A High Performance Intrusion Detection System Using LightGBM Based on Oversampling and Undersampling

Hao Zhang[1,2], Lina Ge[1,2](✉), and Zhe Wang[1,2,3]

[1] School of Artificial Intelligence, Guangxi Minzu University, Nanning, China
66436539@qq.com

[2] Key Laboratory of Network Communication Engineering, Guangxi Minzu University, Nanning, China

[3] Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning, China

Abstract. Intrusion detection system plays an important role in network security, however, the problem with data imbalance limits the detection ability of intrusion detection system. In order to improve the performance of intrusion detection system, this paper proposes to use the adaptive synthetic sampling technique (ADASYN) and random under sampling technique to alleviate the problem of data imbalance in intrusion detection. Firstly, the majority class samples in the dataset are removed by undersampling technology and the minority class samples are oversampled, so the samples can reach a balanced state. Subsequently, a sparse autoencoder (SAE) extracts features from the resampled data to fit the original sample as closely as possible. Finally, LightGBM is applied on the processed dataset for the classification process. Multi-classification experiments were conducted on KDD99 and UNSWNB15 datasets. We compare six models' performance and find LightGBM is superior to other models. Furthermore, we also compare existing methods and the results show that our proposed method outperforms current methods.

Keywords: Intrusion detection systems · Resampling · LightGBM · Autoencoder

1 Introduction

With the advent of the big data era, network traffic is on an exponential growth trend. The growing data makes it increasingly difficult to detect intrusion traffic. As a security device, the intrusion detection system plays a vital role in protecting the country's information infrastructure [1]. According to the detection technology, intrusion detection systems can be divided into misuse-based intrusion detection systems (MIDS) and anomaly-based intrusion detection systems (AIDS). The former detects anomaly traffic based on historical traffic statistics and known attack libraries, while the latter detects anomaly traffic based on statistical deviations [2, 3]. MIDS is unable to detect unknown attacks and is vulnerable when dealing with zero-day attacks. AIDS can detect unknown

https://doi.org/10.1007/978-3-031-13870-6_53

attacks, so it has become a popular research topic. In recent years, machine learning in intrusion detection has achieved success [4]. Typical methods such as decision tree (DT) [5], support vector machine (SVM) [6], random forest (RF) [7], etc., have shown high accuracy in the detection of abnormal traffic. Compared with traditional methods, machine learning technology can efficiently detect attacks in the face of massive traffic data, so it exerts an important influence in intrusion detection. As an extension of machine learning techniques, deep learning has gained widespread attention owing to its ability to extract features [8–11]. It can learn low-dimensional expressions from high-dimensional data so that massive data can be processed. The popular methods are deep belief networks (DBN) [12, 13], Autoencoders (AE) [14, 15], which are efficient in feature extraction.

In the era of big data, the issue that needs to be addressed in intrusion detection is how to identify the attack traffic with only contains a small number of traffic. Generally, in the network flow, most of them are normal traffic, and only a few are anomaly traffic. It is also in line with people's observations in life. The currently available public datasets for intrusion detection include binary and multiclass. Most of these datasets do not have imbalance problems on binary classification. However, different types of attacks require different defense mechanisms [16], and simply dividing traffic into normal and abnormal traffic cannot allow managers to take targeted defense measures. Therefore, it is necessary to identify the traffic to a specific type. Although machine learning technology can be handy in coping with massive data, it is inefficient in the detection of minority classes.

An efficient way is to use resampling techniques [17, 18]. In the multi-classification task, only a few data are attack classes, which makes the classifier tend to the majority classes when training data. Although it has high performance in general, it is especially low when detecting the minority classes. Resampling technology can address this problem. For the minority classes, the oversampling technology is used to increase the number of the minority class data. At the same time, the undersampling technology is used to reduce the number of majority class data. After data resampling, the number of majority class and minority class samples reaches a balanced state, so that the classifier can detect minority class samples. However, how to ensure that the resampling samples come from the same data distribution as the real samples is a key issue of resampling techniques. In general, the resampling technology adopts the Euclidean distance to judge whether the data come from the same distribution. The method assumes that spatially close samples are from the same distribution. Whereas the use of Euclidean distance ignores the outliers, making the resampling technique have errors in fitting the real samples. Deep learning techniques provide a solution for this problem. Using deep learning technology for feature extraction can alleviate the problems caused by data fitting.

There have been many studies on data imbalance [16, 19–23]. Particularly, in this paper, an intrusion detection method based on data imbalance is presented. The method adopts ADASYN oversampling and random undersampling techniques. The ADASYN technique oversamples the minority class samples and can help the classifier to learn a better decision boundary. In addition, a random undersampling technique is used in this paper to remove the samples from most classes. A combination of oversampling and undersampling methods enables the classifier to detect minority class samples. After oversampling and undersampling, a sparse autoencoder is designed to extract features.

