Data Balancing Technique Based on AE-Flow Model for Network Instrusion Detection

Xuanrui Xiong[1], Yufan Zhang[1(✉)], Huijun Zhang[2], Yi Chen[1], Hailing Fang[1], Wen Xu[1], Weiqing Lin[1], and Yuan Zhang[3]

[1] College of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
xiongxr@cqupt.edu.cn, zhanghj@ctbu.edu.cn, s200101041@stu.cqupt.edu.cn
[2] College of Environmental Resources, Chongqing Technology and Business University, Chongqing 400067, China
[3] School of Computing, Chongqing Institute of Engineering, Chongqing 400056, China
zhangyuan@cqie.edu.cn

Abstract. In network intrusion detection, the frequency of some rare network attacks is low, and such samples collected are relatively few. It results in an imbalanced proportion of each category in the dataset. Training the classifier with imbalanced datasets will bias the classifier to majority class samples and affect the classification performance on minority class samples. In response to this problem, researchers usually increase minority class samples and reduce majority class samples to get a balanced dataset. Therefore, we propose a data balancing technique based on AutoEncoder-Flow (AE-Flow) Model. Firstly, we use AutoEncoder (AE) to improve the deep generative model-Flow, obtaining AE-Flow. Then we use it to learn the distribution of minority class samples and generate new samples. Secondly, we use K-means and OneSidedSelection (OSS) algorithms to finish the undersampling of majority class samples. Finally we get a balanced dataset and use machine learning (ML) classifier to finish intrusion detection. We conducted comparative experiments on NSL-KDD dataset. The experimental results show that the balanced dataset obtained by our proposed method can effectively improve the Recall rate on minority class samples and the classification performance on overall samples.

Keywords: Imbalanced data · Deep generative model-Flow · AutoEncoder · Network Intrusion Detection

Supported by the National Natural Science Foundation of China (51808079), Chongqing Research Program of Basic Research and Frontier Technology (cstc2017jcyjAX0470, cstc2017jcyjAX0135), the Science and Technology Research Program of Chongqing Municipal Education Commission (No. KJQN201801908).

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2023
Published by Springer Nature Switzerland AG 2023. All Rights Reserved
F. Gao et al. (Eds.): ChinaCom 2022, LNICST 500, pp. 174–190, 2023.
https://doi.org/10.1007/978-3-031-34790-0_14

1 Introduction

The task of network intrusion detection is to identify the abnormal traffic in network and judge which attack it belongs to. Therefore, network intrusion detection is essentially a multi-classification task. Now classical machine learning classifiers such as Decision Tree (DT), Random Forest (RF), Logistic Regression (LR) and eXtreme Gradient Boosting (XGBoost) are widely used in network intrusion detection [1]- [2]. However, the frequency of various types of intrusions is different in network, so the number of different categories samples in collected network intrusion records varies greatly, which makes the relative datasets have the problem of category imbalance. Because the quantity of some rare attacks is too small, it is difficult for classifier to learn its general characteristics during training. Therefor, it is easy to misjudge minority class samples into majority class during testing, which makes the recall rate of minority class samples especially low.

Training models with imbalanced datasets has always been a challenge for researchers. The traditional ways dealing with imbalanced datasets are oversampling of minority class and undersampling of majority class. The commonly used undersampling methods are generally Random Undersampling and its improvements based on various clustering algorithms. While the oversampling algorithms are generally various improvements based on Synthetic Minority Oversampling Technique (SMOTE) algorithm [3]. But SMOTE is easy to generate redundant samples and noise samples, which will affect the classification performance. With the development of deep learning, many researchers choose deep generative model to generate new samples.

In recent years, Generative Adversarial Network (GAN), Variational Auto-Encoders (VAE) and Flow-based generative model are widely used in the fields of image, speech and text generation. Now both VAE and GAN have been widely used in network intrusion detection domain. In literature [4], the authors used Conditional VAE model to learn the feature of real samples for generating more samples. In this way they rebalanced the data proportion of training datasets. In [5], the authors improved VAE by using overall covariance Gaussian distribution as posterior probability distribution, obtaining the Variational Laplace AutoEncoder (VLAE) model. It enhanced the expressiveness of posterior data and can generate high-quality minority samples. In [6], the authors used conditional generative adversarial network (CGAN) to input samples and their labels simultaneously into its generator network for training, which can specify the category of new samples generated.

The deep generative model uses its powerful learning ability to learn the distribution of samples through training, and generates new samples that conforms to this distribution [7]. The generated samples can effectively expand the quantity and the diversity of original data. Therefore, we select the Flow-based model to generate minority class samples. We propose an AE-Flow model combined with K-means and OSS algorithms to achieve the goal of data balancing, and test the balanced dataset by using ML classifier. The main contributions of this paper are as follows:

1. Aiming at the problem that Flow-based model has a long training time and a huge structure, we introduce AE to simplify it by sharing the tasks of it. In this way we get the AE-Flow model and train it to generate high-quality minority class samples.
2. We use K-means to cluster majority class samples and undersampling from them for removing redundant ones in them. And we use OSS to remove majority class noise samples located near the boundary of minority class samples.
3. A balanced dataset can be obtained by applying aboved methods. The effectiveness of our proposed method is proved by comparing the classification performance of the classifiers before and after the data balancing operation.
4. We compared our proposal with other data balancing methods: SMOTE, VAE, GAN. The experimental results demonstrate the superiority of our proposal.

