Enhancing Network Intrusion Detection with Deep Oversampling and Convolutional Autoencoder for Imbalanced Dataset

Xuanrui Xiong[1], Junfeng Li[1(✉)], Huijun Zhang[2], Han Shen[1], Mengru Liu[1], Wei Peng[1], Qi Huang[1], and Yuan Zhang[3]

[1] School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
xiongxr@cqupt.edu.cn, s210101076@stu.cqupt.edu.cn

[2] College of Environmental Resources, Chongqing Technology and Business University, Chongqing 400067, China
zhanghj@ctbu.edu.cn

[3] School of Computing, Chongqing Institute of Engineering, Chongqing 400056, China
zhangyuan@cqie.edu.cn

Abstract. Network intrusion detection is confronted with a shortage of intrusion samples about uncommon attacks, resulting in an imbalance in data distribution across most network intrusion detection datasets. Traditional machine learning methods encounter challenges in effectively handling unbalanced massive high-dimensional data, resulting in a low detection rate for minority attack classes. We propose a data generation method based on Deep Convolutional Autoencoder-SMOTE (DCAES) generation model. We first generate new attack samples by feeding minority class samples into the DCAES generative model. This process aims to increase minority class samples to balance the dataset. Furthermore, we use DBSCAN clustering undersampling and Tomek Links methods to refine the dataset to eliminate redundant and noisy samples from the majority classes. Finally, we obtain a dataset that shows relative balance and high quality. To assess the efficacy of our approach, we performed comparison experiments using the UNSW-NB15 dataset. The experimental findings demonstrate that the proposed strategy yields a balanced dataset that can be utilized well for classification learning. Furthermore, the detection rate of minority class attacks has also been greatly improved.

Keywords: Network Intrusion Detection · Imbalanced dataset · Convolutional autoencoder · Data enhancement

Supported by the National Natural Science Foundation of China (51808079), Chongqing Research Program of Basic Research and Frontier Technology (cstc2017jcyjAX0135), the Science and Technology Research Program of Chongqing Municipal Education Commission (KJQN201801908).

F. Gao et al. (Eds.): Chinacom 2023, LNICST 590, pp. 215–232, 2024.
https://doi.org/10.1007/978-3-031-67162-3_14

1 Introduction

Network traffic has increased tremendously as computers and the Internet have become more widespread. Cyberattacks have become more common and sophisticated, posing serious threats to the energy, healthcare, communications, and financial industries, etc. Network intrusion detection technology can effectively identify and promptly report unauthorized or abnormal network activities, and protect computer systems, networks, and applications from unauthorized access, intrusion attacks, destruction, and other malicious behaviors. It is one of the important means of security protection. In intrusion detection, traffic data can be categorized using trained machine learning models to determine the presence or absence of unusual activities. A machine learning model is performance strongly depends on the dataset's comprehensiveness for training. Training machine learning models on imbalanced datasets can lead to model bias. Not only is the training difficult, but focusing on minimizing empirical risk as the training objective will cause model to lean toward the majority class. Therefore, in datasets with uneven class distribution, detection of minority classes has attracted the attention of researchers.

Researchers frequently examine the data imbalance issue from the following viewpoints. From the perspective of algorithm, it mainly uses the cost-sensitive factor [1] to improve the existing algorithm, and integrates the classification results of multiple base classifiers to ensemble learning [2]. At the data level, current mainstream solutions process datasets to reduce the quantitative differences between different categories. The commonly used methods include undersampling [3], oversampling [4], or a combination of oversampling and undersampling [5], but these methods are prone to overfitting problems or loss of potentially useful information. In recent years, deep learning technology has continued to develop iteratively, and deep generative models for data generation are also extensively used. It has obtained excellent results in many practical scenarios, such as images, videos, texts, and voices. Resolving class imbalance in datasets using deep generative models. The above shortcomings can be avoided and the minority class sample data can be better expanded.

Therefore, we consider applying deep generative models and undersampling methods to the samples in the dataset. We expect to alleviate the bias problem caused by imbalanced datasets and improve the model's detection performance for minority classes. To achieve this goal, we propose a balanced dataset approach that combines the DCAES generative model, DBSCAN cluster undersampling [6], and the Tomek Links algorithm [7]. Specifically, this article makes the following contributions:

1. We design a DCAES generative model using convolutional autoencoder and deep oversampling, and we generate high-quality minority class attack samples by training the model.
2. We apply DBSCAN clustering undersampling to reduce superfluous samples in majority class data. And we utilize Tomek Links to enhance the boundaries between categories and reduce category overlap.

3. We conducted a classification comparison experiment. The effectiveness and superiority of our method compared with other data balancing methods are analyzed in detail.

The rest of the paper is arranged as follows. Section 2 reviews data balancing method. Section 3 outlines our balancing strategy. In Sect. 4, experimental and evaluation results are described. Finally, Sect. 5 draws conclusions.

2 Related Work

We review relevant prior work in this section, including various sampling methods as well as deep learning techniques for imbalanced data in network intrusion detection.

