Applied Intelligence (2023) 53:12416–12436
https://doi.org/10.1007/s10489-022-03995-2

Network intrusion detection based on conditional wasserstein variational autoencoder with generative adversarial network and one-dimensional convolutional neural networks

Jiaxing He[1] · Xiaodan Wang[1] · Yafei Song[1] · Qian Xiang[1] · Chen Chen[1]

Accepted: 12 July 2022 / Published online: 27 September 2022

Abstract

There is a class-imbalance problem that the number of minority class samples is significantly lower than that of majority class samples in common network traffic datasets. Class-imbalance phenomenon will affect the performance of the classifier and reduce the robustness of the classifier to detect unknown anomaly detection. And the distribution of the continuous features in the dataset does not follow the Gaussian distribution, which will bring great difficulties to intrusion detection. We propose Conditional Wasserstein Variational Autoencoders with Generative Adversarial Network (CWVAEGAN) to solve the class-imbalance phenomenon, CWVAEGAN transform the original dataset through data preprocessing, and then use the improved VAEGAN to generate minority class samples. According to the CWVAEGAN model, an intrusion detection system based on CWVAEGAN and One-dimensional convolutional neural networks (1DCNN), namely CWVAEGAN-1DCNN, is established. By using the examples generated by CWVAEGAN, the problem of intrusion detection on class unbalanced data is solved. Specifically, CWVAEGAN-1DCNN consists of three modules: data preprocessing module, CWVAEGAN, and deep neural network. We evaluate the performance of CWVAEGAN-1DCNN on two benchmark datasets and compared it with the other 16 methods. Experiment results suggest that the performance of CWVAEGAN-1DCNN is better than class-balancing methods, and other advanced methods.

Keywords Intrusion detection system (IDS) · Class-balancing method · Generative adversarial network · Variational autoencoder · Gaussian mixture model

1 Introduction

As the basic infrastructure of modern society, there are also great difficulties to Cyberspace Security. Timely and accurate detection and response to network intrusion are of great strategic significance to cybersecurity. Although many mechanisms have been proposed to improve network defense capability [1], the security of existing methods is still insufficient due to the continuous change of attack forms.

Intrusion detection systems (IDS) adopt the method of active defense, it can detect intrusion and make timely responses. Now, this technology has become an important method to maintain the security of cyberspace. Based on the function position in the network system, IDS can be divided into network-based intrusion detection system (NIDS) and host-based intrusion detection system (HIDS) [2]. IDS is designed to automatically discriminate malicious activities and violations. A variety of IDSs based on machine learning have been studied, such as Naive Bayes [2], and random and support vector machine [3]. These intrusion detection methods achieved good detection results.

But in the actual network activities, the normal traffic and behaviors are dominant, and the number of abnormal behavior is less. The class-imbalance problem between normal activities and attacks and between different attack classes is common in intrusion detection datasets, which greatly affects the detection performance of IDS [4]. Most of the classification methods only focus on the majority of dataset, but ignore minority classes. These models can't learn enough information about minority classes from a limited number of samples, which leads to a bias prediction to the majority

✉ Xiaodan Wang
afeu_wang@163.com

Jiaxing He
hejiaxing@stu.xidian.edu.cn

[1] Air Force Engineering University, Changle East Road, Xi'an City, Shaanxi Province, China

classes. It is crucial for IDS to detect minority classes samples. The loss caused by wrong detection of attack is much more serious than misdetection of normal behavior.

After the class-imbalance problem, intrusion detection datasets also have the problem of multiple modes of continuous columns. Srivastava [5] points out that the vanilla GAN cannot simulate all the mixture components of Gaussian mixture distribution on a 2D dataset, and the learned data distribution is distorted. For example, the common intrusion detection dataset NSL-KDD [6] contains three discrete columns and 38 continuous columns, a total of 41-dimensional feature columns. We use kernel density estimation method to estimate all the continuous columns and found that 22 of 38 continuous columns follow Gaussian mixture model.

To solve class-imbalance problem and multimodal distribution problem of continuous columns in intrusion detection dataset mentioned above, this article proposes a method of generating conditional Wasserstein variational autoencoder generative adversarial network (CWVAEGAN) and one-dimensional Convolutional neural network (1D-CNN). Firstly, CWVAEGAN-1DCNN filters the original dataset and selects the minority classes. Secondly, the continuous columns in the dataset are decomposed by the variational Gaussian mixture model (VGM) [7]; Finally, CWVAEGAN learn the processed dataset and generate new minority data. To rebalance the training dataset and improve the performance of IDS.

This paper has the following contributions

(1) Aiming at the class-imbalance problem in IDS, we propose a new method based on CWVAEGAN to generate minority class samples. One-dimensional convolution layers are applied into the encoder, decoder, and discriminator to enhance the expression ability of the whole network. Therefore, condition information and WGAN-GP [8] loss are introduced [9].
(2) On the basis of CWVAEGAN, CWVAEGAN-1DCNN is proposed, which can detect attacks more accurately. A balanced training dataset is generated by CWVAEGAN, and a one-dimensional CNN network is used for classification, which can more accurately carry out intrusion detection. We establish CWVAEGAN-1DCNN to solve the intrusion detection problem.
(3) Experimental results on two intrusion detection datasets prove that CWVAEGAN-1DCNN is superior to the existing class-balancing methods.

The rest of article is organized as follows. In Section 2, we summarize the development of deep learning in the field of intrusion detection and the related contents of class balancing method. Section 3 introduces the details of CWVAEGAN model and CWVAEGAN-1DCNN. Section 4 introduces the comparative study and ablation experiment, and analyzes the experimental results. Finally, we draw conclusions for our work in Section 5.

2 Related work

In this section, we firstly provide background of IDSs based on deep learning methods and class-balancing methods. Then, we introduce the basic theory of VAEs and GANs.

2.1 IDS based on machine learning

IDS plays an important role in network system. It can detect abnormal behavior in many ways to improve the security of network systems.

Some researches introduce the existing technology to achieve intrusion detection. Panda et al. [2] applied Naive Bayes in NIDS. As a classical machine learning algorithm, Support vector machine (SVM) is also applied in IDS [3, 10, 11]. However, these traditional methods are limited by the lack of data expressing ability, and it is difficult to deal with imbalanced data.

Based on multi-layer perceptron (MLP), Moradi et al. [12] constructed neural networks for IDS.

Li et al. [13] used a convolutional neural network (CNN) to the representation learning of IDS. First the NSLKDD dataset was converted to images, to learn the features of graphical NSL-KDD dataset. Then used ResNet and GoogleNet to carry out intrusion detection. Experimental results indicated that the CNN is sensitive to the images transferred from the attack data. ResNet and GoogleNet achieved 79.14% and 77.04% accuracy on the KDDTest+ and KDDTest-21 test sets, respectively.

Ma et al. [14] proposed an intrusion detection method composed of spectrum clustering and deep neural network, which was called SCDNN. According to the sample similarity, the dataset is divided into k subsets using spectrum clustering. The similarity feature then is used to measure the distance between the data points in the testset and the training set, as well as it is used as the input of the deep neural network algorithm. The recognition accuracy of this method on the KDDTest+ and KDDTest-21 datasets reached 72.64% and 44.55%, respectively.

Yin et al. [15] proposed an IDS based on recurrent neural network. They study the binary classification and multi-classification performances of RNN-IDS. Experiments based on the NSLKDD show that RNN-IDS is an excellent IDS with high accuracy. It achieves 83.28% and 68.55% accuracy on the KDDTest+ and KDDTest-21 respectively.

GAN generates data through confrontation game training, which provides another new method to solve the problem of class-imbalance problem in IDS.

Tama et al. [16] proposed a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system (TSE-IDS). First, the hybrid feature selection technique is used to screen the features, and then a two-level classifier is adopted for classification. The classification accuracy of TSE-IDS has achieved 85.79%, 72.52% and 91.27% on KDDTest+, KDDTest-21 and UNSW-NB15 datasets.

Bedi et al. [17] proposed Improved Siam-IDS (I-SiamIDS), which is a two-layer ensemble. ISiamIDS used an ensemble of binary eXtreme Gradient Boosting, Siamese Neural Network and Deep Neural Network classifiers at the first layer. This layer performs hierarchical filtration of network data into benign and malicious samples. Filtration of incoming data multiple times through different classifiers minimizes the chances of malicious traffic going undetected by I-SiamIDS. The attack samples identified were input to the second layer of I-SiamIDS comprising of multi-class eXtreme Gradient Boosting for classification into four main attack categories. I-SiamIDS was trained and tested using the NSL-KDD datasets, and it achieved 80% accuracy on KDD Test+ dataset.

Khan et al. [18] proposed a novel two-stage deep learning (TSDL) model, based on a stacked auto-encoder, for intrusion detection. The model comprises two decision stages: an initial stage responsible for classifying network traffic as normal or abnormal, and the final decision stage for detecting the normal state and other classes of attacks. Experiments are conducted on KDD99 and UNSW-NB15 datasets, TSDL achieved 89.134% for the UNSW-NB15 datasets. The experiments result on KDD99 dataset achieved 99.996%, this dataset is too old and full of redundant samples.

Yang et al. [19] proposed a intrusion detection model that combines an improved conditional variational AutoEncoder (ICVAE) with a DNN, namely ICVAE-DNN. ICVAE is used to learn potential sparse representations between network data features and classes. ICVAE's decoder generated new attack samples according to balance the training data. The trained ICVAE encoder is not only used to automatically reduce data dimension, but also to initialize the weight of DNN hidden layers. The accuracy of ICVAE-DNN on NSLKDD and UNSWNB15 datasets are 85.97% and 75.43% respectively.