The structure of this paper is organized as follows: Sect. 2 briefly introduces the theories related to our approach. Section 3 details the proposed method. Section 4 presents the experimental results and analysis, and we conclude in Sect. 5.

2 Backgroud

The method proposed in this paper has mainly used the deep generative model-Flow, the unsupervised model-AutoEncoder. Next, we will introduce their theories and usages in detail.

2.1 Flow-Based Model

The goal of generative model is to use a known probability density model to fit the distribution of real data samples. But the distribution of real sample is complex, so the flow-based model applies a series of invertible transformation to convert a simple distribution to a complex distribution. It means Flow performs multiple invertible transformations on the input samples, making it become a latent variable of a known distribution (generally Gaussian distribution), completing the mapping from real data space to latent variables space [8]. Since each step of transformation in flow model is invertible, when the model training is completed, it can sample a random vector from the latent variable space and transform it to a new data sample through multiple inverse transformations.

If we represent the real sample and its probability distribution as $x \sim p_\theta(x)$, the latent variable and its probability distributions as $z \sim p_\varphi(z)$, and the $p_\varphi(z)$ taking a Gaussian distribution, then the flow model is to require the reversible transformation to make

$$\begin{aligned} z &= F(x) \\ x &= F^{-1}(z) \end{aligned} \tag{1}$$

where F represents the invertible transformation. Because the single-step reversible transformation cannot achieve a strong nonlinearity, but the entire

model needs a strong nonlinear transformation to complete the mapping from x to z. So Flow requires many steps of invertible transformations to be coupled [9]. Hence, F is a combination of multi-step invertible transformations: $F = f_1 * f_2 \cdots * f_n$. Then the mapping from x to z is

$$x = h_0 \overset{f_1}{\leftrightarrow} h_1 \overset{f_2}{\leftrightarrow} h_2 \cdots \overset{f_i}{\leftrightarrow} h_i \cdots \overset{f_n}{\leftrightarrow} h_n = z \tag{2}$$

where h_i is the hidden variable of middle layer output after each step of invertible transformation f_i .

Because the density of z is known as $p_\varphi(z)$,and $z = F(x)$, according to the variation theorem, the probability density function of x can be expressed as

$$p_\theta(x) = p_\varphi(z) |\det \frac{\partial F(x)}{\partial x}| \tag{3}$$

where $\frac{\partial F(x)}{\partial x}$ is the Jacobian matrix of $F(x)$ at x .

The task of Flow is to maximize the log-likelihood probability density $p_\theta(x)$ by updating the parameters of invertible transformations through training, which means

$$\max \log p_\theta(x) = \max \log p_\varphi(z) + \log |\det \frac{\partial F(x)}{\partial x}| \tag{4}$$

So the loss function of Flow is to minimizing the negative log-likelihood as

$$\begin{aligned} L_{Flow} &= -\log p_\theta(x) \\ &= -(\log p_\varphi(z) + \log |\det \tfrac{\partial F(x)}{\partial x}|) \end{aligned} \tag{5}$$

Where $\log p_\theta(x)$ is maximized indirectly by reducing the loss function through model training. However, the computational cost of Jacobian matrix determinant in formula(5) can be very high. If the transformation can be designed to make the Jacobian be an upper or a lower triangular, the determinant can easily be computed as the product of its diagonal terms. In order to make this Jacobian easy to compute, researchers devised different methods to solve this problem. Bengio, the authors of NICE [10] has divided the input samples into two parts by designing an additive coupling layer, and performed the following transformations as

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= x_1 + f(x_2) \end{aligned} \tag{6}$$

where f represents a one-step reversible transformation. This way makes the Jacobian matrix be a triangular matrix, and the diagonal elements are all 1. The whole transformation is also invertible [10]. Its inverse transformation is

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= y_2 - f(y_1) \end{aligned} \tag{7}$$

So far, the two major requirements of flow that transformation is invertible and the corresponding Jacobian determinant is easy to calculate has been solved by this design. Later, Bengio proposed a more generalized affine coupling layer in model Real-NVP [11] to improve NICE.

2.2 AutoEncoder

AutoEncoder is a typical unsupervised neural network model that mainly includes two modules: Encoder and Decoder, which is often used for feature extraction and dimensionality reduction [12]. Its network structure is shown in Fig. 1.

