted traffic generation method based on GAN to generate minority class samples and balance this dataset.

In terms of algorithm, Gupta et al. [34] proposed a method to weight samples and use cost-sensitive DNN for classification, to reduce the impact of the unbalanced dataset. Sharma et al. [35] proposed a weighted extreme learning machine to weight each classifier and alleviate dataset imbalance. Li et al. [36] proposed a method called HM-loss cost, which pays more attention to the misclassified samples in minority classes when calculating the loss. Hu et al. [37] proposed a method of batch balancing datasets based on deep learning to ensure that the number of samples in each class is equal in each batch, so as to achieve the balance of datasets.

We summarize all of the above work of balanced datasets, as shown in Table 3.

3. Proposed Model

In this section, we introduce the proposed framework and its workflow. The framework runs through the overall process of network traffic detection and has a high detection rate for various kinds of traffic. As shown in Figure 1, the framework consists of four main modules as follows:

(1) Preprocessing module: raw data are preprocessed to make the processed data easier to understand and process.

(2) Data sampling module: the number of samples in each class is counted, and data are balanced using the proposed SD sampling algorithm.

(3) Feature selection module: high-weight features are selected from all features using the random forest, which will facilitate training processing.

(4) Classification module: A two-layer structure combined with XGBoost and the random forest is used to classify traffic.

3.1. Preprocessing Module.

This module focuses on preprocessing raw data. First, missing and duplicate values are deleted from the data. Second, the data are undersampled to obtain a portion of the data for model training. Then, the dataset is divided into the train set and the test set. Finally, the train set and the test set are normalized and one-hot encoded.

3.2. Data Sampling Module.

This module focuses on balancing classes using the sampling algorithm. The SMOTE algorithm is a commonly used class balancing algorithm. However, the SMOTE algorithm has some limitations, so this paper proposes a new sampling algorithm SD sampling based on SMOTE, which can mitigate the defects of the SMOTE algorithm to some extent and get better results. The specific process of the SMOTE algorithm and SD sampling algorithm is introduced in the following sections.

3.2.1. SMOTE: Synthetic Minority Oversampling Technique.

SMOTE is an oversampling algorithm, which can automatically calculate the ratio of the majority class sample to the minority class sample and oversample the samples of the minority class based on the distance metric. Its sampling strategy is to choose one sample randomly among k nearest neighbors of the sample of each minority class and then select a random point on the line between these two samples as the newly synthesized minority class sample.

The specific process is as follows:

(1) For each sample x in a minority class, we calculate the Euclidean distance of all samples in that class and obtain the k nearest samples of that sample by comparison.

(2) We selecting a sample randomly from k nearest samples and denote it as x_{neighbor}.

(3) We generate the new sample x_{new}. The new sample x_{new} is

$$x_{\text{new}} = x + \text{rand}(0,1) * \left(x_{\text{neighbor}} - x\right). \quad (1)$$

In the formula, $\text{rand}(a,b)$ represents generating a random number between a and b.

The SMOTE algorithm process is concise and has been widely used, but it has several disadvantages:

(1) The algorithm is prone to the problem of overgeneralization. Minority classes are sparser than majority classes, so there is a high chance of class mixing when sampling a minority class, which will make the boundary of them more and more blurred and increase the difficulty of classification.

(2) The algorithm simply oversamples minority class samples indiscriminately and does not consider its

Table 3: Related works of balanced datasets.

Detailed method	Literature	Description	Dataset	Best accuracy (%)
Data	[26]	The variational autoencoder generates samples	UNSW-NB15	96.13
	[27]	The wGAN-GP method generates samples	NSL-KDD/UNSW-NB15/CICIDS2017	86.69/94.90/99.84
	[28]	A three-point domain sample generation method based on the SMOTE algorithm	NSL-KDD	99.00
	[29]	The SVR model predicts SMOTE sampling proportion	KDD Cup 1999	98.10
	[30]	The SMOTE algorithm combining clustering and instance hardness	DoHBrw2020/CIC_Bot/CIC_Inf/DOS2017/UNSW/Botnet2014	AUC = 89.60/90.21/92.09/75.92/93.19/73.81
	[31]	A technique for sampling samples based on the difficulty of sample classification	NSL-KDD/CICIDS2018	82.84/96.99
	[32]	A method combining TGAN and slow start	KDD cCup 1999	93.98
	[33]	An encrypted traffic generation method based on GAN	ISCX	99.10
Algorithm	[34]	A cost-sensitive deep neural network	NSL-KDD/CIDDS-001/CICIDS2017	92.00/99.00/92.00
	[35]	A method of weighted extreme learning machine	UNSW-NB15/KDD cup 1999	96.12/99.71
	[36]	The HM-loss cost method	Personal real data	$F1 = 87.00$
	[37]	A method of batch balancing datasets based on deep learning	CHB-MIT/BonnEEG/FAHXJU	95.96/100.00/87.93

spatial distribution. Therefore, the sampling algorithm will not be targeted.

(3) The algorithm can only oversample minority class samples and does not perform any processing on majority class samples.

3.2.2. SD Sampling. Based on the above limitations of the SMOTE algorithm, a new sampling algorithm called SD sampling is proposed in this paper. SD sampling refers to literature [31] and makes improvements on its basis. The standard of the literature's algorithm for selecting a difficult classified dataset is too strict, and the method for sampling minority classes is too simple. In response to these problems, we propose the concept of instance hardness (IH), which makes the selection criteria more flexible, and use the SMOTE algorithm to oversample minority classes. In addition, the SD sampling algorithm combines the ideas of oversampling and undersampling and takes into account the spatial distribution characteristics of each sample so that it also mitigates the disadvantages of SMOTE to some extent.

The SD sampling algorithm starts from the fact that the SMOTE algorithm cannot deal with the data distribution problem of the unbalanced dataset. As shown in Figure 2, samples are first divided into two parts according to their spatial distribution, one is an easily classified dataset, denoted as SE, and the other is a difficult-to-be classified dataset, denoted as SD. We proposed a concept called instance hardness (IH) as a criterion to assess whether the sample is easy to classify. The higher the IH, the harder it is to classify sample x. IH is calculated as

$$\mathrm{IH} = \frac{\sum_{\text{neighbor}=1}^{k}\left(\text{label}\left(x_{\text{neighbor}}\right)\right)! = \text{label}\,(x)}{k}, \tag{2}$$

where the label(x) represents the label of x, x_{neighbor} represents the near neighbor sample of x, and k represents the number of neighbors of the sample x.

Keeping SE constant, for SD, since most samples in the majority class are redundant, K-means clustering is performed for those majority class samples, and then, cluster centers are used to replace samples in the cluster. In addition, SMOTE oversampling is performed for those minority class samples in SD. The SDsampling algorithm is written as Algorithm 1.

3.3. Feature Selection Module. This module focuses on selecting features. It is difficult for machine learning algorithms to learn from high-dimensional data. Feature selection is a useful method to solve these problems, and it selects features with high weights for training in advance, which can improve performance and save computational resources at the same time.