2.1 Sampling Methods for Imbalanced Data

With the beginning of the age of big data, Massive amounts of data are pouring into the network, showing high-dimensional and unbalanced feature. Before intrusion detection, we need to do some balancing of the dataset. The easiest way is to synthesize new samples. The Synthetic Minority Over-sampling Technique (SMOTE) [8] is one of the most prominent oversampling techniques. This method generates new samples by randomly interpolating between adjacent samples. But it introduces an element of uncertainty regarding the selection of nearest neighbor interpolation points, which may result in the obscuration of sample boundaries. The Borderline-SMOTE (B-SMOTE) method, as described in literature [9], exclusively focuses on class instances at the decision boundary between categories, solving the blindness of smote, but losing the information contained in the other minority class samples. The literature [10] proposed the ADASYN algorithm. This algorithm mainly adjusts the number of generated samples based on the density of minority class samples. However, if the minority class samples are too sparse or have too much noise, too many noise samples may be generated, leading to model overfitting.

2.2 Deep Learning Techniques for Imbalanced Data

Deep generative models have gained favor in various fields in recent years, including variational autoencoders (VAE), generative adversarial networks (GAN), etc. These models have gained widespread attention and applications in different fields, such as image processing, speech analysis, and text generation. The literature [11] systematically established a data-driven research solution, mainly using various VAE models to balance data. They used the CSE-CIC-IDS2018 dataset to conduct experiments and verified effectiveness of these data processing schemes. Dablain D et al. [12] pointed out in their research that DeepSMOTE is a unique data enhancement method targeting the data feature space. It first encodes the data into the feature space to obtain the feature vector, and then

feeds the feature vectors into the decoder. The vector is decoded to obtain a new image, which is an efficient oversampling solution. Ullah I et al. [13] established an IoT anomaly detection framework based on conditional GAN (cGAN), through which the distribution of minority class data is learned, and then the minority class data is generated to improve the class imbalance in the dataset.

The above research conducted on the classification problem in the presence of data imbalance has yielded some results. However, several shortcomings still exist in the research, Especially for minority and unknown types of attack detection.

3 Proposed Method

Our method can be carried out according to the steps in Fig. 1. As per the procedure, a balanced dataset can be acquired in order to train the classifier. Subsequently, classification and detection operations can be executed on the test set. Each module (process) and the proposed method will be detailed below.

3.1 DCAES Based Minority Class Generating

The DCAES generative model is a highly intricate construct with a multitude of interdependent components contributing to its efficacy. To ensure seamless interplay among these components, the model employs an encoder/decoder framework that is adept at processing complex input samples, thus forming the model's backbone and enabling the generation of desirable outputs. Central to the model's operations is a SMOTE-based oversampling method, which enhances the complexity of its output. Additionally, the DCAES generative model incorporates a reconstruction loss and a penalty loss for the samples, which work synergistically to optimize the model's performance. These various components coalesce to generate high-quality data samples, conferring on the model a degree of generalization that is often absent in other generative models.

DCAES Main Framework. The basic network of the DCAES generative model draws on the Deep Convolutional Generative Adversarial Network (DCGAN) [14] architecture, which was originally proposed by the famous researcher Radford. The DCGAN cleverly combines CNN and GAN, which improves sample oscillation and model instability to a certain extent. The encoder/decoder framework of the DCAES generative model also mirrors the characteristics of the discriminator/generator in the DCGAN architecture. Essentially, an encoder takes input data and transforms it into hidden latent variables that capture the most salient information, while a decoder deciphers latent variables from hidden layers to produce the final output. A schematic illustration of the DCGAES generative model is presented in Fig. 2.

The training process of DCAES generative model involves two distinct and complex parts. The first part requires the encoder to batch-extract minority class attack samples from the dataset. They are then utilized to learn the underlying reconstruction patterns inherent to minority class samples so that the decoder