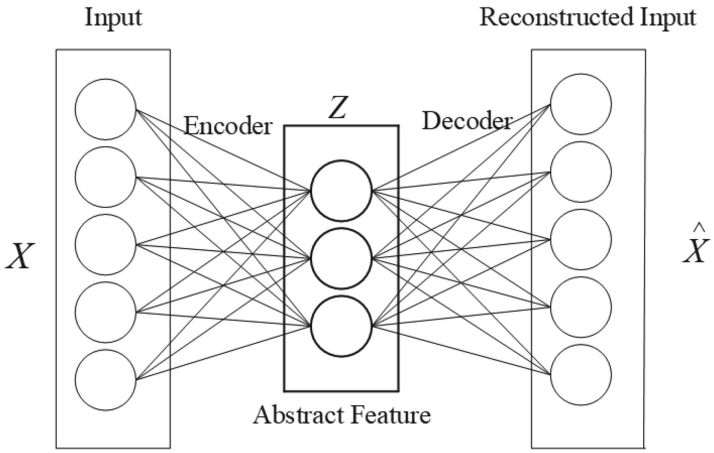

Fig. 1. AutoEncoder model

As shown in Fig. 1, the model maps the input data X to feature space through Encoder to obtain the feature vectors Z, and then maps the feature vectors Z back to original data space through Decoder to obtain the reconstructed samples $\hat{X}$. Its loss function is the reconstruction error between reconstructed data and the original data, which is the mean-square error (MSE) of $\hat{X}$ with X. During model training, the Encoder and Decoder are optimized simultaneously by minimizing the reconstruction error, so as to learn the feature vectors for the input data.

3 Proposed Method

We propose an AE-Flow model to learn the distribution of minority class samples and generate new samples, combined with K-means and OSS algorithms to obtain a balanced dataset. After that, we train the classifier with the balanced dataset, and perform classification detection on test set. The process of the entire network intrusion detection is shown in Fig. 2.

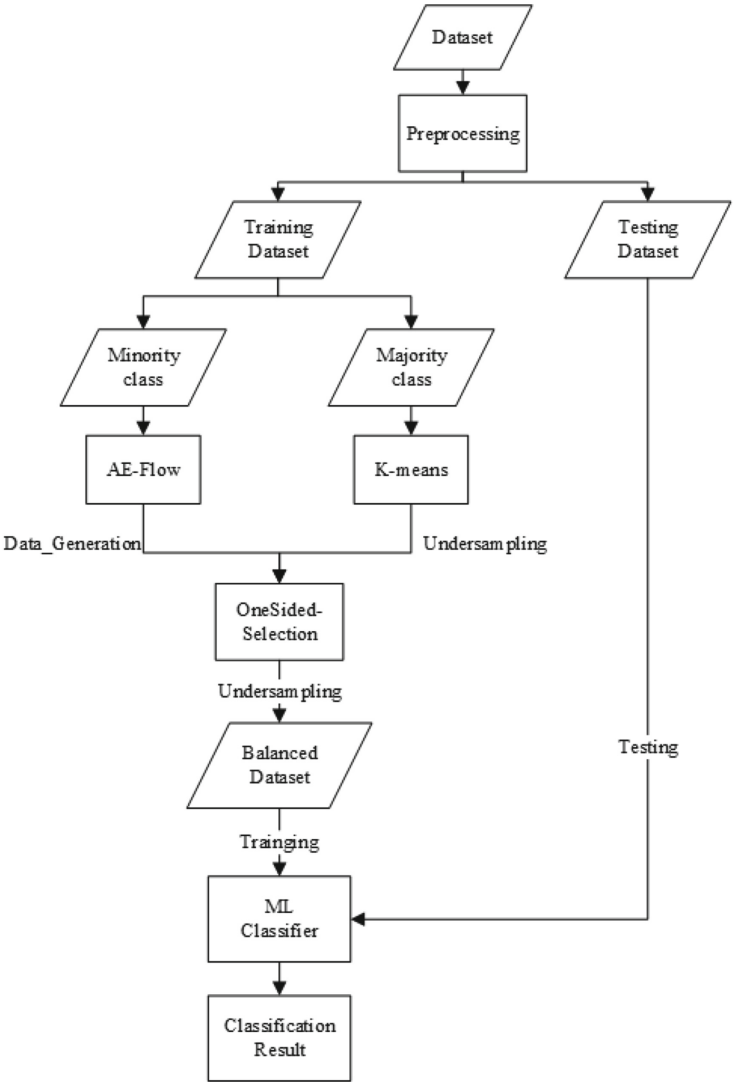

Fig. 2. Flowchart of the proposed network intrusion detection method.

3.1 AE-Flow Based Minority Class Generating

Flow needs multi-step invertible transformations coupled to achieve a strong non-linear transformation for completing the mapping from sample space to latent variable space. Therefor, its model usually has a large structure and the training time is especially long [13]. While network intrusion records are high-dimensional

data with multiple attributes, if we use Flow to generate new samples from original data directly, it needs Flow to complete the mapping from high-dimensional data space to high-dimensional latent variable space. It is such a huge task that will result in a huge model structure and a long training time. Therefore, it is necessary to simplify the Flow model.

Since each step transformation of Flow is invertible, the output latent variable has the same dimension as the original sample [10]. If we use the low-dimensional features extracted from network data as the input of Flow instead of the original sample, since the dimensions of the input and output are same before and after the invertible transformation, the task of Flow has changed as the mapping from the low-dimensional sample feature space to the low-dimensional latent variable space. It is clear that using low-dimensional features as the input of Flow can makes the task much easier, which can allow the Flow to be simplified.