This module uses a random forest-based feature selection method, which can evaluate the weight of each feature

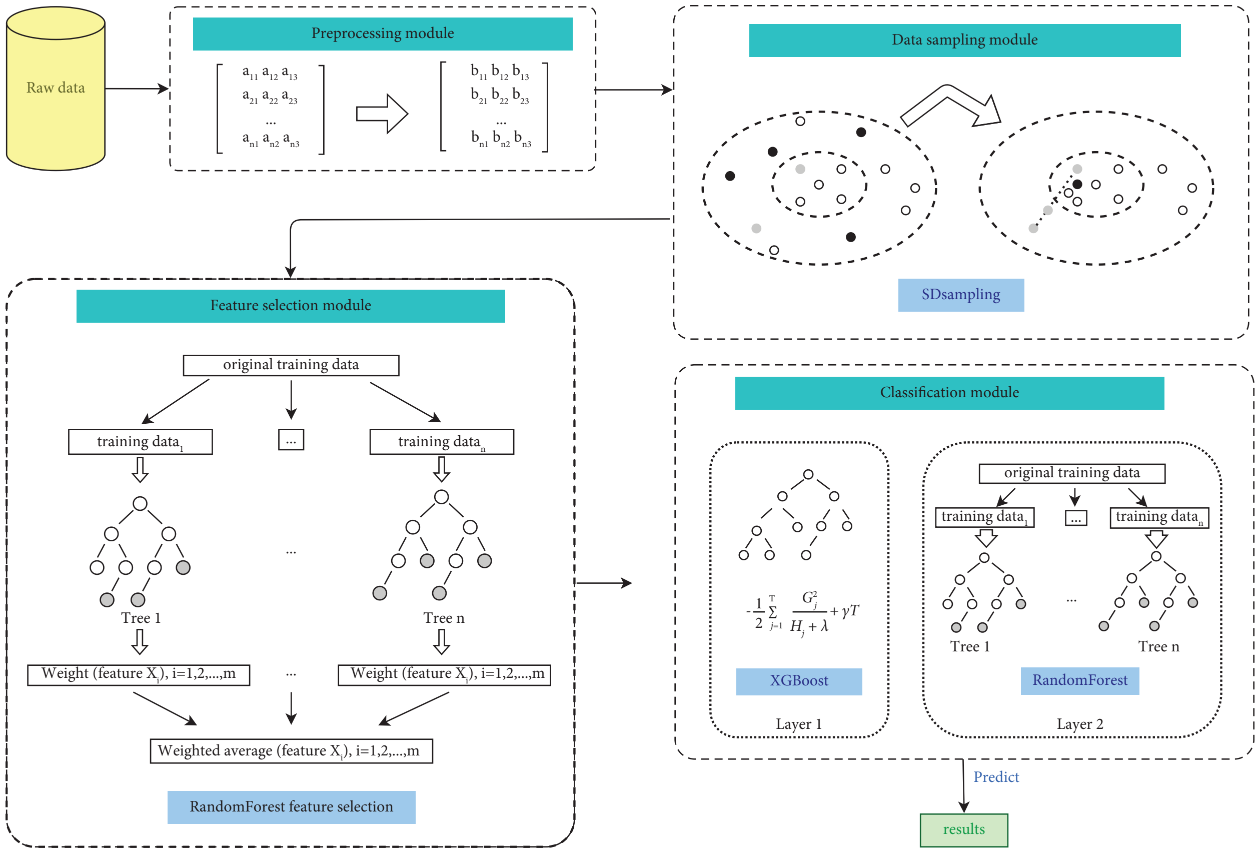

Figure 1: Proposed framework.

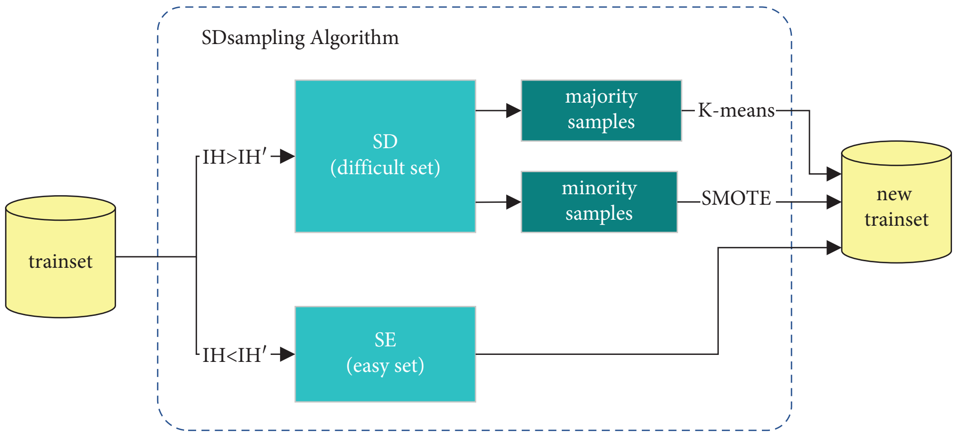

Figure 2: SD sampling algorithm.

by the Gini index. The random forest is composed of many CART trees [38], and its final classification result is decided by these CART trees through voting. The Gini index describes purity, and the smaller the value, the higher the purity. Therefore, in CART trees, the Gini index is used as an assessment of the change in the purity of nodes before and after using feature splitting nodes, and the smaller the value, the better the feature. For the sample set D, suppose there are K classes, the sample size of the k^{th} class is $|C_k|$ and the size of D is $|D|$, and then, the Gini index expression for the sample set D is

$$\operatorname{Gini}(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|}\right)^2. \tag{3}$$

For a CART tree, the number of samples in a node q is N_q, and the number of samples in a node q with a class k is N_{qk}. The Gini index of the node q of the i^{th} tree is

$$\mathrm{Gini}_q^i = 1 - \sum_{k=1}^{K} \left(p_{qk}^i\right)^2, \tag{4}$$

where p_{qk} is the proportion of the class k in the node q, calculated as N_{qk}/N_q.

For the feature X, the importance of the jth feature at the ith tree node q, that is, the amount of change in the Gini index before and after splitting the node q, is

$$\Delta\mathrm{Gini}_{jq}^i = \mathrm{Gini}_q^i - \mathrm{Gini}_l^i - \mathrm{Gini}_r^i, \tag{5}$$

where Gini_l^i and Gini_r^i denote the Gini index of two new nodes after branching.

The set of nodes in which the feature X_j appears in the ith tree is Q, and then, the importance of the feature X_j in the ith tree is

$$\mathrm{Gini}_j^i = \sum_{q\in Q} \Delta\mathrm{Gini}_{jq}^i. \tag{6}$$

There are I trees in the random forest, and feature importance is denoted as FI. Then, the importance of the feature X_j is

$$FI_j = \sum_{i}^{I} \mathrm{Gini}_j^i. \tag{7}$$

Finally, all the obtained importance scores are normalized and calculated as

$$\mathrm{FI}_j^{\mathrm{normalized}} = \frac{\mathrm{FI}_j}{\sum_{j'=1}^{J} \mathrm{FI}_{j'}}, \tag{8}$$

where J is the number of features.