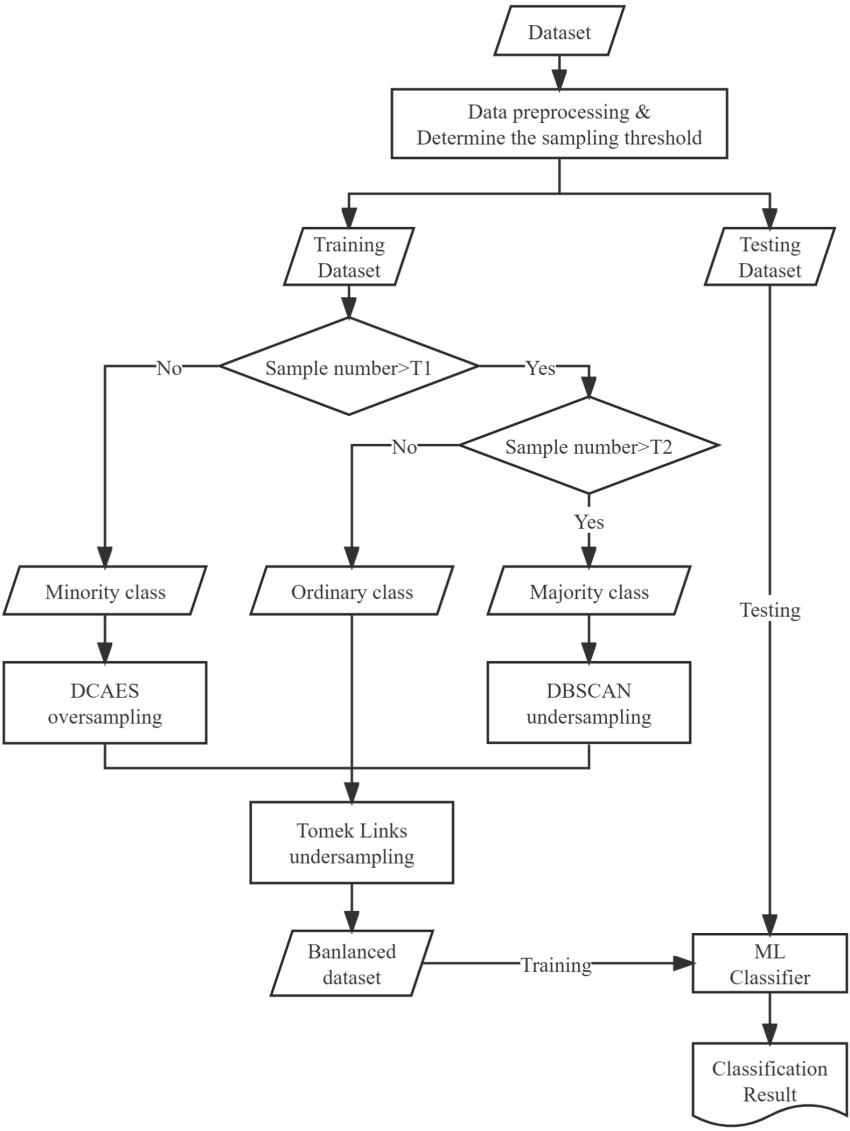


Fig. 1. Flowchart of the proposed data balancing method.

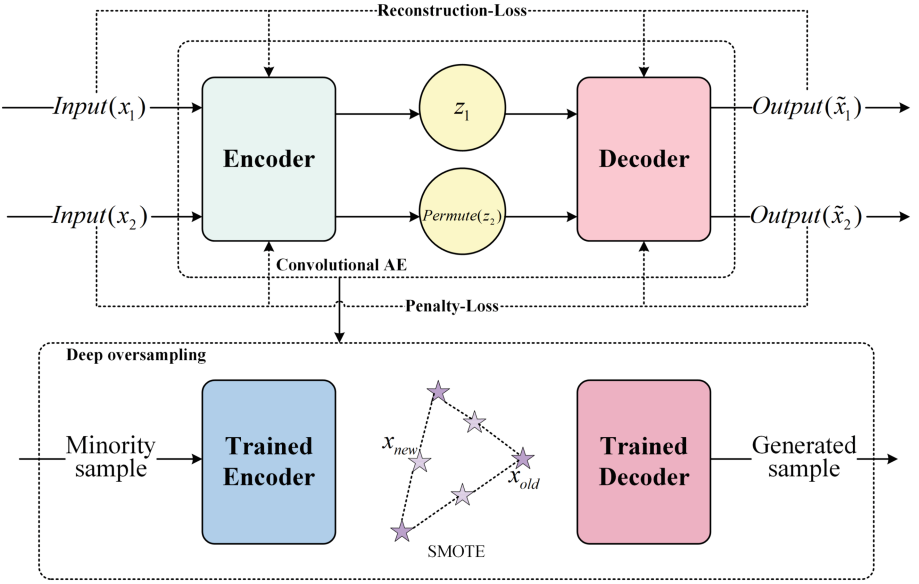


Fig. 2. Schematic diagram of the DCAES generative model

can efficiently reconstruct minority class attack samples. The reconstruction loss function is shown in formula (1).

$$R_L = Smooth_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{1}$$

where $x = D_b - b_j$, b_j is the original data sample, E_b is the encoded sample and D_b is the decoded sample.

In this reconstruction process, the loss is calculated for each batch of samples and reconstructed samples, ensuring that the generated sample distribution remains consistent with the original sample distribution throughout the training process.

The second part of training is more complicated. Because DCAES needs to consider Smote at the feature level, the Smote algorithm is directly placed between the Encoder and Decoder for sample generation. However, the generated samples must still adhere to the constraints imposed by the input data. To this end, the Permute Sample Order operation is introduced in the process of training the model as an 'approximate replacement' of SMOTE to generate variance, thereby training an encoder-decoder model with sufficient decoding capabilities. Then, use SMOTE to replace the Permute Oder operation during the inference process. In this way, new minority class samples can be synthesized.