The autoencoder is trained from raw data, so it is helpful for data fitting problems caused by data resampling. The LightGBM is used as the classifier. LightGBM is a boosting ensemble method, which is improved based on GBDT. LightGBM gives more weight to the misclassified samples so that they can get more attention in the weak classifiers later. For minority class samples, LightGBM can improve the detection rate. Histogram and feature bundling methods were introduced into the LightGBM, which can reduce the training time while obtaining high accuracy. Therefore, the LightGBM model is used for classification in this paper. The main contributions of this study are as follows:

(1) To solve the data imbalance problem, we proposed a hybrid sampling method to process the dataset and used the LightGBM for classification. This method alleviates the imbalance problem.
(2) We adopted sparse autoencoders to perform feature extraction. In addition, we designed a resampling method to make the resampled data closer to the actual distribution. It is shown in Sect. 3.2.
(3) We adopted several metrics, including accuracy rate, precision rate, detection rate, and F1 score to evaluate the performance of the method. Furthermore, the proposed method was compared not only with classical methods but also with current advanced methods.

The rest of the paper is organized as follows: Sect. 2 introduces the theory related to ADASYN techniques and LightGBM. Section 3 presents the proposed method in detail. Section 4 is the experimental part of the paper, which describes and analyzes the experimental results. The final part is the conclusion and outlook.

2 Related Works

2.1 Adasyn

The ADASYN method oversamples minority classes on the boundary to move the decision boundary to the minority classes that are difficult to learn [24]. The specific steps of the method are as follows: Assume that the training set D_{tr} has m samples $\{x_i, y_i\}^m$,$y_i \in \{+1, -1\}$. There are m_s minority class samples and m_l majority class samples in the training set D_{tr}, $m_s + m_l = D_{tr}$.

(1) Calculate the imbalance ratio in the D_{tr}:

$$d = \frac{m_s}{m_l} \tag{1}$$

(2) If d $< d_{th}$ (d_{th} is the preset threshold), enter the loop:

   a. Calculate the number of minority class samples that need to be generated (β ∈ [0, 1], is a balance parameter):

$$G = (m_l - m_s) \times \beta \tag{2}$$

b. For each minority class sample, K nearest neighbor samples are picked based on the euclidean distance (Δ_i represents the number of samples that K nearest neighbor samples belong to the majority class)

$$r_i = \frac{\Delta_i}{K}(i = 1, 2, \ldots m_s) \tag{3}$$

c. Calculate the weights:

$$\hat{r}_i = r_i / \sum_{i}^{m_s} r_i \tag{4}$$

(3) Calculate the number of samples that need to be synthesized for each minority class sample:

$$g_i = \hat{r}_i \times G \tag{5}$$

(4) For each minority class sample x_i, randomly select a sample x_{zi} from its K nearest neighbor samples to generate a new sample according to the following formula:

$$s_i = x_i + (x_{zi} - x_i) \times \lambda(\lambda \in [0, 1]) \tag{6}$$

2.2 LightGBM

Among the tree-based ensemble algorithms, GBDT, XGBoosting, and LightGBM are recognized as the best performing algorithms. In recent years, they have made a big splash in Kaggle competitions. GBDT and XGBoost algorithms need to traverse the entire dataset when dividing nodes, which makes them have a large time overhead. Compared with other algorithms, LightGBM has a greater advantage in training overhead. LightGBM adopts gradient-based one-sided sampling (GOSS) method and exclusive feature bundling (EFB) algorithm [25], making it significantly smaller in training time than GBDT and XGBoosting.

Assuming that each sample is associated with its gradient information, it is generally believed that a sample with a small gradient has a low training error. For the following classification or regression tasks, samples with small gradients have smaller contribution values. Therefore, the GOSS method sorts the gradients of all samples according to their absolute value and saves all samples with large gradients. Subsequently, randomly selects some samples from small gradient samples for preservation. Suppose there are n training instances in the training set, denoted as $\{x_1, \ldots, x_n\}$. After the gradient boosting calculation, the negative gradient of the output of the model is recorded as $\{g_1, \ldots, g_n\}$. The information gain of the split feature j at point d is defined as:

$$V_{j|O}(d) = \frac{1}{n_O}\left(\frac{\left(\sum_{\{x_i \in O: x_{ij} \le d\}}\right)^2}{n^j_{l|O}(d)} + \frac{\left(\sum_{\{x_i \in O: x_{ij} > d\}}\right)^2}{n^j_{r|O}(d)}\right) \tag{7}$$

where O is the parent node of the decision tree division, n_l is the left node of the decision tree, and n_r is the right node of the decision tree, and $n_O = \sum I[x_i \in O], n^j_{l|O}(d) = \sum I\left[x_i \in O : x_{ij} \le d\right]$ *and* $n^j_{r|O}(d) = \sum I\left[x_i \in O : x_{ij} > d\right]$.