Tian et al. [20] proposed an intrusion detection approach based on improved DBN, called KG-DBN. Probabilistic mass function (PMF) and Min-Max methods are introduced to simplify the process of data preprocessing. The combined sparsity penalty term is introduced into the likelihood function of the unsupervised learning phase of the model. And the sparse distribution of the dataset is obtained through the sparse constraint, inducing the sparse state of the hidden layer neurons and avoiding feature homogeneity and network overfitting. But there exists uncertainty on the selection of DBN parameter values, the detection accuracy may be affected to some extent. The accuracy of KG-DBN achieves 86.49% on UNSWNB15 dataset.

Lee et al. [21] proposed an IDS based on GAN and random forest, called GAN-RF. GAN overcomes the disadvantages of oversampling technology, such as over fitting and class overlapping. Experimental results also show that the GAN-RF shows good performance on imbalanced intrusion detection data. But the model structure of GAN-RF is simple and the experiments are performed on CICIDS-2017 dataset, which all methods can achieve high accuracy. The recognition accuracy of GAN-RF on the CICIDS-2017 achieved 99.93% .

Yang et al. [22] proposed an IDS based on Supervised Adversarial Variational Autoencoder, called SAVAER-DNN. The model uses the decoder of SAVAER to synthesize minority class samples. The encoder of SAVAER was used to extract the advanced features of the original samples. And uesd a DNN for classification.

Huang et al. [23] proposed an IDS based on GAN, called IGAN-IDS. It uses the samples generated by GAN to solve the class-imbalance intrusion detection problem. They use GAN generates samples for minority classes and the deep neural network performs the final intrusion detection. IGAN-IDS only adopted the vanilla GAN, which can only synthesis distorted inadequate minority class samples. The recognition accuracy of IGAN-IDS on the KDDTest+ and UNSWNB15 dataset achieved 84.45% and 82.53% respectively.

The recognition accuracy, detection rate, and F1 score of the methods mentioned above on NSL-KDD (KDDTest+, KDDTest-21) and UNSWNB15 datasets is summarized in Table 1.

2.2 IDS based on class-balancing method

Class imbalance phenomenon is a long-standing problem in the field of machine learning, which also brings great difficulties to IDS. When the data is lopsided, majority classes may "drown" the whole algorithm, while minority classes will be ignored. Common class-balancing methods include random undersampling (RUS) [24], random oversampling (ROS) [24].

Aiming at the class-imbalance problem in IDS, Kuang [25] and Abdulhammed [26] use random under-sampling (RUS) and random over-sampling (ROS) method respectively to solve the class-imbalance problem in IDS. Cieslak [27] further combines the technology of Rus and ROS to detect attacks. However, RUS may lead to the loss of useful information, and ROS is just a simple copy of the original samples, that may lead to overfitting phenomenon. Synthetic minority oversampling technology (SMOTE) [28] performs well in the field of data generation.

Table 1 The recognition accuracy, detection rate, and F1 (%) of the different methods on 3 datasets

Datasets	Models	Acc	DR	F1
NSL-KDD (KDDTest+)	SAVAER-DNN [22]	89.36	95.98	90.08
	ResNet [21]	79.14	69.41	79.12
	GoogLeNet [21]	77.04	65.64	76.50
	TSE-IDS [16]	85.79	86.80	/
	RNN-IDS [15]	83.28	73.12	83.22
	ICVAE-DNN [19]	85.97	95.39	86.27
	SCDNN [14]	72.64	57.48	/
	IGAN-IDS [23]	84.45	84.17	
NSL-KDD (KDDTest-21)	SAVAER-DNN [22]	80.30	95.19	86.92
	TSE-IDS [16]	72.52	72.50	/
	RNN-IDS [15]	68.55	/	/
	ICVAE-DNN [19]	75.43	96.2	68.62
UNSWNB15	SAVAER-DNN [22]	93.01	91.94	93.54
	TSE-IDS [16]	91.27	91.30	/
	TSDL [18]	89.13	/	/
	ICVAE-DNN [19]	89.08	86.05	90.61
	IGAN-IDS [23]	82.54	97.09	
	KG-DBN [20]	86.49	/	/

Some researches [29, 30] applies SMOTE technology in IDS to balance dataset and improve detection efficiency. However, SMOTE relies on interpolation for oversampling, and the fitting degree of generated samples is relatively low. When the distance between different classes is very close, the synthetic sample of SMOTE may duplicate the existing data, and even make the classification result worse.

2.3 Variational autoencoder (VAE)

The generative model mainly includes two forms: variational autoencoder VAE and GAN. VAE is a generative deep learning model based on variational thought. VAE is a generative network structure based on variable Bayes (VB) inference proposed by Kingma et al. [31] in 2014 (Fig. 1). The loss function of the original VAE [9] can be expressed as:

$$L_{VAE} = -E_{q(z|x)}[log\frac{p(x|z)p(z)}{q(z|x)}] = L_{llike}^{pixel} + L_{prior} \quad (1)$$

$$L_{llike}^{pixel} = -E_{q(z|x)}[logp(x \mid\mid z)] \quad (2)$$

$$L_{prior} = D_{KL}(q(z \mid x)p(z) \quad (3)$$

Where L_{llike}^{pixel} is the decoding process $p(x|z)$, $p(z)$ is prior distribution, $q(z|x)$ is the approximate posterior distribution, z follows $q(z|x)$; L_{VAE} is the KL divergence between $q(z|x)$ and $p(z)$.

2.4 Generative adversarial networks (GAN)

GAN is composed of generator (G) and discriminator (D) [32]. Figure 2 shows GAN's network structure.

The loss function of GAN is:

$$\min_G \max_D V(G, D) = \min_G \max_D E_{x\ p_r}[log D(x)] + E_{z\ pz}[log(1 - D(G(z)))] \quad (4)$$

Where, z is the random noise, x is the original sample, y is the category information, p_z is the distribution of z, p_r is the distribution of real data x, $G(z)$ is the fake data generated by G, D(x) is the score given by D, $E()$is the expected value. G and D are optimized iteratively in the confrontation, to identify the data source more accurately and to generate excellent fake samples.

Conditional generative adversarial networks (CGAN) introduces class information into the vanilla Gan to generate samples of specified classes. The generator takes noise and class information as input at the same time, and the discriminator will receive corresponding class information when receiving samples. The loss function of CGAN [33] is:

$$\min_G \max_D V(G, D) = \min_G \max_D E_{x\ p_r}[log D((x|y))] + E_{z\ pz}[log(1 - D(G(z|y)))] \quad (5)$$

The network structure of CGAN is shown in Fig. 3.

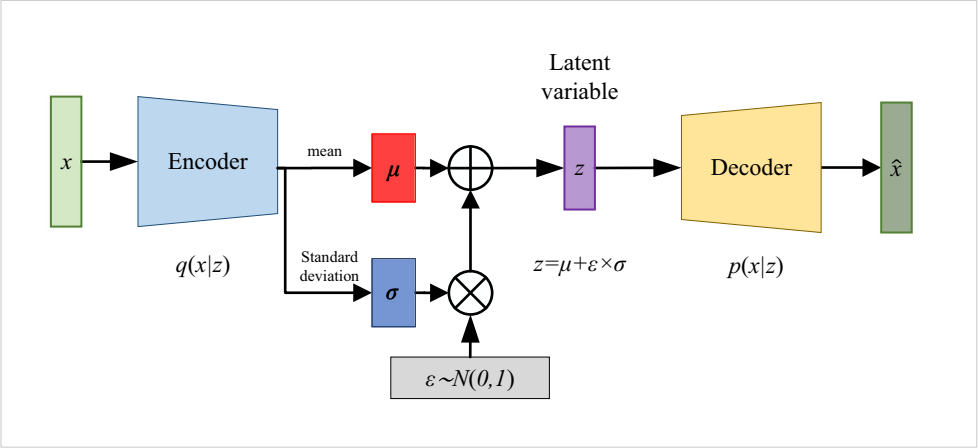


Fig. 1 Structure of VAE

Meanwhile, Mirza et al. [33] pointed out that the Jensen-Shannon divergence used in GAN and CGAN can lead to mode collapse and gradient disappearance.

Arjovsky et al. [34] proposed Wasserstein GAN (WGAN)) in 2017. WGAN changed the JS divergence of loss function to earth mover (EM) distance, which solved the problem of gradient disappearance and mode collapse in GAN and CGAN. The objective function of WGAN is:

$$V(G,D)_{WGAN} = \max_{D\in 1-lipschitz}\left\{E_{x\sim p_r}[D(x)] - E_{x\sim(p_g)}[D(x)]\right\} \quad (6)$$

Where p_g is the generated samples' distribution, $D \in 1-lipschitz$ means that any function $f(x)$ which satisfies $f(x_1)-f(x_2) \leq x_1-x_2$. that is, the discriminator D is smooth enough to prevent the dependent variable from changing too fast with the change of independent variable.