This change in decoding order results in discernible differences between encoded and decoded samples, distinct from the differences experienced during sample reconstruction. This difference is crucial for the generation of minority

class samples during oversampling and can be represented by a penalty loss. The complexity of the whole process highlights the importance of DCAES generative model training in high-quality sample generation. The penalty loss function is shown in Eq. (2).

$$P_L = Smooth_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{2}$$

where $x = D_S - S_b$, S_b is the original data sample, E_S is the encoded sample, P_E is the disordered encoded sample and D_S is the decoded sample.

Finally combining the loss function of the reconstructed sample and the penalised loss function after introducing the variance gives the total loss function as shown in Eq. (3).

$$T_L = R_L + P_L \tag{3}$$

Algorithm 1 presents a detailed process for DCAES generative model training and new sample generation. Where M_i indicates a minority class in M, b_j indicates the j-th batch in M_i.

Deep Oversampling. The deep oversampling process also needs to introduce variance, but the method used is different from the training phase. During the training phase, the Permute Sample Order operation is introduced to generate variance. However, when it comes to deep oversampling, the SMOTE oversampling method is used to introduce variance. The deep oversampling process uses the trained DCAES generative model. Then the trained encoder can be used in the oversampling process to obtain high-quality embeddings, facilitating the discovery of optimal data representations for oversampling. The SMOTE method generates data on the best data and finally decodes the generated data through a trained decoder to generate high-quality minority class attack samples. It can thus be said that deep oversampling enhanced by the SMOTE oversampling method represents an innovative and effective approach to acquiring high-quality embeddings and generating high-quality minority class attack samples.

3.2 DBSCAN Based Undersampling

After processing the minority class, we also need to undersample the majority class samples to remove redundant samples. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is an algorithm for density clustering that is employed to group data points exhibiting high density and detect noise points situated in low-density regions. Clustering is executed according to the density of data points; the number of clusters is not predetermined, and clusters of any shape can be identified. Cluster class samples are therefore similar to some extent, and samples within clusters can be sampled proportionally to achieve the goal of removing redundant samples while preserving the diversity of cluster class samples.

Algorithm 1: DCAES training and new sample generation

Input: minority class sample sets $M = \{M_i, i = 1, 2, ..., n\}$, a minority class $M_i = \{b_j, j = 1, 2, ..., l\}$

Output: Generated minority class samples G_m

Train the Encoder/Decoder;
for $e \rightarrow epochs$ **do**
  for $i = 1 \rightarrow n$ **do**
    for $j = 1 \rightarrow l$ **do**
      $E_b \Leftarrow encode(b_j)$;
      $D_b \Leftarrow decode(E_b)$;
      $R_L = Smooth_{L_1}(x)\ where\ x = D_b - b_j$ //Reconstruction Loss;
      $S_b \Leftarrow randmly\ sample\ |b|\ instances\ from\ M_i$;
      $E_S \Leftarrow encode(S_b)$;
      $P_E \Leftarrow permute\ order(E_S)$;
      $D_S \Leftarrow decode(P_E)$;
      $P_L = Smooth_{L_1}(x)\ where\ x = D_S - S_b$ //Penalty Loss;
      $T_L = R_L + P_L$;
      $\theta := \theta - \alpha \frac{\partial}{\partial \theta} T_L$;
    end
  end
end
Generate Samples;
for $i = 1 \rightarrow n$ **do**
  $E_m \Leftarrow encode(M_i)$;
  $O_m \Leftarrow SMOTE(E_m)$;
  $G_m \Leftarrow decode(O_m)$;
end

The DBSCAN algorithm has two important parameters for dividing sample clusters: Eps (neighborhood radius) and $MinPts$ (minimum number of points in the neighborhood). These two parameters can be used to define the ε-neighbourhood of p with point p as the center.

$$N_{Eps} = q \in D \quad and \quad dist(p, q) < Eps \tag{4}$$

The object's database is denoted as D, and the mutual distance between points p, q is represented by $dist(p, q)$. The core point is defined as the number of points within the radius Eps of point P that exceeds $MinPts$.

$$N_{Eps}(P) > MinPts \tag{5}$$

DBSCAN checks the number of data points in a neighborhood centered on a point and with a radius of Eps. A core point requires that the number of points in its neighborhood is at least $MinPts$. If the number of points in the neighborhood is less than $MinPts$ but greater than 0, the point is a boundary point and does not belong to the core point. Points that are neither core points

nor boundary points are noise points. Repeat the above steps until all points have been visited.

3.3 Tomek Links Based Undersampling

After removing redundant samples, the class distributions also overlap. Part of the majority class samples are embedded in the dense area of minority class samples, resulting in blurred boundaries and difficult classification. Tomek links refer to the nearest neighbor pairs between two samples that are adjacent in the feature space but belong to different categories. For each sample point, calculate the distance between it and all other sample points and find the nearest heterogeneous sample. Some of these nearest neighbors will form Tomek links. Then, specific samples within it are deleted based on the rules of the Tomek links. The purpose of this is to enhance the boundaries between categories and reduce category overlap, thereby making classification easier.