Because AE can learn abstract features z for input samples x, and the dimension of z is lower than x. We propose to introduce AE into Flow to obtain a new generative model AE-Flow. The model structure is shown in Fig. 3.

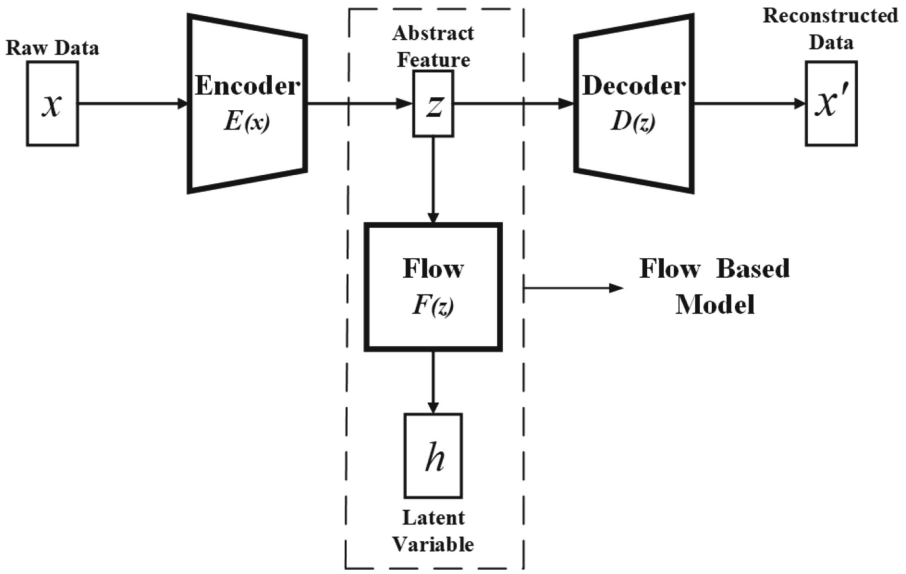

Fig. 3. The proposed AE-Flow model.

As shown in Fig. 3, the data input into AE-Flow firstly pass through the encoder network to become dimensionality-reduced feature vector z. Then on the one hand, through the Flow model, z becomes a latent variable h ,which is of the

same dimension and has a known probability density. And on the other hand, z becomes a reconstructed sample x' through the decoder network.

So the loss function of AE from x to x' is:

$$\begin{aligned} L_{AE} &= ||x - x'||^2 \\ &= ||x - D(z)||^2 \\ &= ||x - D(E(x))||^2 \end{aligned} \tag{8}$$

Where z represents the encoded features of data sample, E represents the encoder network, and D represents the decoder network.

Well the loss function of the Flow model can be expressed as:

$$\begin{aligned} L_{Flow} &= -\log p_\theta(z) \\ &= -(\log p_\varphi(h) + \log|\det \tfrac{\partial F(z)}{\partial z}|) \end{aligned} \tag{9}$$

Where h represents the latent variable that conforms to the Gaussian distribution, F represents the Flow network.

Finally the loss function of the entire AE-Flow model is the sum of L_{AE} and L_{Flow}, which can be expressed as:

$$\begin{aligned} L_{AE-Flow} &= L_{AE} + L_{Flow} \\ &= ||x - D(z)||^2 - (\log p_\varphi(h) + \log|\det \tfrac{\partial F(z)}{\partial z}|) \end{aligned} \tag{10}$$

During model training, AE and Flow are simultaneously optimized by minimizing the loss function $L_{AE-Flow}$. When the training is completed, we can sample a random vector from the latent variable space. And after a series of inverse transformations of Flow, it becomes a low-dimensional feature, then the feature through reconstruction by decoder to becomes a new sample that conforms to the distribution of original data.

We use the proposed AE-Flow model to generate new samples for minority class: R2L and U2L in NSL-KDD dataset, and add them to original training set to complete data augmentation.

3.2 K-means Based Undersampling

Besides expanding the quantity of minority class samples, we also need to eliminate the redundant samples of majority class, to avoid the bias of classifier to this class. We use the K-means algorithm to cluster the majority class samples, and determine the sampling proportion from each cluster according to its density. If the density of a sample cluster is very high, samples in this cluster are with high similarity, and a large number of redundant samples need to be removed. On the contrary, if the density of a sample cluster is very low, it means there are few samples with high similarity in this cluster, so we need to keep them in the whole cluster.

3.3 OSS Based Undersampling

In addition to removing redundant majority class samples, we also need to pay attention to the majority class samples that are at the boundaries of minority class and majority class. Because they are relatively close to the minority class samples, it is easy to interfere with the judgment of minority class for classifier. They are noisy samples for minority class and we need to eliminate them. The OSS algorithm can find the nearest neighbor for each minority class sample located near the class boundary [14]. If its nearest neighbor belongs to majority class, the majority class sample is removed correspondingly. By applying OSS algorithm, we can obtain a dataset with a clear class boundary.