Finally, top i features with the highest weight are selected for classification.

3.4. Classification Module. This module focuses on classifying traffic. We propose a two-layer structure combined with XGBoost and the random forest. The first layer uses the XGBoost model and distinguishes between normal and abnormal samples in the dataset and the second layer uses the random forest model to distinguish the type of attack for each abnormal sample.

XGBoost is a boosted tree model, which is a combination of many tree models together to form a very powerful integrated classifier. The idea of XGBoost is to train K trees, and the final prediction result is the sum of the predicted values of those K trees. It is an improvement on the gradient boosting algorithm, which can get a high accuracy rate in a very short time.

The random forest is an ensemble model that uses many decision trees to classify samples for prediction and finally votes on the classification result. The randomness of the random forest is reflected in random and unreleased data sampling and random feature selection, which leads to faster training speeds and higher accuracy.

The two-layer structure combined with XGBoost and the random forest uses a hierarchical approach to multiclassify traffic, and the hierarchical approach refers to two pieces of literature [39, 40] on traffic multiclassification. However, different from both of them, this structure is mainly used to identify abnormal traffic types rather than application traffic types. Its workflow is shown in Figure 3, which mainly includes three steps as follows:

(1) Dataset construction: The train set and the test set are replicated into two copies, and labels are recoded. For the train set and test set, the first part, denoted as $\mathrm{trainset}_1$ and $\mathrm{testset}_1$, is labeled with all normal samples as 0 and abnormal samples as 1 using $\mathrm{RecodeMethod}_1$.

$$\begin{aligned}\mathrm{trainset}_1 &= \mathrm{RecodeMethod}_1(\mathrm{trainset}),\\ \mathrm{testset}_1 &= \mathrm{RecodeMethod}_1(\mathrm{testset}).\end{aligned} \tag{9}$$

The second part, denoted as $\mathrm{trainset}_2$ and $\mathrm{testset}_2$, is labeled with all normal samples as 0 and abnormal samples sequentially coded as $1, 2, 3, \ldots, m$ according to categories using $\mathrm{RecodeMethod}_2$, where m is the number of abnormal categories.

$$\begin{aligned}\mathrm{trainset}_2 &= \mathrm{RecodeMethod}_2(\mathrm{trainset}),\\ \mathrm{testset}_2 &= \mathrm{RecodeMethod}_2(\mathrm{testset}).\end{aligned} \tag{10}$$

(2) Model training: The XGBoost model is used to train on $\mathrm{trainset}_1$ to obtain a binary classifier, denoted as clf_1, and the random forest model is used to train on $\mathrm{trainset}_2$ to obtain a multiclass classifier that can distinguish the types of abnormal traffic, denoted as clf_2.

(3) Classification: The trained classifier clf_1 is used to classify $\mathrm{testset}_1$, and the samples classified as normal are noted as data_0 and those classified as abnormal are noted as data_1.

$$\mathrm{data}_0, \mathrm{data}_1 = \mathrm{clf}_1(\mathit{testset}_1). \tag{11}$$

We select the data contained in data_1 from $\mathrm{testset}_2$. Then, the trained classifier clf_2 is used to classify and predict $\mathrm{testset}_2$, and the samples classified as normal are noted as added to data_0 and those classified as abnormal are noted as data_1, data_2, ..., data_m by class.

$$\mathrm{data}_0, \mathrm{data}_1, \ldots, \mathrm{data}_m = \mathrm{clf}_2(\mathrm{testset}_2). \tag{12}$$

In this way, the result of multiclassification of traffic is obtained.

4. Experiments

In this section, we evaluate the classification performance of the framework through experiments. First, we introduce the experimental environment, the CICIDS2017 dataset, and evaluation metrics in detail. Then, we adjust some of the parameters used in the proposed framework and finally compare and analyze the classification results.