WGAN-GP [8] adds a gradient penalty term on the basis of WGAN to satisfy the Lipschitz constraint and avoid the situation that the discriminator cannot converge in the training process. The objective function of WGAN-GP is :

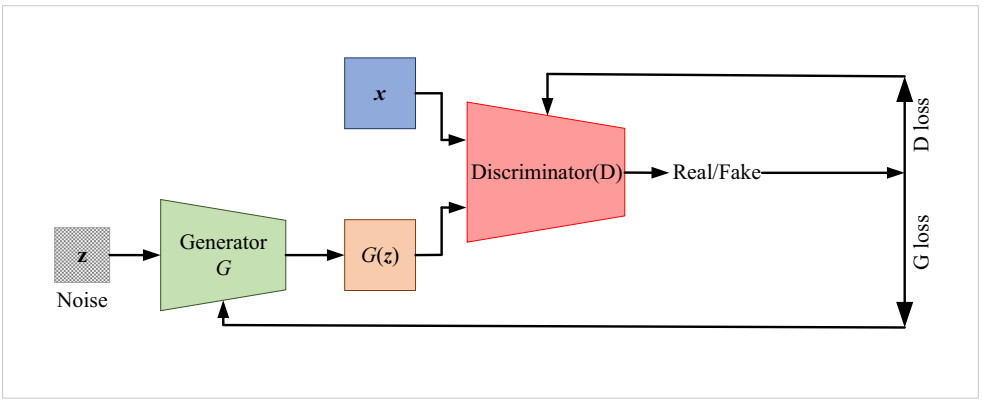
$$V(G,D)_{WGAN-GP} = \max_{D\in 1-lipschitz}\{E_{x\sim p_r}[D(x)] - E_{x\sim p_g}[D(x)] - \lambda E_{x\sim p_{penaty}}[\nabla_x D(x)-1]^2\} \quad (7)$$

Where λ it is a parameter; $\nabla_x D(x)$ represents the computing paradigm $D(x)$ of alignmentx; $x \sim p_{penaty}$means to take the middle position x from the line between a point on p_r and a point on p_g.

3 Methods

In this section, we propose an CWVAEGAN model to generate new minority samples. And we establish an CWVAEGAN-based Intrusion Detection System (CWVAEGAN-IDS) to perform class imbalance intrusion detection.

Fig. 2 Model Structure of GAN



3.1 Conditional wasserstein variational autoencoders with GAN (CWVAEGAN)

Larsen et al. [9] called the joint training of VAE and Gan as VAE-GAN, which uses the representation learned by GAN'S discriminator to measure the distance between dataspaces. The decoder of VAE is combined with GAN, and training is carried out simultaneously. The feature-wise error is used to replace the element-wise error to generate higher quality image samples.

On the basis of VAE-GAN, we add the condition vector and WGAN-GP loss function, to overcome the difficulty in GAN training and improve the non-convergence phenomenon of the model. New training samples are generated for minority classes to alleviate the class-imbalance problem. The condition vectors are adopted to control the class of the generated samples, and only minority class samples are generated. We also improve the model structure of VAE-GAN to raising the quality of generated samples.

3.1.1 CWVAEGAN structure

Figure 4 shows the structure of CWVAEGAN, which includes three parts: encoder E, generator G and discriminator D, in which the G of GAN also acts as a decoder in VAE. E encodes the original sample data x into the implicit vector z, and calculates the mean value and variance of the input dataset through encoder. G then attempts to generate samples from the reparameterized z or random noise z. Finally, D gives scores for the original sample, the reconstructed sample and the newly generated sample. The combination of E and G is used as the data generation part to conduct adversarial learning with D. In conclusion, CWVAEGAN takes the original training sample $s = (r, y)$ as the input and outputs the synthesized sample $s_G = [G(z, y'), y']$, where r, y, z and y' respectively represent the original data feature, the original label, the noise and minority class label.

Encoder E As shown in Fig. 4, the discrimination model E consists of one-dimensional convolution layer and fully connected layer. In adversarial learning, the input of E is minority classes vector r' and the corresponding label y'. For a particular input $s = (r, y)$, E translates it into a hidden vector z. As shown in (1), the original VAE loss function consists of reconstruction loss L_{llike}^{pixel} and prior loss L_{prior}. In VAEGAN, L_{llike}^{pixel} is replaced by the feature representation $L_{llike}^{Dis_l}$ learned by the discriminator

$$L_{llike}^{pixel} = -E_{q(z|s)}[\log p(Dis_l(s)|z)] \tag{8}$$

in which $Dis_l(s)$ represents the hidden representation of s in layer l of D, and $p(Dis_l(s)|z)$ can be expressed as a Gaussian distribution with mean value $Dis_l(\widetilde{s})$ and identity covariance [9]

$$p(Dis_l(s)|z) = \mathcal{N}(Dis_l(s) \mid Dis_l(\widetilde{s}), I) \tag{9}$$

In the process of training, we use the advanced feature representation of the hidden layer in D to measure the distance between the generated samples and the original samples. The loss function of encoder is

$$L(E) = L_{llike}^{Dis_l} + L_{prior} \tag{10}$$

Discriminator D In the process of training, the input of D is the mixture of s', s_R and s_G, and the output is the prediction probability of the samples. For a particular input (r, y), D needs to determine the probability $D(r, y)$ whether it comes from s' other than s_R or s_G. The loss function of D is

$$\begin{aligned} L(D) = {} & E_{s\sim p_r}[D(s)] - E_{s\sim p_g}[D(s)] \\ & -\lambda E_{s\sim p_{penaty}}[\| \nabla_s D(s) \| - 1]^2 \end{aligned} \tag{11}$$

Among them, the first item of the loss function is the discrimination result of the real samples, the second and third items are the discrimination result of the reconstructed samples and the newly generated samples respectively, and the fourth item is the gradient penalty term. After the loss value is obtained, the gradient can be calculated η_{θ_D} and the network parameters θ_D of the discriminator can be updated by the ADAM algorithm.

Generator G In the training process, the class information and the input vector are connected as (z, y') or (z', y'), and the reconstructed samples $s_R = [G(z, y'), y']$ or the newly

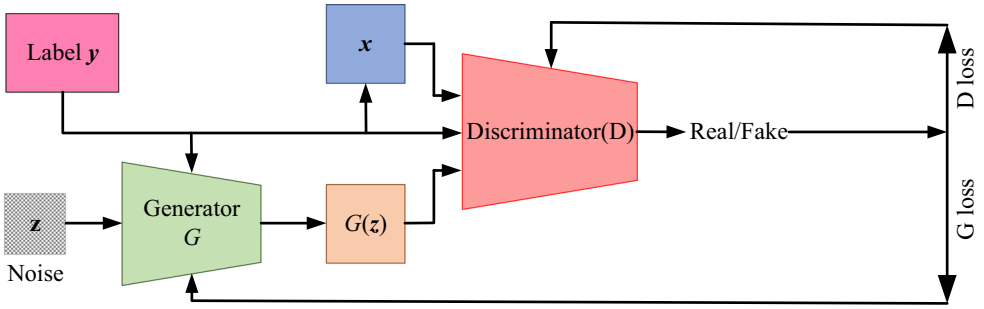


Fig. 3 Model structure of CGAN

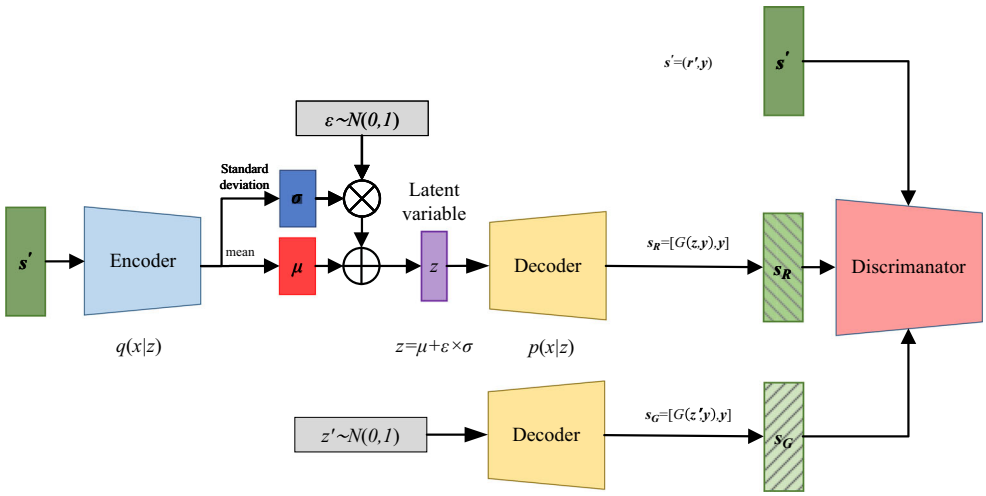


Fig. 4 Model Structure of CWVAEGAN

generated samples $s_G = [G(z, y'), y']$ are obtained through G. In the generator training process, we will also add $L_{llike}^{Dis_l}$ to the generator's loss function as an regularization, the loss function of G is

$$L(G) = E_{s\sim p_g}[D(s)] - L_{llike}^{Dis_l} \tag{12}$$

3.1.2 Imbalanced adversarial training

During the training of CWVAEGAN, generator and discriminator are trained alternately. The main steps of training are as follows:

1. The original data is encoded by E to get the hidden vector z.
2. The z' or z concatenated with minority classes label y' are input, then G are trained and then generated s_R.;
3. Fixed E and G, trained D, updated θ_D;
4. Fixed discriminator D, training E and G, updating θ_E and θ_G;
5. Before the loss value of D reaches 0.5, the cycle performs steps (1) to (4), E, G and D alternately, which makes the generated samples close to the real samples.

The detailed training process of G and D in the above steps is described in detail in chapters 2.2.1 and 2.2.2. The whole training process of CWVAEGAN is shown in Algorithm 1, among which θ_E, η_{θ_E}, θ_G, η_{θ_G}, θ_D, η_{θ_D} are network parameters and gradients of encoder, generator and discriminator respectively.

Algorithm 1 minority classes data generation based on CWVAEGAN and VGM.