3.4 Classification

Combining the processing of above methods, we obtain a balanced dataset. We use the original training set and the balanced set to train the classifier, and compare the classification performance of classifier on the test set. If the balanced dataset obtained by our proposal can improve the performance of classifier, it proves that our proposal is effective.

4 Experiments and Analysis

4.1 Dataset

We conduct experiments with the NSL-KDD dataset, which is a publicly available dataset in the field of network intrusion detection. This dataset contains 5 categories: Normal, Dos, Probe, U2R and R2L. Each data record has 41-dimensional features and a label. The quantity of each category in the training set KDD Train+ and the test set KDD Test+ is shown in Table 1.

Table 1. The quantity of each category in KDD Train+ and KDD Test+.

Dataset	Normal	Dos	Probe	R2L	U2R	Total
KDD Train+	67343	45927	11656	995	52	125973
KDD Test+	9710	7457	2421	2754	200	22544

It can be seen that the quantity of R2L and U2L samples in training set is far less than that of Normal and Dos. Therefore, R2L and U2R are regarded as minority classes, and Normal and Dos are regarded as majority classes. Training a classifier model with such imbalanced data will bias the model towards majority class, thus requiring processing of imbalanced datasets.

4.2 Data Preprocessing

Numericalization Processing. Each sample in the dataset has 41-dimensional features, including 3 character-type features. Character-type features are inconvenient to input into model to participate in calculation, so they need to be encoded into numerical values. We use the LabelEncoder method to encode 3 character features ("protocol-type", "service", "flag") as numerical features. For example, the "protocol-type" has three attributes: "TCP", "UDP", "ICMP" and they are encoded as "0", "1", "2" respectively. In this way, different attributes can be distinguished by numerical value, and it is easy to input into models to participate in calculation.

Normalization Processing. The 41-dimensional features of original sample data have different value ranges, If these value ranges can be scaled down to around 0 and the variance is 1, the feature value of each data will be greatly reduced, and the operation speed of the entire model will be greatly improved. we choose Z-score standardization method to realize it, and the formula of Z-score is:

$$x^* = \frac{x - u}{\sigma} \tag{11}$$

where x^* is the transformed output value, x is the original data, u and σ is the mean and variance of each feature in original data record.

4.3 Balanced Dataset Processing

Use our proposed AE-Flow model to generate new samples for the two minority class in KDD Train+. We have generated 10,000 new samples for R2L and U2L respectively. Then we perform K-means clustering undersampling on the two majority class-Normal and Dos. The number of Normal has been reduced from 67342 to 16836, and the number of the Dos has dropped from 45972 to 16714. Finally we apply OSS algorithm to the obtained dataset and reduce the number of Normal and Dos data to 16353, 15445 respectively. The data balancing procedure is completed and a balanced dataset is obtained. The number of each category in the dataset before and after data balancing is shown in Table 2. And the proportion of each category in raw dataset and the balanced dataset are shown in Fig. 4.

Table 2. The number of each category in raw dataset and balanced dataset.

Class	Raw dataset	Balanced dataset
Dos	45927	16353
Normal	67343	15445
Probe	11656	11656
R2L	995	11995
U2L	52	11052

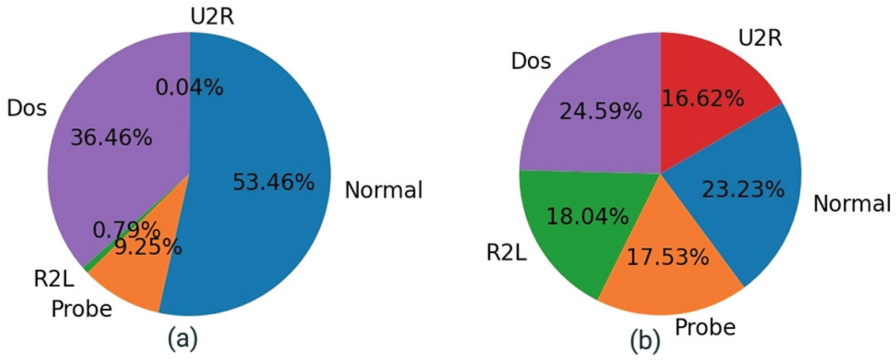

Fig. 4. The proportion of each category in: (a) raw dataset; (b) the balanced dataset.

As shown in Fig. 4, in raw dataset, the proportion of each category is extremely imbalanced. While after our data balancing processing, each category counts a balanced proportion in balanced dataset.