Let a and b be the sampling ratios of large gradient and small gradient instances, respectively. According to the sorted instance gradient values, the first a × 100% large gradient sample is selected, and then randomly selects b × 100% small gradient samples from the rest of the data. After many iterations, the final calculated information gain is:

$$\widetilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n^j_l(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n^j_r(d)}\right) \tag{8}$$

where $A_l = \left\{x_i \in A : x_{ij} \le d\right\}, A_r = \left\{x_i \in A : x_{ij} > d\right\}, B_l = \left\{x_i \in B : x_{ij} \le d\right\}, B_r = \left\{x_i \in B : x_{ij} > d\right\}$.

EFB method exploits the sparsity of high-dimensional spaces to reduce features. In high-dimensional space, the values between two instances are mostly mutually exclusive, that is, they are not all zero in the same row. Therefore, an undirected graph with weights can be further constructed. The nodes of the graph represent a feature, and the weights are expressed as the number of mutually exclusive values between two features. It is then transformed into a graph coloring problem and solved using a greedy algorithm.

3 Methodology

Network flow presents the characteristics of massive and high dimensions. In order to improve the detection ability, an intrusion detection method based on oversampling and undersampling is designed in this paper. The schematic diagram of the method is shown in Fig. 1. The proposed method includes data preprocessing module, resampling module, feature extraction module, and classification module.

In the data preprocessing module, these methods including numeralization, one-hot coding, and normalization are used for data processing. After data preprocessing, it is partitioned into a training set and test set. It is described in Sect. 3.1.

In the resampling module, this paper uses the ADASYN and random undersampling technology to make the training data reach a relatively balanced state. This is described in detail in Sect. 3.2.

In the feature extraction module, a sparse autoencoder is trained from the original data. Then the autoencoder is used to feature extraction on the resampled data, which reduces the data dimension while decreasing the fitting error caused by data resampling. The details of feature extraction are presented in Sect. 3.3.

In the classification module, the LightGBM model is used to train the data after processing so that the trained LightGBM model is obtained. Finally, the test set is input into the LightGBM model for testing, and the final results are output.

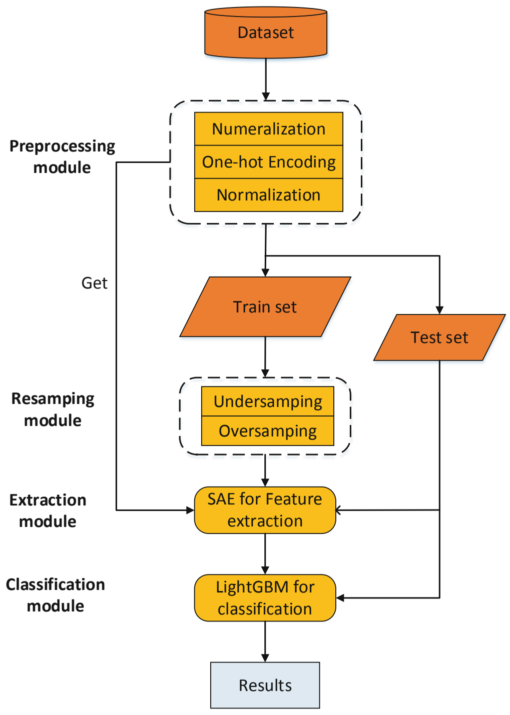

Fig. 1. A schematic diagram of the intrusion detection system

3.1 Dataset Description and Preprocessing

The KDD99 dataset comes from the intrusion detection project at Lincoln laboratory in the United States. A total of nine weeks of network connection data was collected in the dataset, which contained 7 million records. The dataset contains a total of 39 attack types, of which 22 appear in the training set and 17 in the test set. These 39 attack types can be classified into four categories, namely Probe, Dos, R2L, and U2R. 10% of the KDD99 dataset is used in this paper. On the basis of this dataset, it is further divided into training set and test set with a ratio of 7:3.

The UNSWNB15 dataset was developed by the Australian Cyber Security Centre in 2015 using the IXIA tool. The dataset contains 2 million records, which are saved in four CSV files. The dataset contains a total of 9 attack types. In this paper, all the data in this dataset are used for experiments. Similarly, the UNSWNB15 dataset is also divided into training set and test set in a ratio of 7:3.