Input: $s = (r, y)$
Output: $s_G = [G(z, y'), y']$

$c_{i,j} = v_{i,j} \oplus s_{i,j}$/* Continuous columns decomposition */
$r_{j'} = v_{1,j} \oplus a_{1,j} \oplus \ldots \oplus v_{N_c,j} \oplus a_{N_c,j} \oplus d_{1,j} \oplus \ldots \oplus d_{N_d}$/* Feature column decomposition */
while D has not converged to 0.5 do/*CWVAEGAN training*/ **do**
    /*Optimization of D */
    Sample $(r_i', y_i')_{i=1}^{n_z}$ from $p_{data}(r', y')$
    $z_i = E(r_i', y_i')$
    $Sample(z_i')_{i=1}^{n_z} from p_z(z)$
    $\eta_{\theta_D} \leftarrow \nabla\theta_D[\frac{1}{n_z}\sum_{i=1}^{n_z} D\left(r_i', y_i'\right) - D\left(G\left(z_i, y_i'\right), y_i'\right) - D(G(z_i', y_i'), y_i')\ \lambda E_{(r_i', y_i')\sim p_{penaty}}[\| \nabla_s D\left(r_i', y_i'\right) \| - 1]^2]$
    $\theta_D \leftarrow \theta_D + \alpha_D \cdot Adam\left(\theta_D, \eta_{\theta_D}\right)$
    /*Optimization of G and E jointly*/
    Sample $\{(z_i)\}_{i=1}^{n_z}$ from $p_z(z)$
    $\eta_{\theta_E} = -\nabla_{\theta_E}(D_K L(q(z_i \mid (r_i', y_i')) \| p(z_i)) - E_{q(z_i\|(r_i', y_i'))}[log\, p((r_i', y_i') \| z_i)])$
    $\eta_{\theta_G} \leftarrow \nabla\theta_D[\frac{1}{n_z}\sum_{i=1}^{n_z} D(G(z_i', y_i'), y_i')$
    $\theta_G \leftarrow \theta_G + \alpha_G \cdot Adam\left(\theta_G, \eta_{\theta_G}\right)$
    $\theta_E \leftarrow \theta_E + \alpha_E \cdot Adam\left(\theta_E, \eta_{\theta_E}\right)$
end while

3.2 IDS based on CWVAEGAN (CWVAEGAN-IDS)

CWVAEGAN is proposed to deal with class-imbalance problem and simulate unknown exceptions. Furthermore, the IDS based on CWVAEGAN (CWVAEGAN-IDS) is constructed to detect the network abnormal behavior.

3.2.1 IDS structures

IDS has three modules: data processing module, CWVAEGAN module, and CNN module. Figure 5 illustrates the architecture of CWVAEGAN-IDS. First of all, the data preprocessing module filters out minority classes in the dataset, and decomposes the continuous features in the original

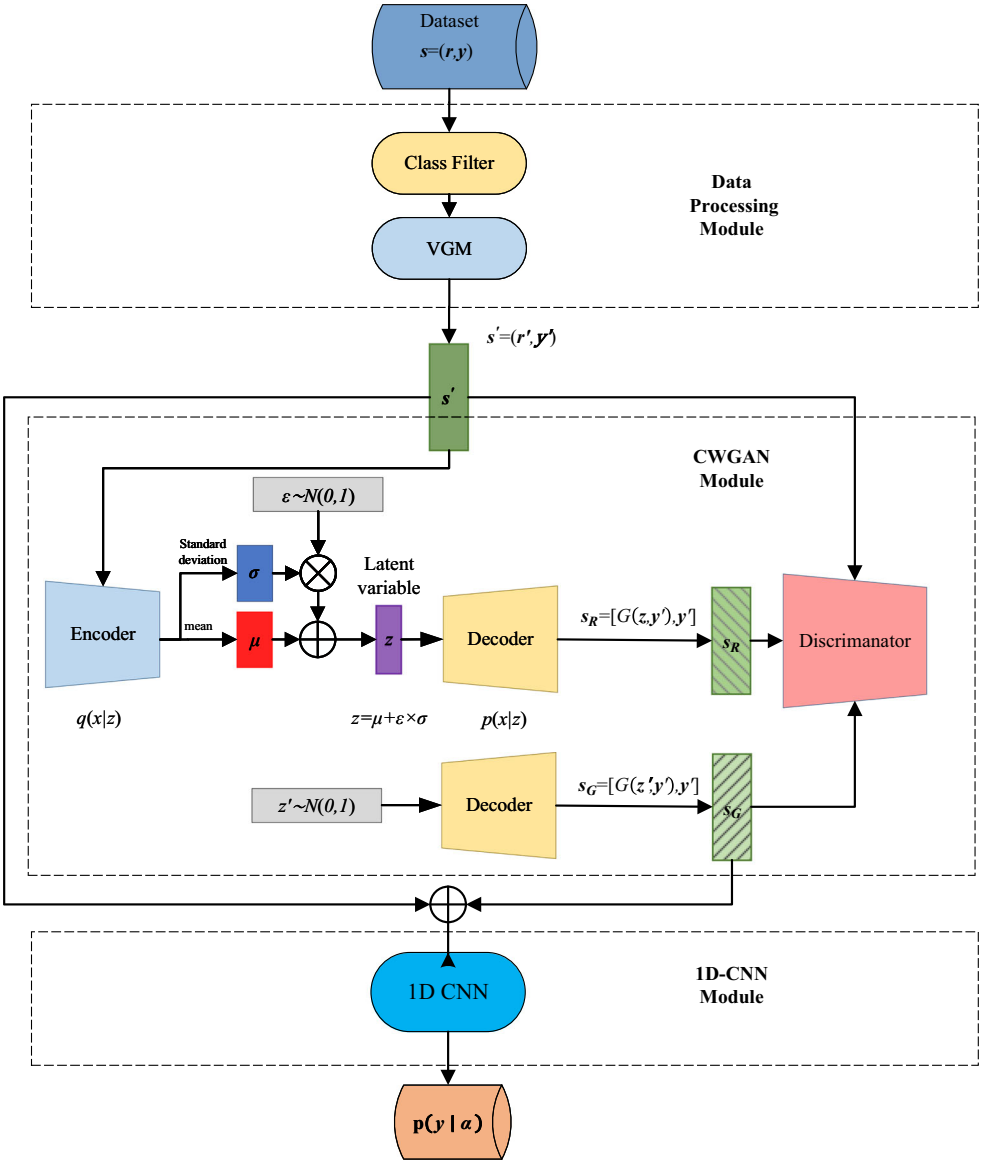


Fig. 5 Model Structure of CWVAEGAN-IDS

dataset by VGM to get the original training dataset. Then, the CWVAEGAN module generates new samples. Finally, CNN module uses the balanced samples for training and performs intrusion detection on the test dataset. In general, CWVAEGAN-IDS takes the rows from intrusion detection dataset as input and predicts their labels $p(y|r)$.

3.2.2 Data preprocessing

In the part of data preprocessing, CWVAEGAN-IDS firstly extract the minority class information from the dataset, and decompose the continuous features of multimodal distribution. Taking the original sample $s = (r, y)$ as the input, the first step is to screen out the minority classes (r, y') that need to be enhanced, and then VGM decomposition is used to get the minority sample data $s' = (r', y')$.

VGM is used to process the continuous features of Gaussian mixture model. In intrusion detection dataset, discrete features can be expressed as one-hot vector [23], while continuous features of Gaussian mixture distribution are difficult to be fully expressed.

We choose VGM to decompose them. Each value in the continuous feature column is converted into a parallel connection of an one-hot vector v_* indicating the component and a scalar a_* indicating the value under the component by VGM. Firstly, for each continuous column C_i, VGM is used to estimate the number of components, and Gaussian mixture model is trained; Secondly, the probability of each value $c_{i,j}$ in each component is calculated; Finally, samples are taken from the sub distribution with the largest probability.

Through VGM decomposition, $c_{i,j}$ is decomposed into $v_{i,j} \oplus a_{i,j}$, which is convenient for GAN to learn the mixed

distribution better. A row in the dataset r_j can be rewritten as a concatenation of continuous features and discrete features.

$$r_{j'} = v_{1,j} \oplus a_{1,j} \oplus \ldots \oplus v_{N_c,j} \oplus a_{N_c,j} \oplus d_{1,j} \oplus \ldots \oplus d_{N_d} \quad (13)$$

3.2.3 CWVAEGAN

We have described the CWVAEGAN part in the Section 3.1. CWVAEGAN module takes raw data or Gaussian noise as input and generates minority classes samples. CWVAEGAN can be expressed as $s_G = G[(z, y')y']$.

3.2.4 One dimensional convolutional neural network (1D-CNN)

1D-CNN module take an one dimensional convolutional neural network (1D-CNN) to perform intrusion detection, as shown in Fig. 2. 1D-CNN model takes the rebalancing training dataset as the input, and outputs the probability distribution $P(y)$ of different classes in the test dataset. Table 1 shows the structure of 1D-CNN.

Refer to [35], we design the structure of 1D-CNN, which uses convolution layers of 32, 64, 128 and 256 kernels respectively, and the activation function is Mish [36]. The output of convolution layer is randomly discarded with a probability of 0.2 to avoid over fitting. Then a 32 unit full-connected layer was set and activated by leaky-Relu function. The size of the output layer is equal to the number of classes. Finally, the probability distributions $p(y|r)$ are given.

4 Experiment and analysis

In this section, CWVAEGAN-IDS is compared with other class-balancing methods and other advanced methods. The results show that CWVAEGAN-IDS has better performance than the most advanced methods.

4.1 Dataset

The performance of CWVAEGAN-1DCNN was evaluated on NSL-KDD dataset [6] and UNSWNB15 dataset [37]. These two datasets are commonly used to evaluate IDS.