3.4 Classification and Detection

Following the above steps, we first preprocess the dataset, then set the threshold T1 to 5000 and T2 to 20000, and divide the UNSW-NB15 training set into three parts: the minority class set, the ordinary class set, and the majority class set. Using the above methods to process each sample set, a relatively balanced dataset with fewer noisy samples can be obtained. Then, merge the three into a new dataset and input them into the classifier for training. Then, classify and detect the test set.

4 Experiments and Evaluations

We have used Python and the machine learning library scikit-learn to conduct the experiments in a computer with the configuration of Intel CORE i5-12400 CPU@2.50 GHz, 32 GB memory, CUDA 11.0-enabled NVIDIA GeForce RTX 3060 12 GB GPU driver and Windows10 operating system.

4.1 Dataset Description

We use the UNSW-NB15 dataset. Its raw traffic was created by the IXIA PerfectStorm tool at the Australian Centre for Cyber Security (ACCS) Cyber Scope Lab to generate a realistic mixture of normal traffic and attack behavior. Table 1 shows all categories and specific quantities of the UNSW-NB15 dataset.

There are 175341 samples in the UNSW-NB15 training set and 82332 samples in the UNSW-NB15 test set. Each sample has 45 features containing traffic and packet-based features, two of which were category-labeled features indicating attack and normal types.

Table 1. UNSW-NB15 dataset class distribution details.

Class	Training set	Testing set
Normal	56000	37000
Fuzzers	18184	6062
Analysis	2000	677
Backdoor	1746	583
DoS	12264	4089
Exploits	33393	11132
Generic	40000	18871
Reconnaissance	10491	3496
Shellcode	1133	378
Worms	130	44
Total	175341	82332

4.2 Data Preprocessing

Numerical Characterisation. In the UNSW-NB15 dataset, there are some character-based features, such as the "proto" attribute, which contains 11 different character features. The model cannot process character data directly and hence requires conversion into numerical features to be used in computations. One-hot encoding is used to preprocess character-based data and convert each character feature into a binary encoding with a length of 11 bits.

Data Normalisation. If the unnormalized data is directly input into the model, it is easy to make the features with a large range of values have higher weights, making them the dominant attributes. However, data with a smaller value range tends to obtain smaller weights, resulting in feature loss. To facilitate arithmetic processing and the elimination of different magnitudes, the min-max normalization method is used to map the range of each feature uniformly and linearly in the [0,1] interval. The formula was calculated as:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{6}$$

where x' is the normalised value, x is the original value, $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of the attribute respectively.

Digitisation of Labels. Use sequential coding to convert the 10 category labels in the dataset into numerical labels.

4.3 Data Balancing Process

The huge UNSW-NB15 dataset suffers from serious and obvious class imbalance. This will cause the model to show certain model preferences for certain categories

during model training. There are very few minority classes in the UNSW-NB15 dataset, so we deployed DCAES generative models for the minority classes in the dataset, generating up to 10000 samples for each of the minority classes, namely Analysis, Backdoor, Shellcode, and Worms. In order to keep the total number of training sets roughly unchanged, we performed cluster undersampling on the majority class. The data distribution in the Generic class cluster is relatively concentrated and the sampling ratio is larger. The distribution of Normal and Exploits cluster data is not concentrated, and the sampling ratio is small. Through the above steps, we successfully reduced the majority class samples. Finally, to further refine the dataset, we apply the Tomek Links method in all categories to remove noisy samples from the dataset. We can see the details of each category after balancing from Table 2.

Table 2. Distribution of each class before and after balancing the training set.

Classes	Training set	CIR	Balanced set	CIR
Normal	56000	1	47926	1
Fuzzers	18184	0.3247	15915	0.3321
Analysis	2000	0.0357	10000	0.2087
Backdoor	1746	0.0312	9848	0.2055
DoS	12264	0.2190	11455	0.2390
Exploits	33393	0.5963	28300	0.5905
Generic	40000	0.7143	17818	0.3718
Reconnaissance	10491	0.1873	9417	0.1965
Shellcode	1133	0.0202	9689	0.2022
Worms	130	0.0023	9969	0.2080

In the UNSW-NB15 dataset, the Class Imbalance Rate (CIR) is the ratio of the number of classes to the number of Normal, which can well reveal the imbalance in the dataset. In Table 2, the majority class Generic ratio in the UNSW-NB15 training set is as high as 0.7143, while Exploits follows closely at 0.5963. On the contrary, the ratios of the minority classes Analysis, Backdoor, Shellcode, and Worms are almost negligible, which are 0.0357, 0.0312, 0.0202, and 0.0023, respectively. The dataset is extremely unbalanced. However, after working hard to balance the training set, the class imbalance ratios are significantly and substantially improved. Especially for the minority classes, where they all reach around 0.2, greatly improving the dataset class imbalance.