The KDD99 dataset contains 41-dimensional features, of which the three features of protocol, service, and flag are symbol types, and the others are numeric types. Since the classifier cannot process symbolic data, the symbolic data is first converted into numeric types. Then the data are sparsely encoded by one-hot encoding. Taking the KDD99 dataset as an example, the protocol feature contains three values: TCP, UDP, and ICMP, which are encoded as 100, 010, and 001 respectively. The service feature has 66 values, so it is represented using 66 numbers consisting of 0 and 1. The flag feature has 11 values, which are represented by 11 numbers containing 0 and 1. After one-hot coding, the dimensionality of the KDD99 dataset is extended to 118. Similarly, the UNSWNB15 dataset is processed in the same way.

Dataset normalization processing. To speed up the training, the data normalization method is required. In this study, the maximum and minimum normalization method is used to scale the data to [0,1]. The maximum-minimum normalization method is calculated as follows:

$$MaxMinScaler = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{9}$$

Among them, x_{max} and x_{min} represent the maximum and minimum values of the column where the feature x is located, respectively.

3.2 Resampling Method

In order to obtain a proper representation of the data distribution in practice, this paper does not simply resample all data to the same amount. Instead, the majority and minority class data are resampled to a relatively balanced state. First, a threshold θ is preset, which represents an acceptable balance ratio. In general, the threshold is set to 0.5. For a dataset, its balanced degree is determined by the following equation:

$$d = \frac{|T_i|}{|T|} \tag{10}$$

where $|T_i|$ represents the number of instances of the i-th type of data, and $|T|$ represents the total number of samples.

For each class of samples, its balance degree is calculated according to the formula (10). If the balance degree exceeds the threshold θ, the samples are undersampled. The sample size is calculated as:

$$\left|T_i^{'}\right| = \frac{|T_i|}{C} \tag{11}$$

where C represents the number of categories in the dataset, and $\left|T_i^{'}\right|$ is the number of samples of this category after undersampling.

After that, sort all $\left|T_i^{'}\right|$ and save the smallest $\left|t_i^{'}\right| = \mathrm{Min}\left(\left|T_i^{'}\right|\right)$. For the minority class whose balance degree is less than θ, it needs to be oversampled. The sample size for oversampling is calculated as:

$$\left|T_i^{'}\right| = |T_i| + 10^N \tag{12}$$

where N is the number of digits of $\left|t_i^{'}\right|$.

After the above calculations, the number of samples required for each class is obtained. To ensure that the number of samples for all classes is in the same order of magnitude, it needs to be traversed. For classes that are not in the same order of magnitude, they are further processed so that they are in the same order of magnitude as other samples.

3.3 Feature Extraction and Classification

After data resampling, a sparse autoencoder is used to extract features from the data. First, in the data preprocessing module, training a sparse autoencoder so that the autoencoder can extract the low-dimensional representation of the data. The trained sparse autoencoder will be used to recover a low-dimensional representation of the resampled data. The structure of the sparse autoencoder used in this paper is shown in Fig. 2. A sparse autoencoder consists of an input layer, an encoding layer, a middle layer, a decoding layer, and an output layer. The size of the input and output layers is determined by the dimensionality of the input data. Both the encoding and decoding layers are composed of two neural network layers with sizes of 80 and 50, respectively. The middle layer consists of neurons with size 16 and is a low-dimensional representation of original data. After feature extraction, the data will be input into the LightGBM classifier training. Finally, the test data will be input into the LightGBM model for testing.

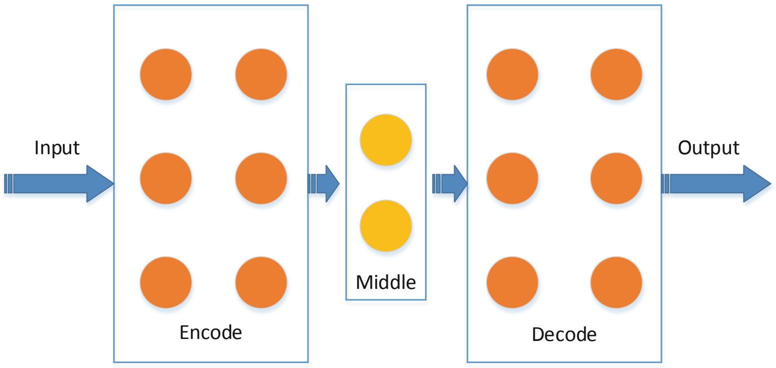

Fig. 2. The structure of the sparse autoencoder

4 Experiment and Discussion

This part presents the performance of the LightGBM model before and after the data resampling. Furthermore, to validate the performance of LightGBM, the paper compared with support vector machine, random forest, GBDT, and so on. The experiments were deployed on a Dell host with 32G memory and were programmed using Python 3.8. The sklearn library and the keras framework were used to build the model.