4.1. Experimental Environment. The details of all experimental implementation configurations are shown in Table 4.

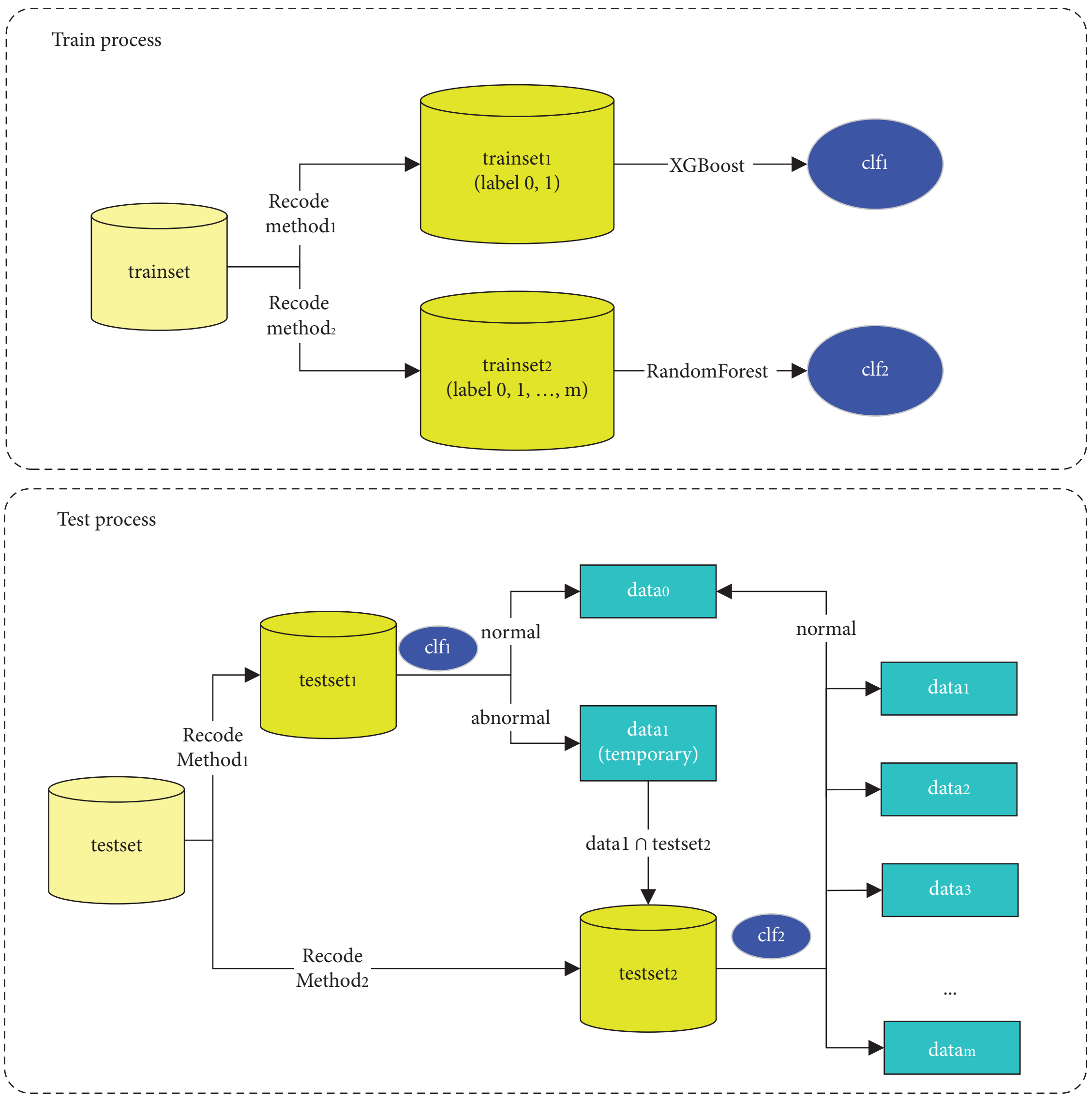

Figure 3: Workflow of the two-layer structure combined with XGBoost and the random forest.

4.2. Dataset. The CICIDS2017 dataset is a widely used dataset collected by the Canadian Institute for Cybersecurity in 2017. It contains both normal and abnormal traffic, and it is generated by simulating in a real network environment, making it closer to the realistic situation and more reliable.

The CICIDS2017 dataset provides the original pcap package, and we extracted the statistical features from them. Since the transmission content is mostly encrypted, semantic features are difficult to obtain from traffic. However, the statistical distribution of normal traffic packets and abnormal traffic packets in a session is different, such as the number and length of packets. Therefore, we use statistical features instead of original traffic for classification in this paper. Note that all the packets of flow/biflow need to be collected at the end of the session, and then, the statistical features of the flow can be calculated.

At the sample size level, the CICIDS2017 dataset contains a total of 2,830,743 records, including 2,273,097 records for normal traffic and 557,646 records for abnormal traffic, which are extremely unbalanced. Therefore, after performing regular operations (such as deleting missing values, normalizing data, and encoding labels), we undersample the CICIDS2017 dataset, which not only mitigates the negative impact of the unbalanced dataset but also reduces the training time. The category distribution of the processed dataset is shown in Figure 4.

At the feature size level, the CICIDS2017 dataset is a high-dimensional dataset, which contains 84 feature columns and 1 label column. In order to improve the generalization ability of the model, we remove the "Flow ID," "Source IP," "Source Port," "Destination IP," and "Time stamp" features. As a result, the final dataset contains only 79

```
     Input: Imbalanced train set $S$, scaling factor $K$, instance hardness threshold $IH'$, and sample threshold UB
     Output: New train set $S_N$
(1)  Step1: Distinguish between easy sets and difficult sets for each sample$\in S$ do
(2)    Compute its $K$ nearest neighbors and IH if $IH > IH'$ then
(3)      Put the samples into the difficult set
(4)    end
(5)  end
(6)  Difficult set $S_D$ and easy set $S_E = S - S_D$
(7)  Step2: Compress the majority samples in the difficult set by the cluster centroid
(8)  Take all the majority samples from $S_D$ and set it as $S_{Maj}$
(9)  Use the $K$-means algorithm with $K$ cluster
(10) Use the coordinates of $K$ cluster centroids and replace the majority samples in $S_{Maj}$
(11) Compressed the majority sample set $S_{Maj}$
(12) Step3: Sample the minority samples in the difficult set using the SMOTE algorithm
(13) Take all the majority samples from $S_D$ and set it as $S_{Min}$
(14) for each sample $\in S_{Min}$ do
(15)   Using SMOTE sampling, the sampling threshold is set to UB
(16)   Putting new samples into $S_Z$
(17) end
(18) Step4: Merge sample sets
(19) New train set $S_N = S_E + S_{Maj} + S_{Min} + S_Z$
```

Algorithm 1: SDsampling Algorithm.

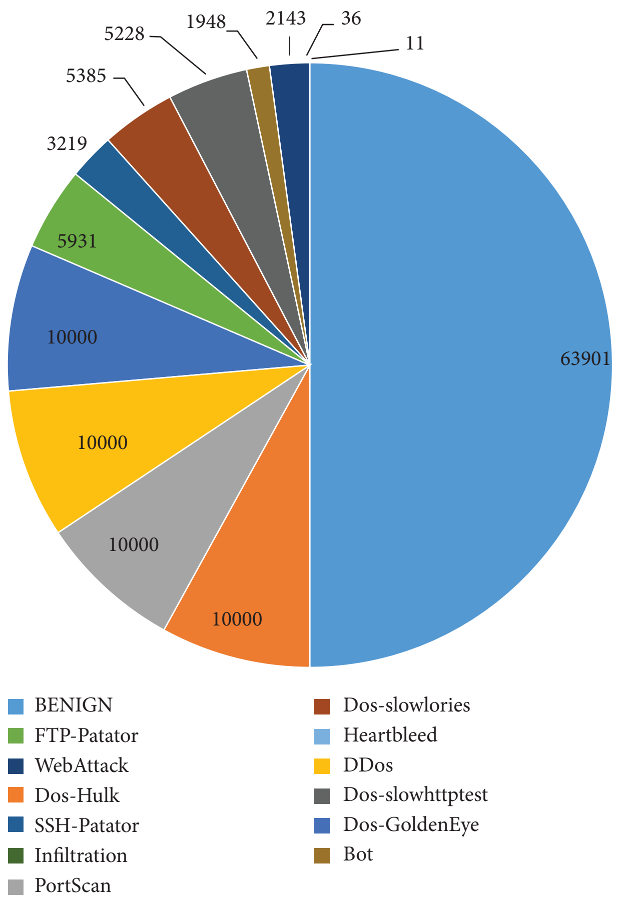

Figure 4: Undersampled CICIDS2017 dataset.

feature columns and 1 label column. These features are listed and classified in Table 5.

Finally, the dataset is divided into a train set and a test set in a ratio of 3 : 1.

4.3. Evaluation Metrics and Baseline Methods.

In order to accurately evaluate the two-layer structure combined with XGBoost and the random forest, we use the six evaluation criteria: accuracy, recall, precision, $F1$ score, false-negative rate (FNR), and false-positive rate (FPR).

As shown in Table 6, we first calculate the confusion matrix according to real labels and predicted labels. True positive (TP) refers to the sample with both positive real values and predicted values, whereas false positive (FP) refers to the sample with negative real values and positive predicted values. False negative (FN) refers to the sample with positive real values and negative predicted values, whereas true negative (TN) refers to the sample with both negative true values and predicted values.