4.4 Evaluation Metric

Network intrusion detection is a multi-classification problem. When judging a category of samples, this category can be regarded as a positive example, and the other types of samples are negative examples, then the classified data can be divided into four types: True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). Therefore, this paper uses Precision, Recall, and F1-score as evaluation metrics. The calculation formulas of them are as follows.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{14}$$

As for the evaluation of overall Precision, Recall, and F1-score of all samples in test set, it is generally calculated by the average of the metric of each category. Now there are two main ways to calculate the average: $Macro$ and $Weighted$. $Macro$ is calculated by arithmetic mean. It means each category has the same weight. $Weighted$ takes the proportion of each category in the entire data set as the weight of this class when calculating the overall mean. Because the quantity of each category in our test set KDD Test+ is still imbalanced (as shown in Table 1), to calculate the average of above metrics on all samples in entire dataset. The formulas are as follows.

$$Weighted\_P = \sum_{i=1}^{n} \frac{Support_i}{Total} * Precision_i \tag{15}$$

$$Weighted\_R = \sum_{i=1}^{n} \frac{Support_i}{Total} * Recall_i \tag{16}$$

$$Weighted\_F1 = \sum_{i=1}^{n} \frac{Support_i}{Total} * F1 - score_i \tag{17}$$

$Support_i$ refers to the quantity of the ith class samples in test set, and $Total$ refers to the total number of all class samples in the test set. $Precision_i$, $Recall_i$, $F1 - score_i$ refer to the Precision, Recall, and F1-score value of the ith class samples. This calculation method can avoid the bad influence of imbalanced data in test set.

4.5 Experimental Results and Analysis

Classification Performance Comparison Before and After Data Balancing. We select the commonly used classifiers in machine learning: DT, RF, LR and XGBoost to verify the effectiveness of our proposed method. These classifiers are trained by original training set KDD Train+ (before data balancing) and the Balanced Dataset obtained in Sect. 4.5(after data balancing) respectively, then tested on KDD Test+. Comparing the $Weighted\_P$, $Weighted\_R$, $Weighted\_F1$ before and after data balancing, the result is shown in Fig. 5.

From Fig. 5, it can be seen that compared with original training set, training the classifier with balanced dataset makes each classifier have a greater improvement in each evaluation metric on the test set, and it has the most obvious improvement on the performance of LR and XGBoost. So we propose to use VotingClassifier [15] to combine the advantages of these two classifiers by making a decision fusion from their classification results. This means they are trained in parallel and the final result is voted upon their output classifications. We conduct comparative experiments on the VotingClassifier (XGBoost+LR). The experimental results are shown in Table 3.

As can be seen from Table 3, compared with the original training set, the Recall of the minority class samples has been greatly improved. The Recall of R2L has increased from 1.38% to 34.50%, and it has increased from 0.5% to 58.50% for U2R. And on the weighted average of overall classification metric, the Precision increased by 3.48%, the Recall increased by 7.37%, and the F1-score increased by 10.89%.

In addition, compared with the experimental results of using LR or XGBoost alone in Fig. 5, the classification performance of VotingClassifier combining them is better than both. This proves that VotingClassifier can combine the advantages of both XGBoost and LR to achieve high-performance network intrusion detection.

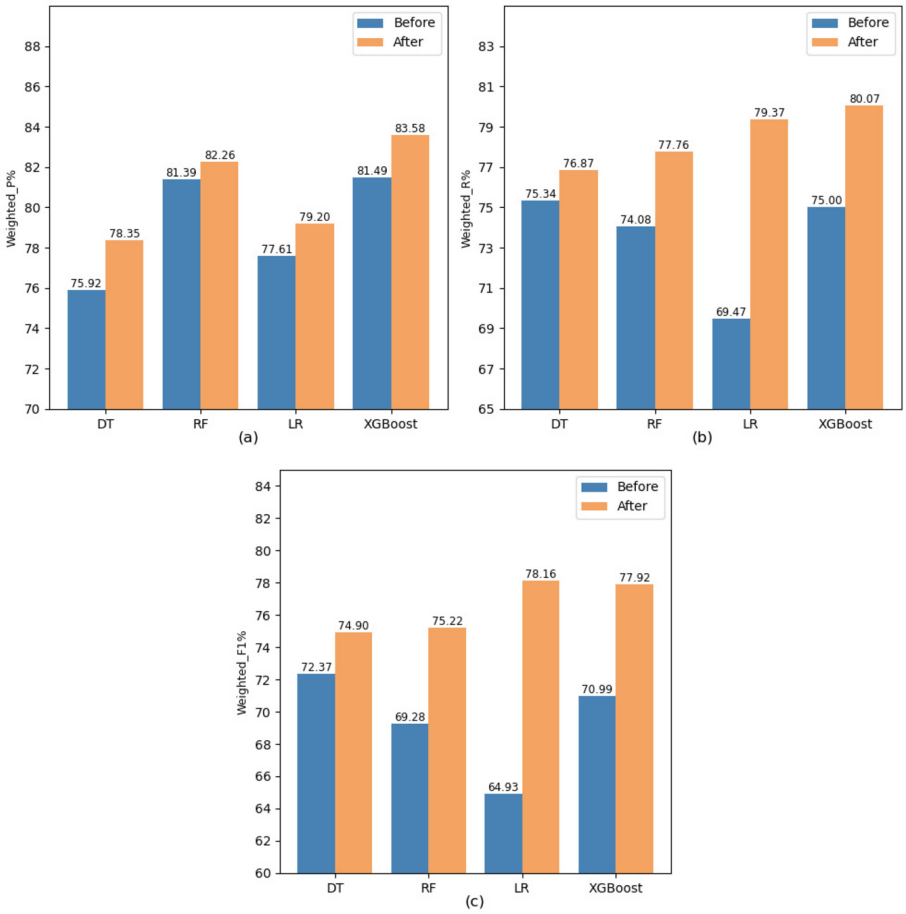

Fig. 5. Classification performance of each classifier on test set before and after data balancing: (a) $Weighted\_P$; (b) $Weighted\_R$; (c) $Weighted\_F1$.