NSL-KDD dataset is improved based on KDD99 dataset. It removes large amount of redundant data, and adjusts the composition of training set and test set. It is one of the most classic benchmark datasets for IDSs' evaluation. It enables researchers to intuitively compare the efficiency of various methods. NSL-KDD includes 41 features. NSL-KDD dataset contains four kinds of abnormal behaviors: DOS, Probe, U2R, and R2L. NSL KDD is divided into the training set KDDTRAIN+20, as well as test set KDDTEST+ and KDDTEST21 (Table 2).

Table 2 Sample distribution of NSL-KDD

Label	KDDTrain+20	KDDTest	KDDTest21
Normal	13,449	9,711	2,152
DoS	9,234	7,458	4,342
Probe	2,289	2,421	2,402
U2R	11	200	200
R2L	209	2,754	2,754
Total	25,192	22,544	11,850

Moustafa et al. [37] proposed UNSWNB15 dataset in 2015, which is more suitable for modern network intrusion detection as a mixture of nine kinds of abnormal activities. Different IDSs can be fully tested through various types of attacks in UNSWNB15. UNSWNB15 contains 42 network features and two label columns. There are 175,341 and 82,332 records in training set and test set respectively, as shown in Table 3.

In the training dataset KDDTrain 20percent, the number of U2R and R2L samples are extremely few. The training set KDDTrain 20percent, there are 23 kinds of exceptions, while there are 38 kinds of exceptions in KDDTest+ and KDDTest21. Class imbalance data and 15 unknown exceptions bring great challenges to intrusion detection. In UNSWNB15, there are only 5010 samples for the least four classes, worms, backdoor, shellcode and analysis, and only 130 training samples for the least classes worms. The two largest categories of normal and generic add up to more than 100,000 records.

We perform multi-classification test on both datasets, trying to detect the attack class of each network traffic data, instead of simply judging whether it is normal. Intrusion detection is regarded as a multi-classification task, and do 5 classification on NSL-KDD and 10 classification on UNSWNB15.

Table 3 Sample Distribution of UNSWNB15

Class	Training set	Testing set
Normal	56,000	37,000
Generic	40,000	18,871
Exploits	33,393	11,132
Fuzzers	18,184	6,062
DoS	12,264	4,089
Reconnaissance	10,491	34,96
Analysis	2,000	677
Backdoor	1,746	583
Shellcode	1,133	378
Worms	130	44
Sum	175,341	82,332

4.2 Metrics

We adopt accuracy, precision, recall, F1 score and G-means as the main metrics to evaluate the classification performance of the CWVAEGAN-IDS. Accuracy indicates the overall performance of CWVAEGAN-1DCNN, precision quantifies the specific classification capacity of each class, while recall indicates the detection rate for a specific class. These metrics are defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

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

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

where TP, TN, FP and FN [23] are true positive, true negative, false positive and false negative respectively.

F1 score is the harmonic mean value of precision and recall. G-means is a comprehensive parameter of positive and negative class accuracy.

$$F1 = 2 \times \frac{Precision \times recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{17}$$

$$G - \text{mean} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{18}$$

4.3 Experimental procedure

4.3.1 Experimental environment and parameter setting

As we described in Section 3.2.1, the original samples are first input to the data processing module. Data processing module firstly selects the minority classes, remove the normal network activities with large number of samples. We implement VGM to decompose the features of continuous columns, and use Bayesian Gaussian mixture method from sklearn to achieve VGM. The Gaussian mixture model are fitted with 10 components to approximate the original distribution, and the feature vectors processed from the data processing module are input into CWVAEGAN module. The experiments are performed using Python 3.7 on a Windows 10 personal computer with an AMD Ryzen 9 5900X@4.4GHz CPU, NVIDIA GeForce RTX 2080 Ti GPU and 32 GB RAM.

For CWVAEGAN module, the architectures of E, D and G are flexible, which should be set based on the specific situation. After selecting the appropriate parameters, CWVAEGAN can achieve enough expressive power with minimum overhead.

According to the VAE-GAN model proposed by Larsen [9], we adjust the network structure. Two convolution layers are taken to down sampling the original data to get multi-channel hidden features, and the fully connected layers are used to transform them into 256 dimensional single channel features. The first 128 dimensions represent the mean value of the sample, and the last 128 dimensions represent the standard deviation of samples. By fitting the encoder to get mean value and standard deviation of samples, the 128 dimensional hidden vector z is reconstructed with the multiple reparameterization technique.

The generator (encoder) G is composed of fully connected layers and one-dimensional transposed convolution layers. z is used to reconstruct the original samples. In the fully connected layer, 128 dimensional features are expanded and sent to 64 different channels, and then one-dimensional transposed convolution layer is used for up sampling. The hidden vector z gets the reconstructed sample S_R through G. Or the Gaussian noise z'can be input into G to get the new sample S_G.

Discriminator D is composed of one-dimensional convolution layers and fully connected layers. It attempts to identify the original samples and generated samples, which is equivalent to a classifier performing two classification tasks.

There is no automatic parameter tuning algorithm at this stage. We conduct large amount of experiments on the basis of existing references to compare and analyze the influence of different parameters. The final parameters of the model are determined through a large number of experiments. All hidden layer activation are ReLU, the generator output layer adopts sigmoid activation, and the encoder and discriminator output layer adopts linear activation. The training parameter settings are given in Table 4.

4.3.2 Experimental setup

We conduct the following experiments to test the performance of CWVAEGAN-1DCNN:

Experiment 1: The training experiment of CWVAEGAN model.

Experiment 2: The training experiment of 1DCNN model.

Experiment 3: Performance comparison between CWVAEGAN-1DCNN and different data class-balancing methods.

The selected comparison algorithms include ROS, SMOTE [23] and adaptive synthetic (ADASYN) [33], and are combined with CNN described in 2.3. Class balance methods generate samples, and then the balanced samples are input to 1DCNN for intrusion detection. These methods are recorded as ROS + CNN, SMOTE +CNN and ADASYN + CNN respectively. The CNN

Table 4 Network structure of CWVAEGAN

Model	#	Layer	Filters	Kernel Size	Output Size	Activation
Encoder	1	Input	–	–	42*1	–
	2	Conv1d	32	3	40*32	Relu
	3	Conv1d	64	3	38*64	Relu
	4	Flatten	–	–	2432	
	5	Dense	–		256	–
Decoder	1	Input	–	–	128	
	2	Dense	–	–	2432	Relu
	3	Reshape	–	–	38*64	–
	4	Conv1dtrans	64	3	40*64	Relu
	5	Conv1dtrans	32	3	42*32	Relu
	6	Conv1dtrans	1	1	42*1	Sigmoid
Discrimator	1	Input	–	–	42*1	
	2	Conv1d	32	3	40*32	Relu
	3	Conv1d	64	3	38*64	Relu
	4	Flatten	–	–	2432	–
	4	Dense	–	–	32	Relu
	5	Dense	–	–	1	–

parameters in these methods are consistent, which can be set according to Table 5.

Experiment 4: Performance comparison between CWVAEGAN-1DCNN and existing intrusion detection models.

The intrusion detection models employed as the comparison methods include RNN-IDS [15], SAVAER-DNN [22], I-SiamIDS [17], TSE-IDS [16], GFBLS [40], LSTM4 [40], IGAN-IDS [23], TSDL [18], ICVAE-DNN [19], DBN [39]. All models use the same dataset.

Experiment 5: Visualization Comparison of class-balancing methods.

We compare the samples generated by 4 kind of class-balancing methods through visualization Park et al. [41] proposed a method to evaluate the synthetic data. Through compare the statistical characteristics of the new samples with the original samples, we evaluate the effects of different class-balancing methods. Visualization of various statistical features can intuitively compare the effects of samples generated by different methods.

Experiment 6: Statistical test.

4.4 Experiment results and analysis

4.4.1 Experiment 1: the training experiment of CWVAEGAN model

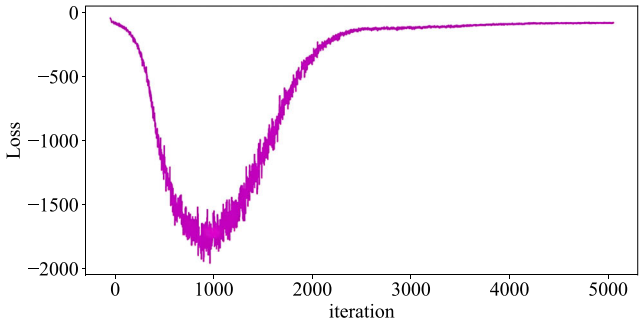
CWVAEGAN is used to generate samples for minority classes. The loss curve of discriminator D during training is shown in Fig. 6. Three subgraphs (a), (b) and (c) in Fig. 6 correspond to Probe, U2R and R2L classes.From the curves, we observe that CWVAEGAN converges after 3000, 5000 and 3500 training on Probe, U2R and R2L classes respectively.

After convergence, the loss curves of the three classes have different degrees of oscillation, among which U2R is the most obvious, R2L is the second, and Probe is the smallest. This is because there are only 11 samples of U2R class in KDDTRAIN + dataset, which is difficult to converge.