Accuracy refers to the percentage of correctly predicted samples in the total sample, which can represent the overall predictive ability of the model as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \quad (13)$$

Recall is the ratio of the number of samples that are predicted to be positive to the number of samples with positive real values, which can represent the coverage rate of prediction as follows:

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (14)$$

Table 4: Experimental environment.

Software/hardware	Details
OS	Windows 10
CPU	Intel (R) Core (TM) i7-8750H CPU @2.20 GHz
Memory	8.0 GB
Disk	1.0 TB
Python	3.6.9
Framework	Sklearn 0.24.2 + torch 1.10.0

Table 5: The feature set of CICIDS2017.

Category	Feature name	Count
Flow label	Destination port, protocol, flow duration	3
IAT	Flow IAT mean, flow IAT std, flow IAT max, flow IAT min, fwd IAT total, fwd IAT mean, fwd IAT std, fwd IAT max, fwd IAT min, bwd IAT total, bwd IAT mean, bwd IAT std, bwd IAT max, bwd IAT min	14
Forward/backward traffic packets	Total fwd packets, total backward packets, total length of fwd packets, total length of bwd packets, fwd packet length max, fwd packet length min, fwd packet length mean, fwd packet length std, bwd packet length max, bwd packet length min, bwd packet length mean, bwd packet length std, min packet length, max packet length, packet length mean, packet length std, fwd header length, bwd header length, packet length variance, average packet size, avg fwd segment size, avg bwd segment size, fwd header length, subflow fwd packets, subflow fwd bytes, subflow bwd packets, subflow bwd bytes, Init_Win_bytes_forward, Init_Win_bytes_backward, act_data_pkt_fwd, min_seg_size_forward	31
Flags	Fwd PSH flags, bwd PSH flags, fwd URG flags, bwd URG flags, FIN flag count, SYN flag count, RST flag count, PSH flag count, ACK flag count, URG flag count, CWE flag count, ECE flag count	12
Flow rate	Flow bytes/s, flow packets/s, fwd packets/s, bwd packets/s, fwd avg bytes/bulk, fwd avg packets/bulk, fwd avg bulk rate, bwd avg bytes/bulk, bwd avg packets/bulk, bwd avg bulk rate	10
Other	Down/up ratio, active mean, active std, active max, active min, idle mean, idle std, idle max, idle min	9

Precision is the ratio of the number of samples with positive real values to the number of samples predicted to be positive, which can represent the ability of the model to predict positive samples as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{15}$$

The $F1$ score is the harmonic mean of precision and recall and can be represented as follows:

$$F1 - \text{score} = \frac{2 \cdot \text{precision} \cdot r\text{ecall}}{\text{precision} + r\text{ecall}}. \tag{16}$$

FNR is the ratio of the number of samples with positive predicted values and negative real values to the number of samples with positive predicted values and can be represented as follows:

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} = 1 - \textit{precision}. \tag{17}$$

FPR is the ratio of the number of samples that are not predicted to be positive to the number of samples with positive real values and can be represented as follows:

$$\text{FPR} = \frac{\text{FP}}{\text{TP} + \text{FP}} = 1 - \textit{recall}. \tag{18}$$

Additionally, we also use four commonly used machine learning models including the k-nearest neighbor (KNN), decision tree (DT), support vector machine (SVM), and deep neural network (DNN) to conduct comparative experiments, and their parameters are shown in Table 7.

4.4. Parameter Selection. In order to achieve the best classification effect of the model, we select the important parameters in each module and select the optimal results to improve model accuracy.

In the feature selection module, we use a feature selection method based on the random forest. We need to select the top i features with the highest weight as the final result and set i to 10, 20, 30, 40, 50, 60, and 70 for comparative experiments. Finally, we use the random forest and XGBoost to classify the processed dataset and use the average value of the $F1$ score to select the optimal number of features.

As shown in Figure 5, the average value of the $F1$ score of the random forest and XGBoost is obtained by using the feature selection method based on the random forest, and the value is the highest when the number of features is 50. Therefore, the optimal feature number is 50.

In the classification module, for the random forest and XGBoost models used in the two-layer structure, we adjust the important parameters of the two models, respectively, and obtain the optimal results, and the selected parameters are shown in Table 8.

4.5. Classification Results and Analysis. Our experiment aims to explore the advantages of the SD sampling algorithm and two-layer structure combined with XGBoost and the random forest in binary and multiclass classification.

4.5.1. Influence of the SD Sampling Algorithm. First, we sample the CICIDS2017 dataset in four different modes: no

Table 6: Confusion matrix.

Confusion matrix		Predicted	
		1	0
Real	1	TP	FN
	0	FP	TN

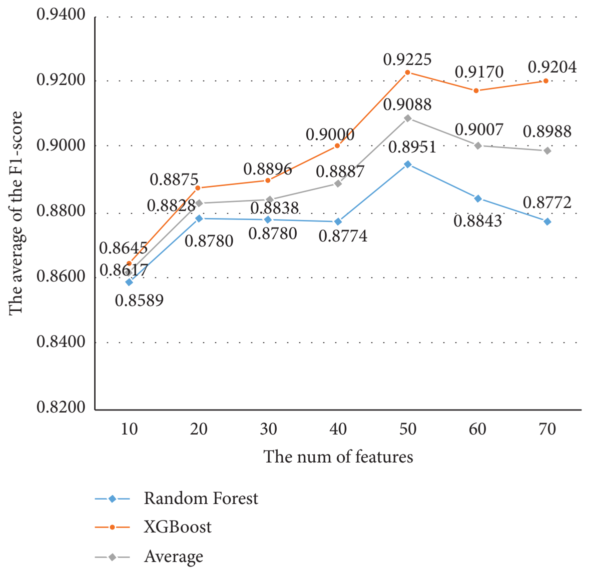

Figure 5: Feature selection based on the random forest.

Table 7: Model parameters of comparative experiments.

Model	Parameters
KNN	n_neighbors': 5
DT	min_samples_leaf': 3 max_features' = none criterion' = "entropy"
SVC	C': 1.0 "kernel": "rbf" "degree": 3 gamma': "scale"
DNN	Epoch': 1000 batch_size': 10000 learning_rate': 0.01

sampling, SMOTE sampling, random sampling, and SD sampling. Then, we classify the processed dataset with a two-layer structure. Finally, we evaluate the results as per the six evaluation criteria.

As shown in Tables 9–12, it is found that the classification performance of the above four sampling modes is roughly the same in the case of binary classification. However, in the case of multiclass classification, the classification ability of the model for minority samples in the dataset sampled by the SD sampling algorithm has significantly improved, and the identification ability of each attack can reach more than 99%. The reason is that the SD sampling algorithm oversamples minority samples and clusters majority samples in SD. In this way, the imbalance ratio between majority samples and minority samples will be reduced, and the sampled dataset is more conducive to the model to distinguish the abnormal flow with minority samples.