4.1 Evaluate Metrics

In this paper, the four metrics of accuracy rate (AC), precision rate (P), detection rate (DR), and F1 scores (F1) are used to evaluate the model.

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

$$P = \frac{TP}{TP + FP} \tag{14}$$

$$DR = \frac{TP}{TP + FN} \tag{15}$$

$$F1 = 2/\left(\frac{1}{P} + \frac{1}{DR}\right) \tag{16}$$

4.2 Results and Analysis

We conducted multi-classification experiments on KDD99 and UNSWNB15 datasets. The experiments compared the LightGBM's performance before and after data resampling. To ensure the comparability of the methods, the random_state was set to 42. Note that all results were rounded to two decimal places.

The performance of LightGBM is shown in Fig. 3. As shown in Fig. 3, the accuracy, precision, and F1-score of the model on the KDD99 dataset are improved by 0.46% after data resampling. In addition, the accuracy, precision, and F1-score of the model on the UNSWNB15 dataset are improved by 0.63%, 1.13%, and 0.83% respectively. It can be concluded that the performance of the model has improved after data resampling.

Figure 4 and Fig. 5 show the detection rates of each class on the two datasets before and after data resampling. It can be seen from Fig. 4 that the detection rates of Normal and Dos classes are above 90% before resampling because of their high proportion in the original dataset. The detection rates of Probe and R2L classes are lower than 80% before resampling. After resampling, the detection rate increased significantly. In particular, the detection rate of U2R before resampling is zero. The reason is that the number of U2R in the original data is rare, with only 52 records. After resampling, the detection rate of U2R reaches 94.12%. The performance of the model on the UNSWNB15 dataset is shown in Fig. 5. After resampling, the detection rate of Exploits decreases, indicating that the boundary between this data and adjacent samples becomes blurry. However, the model improves the detection rate to varying levels in the remaining attack classes. In general, the detection of minority classes is improved after resampling.

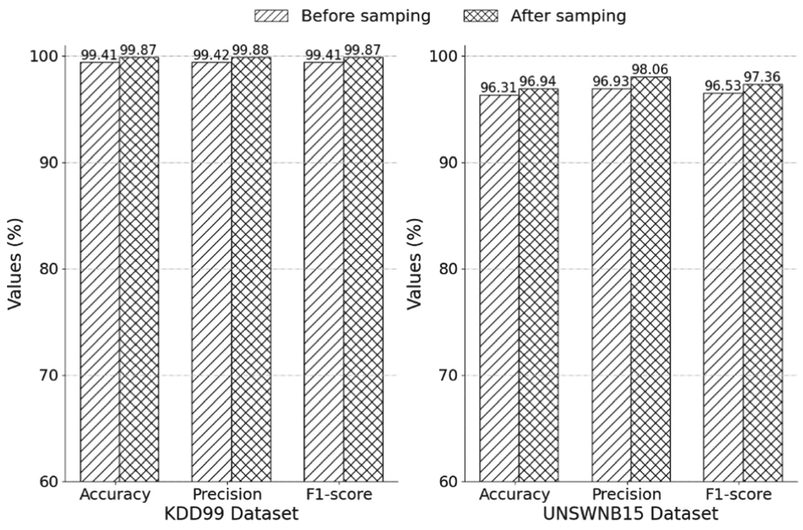

Fig. 3. The performance of model before and after resampling for KDD99 and UNSWNB15 datasets