4.4 Evaluation Metric

In order to conduct a more nuanced assessment of DCAES, the measurement techniques have been categorized into two distinct types: single-type traffic measurement and overall measurement. The evaluation metrics for a certain category

of traffic encompass Precision, Recall, and F1-score. Precision evaluates the system's capability to accurately identify just the assaults, while Recall assesses the system's ability to detect all the attacks. F1-score takes both into account and is a comprehensive indicator.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{9}$$

Because the test set is unbalanced, we also evaluate the performance of DCAES from a macro perspective. Including Macro_Precision (Macro_P): the average precision across multiple classification tasks; Macro_Recall (Macro_R): the average recall across multiple classification tasks; Macro_F1: the average F1-score across multiple classification tasks.

$$Macro\_P = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{10}$$

$$Macro\_R = \frac{1}{n}\sum_{i=1}^{n} R_i \tag{11}$$

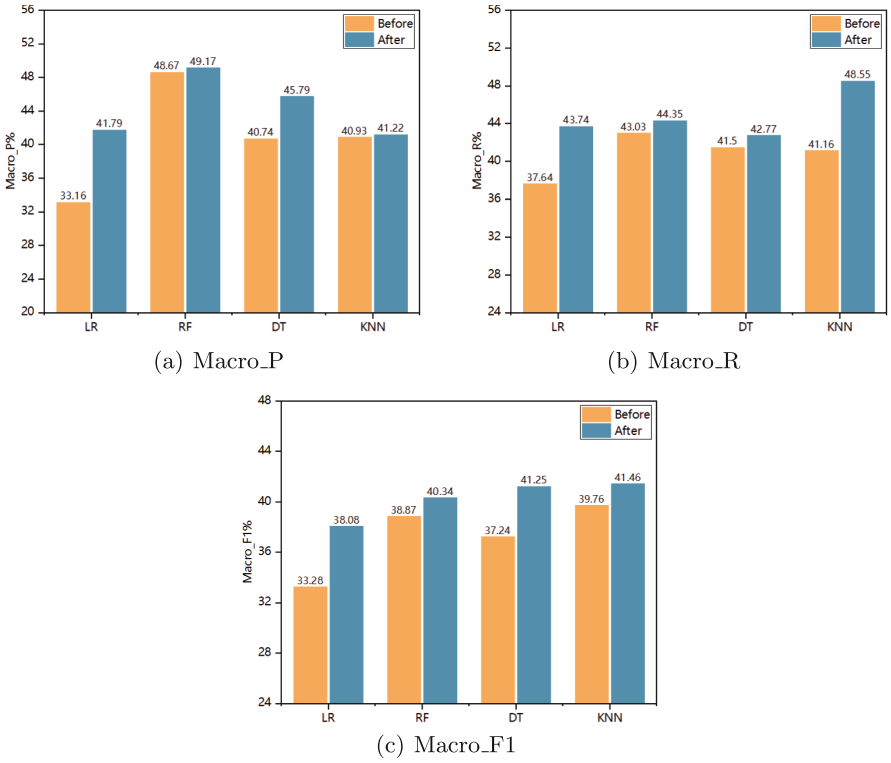
$$Macro\_F1 = \frac{1}{n}\sum_{i=1}^{n} F1_i \tag{12}$$

where n denotes the number of categories in the dataset and P_i, R_i, $F1_i$ are P_i, R_i, $F1_i$ are the specific indicators of the i-th class.

4.5 Analysis of Results

Experimental Comparative Analysis After Data Balance. To evaluate the effectiveness of our proposed approach, we use widely used machine learning classifiers such as logistic regression (LR), random forest (RF), decision tree (DT), and K-nearest neighbors (KNN). Each classifier is trained separately, and then classified and detected on the test set.

The results in Fig. 3 show significant enhancements for each classifier on the metrics Macro_P, Macro_R, and Macro_F1. It is worth noting that the DT classifier has a higher improvement on Macro_P and Macro_F1 by 5.05% and 4.01% respectively. The KNN classifier has a 7.39% improvement in Macro_R. The classification performance of the RF classifier is also improved after dataset balancing, but the improvement is not outstanding compared to other classifiers. The basic performance of the LR classifier is obviously poor, so after the data is balanced, all the indicators have been greatly improved to varying degrees.



(a) Macro_P (b) Macro_R

(c) Macro_F1

Fig. 3. Classification results of the base classifier after using the data balancing method: (a) Macro_P; (b) Macro_R; (c) Macro_F1.

Table 3. Performance of the voting classifier (DT+KNN) using the data balancing method.

Classes	Precision(%)		Recall(%)		F1-score(%)	
	Raw	Balanced	Raw	Balanced	Raw	Balanced
Normal	91.64	**92.38**	78.37	76.28	84.49	83.56
Fuzzers	24.02	**25.34**	44.84	**46.32**	31.28	**32.76**
Analysis	4.08	**7.13**	9.16	**31.76**	5.65	**11.64**
Backdoor	4.75	**9.03**	8.92	**46.31**	6.20	**15.11**
DoS	29.01	**34.59**	30.79	24.85	29.87	28.92
Exploits	64.64	**69.83**	65.68	62.68	65.15	**66.06**
Generic	99.92	**99.96**	96.22	96.18	98.04	98.03
Reconnaissance	60.45	**66.86**	47.97	**66.13**	53.49	**66.49**
Shellcode	48.72	**57.46**	20.11	**20.37**	28.46	**30.08**
Worms	50.00	**58.33**	6.82	**15.91**	12.00	**25.00**
Macro avg	47.72	**52.09**	40.89	**48.68**	41.46	**45.77**

Considering the inherent unique strengths of each classifier, we combined the classification strengths of DT and KNN to create a powerful voting classifier. Use the voting classifier to carry out the classification experiment again.