4.5.2. Influence of the Two-Layer Structure. In order to verify that the two-layer structure is better than other models, we conduct comparative experiments using six models, including the four baseline models mentioned in Section 4.3 and the single-layer XGBoost and random forest model, and the experimental results are shown in Table 13. It is found that under the same sampling mode, the six evaluation metrics of the two-layer structure are significantly higher than those of the other four models.

Table 8: Model parameters of the two-layer structure.

Model	Parameters			
	Original	SMOTE	Random sampling	SD sampling
XGBoost	n_estimators': 90 max_depth': 8 learning rate': 0.15 gamma': 0.01	"n_estimators": 80 "max_depth": 8 "learning_rate": 0.2 "gamma": 0.001	"n_estimators": 80 "max_depth": 8 "learning_rate": 0.2 "gamma": 0.001	"n_estimators": 80 "max_depth": 8 "learning_rate": 0.2 "gamma": 0.001
RF	"n_estimators": 20 "min_samples_leaf": 1 "max_features": none	"n_estimators": 60 "min_samples_leaf": 1 "max_features": "log2"	"n_estimators": 60 "min_samples_leaf": 1 "max_features": "log2"	"n_estimators": 20 "min_samples_leaf": 1 "max_features": none

Table 9: Classification performance of original datasets.

	Precision	Recall	*F*1 score	FNR	FPR
Original layer 1					
Benign	0.9992	0.9980	0.9986	0.0008	0.0020
Abnormal	0.9980	0.9992	0.9986	0.0020	0.0008
Accuracy					0.9986
Macro avg	0.9986	0.9986	0.9986	0.0014	0.0014
Weighted avg	0.9986	0.9986	0.9986	0.0014	0.0014
Original layer2					
Benign	0.9992	0.9980	0.9986	0.0008	0.0020
DoS hulk	0.9846	0.9988	0.9917	0.0154	0.0012
DDoS	1.0000	1.0000	1.0000	0.0000	0.0000
PortScan	0.9996	0.9996	0.9996	0.0004	0.0004
DoS goldeneye	1.0000	0.9988	0.9994	0.0000	0.0012
FTP-patator	0.9993	1.0000	0.9997	0.0007	0.0000
DoS slowloris	0.9948	0.9941	0.9944	0.0052	0.0059
DoS slowhttptest	0.9969	0.9969	0.9969	0.0031	0.0031
SSH-patator	1.0000	0.9988	0.9994	0.0000	0.0012
Bot	0.9979	0.9918	0.9949	0.0021	0.0082
Infiltration	1.0000	0.5556	0.7143	0.0000	0.4444
Heartbleed	1.0000	0.6667	0.8000	0.0000	0.3333
Web attack	0.9906	0.9813	0.9859	0.0094	0.0187
Accuracy					0.9978
Macro avg	0.9972	0.9369	0.9596	0.0028	0.0631
Weighted avg	0.9978	0.9978	0.9978	0.0022	0.0022

Table 10: Classification performance of the sampled dataset with the SMOTE algorithm.

	Precision	Recall	*F*1 score	FNR	FPR
SMOTE layer1					
Benign	0.9999	0.9977	0.9988	0.0001	0.0023
Other	0.9977	0.9999	0.9988	0.0023	0.0001
Accuracy					0.9988
Macro avg	0.9988	0.9988	0.9988	0.0012	0.0012
Weighted avg	0.9988	0.9988	0.9988	0.0012	0.0012
SMOTE layer2					
Benign	0.9999	0.9977	0.9988	0.0001	0.0023
DoS hulk	0.9819	1.0000	0.9909	0.0181	0.0000
DDoS	1.0000	1.0000	1.0000	0.0000	0.0000
PortScan	1.0000	1.0000	1.0000	0.0000	0.0000
DoS goldeneye	0.9988	0.9980	0.9984	0.0012	0.0020
FTP-patator	1.0000	1.0000	1.0000	0.0000	0.0000
DoS slowloris	0.9955	0.9941	0.9948	0.0045	0.0059
DoS slowhttptest	0.9962	0.9946	0.9954	0.0038	0.0054

Table 10: Continued.

	Precision	Recall	$F1$ score	FNR	FPR
SSH-patator	1.0000	0.9975	0.9988	0.0000	0.0025
Bot	1.0000	0.9979	0.9990	0.0000	0.0021
Infiltration	1.0000	0.8889	0.9412	0.0000	0.1111
Heartbleed	1.0000	1.0000	1.0000	0.0000	0.0000
Web attack	0.9925	0.9907	0.9916	0.0075	0.0093
Accuracy					0.9979
Macro avg	0.9973	0.9892	0.9930	0.0027	0.0108
Weighted avg	0.9980	0.9979	0.9979	0.0020	0.0021

Table 11: Classification performance of the sampled dataset with the random sampling algorithm.

	Precision	Recall	$F1$ score	FNR	FPR
Random sampling layer1					
Benign	0.9993	0.9981	0.9987	0.0007	0.0019
Abnormal	0.9981	0.9993	0.9987	0.0019	0.0007
Accuracy					0.9987
Macro avg	0.9987	0.9987	0.9987	0.0013	0.0013
Weighted avg	0.9987	0.9987	0.9987	0.0013	0.0013
Random sampling layer2					
Benign	0.9993	0.9981	0.9987	0.0007	0.0019
DoS hulk	0.9850	0.9992	0.9921	0.0150	0.0008
DDoS	0.9992	0.9996	0.9994	0.0008	0.0004
PortScan	1.0000	0.9984	0.9992	0.0000	0.0016
DoS goldeneye	0.9980	0.9972	0.9976	0.0020	0.0028
FTP-patator	1.0000	1.0000	1.0000	0.0000	0.0000
DoS slowloris	0.9948	0.9941	0.9944	0.0052	0.0059
DoS slowhttptest	0.9962	0.9946	0.9954	0.0038	0.0054
SSH-patator	1.0000	0.9963	0.9981	0.0000	0.0037
Bot	0.9979	0.9979	0.9979	0.0021	0.0021
Infiltration	1.0000	0.7778	0.8750	0.0000	0.2222
Heartbleed	1.0000	1.0000	1.0000	0.0000	0.0000
Web attack	0.9944	0.9869	0.9906	0.0056	0.0131
Accuracy					0.9977
Macro avg	0.9973	0.9800	0.9876	0.0027	0.0200
Weighted avg	0.9978	0.9977	0.9977	0.0022	0.0023

Table 12: Classification performance of the sampled dataset with the SD sampling algorithm.