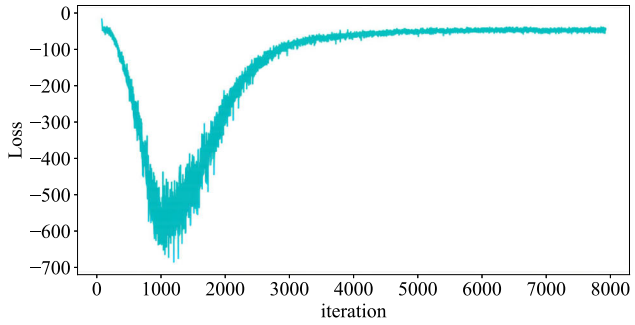
Take the loss curve of Probe in Fig. 6(b) as an example. In 0 1,000 iterations, G can not generate homologous samples, D can easily distinguish generated samples, and

Table 5 Parameters of CWVAEGAN

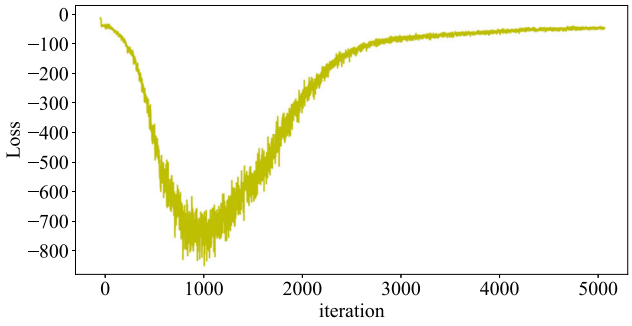
CWVAE-GAN	$epoch_0$	1000	1D-CNN	$epoch_1$	300
	$batchsize_0$	500		$batchsize_1$	512
	lr_{enc}	0.001		lr_{CNN}	0.005
	lr_{dec}	0.001		activation	Adam
	lr_{disc}	0.0001			
	activation	Adam			



(a) Probe



(b) U2R



(c) R2L

Fig. 6 Convergence curve of discriminatorD on NSLKDD's three minority class: (a)Curve on Probe. (b)Curve on U2R. (c)Curve on R2L

the loss value is large; In the process of 1000 3000 iterations, the generation ability of G is constantly improved, the score of the generated samples given by D is closer to the score of the original samples, the loss value is constantly reduced, and reaches dynamic stability after 4000 iterations.

4.4.2 Experiment 2:The training experiment of 1DCNN model

For CNN, appropriate convolution kernel number and convolution layer number can improve the performance of classifier. According to the paper, the number of candidate convolution layers is 2, 3, 4. The number of convolution kernels of each convolution layer is twice that of the previous convolution layer. According to the different number of convolution layers and kernels of the initial convolution layer, we can determine the network structure of a classifier. The four layer convolution network with 64 convolution kernels of the initial convolution layer is recorded as $c4\_64$ and the number of network channels is 64-128-256-512. We carry out experiments for different network layers and channel number of initial layer. On the NSLKDD dataset, all the classifiers can learn the features of the training set better, and the difference of different model parameters is not big, so we conduct this experiment on UNSWNB15's training set.

Firstly, the convolution kernel number of the initial layer is set to 64, and different convolution layers $c2\_64$, $c3\_64$, $c4\_64$, $c5\_64$ are studied. In Fig. 7, when the convolution layers are $c4\_64$, the classification effect of the model is the best and the convergence speed is the fastest.

Then we fix the number of convolution layers as 4, and the initial layer convolution kernel numbers are $c4\_8$, $c4\_16$, $c4\_32$, $c4\_64$, $c4\_80$, $c4\_100$. In Fig. 8, we can see that with the continuous expansion of the model parameters, the overall detection performance is improved and the convergence speed is also accelerated. When the initial number of convolution kernels is equal to or greater than 64, the model can reach 90% before epoch reach 300. At the same time, when the number of convolution cores is increased from 64 to 80 and 100, the performance improvement of less than 1% requires 56% and 140% additional flops.

Therefore, we set the structure of the network to include four one-dimensional convolution layers. The number of convolution cores of each layer is 64, 128, 256 and 512 respectively. The input of each layer is not zero filled, and the convolution core size and step are 5 and 1 respectively. At the same time, the output is batch normalized and then activated by the Mish [36] function. The window size and step length of max-pooling layer are 3.

The new training set of NSL-KDD and UNSWNB15 are shown in Tables 6 and 7, respectively.

4.4.3 Experiment 3: Comparison of class-balancing methods

The selected comparison algorithms include ROS, SMOTE and ADASYN [38], and the original VAEGAN technology. Based on class-balancing algorithm, we construct five classification models: ROS-CNN, SMOTE-CNN, ADASYN-CNN, VAEGAN-CNN and CWVAEGAN-1DCNN. The type and quantity of attack samples generated by VAEGAN are consistent with CWVAEGAN. The results are shown in Figs. 9, 10, 11, 12 and 13.

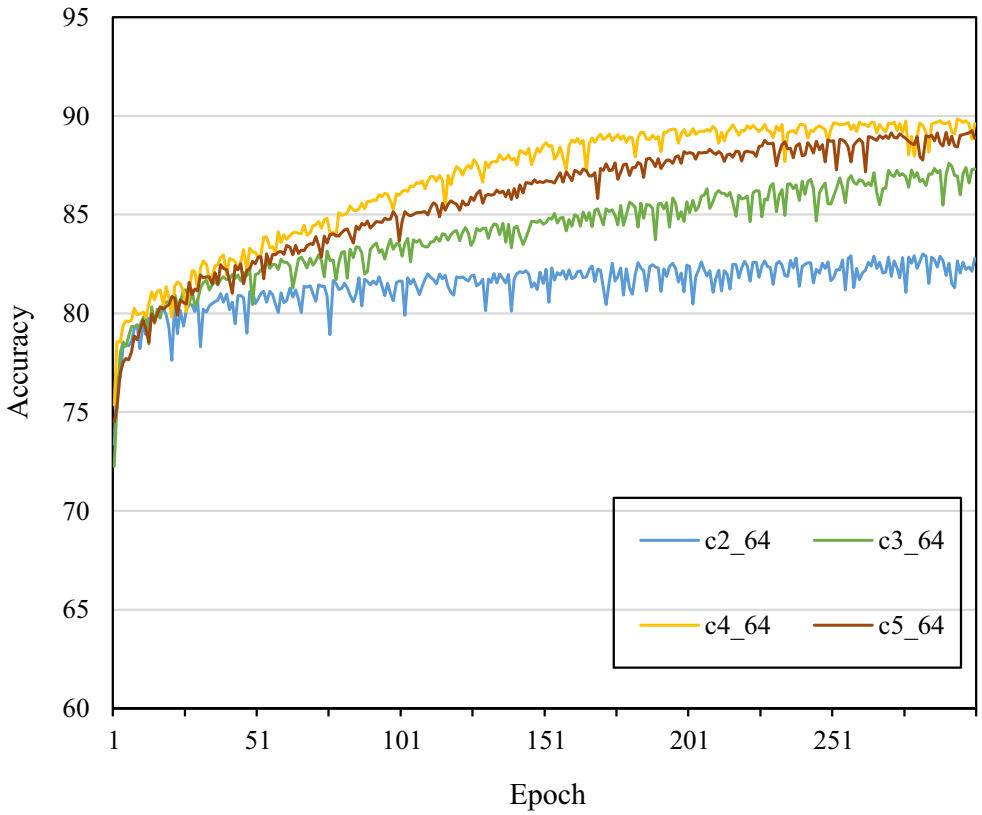


Fig. 7 Accuracy curve of different layers

Table 6 The generated samples for NSL-KDD dataset

Class	Number of original samples	Number of generated samples	Sum
Normal	13449	0	13449
DoS	9234	4215	13449
Probe	2289	3000	5289
U2R	11	1000	1011
R2L	209	2000	2209
Total	25192	10215	35407

Table 7 The generated samples for UNSWNB15 dataset

Class	Number of original samples	Number of generated samples	Sum
Normal	56000	0	56000
Generic	40000	0	40000
Exploits	33393	0	33393
Fuzzers	18184	0	18184
DoS	12264	0	12264
Reconnaissance	10491	0 10491	
Analysis	2000	8000	10000
Backdoor	1746	8000	9746
Shellcode	1133	8000	9133
Worms	130	4000	4130
Sum	175341	124423	203341

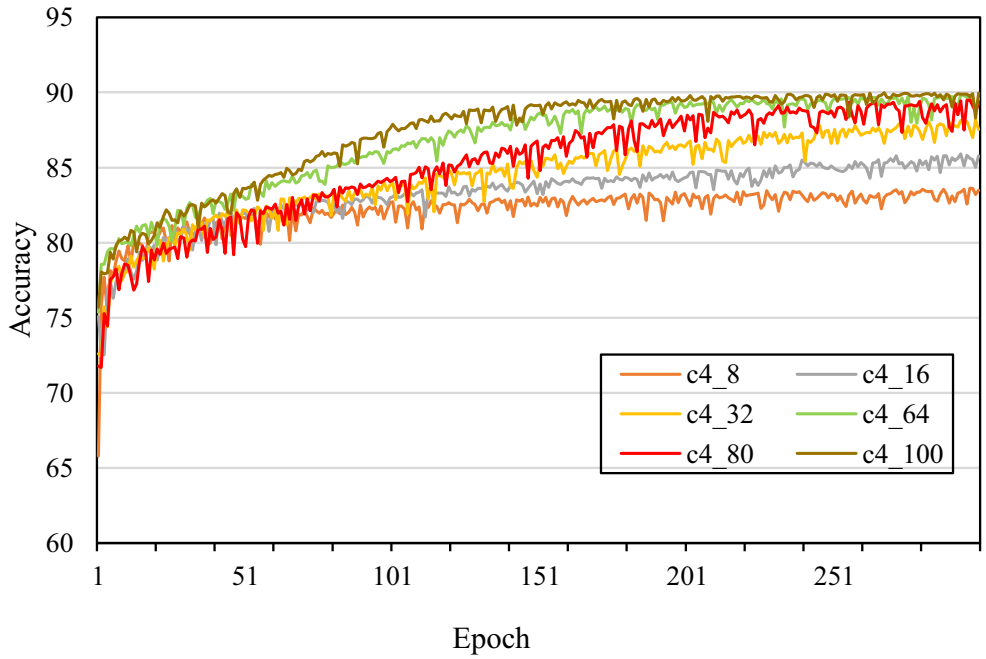
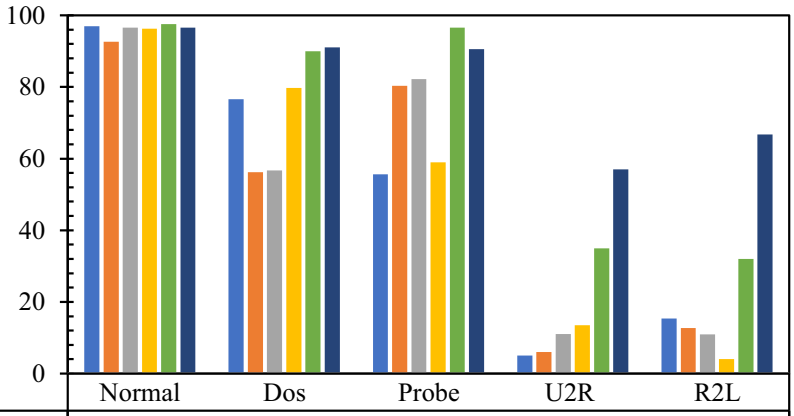