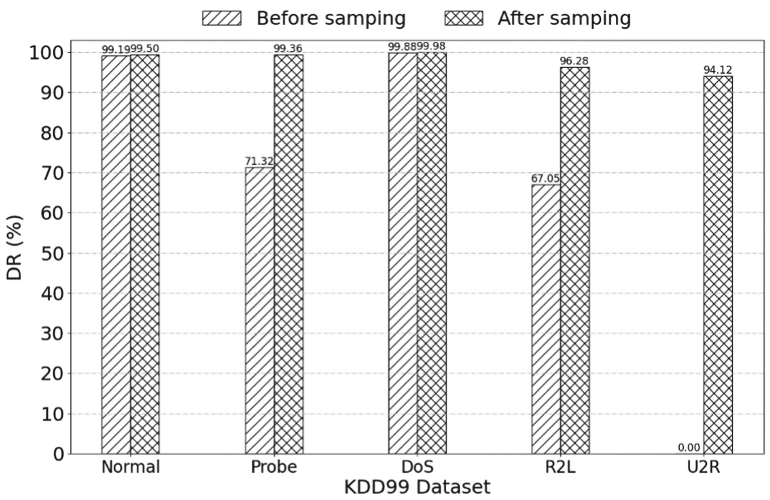

Fig. 4. The detection rates of each class for the KDD99 dataset

Tables 1 and 2 show the performance of different models after resampling. On the KDD99 dataset, the LightGBM model achieves 99.87% accuracy, which is the highest among all models. In addition, the model obtains 99.88% and 99.87% in precision and F1 scores, respectively. The RF model also performs the same in these metrics. However, the training time for the LightGBM is significantly less than for the RF model. Though the DT model has the lowest training time among all models, its accuracy, precision, and F1 scores are lower than the LightGBM model. On the UNSWNB15 dataset, although

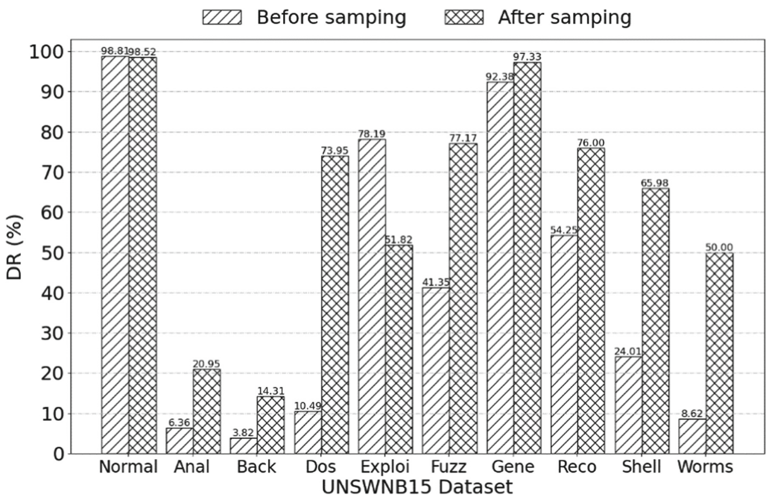

Fig. 5. The detection rates of each class for the UNSWNB15 dataset

the RF and XGBoost models have an outstanding performance in some metrics, the training time of RF is 6 times longer than the LightGBM model and the training time of XGBoost is 7 times longer than the LightGBM model. In general, the LightGBM model balances time overhead and overall performance on both the KDD99 dataset and the UNSWNB15 dataset.

Table 1. The performance of different models on the KDD99 dataset after resampling

Model	SVM	DT	RF	GBDT	XGBoost	LightGBM
AC (%)	99.31	99.66	99.86	99.31	99.84	**99.87**
P (%)	99.69	99.71	**99.88**	99.60	99.86	**99.88**
F1 (%)	99.46	99.68	**99.87**	99.42	99.85	**99.87**
Time (s)	72.66	**1.03**	15.75	156.96	13.97	1.56

Table 2. The performance of different models on the UNSWNB15 datasets after resampling

Model	SVM	DT	RF	GBDT	XGBoost	LightGBM
AC (%)	96.62	96.64	**96.99**	96.73	96.95	96.94
P (%)	97.86	97.73	97.92	97.92	98.02	**98.06**
F1 (%)	97.07	97.10	**97.36**	97.19	**97.36**	**97.36**
Time (s)	8412.03	**3.97**	65.42	735.57	76.41	10.09

Tables 3 and 4 show the detection rate for specific classes. On the KDD99 dataset, all models have identical performance in the detection of the Dos and Probe classes. On the detection of Normal, LightGBM and RF models perform the best. On the detection of R2L, XGBoost and LightGBM have the highest detection rate. GBDT and SVM perform best in the detection of U2R. Overall, the LightGBM model outperforms in all classes, although detection on U2R is not the best. On the UNSWNB15 dataset, for classes of Analysis and Backdoors, all models have a low detection rate, no more than 50%. The possible reason is that the boundaries of Analysis and Backdoor samples are relatively blurred, which makes the model unable to distinguish them effectively. The detection rates of LightGBM models are above 50% for all classes excluding Analysis and Backdoor classes. However, the SVM and GBDT models do not reach 50% in the detection of Exploits. DT, RF, and XGBoost models also do not reach 50% in detecting Worm classes. It means that the LightGBM model is more effective than other models in detecting minority classes.