So far, we have obtained a high-performance network intrusion detection system combined AE-Flow data balancing strategy with VotingClassifier (XGBoost+LR).

Comparison with Other Data Balancing Method. We compared the popular data balancing methods in other papers. They are SMOTE [3], VAE [5] and GAN [6]. We choose Precision, Recall, and F1-score of each category as the evaluation metric for comparison. Then the classifier we used is the VotingClassifier (XGBoost+LR). The experimental results are shown in Table 4, Table 5, and Table 6.

From Table 5 we can find that the Recall of Normal data in Raw dataset is higher than the results of other balanced datasets, which is because it accounts

Table 3. The classification performance of VotingClassifier (XGBoost+LR) before and after data balancing.

Class	Precision(%)		Recall(%)		F1-score(%)	
	before	**after**	**before**	**after**	**before**	**after**
Dos	96.61	96.08	77.91	82.45	86.26	88.74
Normal	64.36	74.63	97.32	95.19	77.48	83.67
Probe	80.30	75.93	59.77	**80.50**	68.53	78.15
R2L	97.44	91.97	1.38	**34.50**	2.72	50.17
U2R	33.33	74.05	0.5	**58.5**	0.99	65.36
Weighted avg	80.50	**83.98**	74.29	**81.66**	69.61	**80.50**

Table 4. Comparison of *Precision* with different data balancing method.

Model	**Dos(%)**	**Normal(%)**	**Probe(%)**	**R2L(%)**	**U2R(%)**
Raw	95.80	66.92	**81.23**	39.34	10.74
SMOTE	**97.07**	68.41	76.12	39.64	66.67
VAE	94.95	71.71	75.83	51.11	50.0
GAN	91.71	**77.49**	62.61	62.97	49.02
Proposed	96.08	74.63	75.93	**91.97**	**74.05**

Table 5. Comparison of *Recall* with different data balancing method.

Model	**Dos(%)**	**Normal(%)**	**Probe(%)**	**R2L(%)**	**U2R(%)**
Raw	73.34	**97.09**	67.74	8.24	8.0
SMOTE	77.38	96.75	62.54	12.09	12.0
VAE	78.69	96.32	72.73	17.50	13.5
GAN	76.03	96.56	74.85	30.07	12.5
Proposed	**82.45**	95.19	**80.50**	**34.50**	**58.5**

Table 6. Comparison of $F1$ with different data balancing method.

Model	**Dos(%)**	**Normal(%)**	**Probe(%)**	**R2L(%)**	**U2R(%)**
Raw	83.08	79.24	73.87	13.63	9.17
SMOTE	86.12	80.15	68.66	18.53	20.33
VAE	86.05	82.21	74.25	26.07	21.25
GAN	83.13	85.98	68.18	40.70	19.92
Proposed	**88.74**	**83.67**	**78.15**	**50.17**	**65.36**

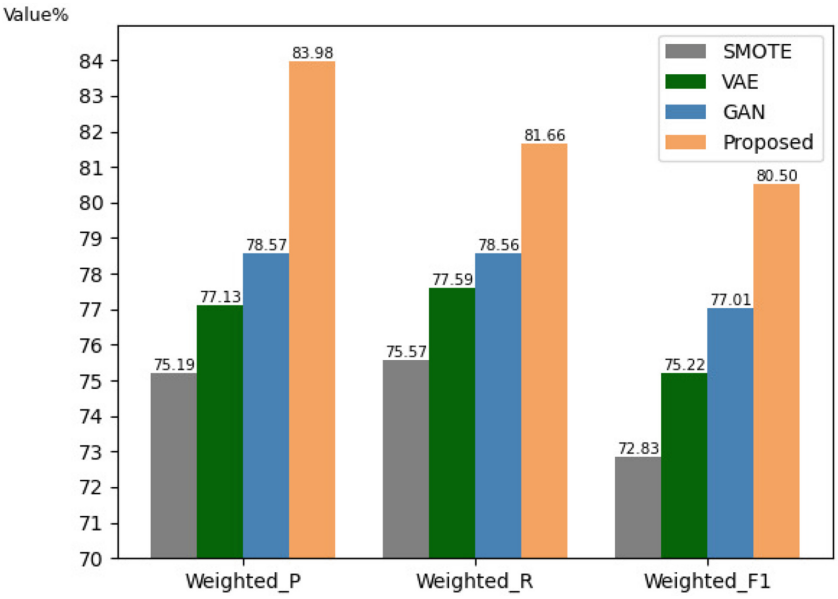

Fig. 6. Comparison of $Weighted\_P$, $Weighted\_R$, $Weighted\_F$ with different data balancing methods.

for the largest number in the original imbalanced data set. This result confirms the phenomenon that training a classifier with an imbalanced dataset biases the classifier towards the majority class. While after applying different data balancing methods, the Recall of the minority class samples R2L and U2R have a certain improvement compared to the raw dataset, moreover our method has the highest improvement.