Fig. 8 Accuracy curve of different kernels

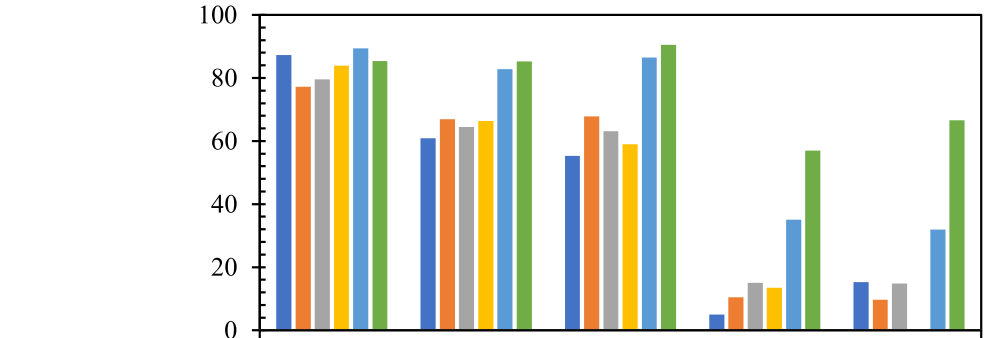
Figures 9 and 10 show the recall of different class-balancing methods for five different types of samples on KDDTEST+ and KDDTEST21 dataset respectively. Compared with other five class-balancing methods, CWVAGAN-CNN has the highest recall for U2R and R2L attack class, and the performance of majority classes is only slightly backward in few metrics. The comparative experimental results show that CWVAEGAN-1DCNN improves the recall for minority classes.

The overall performance of different class-balancing methods on three test sets of KDD and UNSWNB15 is shown in the Figs. 11, 12 to 13. CWVAEGAN-1DCNN get the best results on accuracy, F1 score and G-mean. On the UNSWNB15 dataset, the recall of CWVAEGAN CNN are



	Normal	Dos	Probe	U2R	R2L
CNN	96.98	76.61	55.67	5	15.32
ROS-CNN	92.61	56.26	80.32	6	12.75
SMOTE-CNN	96.59	56.75	82.19	11	10.93
ADSYN-CNN	96.3	79.72	58.94	13.5	4
VAEGAN-CNN	97.52	89.94	96.57	35	32
CWVAEGAN-CNN	96.58	91.08	90.54	57	66.7

Fig. 9 Recall of different class-balancing methods on KDDTEST+



	Normal	Dos	Probe	U2R	R2L
CNN	87.22	60.87	55.32	5	15.25
ROS-CNN	77.18	66.9	67.86	10.5	9.7
SMOTE-CNN	79.55	64.42	63.07	15	14.85
ADSYN-CNN	83.96	66.32	58.99	13.52	0.05
VAEGAN-CNN	89.41	82.79	86.46	35	31.95
CWVAEGAN-CNN	85.41	85.26	90.47	57	66.63

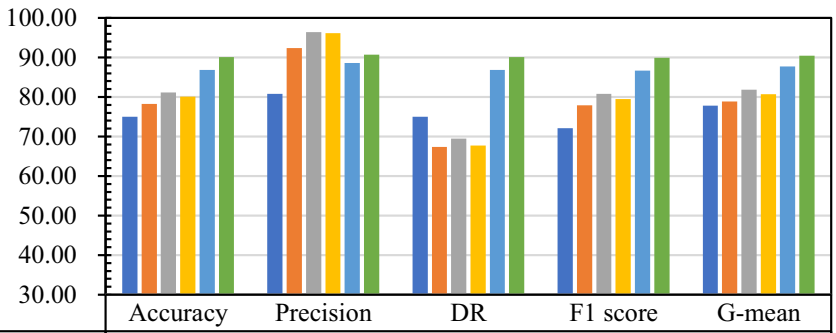
Fig. 10 Recall of different class-balancing methods on KDDTEST21

5% lower than other methods, and on the KDD test set, the precision backward by 2% 9%. The experiment shows that CWVAEGAN is an effective class-balancing method.

All the results reported above demonstrate that CWVAEGAN-1DCNN has better performance than ROS-CNN, SMOTE-CNN and ADASY-CNN. The reason is that ROS only simple replicate the original data, SMOTE and ADASY are random synthesis of the original data according to the k-nearest neighbor principle, and lack of learning the distribution of the original data.

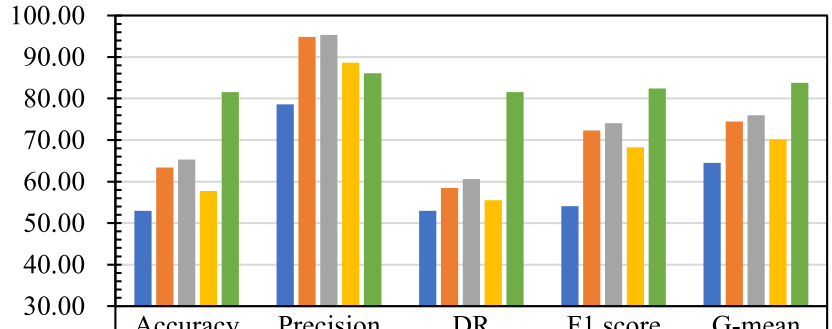
4.4.4 Experiment 4: performance comparison with advanced IDSs

We compare CWVAEGAN-1DCNN with the methods mentioned in Section 4.3.2. The results on KDDTEST+ and KDDTEST21 are shown in Tables 7 and 8 respectively. Table 9 shows the results on the testset of UNSWNB15, best results in each column are boldfaced. In this paper, the performance data given in references of advanced IDSs are used to compare with CWVAEGAN-CNN.



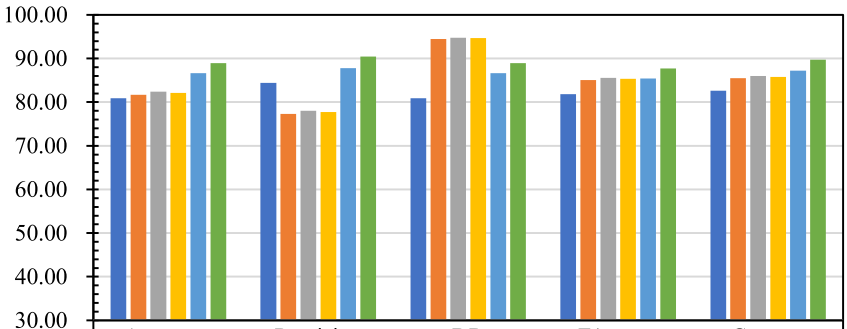
	Accuracy	Precision	DR	F1 score	G-mean
CNN	75.01	80.77	75.01	72.15	77.84
ROS-CNN	78.26	92.34	67.41	77.93	78.90
SMOTE-CNN	81.16	96.43	69.48	80.76	81.85
ADSYN-CNN	80.10	96.16	67.74	79.49	80.71
VAEGAN-CNN	86.84	88.63	86.84	86.71	87.73
CWVAEGAN-CNN	90.11	90.73	90.11	89.9	90.42

Fig. 11 Performance of different class-balancing methods on KDDTEST+



	Accuracy	Precision	DR	F1 score	G-mean
CNN	52.98	78.60	52.98	54.04	64.53
ROS-CNN	63.43	94.89	58.46	72.35	74.48
SMOTE-CNN	65.34	95.37	60.59	74.1	76.02
ADSYN-CNN	57.76	88.65	55.50	68.26	70.14
CWVAEGAN-CNN	81.53	86.11	81.53	82.4	83.79

Fig. 12 Performance of different class-balancing methods on KDDTEST21



	Accuracy	Precision	DR	F1 score	G-mean
CNN	80.88	84.44	80.88	81.83	82.64
ROS-CNN	81.7	77.32	94.49	85.05	85.47
SMOTE-CNN	82.44	78.05	94.77	85.6	86.00
ADSYN-CNN	82.15	77.76	94.65	85.38	85.79
VAEGAN-CNN	86.65	87.79	86.65	85.41	87.22
CWVAEGAN-CNN	88.96	90.46	88.96	87.71	89.71

Fig. 13 performance of different class-balancing methods on UNSWNB15

Table 8 Performance comparison of different IDS on KDDTEST

Model	Acc	Precision	DR	F1	G-mean
RNN-IDS [15]	83.28	73.06	73.12	83.22	**84.09**
IGAN-IDS [23]	84.45	/	/	84.17	/
TSE-IDS [16]	85.79	88	86.8	/	/
SAVAER-DNN [22]	89.36	/	**95.98**	**90.08**	/
I-SiamIDS [17]	80	80.73	86.87	83.66	/
ICVAE-DNN [19]	85.97	**97.39**	77.43	86.27	/
Ours	**90.11**	90.73	90.11	89.9	81.76

Table 9 Performance comparison of different IDS on KDDTEST21

Model	Acc	Precision	DR	F1	G-mean
LSTM4 [40]	66.74	/	/	76.21	/
GFBLS [40]	67.47	/	/	76.29	/
RNN-IDS [15]	68.55	/	/	/	/
TSE-IDS [16]	72.52	85.00	72.50	/	/
SAVAER-DNN [22]	80.30	/	**95.19**	**86.92**	/
ICVAE-DNN [19]	75.43	**96.2**	72.86	68.62	/
Ours	**81.53**	86.11	81.53	82.4	**70.71**

According to Table 7, on the KDDTEST+, the accuracy of CWVAEGAN-1DCNN is 90.11%, which is higher than other methods. Precision, F1 score and G-mean of CWVAEGAN-1DCNN are, 90.73%, 89.9%, and 81.76%, which are close to the best value of other methods CWVAEGAN-1DCNN has better performance on KDDTEST+.