It can be found in Table 3 that the overall classification performance of the voting classifier exceeds that of a single classifier. After data balancing, Macro_P, Macro_R, and Macro_F1 increased to 52.09%, 48.68%, and 45.77%, respectively, which were significantly increased by 4.37%, 7.79%, and 4.31% before data balancing. This result shows that voting classifiers can combine the strengths of different classifiers for voting decisions.

The classification results of each class show that their Precision has improved to varying degrees, and the Recall of the minority classes Analysis and Backdoor have increased significantly by 22.60% and 37.39%, and the Recall of the minority classes Shellcode and Worms have increased respectively by 8.09%, 9.09%, their F1-score also increased by 5.99%, 8.91%, 1.62% and 13.00% respectively. The improvement of minority class classification performance proves that the minority class samples generated by DCAES conform to the original data distribution and can effectively expand the dataset. Unfortunately, the Recall for the majority class has dropped slightly as a consequence of undersampling. Nevertheless, as the Precision improves, when we turn our attention to the final F1-score, the difference between this metric and its counterpart in the original dataset is small. This shows that even though the majority class is undersampled, most of the information is still preserved. In summary, Our method effectively improves the data imbalance problem and improves the detection rate of minority classes.

Ablation Experiment. To further examine the efficiency of the proposed data balancing approach, three modules of the method are verified in this work. Table 4 shows the experimental results of each method.

Table 4. Comparison of ablation results of modules in data balancing method.

Method	Module			Macro avg(%)		
	DCAES	DBSCAN	Tomek Links	Macro_P	Macro_R	Macro_F1
1	×	×	×	47.72	40.89	41.46
2	✓	×	×	51.14	48.34	45.46
3	×	✓	×	51.59	41.73	42.29
4	×	×	✓	51.41	42.17	42.49
5	✓	✓	✓	**52.09**	**48.86**	**45.77**

As can be seen from Table 4, each indicator under Macro_average are improved after each module is applied compared with none of the methods. When using the DCAES generative model, the improvement effect is the largest, and the Macro_R value is increased by 7.45%. Macro_P increased by 3.87% when using DBSCAN. Macro_R increased by 1.28% when using Tomek Links. The experimental results show that the three modules applied to data balancing at the same time have a better detection effect than using one module alone.

Experimental Comparison Analysis with Other Data Balancing Methods. A balanced treatment of datasets is crucial in network intrusion detection. Many papers have delved into this topic, offering a variety of approaches. Among them, ADASYN, B-SMOTE, and CGAN [15] have received special attention. We make a comparative analysis of the above-mentioned dataset balance methods. To comprehensively evaluate the efficacy of each method in addressing the inherent scarcity challenges in minority classes (i.e., Analysis, Backdoor, Shellcode, and Worms), we use the F1-score as a baseline for comparison, and the classifier is a voting classifier (DT+KNN).

Table 5. Comparison of F1-score for minority class with different data balancing methods.

Model	Analysis(%)	Backdoor(%)	Shellcode(%)	Worms(%)
Raw	5.65	6.02	28.46	12.00
ADASYN	8.01	6.49	17.91	11.92
B-SMOTE	9.63	7.87	22.54	17.70
CGAN	1.81	7.93	**37.81**	17.87
DCAES	**11.64**	**15.11**	30.08	**25.00**

According to the experimental results in Table 5, we dominate with the highest F1 score in the minority classes (analytics, backdoors, and worms). Reaching 11.64%, 15.11% and 25.00% respectively. The CGAN method leads only on the minority class Shellcode with an F1 score of 37.81%. However, compared with our method, CGAN is not comprehensive enough to deal with imbalanced datasets and even its performance on Analysis is the worst. The ADASYN and B-SMOTE methods also show some improvements when applied to the original dataset. The F1-score shown by Analysis and Backdoor have risen to varying degrees. The opposite is true for Shellcode, whose F1-score dropped by 10.55% and 5.92%, respectively. New samples for these methods are generated by interpolation of old samples. Therefore, they cannot capture complex feature information.