Then we compared the $Weighted\_P$, $Weighted\_R$, $Weighted\_F$ of each data balancing method. The experimental results are shown in Fig. 6:

As can be seen from Fig. 6, the evaluation results of the data balancing method based on deep generative models (VAE, GAN and AE-Flow) are higher than SMOTE, then GAN outperforms VAE, and our proposal outperforms GAN. It is because SMOTE only synthesizes new samples by "interpolating" sampling between the minority class samples, while the deep generative model can learn the distribution of the original samples to generate new sample data that conforms to this distribution. The result of GAN is better than that of VAE. Because VAE directly assumes the probability distribution of features as a Gaussian distribution, while the distribution of real sample features may be much more complex than Gaussian distribution actually. If the two are far from each other, the VAE will poorly fit the real samples and generate new samples of low quality. GAN distinguishes the real samples with fake samples generated by the generator model through the discriminator model, and modeling data generation (an unsupervised learning problem) as a supervised problem bypasses the challenge

of finding complex distributions of real samples. Our AE-Flow model establishes a probabilistic relationship between real samples and generated samples by training a reversible neural network, and this relationship is deterministic and one-to-one, which is where AE-Flow outperforms VAE and GAN. Therefore, our proposal achieves the best experimental results in the end.

5 Conclusion

Aiming at the problem with poor classification performance of minority class samples caused by the imbalance data in network intrusion detection, this paper proposes a data balancing method based on AE-Flow combined with K-means and OSS. Our method uses AE-Flow to learn the minority class samples for generating this kind of samples, then it uses the K-means algorithm to cluster the majority class samples and completes undersampling. Finally, the OSS algorithm is used to eliminate the majority class samples located near the boundary, and so far a new balanced data set is obtained. Using this dataset as a new training set, the experimental results of the classifier on the test set show that our method can effectively improve the classification performance on the minority class samples and the entire dataset. Comparing with the experimental results of data balancing methods in other papers, our method also outperforms them all.

References

1. Besharati, E., Naderan, M., Namjoo, E.: LR-HIDS: logistic regression host-based intrusion detection system for cloud environments. J. Ambient. Intell. Humaniz. Comput. **10**, 3669–3692 (2019)
2. Dong, R.H., Shui, Y.L., Zhang, Q.Y.: Intrusion detection model based on feature selection and random forest. Int. J. Netw. Security **23**(6), 985–996 (2021)
3. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, P.: SMOTE: Synthetic Minority oversampling Technique. Journal of Artificial Intelligence Research **16**, 321–357 (2002)
4. Yang, Y.Q., Zheng, K.F., Wu, C.H., Yang, Y.X.: Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. Sensors **19**(11), 1–20 (2019)
5. Azmin, S.H., Islam, A.B.: Network Intrusion Detection System based on Conditional Variational Laplace AutoEncoder. In: 7th International Conference on Networking, Systems and Security, pp. 82–87. ACM, Dhaka, Bangladesh (2020). https://doi.org/10.1145/3428363.3428371
6. Dlamini, G., Fahim, M.: DGM: a data generative model to improve minority class presence in anomaly detection domain. Neural Comput. Appl. **33**(20), 13635–13646 (2021). https://doi.org/10.1007/s00521-021-05993-w
7. Liu, X.D., et al.: A GAN and feature selection-based oversampling technique for intrusion detection. Security Commun. Netw. **2021**, 1–15 (2021)
8. Kingma, D.P., Dhariwal, P.: Glow: Generative Flow with Invertible 1x1 Convolutions. arXiv preprint arXiv:1807.03039 (2018)

9. Dinh, L., Krueger, D., Bengio, Y.: NICE: Non-linear Independent Components Estimation. arXiv preprint arXiv:1410.8516 (2014)
10. Rezende, D.J., Mohamed, S.: Variational Inference with Normalizing Flows. In: the 32nd International Conference on Machine Learning, pp. 1530–1538. ACM, Lille, France (2015)
11. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using Real NVP. arXiv preprint arXiv:1605.08803 (2017)
12. Andresini, G., Appice, A., Malerba, D.: Autoencoder-based deep metric learning for network intrusion detection. Inf. Sci. **569**, 706–727 (2021)
13. Grover, A., Dhar, M., Ermon, S.: Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models. In: the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), pp. 3069–3076. AAAI, New Orleans, USA (2018). https://doi.org/10.1609/aaai.v32i1.11829
14. Guo, T., Lu, X.P., Yu, K.P., Zhang, Y.X., Wei, W.: Integration of light curve brightness information and layered discriminative constrained energy minimization for automatic binary asteroid detection. IEEE Trans. Aerosp. Electron. Syst. **2022**, 1–20 (2022)
15. Khan, M.A., Khattk, M.A.K., Latif, S.: Voting Classifier-Based Intrusion Detection for IoT Networks. In: Advances on Smart and Soft Computing ICACIn 2021 (AAAI-18), pp. 313–328. Springer, Casablanca, Morocco (2021). https://doi.org/10.1007/978-981-16-5559-326