Table 3. The detection rates of different models for each class on the KDD99 dataset after resampling

Model	SVM	DT	RF	GBDT	XGBoost	LightGBM
Normal	0.97	0.99	**1.00**	0.97	0.99	**1.00**
Probe	**0.99**	0.98	**0.99**	**0.99**	**0.99**	**0.99**
Dos	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
R2L	0.85	0.92	0.95	0.90	**0.96**	**0.96**
U2R	**1.00**	0.76	0.88	**1.00**	0.94	0.94

Table 4. The detection rates of different models for each class on the UNSWNB15 dataset after resampling

Model	SVM	DT	RF	GBDT	XGBoost	LightGBM
Normal	0.98	0.98	0.99	0.98	0.99	0.99
Dos	0.76	0.55	0.51	0.74	0.70	0.74
Exploits	0.43	0.54	0.61	0.49	0.54	0.52
Fuzzers	0.67	0.71	0.79	0.71	0.78	0.77
Generic	0.97	0.97	0.97	0.97	0.97	0.97
Backdoors	0.16	0.04	0.04	0.12	0.15	0.14
Reconnaissance	0.77	0.63	0.75	0.72	0.76	0.76
Analysis	0.18	0.16	0.19	0.17	0.20	0.21
Worms	0.66	0.24	0.26	0.59	0.40	0.50
Shellcode	0.45	0.52	0.63	0.66	0.67	0.66

Tables 5 and 6 show the detection rates compared to other advanced methods. For the KDD99 dataset, it is clear from Table 10 that FE-SVM has the poor performance. Among all the methods, our proposed method has the best performance in detecting the U2R class. The RU-Smote method performs best in detecting the R2L class. Overall, our proposed method achieves an average detection rate of 97.8% and is more effective on the KDD99 dataset compared to other methods. For the UNSWNB15 dataset, the TSDL, CASCADE-ANN, and DBN methods achieve a 0% detection rate on some classes. By contrast, ICVAE-DNN, SAE, and our proposed method can detect all classes. In particular, our proposed method achieves an average detection rate of 62.6%, which is the highest among all methods. This proves the generalizability of our proposed method.

Table 5. Comparison multi-class classification results with advanced methods on the KDD99 dataset.

Method	FE-SVM[6]	ACAE-RF[7]	PDAE[15]	RU-Smote[18]	MSML[23]	Our method
Normal	0.99	1.00	1.00	0.98	0.86	1.00
Probe	0.70	0.99	0.99	0.99	0.98	0.99
Dos	0.96	1.00	1.00	1.00	1.00	1.00
R2L	0.45	0.88	0.92	**0.97**	0.91	0.96
U2R	0.14	0.47	0.75	0.83	0.73	**0.94**
Avg	0.648	0.868	0.932	0.954	0.896	**0.978**

Table 6. Comparison multi-class classification results with advanced methods on the UNSWNB15 dataset.

Method	TSDL[8]	CASCADE-ANN[10]	ICVAE-DNN[11]	DBN[13]	SAE[14]	Our method
Normal	0.82	0.80	0.81	0.70	0.78	0.99
Dos	0.00	0.00	0.08	0.00	0.06	0.74
Exploits	0.57	0.57	0.71	0.86	0.94	0.52
Fuzzers	0.40	0.00	0.35	0.57	0.52	0.77
Generic	0.61	0.98	0.96	0.96	0.97	0.97
Backdoors	0.00	0.00	0.21	0.00	0.04	0.14
Reconnaissance	0.25	0.34	0.80	0.73	0.81	0.76
Analysis	0.01	0.00	0.15	0.00	0.01	0.21
Worms	0.00	0.00	0.80	0.00	0.11	0.50
Shellcode	0.01	0.00	0.92	0.00	0.59	0.66
Avg	0.267	0.269	0.579	0.382	0.482	**0.626**

5 Conclusions

For the problem that the abnormal traffic in the network flows is less than the normal traffic, we propose adaptive synthetic oversampling technology and random undersampling technology to process imbalanced data. In the experiments with two datasets, the model performance before and after data resampling was compared. After resampling, the LightGBM achieves 99.87% accuracy, 99.88% precision, and 99.87% F1 score on the KDD99 dataset, respectively. On the UNSWNB15 dataset, it achieves 96.94%, 98.06%, and 97.36% for accuracy, precision, and F1 scores respectively. The overall performance of LightGBM is improved after resampling. Meanwhile, the paper also compares the performance of different models after data resampling. The results show that our proposed method not only performs better but also has a low time overhead. Furthermore, the paper compares the proposed methods with state-of-the-art methods, and the results are promising. Although the LightGBM model performs well in the overall performance, it still needs to be improved in detecting minority classes such as Analysis and Backdoors. In the future, we will further optimize the model so that it can exceed 50% detection rate for these classes.