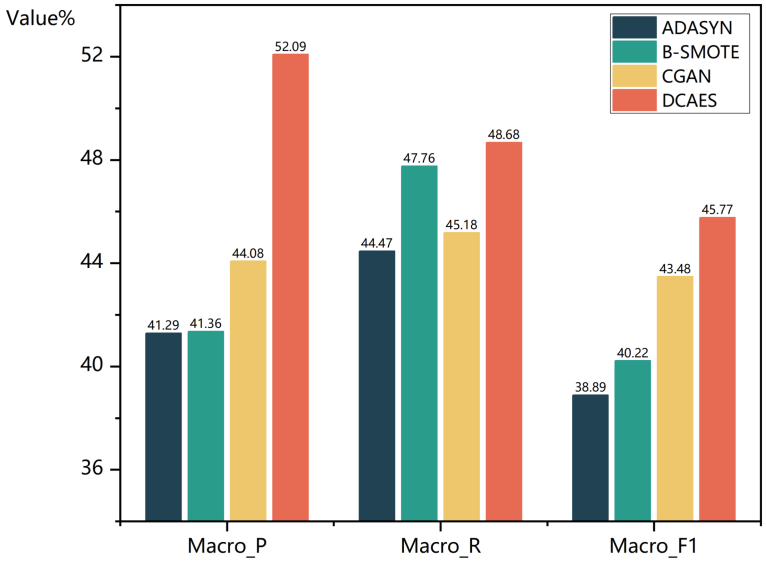


Fig. 4. Comparison of Macro_P, Macro_R, Macro_F1 with different data balancing methods.

Then we compared the performance metrics Macro_P, Macro_R and Macro_F1 of these methods. From Fig. 4, we can see that CGAN outperforms ADASYN and B-SMOTE on Macro_P and Macro_F1. This is because deep generative models can capture the high-order structure and characteristics of data, and generate new representative samples. The B-SMOTE method outperforms ADASYN and CGAN in terms of Macro_R. This method can be attributed to the implementation of the nearest neighbor linear interpolation strategy for boundary samples, which makes the synthesized new samples more likely to reflect the distribution of real data, helping to improve the generalization ability of the model.

However, our proposed method outperforms all compared methods in every metric. The key strengths of our approach are that, on the one hand, it compensates for the inherent shortcomings of interpolation-based sampling methods in conventional oversampling, and on the other hand, it leverages the superior ability of deep generative models to fit samples. The new samples generated by our DCAES generative model greatly improve the imbalance of the dataset. In summary, our method stands out among the comparison methods.

5 Conclusion

Uneven data distribution in network intrusion detection leads to low minority class detection rate, which is a common problem. We proposed a generative model based on convolutional autoencoder and deep oversampling to generate

minority class samples. Compared with existing data generation methods, our model can learn more fine-grained sample features, combined with deep oversampling to better sample generation. In order to obtain a relatively balanced and high-quality dataset, we also performed DBSCAN clustering undersampling and Tomek Links undersampling on the majority class in the dataset and eliminated redundant samples and noise samples in the majority class. In simulations, our method performs well on minority classes and has the best detection performance.

References

1. Wang, N., et al.: Search-based cost-sensitive hypergraph learning for anomaly detection. Inf. Sci. **617**, 451–463 (2022)
2. Bhati, B.S., Chugh, G., Al-Turjman, F., Bhati, N.S.: An improved ensemble based intrusion detection technique using XGBoost. Trans. Emerg. Telecommun. Technol. **32**(6), e4076 (2021)
3. Goyal, S.: Handling class-imbalance with KNN (neighbourhood) under-sampling for software defect prediction. Artif. Intell. Rev. **55**(3), 2023–2064 (2022)
4. Li, J., Zhu, Q., Wu, Q., Fan, Z.: A novel oversampling technique for class-imbalanced learning based on SMOTE and natural neighbors. Inf. Sci. **565**, 438–455 (2021)
5. Ding, H., Chen, L., Dong, L., Fu, Z., Cui, X.: Imbalanced data classification: a KNN and generative adversarial networks-based hybrid approach for intrusion detection. Future Gener. Comput. Syst. **131**, 240–254 (2022)
6. Khan, K., Rehman, S.U., Aziz, K., Fong, S., Sarasvady, S.: DBSCAN: past, present and future. In: The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014), Bangalore, India, pp. 232–238. IEEE (2014). https://doi.org/10.1109/ICADIWT.2014.6814687
7. Pereira, R.M., Costa, Y.M., Silla, C.N., Jr.: MLTL: a multi-label approach for the Tomek Link undersampling algorithm. Neurocomputing **383**, 95–105 (2020)
8. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
9. Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005). https://doi.org/10.1007/11538059_91
10. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, pp. 1322–1328. IEEE (2008). https://doi.org/10.1109/IJCNN.2008.4633969
11. Liu, C., Antypenko, R., Sushko, I., Zakharchenko, O.: Intrusion detection system after data augmentation schemes based on the VAE and CVAE. IEEE Trans. Reliab. **71**(2), 1000–1010 (2022)
12. Dablain, D., Krawczyk, B., Chawla, N.V.: DeepSMOTE: fusing deep learning and SMOTE for imbalanced data. IEEE Trans. Neural Netw. Learn. Syst. **34**, 6390–6404 (2022)

13. Ullah, I., Mahmoud, Q.H.: A framework for anomaly detection in IoT networks using conditional generative adversarial networks. IEEE Access **9**, 165907–165931 (2021)
14. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv2015 arXiv:1511.06434
15. Dlamini, G., Fahim, M.: DGM: a data generative model to improve minority class presence in anomaly detection domain. Neural Comput. Appl. **33**, 13635–13646 (2021)