	Precision	Recall	$F1$ score	FNR	FPR
SD sampling layer1					
Benign	0.9996	0.9972	0.9984	0.0004	0.0028
Abnormal	0.9972	0.9996	0.9984	0.0028	0.0004
Accuracy					0.9984
Macro avg	0.9984	0.9984	0.9984	0.0016	0.0016
Weighted avg	0.9984	0.9984	0.9984	0.0016	0.0016
SD sampling layer2					
Benign	0.9996	0.9972	0.9984	0.0004	0.0028
DoS hulk	0.9811	0.9988	0.9899	0.0189	0.0012
DDoS	1.0000	1.0000	1.0000	0.0000	0.0000
PortScan	1.0000	0.9996	0.9998	0.0000	0.0004
DoS goldeneye	0.9980	0.9988	0.9984	0.0020	0.0012
FTP-patator	1.0000	1.0000	1.0000	0.0000	0.0000
DoS slowloris	0.9955	0.9911	0.9933	0.0045	0.0089
DoS slowhttptest	0.9962	0.9908	0.9935	0.0038	0.0092

Table 12: Continued.

	Precision	Recall	$F1$ score	FNR	FPR
SSH-patator	1.0000	1.0000	1.0000	0.0000	0.0000
Bot	1.0000	0.9959	0.9979	0.0000	0.0041
Infiltration	1.0000	1.0000	1.0000	0.0000	0.0000
Heartbleed	1.0000	1.0000	1.0000	0.0000	0.0000
Web attack	0.9853	0.9981	0.9917	0.0147	0.0019
Accuracy					0.9975
Macro avg	0.9966	0.9977	0.9971	0.0034	0.0023
Weighted avg	0.9976	0.9975	0.9975	0.0024	0.0025

Table 13: The classification performance of each model in the three sampling modes.

	Accuracy	Precision	Recall	$F1$ score	FNR	FPR
KNN	0.9909	0.9390	0.8814	0.8953	0.0610	0.1186
DT	0.9971	0.9432	0.9367	0.9369	0.0568	0.0633
SVC	0.9394	0.8678	0.7709	0.7960	0.1322	0.2291
DNN	0.9359	0.8017	0.8523	0.7945	0.1983	0.1477
Random forest	0.9974	0.9953	0.9367	0.9585	0.0047	0.0633
XGBoost	0.9849	0.9906	0.9068	0.9333	0.0094	0.0932
XGBoost + RF	0.9978	0.9972	0.9369	0.9596	0.0028	0.0631
SMOTE + KNN	0.9906	0.9247	0.9004	0.8974	0.0753	0.0996
SMOTE + DT	0.9964	0.9776	0.9532	0.9615	0.0224	0.0468
SMOTE + SVC	0.9308	0.8759	0.8428	0.8360	0.1241	0.1572
SMOTE + DNN	0.9611	0.8583	0.8489	0.8422	0.1418	0.1511
SMOTE + random forest	0.9969	0.9494	0.9805	0.9606	0.0506	0.0195
SMOTE + XGBoost	0.9977	0.9952	0.9805	0.9868	0.0048	0.0195
SMOTE + (XGBoost + RF)	0.9979	0.9973	0.9892	0.9930	0.0027	0.0108
Random sampling + KNN	0.9895	0.9443	0.8980	0.9080	0.0557	0.1020
Random sampling + DT	0.9967	0.9954	0.9702	0.9802	0.0047	0.0298
Random sampling + SVC	0.9174	0.8651	0.8091	0.8049	0.1349	0.1909
Random sampling + DNN	0.9451	0.8502	0.8934	0.8497	0.1498	0.1066
Random sampling + random forest	0.9968	0.9806	0.9269	0.9473	0.0194	0.0731
Random sampling + XGBoost	0.9928	0.9901	0.9901	0.9900	0.0099	0.0099
Random sampling + (XGBoost + RF)	0.9977	0.9973	0.9800	0.9876	0.0027	0.0200
SD sampling + KNN	0.9895	0.9323	0.8979	0.9015	0.0677	0.1021
SD sampling + DT	0.9957	0.9571	0.9703	0.9632	0.0429	0.0297
SD sampling + SVC	0.9509	0.8696	0.8612	0.8320	0.1304	0.1388
SD sampling + DNN	0.9475	0.8559	0.9318	0.8750	0.1441	0.0682
SD sampling + random forest	0.9962	0.9903	0.9620	0.9711	0.0097	0.0380
SD sampling + XGBoost	0.9982	0.9960	0.9978	0.9969	0.0040	0.0022
SD sampling + (XGBoost + RF)	0.9975	0.9966	0.9977	0.9971	0.0034	0.0023

The reason why the two-layer structure is more effective is that the first layer can first detect normal traffic and remove it from the dataset, and the second layer only detects the traffic that is judged to be abnormal in the first layer. In this way, the proportion of minority samples in the test set increases, reducing the impact of the unbalanced dataset and improving the classification ability of the model for minority samples.

5. Conclusions

This paper presents a novel network intrusion detection framework, which consists of four modules, the preprocessing module, data sampling module, feature selection module, and classification module. In the data sampling module, we propose a new sampling algorithm named SD sampling to balance the dataset. This algorithm overcomes the disadvantages of the SMOTE algorithm by considering the spatial distribution of samples. It also combines the idea of oversampling and undersampling, and finally, it obtains a dataset that is very easy to classify. In the classification module, we propose a two-layer structure combined with XGBoost and the random forest. The first layer is used for binary classification and the second layer is used for multiclassification. Both of them use the ensemble model, which overcomes the defects of low accuracy and poor generalization ability of the single algorithm. Finally, we conduct comparative experiments on the CICIDS2017 dataset using three sampling modes to verify the advantages of the SD sampling algorithm. At the same time, we also use four commonly used machine learning models to conduct comparative experiments, and the results show that the two-layer structure proposed in this paper can be used to classify traffic accurately and that the accuracy rate is up to 99.75%.

In the traffic classification task, it is also important to improve the interpretability of the model to generate some new domain knowledge. Therefore, in the future, we will focus on using explainable artificial intelligence (XAI) [41] tools to help us understand data information and model decision methods, such as simplifying models, estimating the correlation between individual features, visualizing feature importance, and visualizing the reasoning process of deep learning models.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Key Research and Development Program of China (No. 2019QY1300), the Youth Innovation Promotion Association CAS (No. 2021156), and the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDC02040100).

References

[1] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[2] H. Sun, Y. Xiao, J. Wang et al., “Common knowledge based and one-shot learning enabled multi-task traffic classification,” *IEEE Access*, vol. 7, pp. 39485–39495, 2019.

[3] L. Wang, X. Zhou, and R. Gu, “Traffic classification using cost based decision tree,” in *Proceedings of the 2011 International Conference on Computer Science and Network Technology*, vol. 4, Harbin, December 2011.

[4] S. Dong, “Multi class svm algorithm with active learning for network traffic classification,” *Expert Systems with Applications*, vol. 176, Article ID 114885, 2021.

[5] S. Cheng, Y. Niu, and L. I. Jun, *A Study on Network Traffic Classification Model of Knn Based on Network Resources”*, Journal of Hubei University of Technology, Hongshan, Wuhan, China, 2016.

[6] J. Zhang, C. Chen, X. Yang, W. Zhou, and Y. Xiang, “Internet traffic classification by aggregating correlated naive bayes predictions,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 5–15, 2012.