Acknowledgement. This work was supported by the National Natural Science Foundation of China under Grant 61862007, and Guangxi Natural Science Foundation under Grant 2020GXNSFBA297103.

References

1. Bijone, M.: A survey on secure network: intrusion detection & prevention approaches. Am. J, Inf. Sys. **4**(3), 69–88 (2016)
2. Jian, S.J., et al.: Overview of network intrusion detection technology. J. Inf. Secur. China **5**(4), 96–122 (2020)
3. Mahmoud Said, E., et al.: A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. J. Netw. Compu. Appl. **191**, 103160 (2021)
4. Kilincer, I.F., et al.: Machine learning methods for cyber security intrusion detection: datasets and comparative study. Comput. Netw. **188**, 107840 (2021)
5. Ahmim, A., et al.: A novel hierarchical intrusion detection system based on decision tree and rules-based models. In: 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 228–233. IEEE (2019)
6. Dong, K., Shi, J., Guo, L., Yuan, F.: Application of SVM in anomaly detection based on sampling and feature extraction. J. Phys. Conf. Ser. **1629**(1), 012017, IOP Publishing (2020)
7. Ji, S., Ye, K., Xu, C.-Z.: A network intrusion detection approach based on asymmetric convolutional autoencoder. In: Zhang, Qi., Wang, Y., Zhang, L.-J. (eds.) CLOUD 2020. LNCS, vol. 12403, pp. 126–140. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59635-4_9
8. Khan, F.A., et al.: A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access **7**, 30373–30385 (2019)
9. Kasongo, S.M., Sun, Y.: A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Comput. Secur. **92**, 101752 (2020)
10. Baig, M.M., Awais, M.M., El-Alfy, E.M.: A multiclass cascade of artificial neural network for network intrusion detection. J. Intell. Fuzzy Syst. **32**(4), 2875–2883 (2017)

11. Yang, Y., et al.: Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. Sensors **19**(11), 2528 (2019)
12. Zhao, Z., Ge, L., Zhang, G.: A novel DBN-LSSVM ensemble method for intrusion detection system. In: 2021 9th International Conference on Communications and Broadband Networking, pp. 101–107 (2021)
13. Singh, P., Kaur, A., Aujla, G.S., Batth, R.S., Kanhere, S.: Daas: Dew computing as a service for intelligent intrusion detection in edge-of-things ecosystem. IEEE Internet Things J. **8**(16), 12569–12577 (2020)
14. Li, Y., Gao, P., Wu, Z.: Intrusion detection method based on sparse autoencoder. In: 2021 3rd International Conference on Computer Communication and the Internet (ICCCI), pp. 63–68, IEEE (2021)
15. Basati, A., Faghih, M.M.: PDAE: Efficient network intrusion detection in IoT using parallel deep auto-encoders. Inf. Sci. **598**, 57–74 (2022)
16. Zhang, H., et al.: An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset. Comput. Netw. **177**, 107315 (2020)
17. Ahsan, R..: A detailed analysis of the multi-class classification problem in network intrusion detection using resampling techniques. Doctoral Dissertation, Carleton University (2021)
18. Bagui, S., Li, K.: Resampling imbalanced data for network intrusion detection datasets. J. Big Data **8**(1), 1–41 (2021). https://doi.org/10.1186/s40537-020-00390-x
19. Liu, J., Gao, Y., Hu, F.: A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. Comput. Secur. **106**, 102289 (2021)
20. Andresini, G., et al.: GAN augmentation to deal with imbalance in imaging-based intrusion detection. Fut. Gene. Comput. Syst. **123**, 108–127 (2021)
21. Alshamy, R., Ghurab, M., Othman, S., Alshami, F.: Intrusion detection model for imbalanced dataset using SMOTE and random forest algorithm. In: Abdullah, N., Manickam, S., Anbar, M. (eds.) ACeS 2021. CCIS, vol. 1487, pp. 361–378. Springer, Singapore (2021). https://doi.org/10.1007/978-981-16-8059-5_22
22. Gonzalez-Cuautle, D.,et al.: Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusion-detection-system datasets. Appl. Sci. **10**(3), 794 (2020)
23. Yao, H., et al.: MSML: a novel multilevel semi-supervised machine learning framework for intrusion detection system. IEEE Internet of Things J. **6**(2), 1949–1959 (2018)
24. He, H., et al.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328 (2008)
25. Ke, G., et al.: LightGBM: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems, vol. 30, pp. 3146–3154 (2017)