[7] Abid Saber, B. Fergani, and M. Abbas, “Encrypted traffic classification: combining over-and under-sampling through a pca-svm,” in *Proceedings of the 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, Tebessa, Algeria, October 2018.

[8] H. Singh, “Performance analysis of unsupervised machine learning techniques for network traffic classification,” in *Proceedings of the 2015 Fifth International Conference on Advanced Computing & Communication Technologies*, Haryana, India, February 2015.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[10] Y. Song, H. Li, P. Xu, and D. Liu, “A method of intrusion detection based on woa-xgboost algorithm,” *Discrete Dynamics in Nature and Society*, vol. 2022, pp. 1–9, 2022.

[11] C. Wang, T. Xu, and Xi Qin, “Network traffic classification with improved random forest,” in *Proceedings of the 2015 11th International Conference on Computational Intelligence and Security (CIS)*, Shenzhen, China, December 2015.

[12] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” *ICISSp*, vol. 1, pp. 108–116, 2018.

[13] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, “An adaptive ensemble machine learning model for intrusion detection,” *IEEE Access*, vol. 7, pp. 82512–82521, 2019.

[14] H. Xia and Z. Sun, “A Semi-supervised Outlier Detection Model Based on Autoencoder and Integrated Learning,” *Computer Engineering & Science*, vol. 1, pp. 1–9, 2020.

[15] F. T. Liu, K. Ting, and Z. H. Zhou, “Isolation forest,” in *Proceedings of the 2008 Eighth Ieee International Conference on Data Mining*, Beijing, China, December 2008.

[16] Y. Ling, Y. Liu, and B. Jiang, “Intrusion detection method based on double-layer heterogeneous ensemble learner,” *Journal of Cyber Security*, vol. 6, no. 3, pp. 16–28, 2021.

[17] M. Xu, X. Li, H. Liu, Z. Cheng, and J. Ma, “An intrusion detection scheme based on semi-supervised learning and information gain ratio,” *Journal of Computer Research and Development*, vol. 54, no. 10, pp. 2255–2267, 2017.

[18] J. Ren, X. Liu, Q. Wang, H. He, and X. Zhao, “An multi-level intrusion detection method based on knn outlier detection and random forests,” *Journal of Computer Research and Development*, vol. 56, no. 3, pp. 566–575, 2019.

[19] M. Data and M. Aritsugi, “T-DFNN: an incremental learning algorithm for intrusion detection systems,” *IEEE Access*, vol. 9, pp. 154156–154171, 2021.

[20] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, “Multi-classification approaches for classifying mobile app traffic,” *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018.

[21] I. P. Possebon, S. S. Anderson, L. Z. Granville, A. Schaeffer-Filho, and A. Marnerides, “Improved network traffic classification using ensemble learning,” in *Proceedings of the 2019 IEEE Symposium on Computers and Communications (ISCC)*, Barcelona, Spain, June 2019.

[22] X. Yang and S. Zhang, “Anomaly detection model based on multi-grained cascade isolation forest algorithm,” in *Proceedings of the 2022 IEEE 5th International Conference on Computer and Communication Engineering Technology (CCET)*, Beijing, China, August 2019.

[23] Y. Qiu, X. Chang, Q. Qiu, P. Cheng, and S. Su, “Stream data anomaly detection method based on long short-term memory network and sliding window,” *Journal of Computer Applications*, vol. 40, no. 5, Article ID 1335, 2020.

[24] A. Giuseppe, D. Ciuonzo, A. Montieri, and A. Pescapè, “Mimetic: mobile encrypted traffic classification using multimodal deep learning,” *Computer Networks*, vol. 165, Article ID 106944, 106944 pages, 2019.

[25] J. Gu and S. Lu, "An effective intrusion detection approach using svm with native bayes feature embedding," *Computers & Security*, vol. 103, Article ID 102158, 2021.

[26] R. Zhang, W. Chen, M. Hang, and L. Wu, "Detection of abnormal flow of imbalanced samples based on variational autoencoder," *Computer Science*, vol. 48, no. 7, pp. 62–69.

[27] X. Liu, T. Li, R. Zhang, D. Wu, Y. Liu, and Z. Yang, "A gan and feature selection-based oversampling technique for intrusion detection," *Security and Communication Networks*, vol. 2021, Article ID 9947059, 15 pages, 2021.

[28] B. Yan and G. Han, "Combinatorial intrusion detection model based on deep recurrent neural network and improved smote algorithm," *Chinese Journal of Network and Information Security*, vol. 4, no. 7, pp. 48–59, 2018.

[29] J.-H. Seo and Y.-H. Kim, "Machine-learning approach to optimize smote ratio in class imbalance dataset for intrusion detection," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–11, 2018.

[30] Y. Wang and G. Sun, "Oversampling method for intrusion detection based on clustering and instance hardness," *Journal of Computer Applications*, vol. 41, no. 6, p. 2021, 1709.

[31] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE Access*, vol. 9, pp. 7550–7563, 2021.

[32] S. Park and P. Hyunhee, "Combined oversampling and undersampling method based on slow-start algorithm for imbalanced network traffic," *Computing*, vol. 103, no. 3, pp. 401–424, 2021.

[33] Z. X. Wang, P. Wang, X. Zhou, S. H. Li, and MoX. Zhang, "Flowgan: unbalanced network encrypted traffic identification method based on gan," in *Proceedings of the 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, Xiamen, China, Decembre 2019.

[34] N. Gupta, V. Jindal, and P. Bedi, "Cse-ids: using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems," *Computers & Security*, vol. 112, Article ID 102499, 2022.

[35] J. Sharma, C. Giri, O. C. Granmo, and M. Goodwin, "Multilayer intrusion detection system with extratrees feature selection, extreme learning machine ensemble, and softmax aggregation," *EURASIP Journal on Information Security*, vol. 2019, no. 1, pp. 15-16, 2019.

[36] W. Li, S. Sun, S. Zhang, H. Zhang, and Y. Shi, "Cost-sensitive approach to improve thehttp traffic detection performance on imbalanced data," *Security and Communication Networks*, vol. 2021, 2021.

[37] J. Hu, H. Zhang, Y. Liu, R. Sutcliffe, and J. Feng, "Bbw: a batch balance wrapper for training deep neural networks on extremely imbalanced datasets with few minority samples," *Applied Intelligence*, vol. 52, no. 6, pp. 6723–6738, 2022.

[38] W.-Y. Loh, "Classification and regression trees," *WIREs Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[39] Ji-hye Kim, S.-Ho Yoon, and M.-S. Kim, "Study on traffic classification taxonomy for multilateral and hierarchical traffic classification," in *Proceedings of the 2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Seoul, September 2012.

[40] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescapé, "A dive into the dark web: hierarchical traffic classification of anonymity tools," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1043–1054, 2020.

[41] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "Xai meets mobile traffic classification: understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225–4246, 2021.