In Table 8, on the KDDTEST21, the accuracy and precision of CWVAEGAN-1DCNN are 81.53% and 86.11% respectively, which are higher than other methods. Recall and F1 score of CWVAEGAN-1DCNN are 81.53% and 82.4% respectively, which ranked second in all methods. CWVAEGAN-1DCNN has better performance on KDDTEST21.

Table 9 shows that CWVAEGAN-1DCNN achieves 88.86% accuracy, 90.46% precision, 88.96% Dr, 87.71% F1 score, and 80.48% G-mean on UNSWNB15, which are lower than other IDS sightly.

CWVAEGAN-1DCNN's results in accuracy, F1 score, and DR ranked third, but there are only small gaps. CWVAEGAN CNN performance on UNSWNB15 dataset is not as good as that on NSLKDD. This is because the learning effect of 1DCNN classifier on UNSWNB15's training set isn't good. Figure 8 illustrates that the learning effect of CWVAEGAN-1DCNN on UNSWNB15's training set can only reach 90%, which limits the performance of the overall IDS.

All the above results show that CWVAEGAN-1DCNN's performance is better than other methods. We can conclude that CWVAEGAN can effectively solve the class-imbalance problem, and CWVAEGAN-1DCNN has an efficient intrusion detection capability, but there is still a slight gap compared with advanced methods. When there are a large number of samples in training set, the classifier can not fully learn, which limits the performance of the overall IDS.

4.4.5 Experiment 5: visualization comparison of class-balancing methods

Starting with the first evaluation, we take a look at the mean and standard deviations of the real and fake dataset, the log transformed values of all the numeric columns are. The assumption is made that if a synthesizer already fails to capture these basic properties, more derived features will likely suffer the same fate. In the NSL-KDD dataset (Fig. 14). We observe that all of the five synthesizers capture these properties, while VAE has a hard time reproducing these values.

Then we draw the column correlation matrix between the datasets synthesized by different methods and the original dataset, and compare the absolute difference between the synthesizer and the original dataset correlation matrix. We can intuitively see the difference between data generated and original data. Figure 15 shows the column wise correlations of the NSL-KDD dataset.

Then we compare the absolute difference of 4 synthesizers. In Fig. 16, it can be seen intuitively that CWVAEGAN has the best effect among all synthesizers. SMOTE and ROS have good fitting effect on discrete features, that the difference between generated data is small, but they do not

Table 10 Performance comparison of different IDS on UNSWNB15

Model	Acc	Precision	DR	F1	G-mean
IGAN-IDS [23]	82.53	/	/	82.86	/
TSDL [18]	89.13	/	/	/	/
TSE-IDS [16]	91.27	/	91.30	/	/
SAVAER-DNN [22]	**93.01**	/	91.94	**93.54**	/
ICVAE-DNN [19]	89.08	86.05	**95.68**	90.61	/
KG-DBN [20]	86.49	/	86.49	/	/
Ours	88.86	**90.46**	88.96	87.71	**80.48**

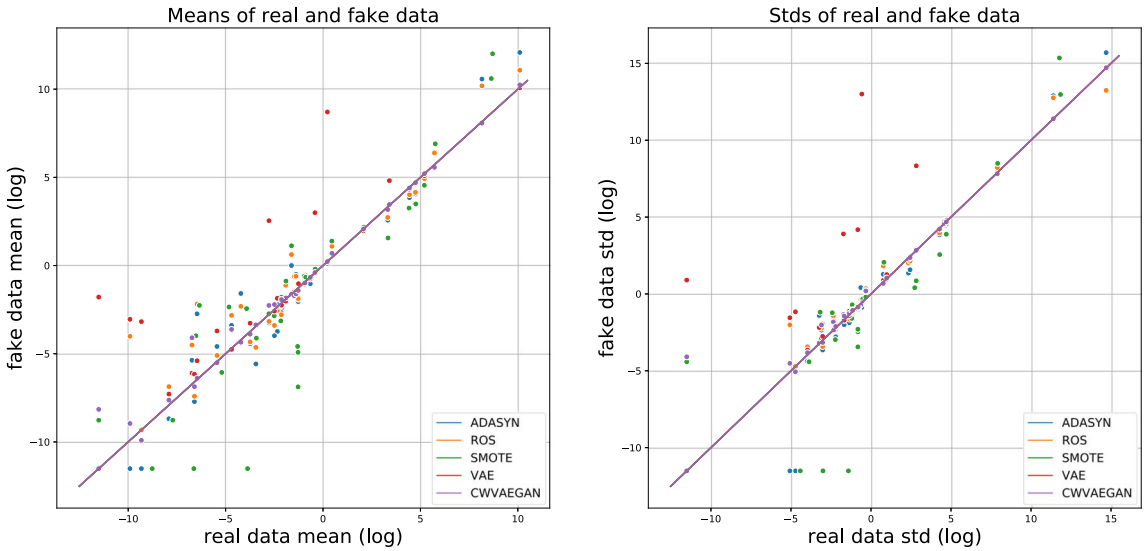


Fig. 14 Mean and standard deviations of each column of the real and synthetic NSL-KDD dataset

perform well on the continuous data. On the contrary, VAE performs well on continuous data sets, while the fitting effect on discrete features is general. Only CWVAEGAN performed best among the five methods.

4.4.6 Experiment 6: statistical test

In order to illustrate the superiority of the methods proposed in this paper, we use statistical test to see if the performance of the four methods in Tables 8, 9 and 10 on three datasets are significantly different. RNN-IDS, IGAN-IDS and I-siamIDS accuracy values are missing, so we discard them and carry out significance test on the remaining four methods. We take the accuracy of the four methods on the three datasets as the standard of significance test. First, we use Friedman test [42] to detect whether there are significant differences between the four methods. Friedman test is a sort based statistical method, When the $p-value$ obtained by Friedman test is less than 0.05, we think that these methods have significant differences. The $p-value$ of these methods is 0.532, which is greater than 0.05, and we think that these four methods are comparable.

Similarly, we compared the six class-balancing methods mentioned in Experiment 3, and through Friedman test obtained a $p-value$ of 0.0121, which is obvious that there are significant differences between the six methods. Friedman test can only be used to determine whether there is a significant difference between the measurement results of multiple models, but it does not know whether there is a difference between any two models, which is what post-hoc nimenyi test [42] aims to solve. We use post-hoc nimenyi test and get a $p-value$ matrix, as Table 11

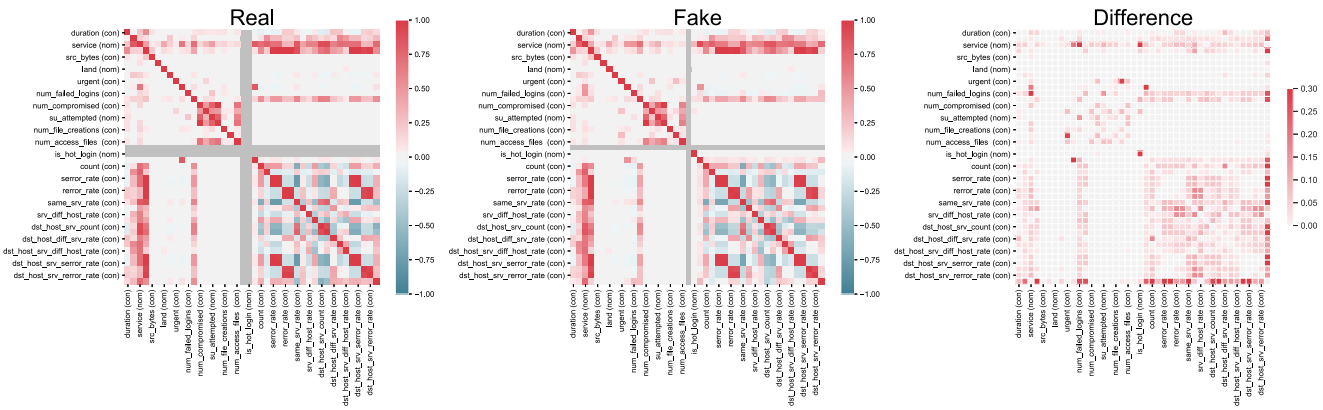


Fig. 15 Associations per column from the real dataset and CWVAEGAN. The third one is the absolute difference between the synthesizer and the original

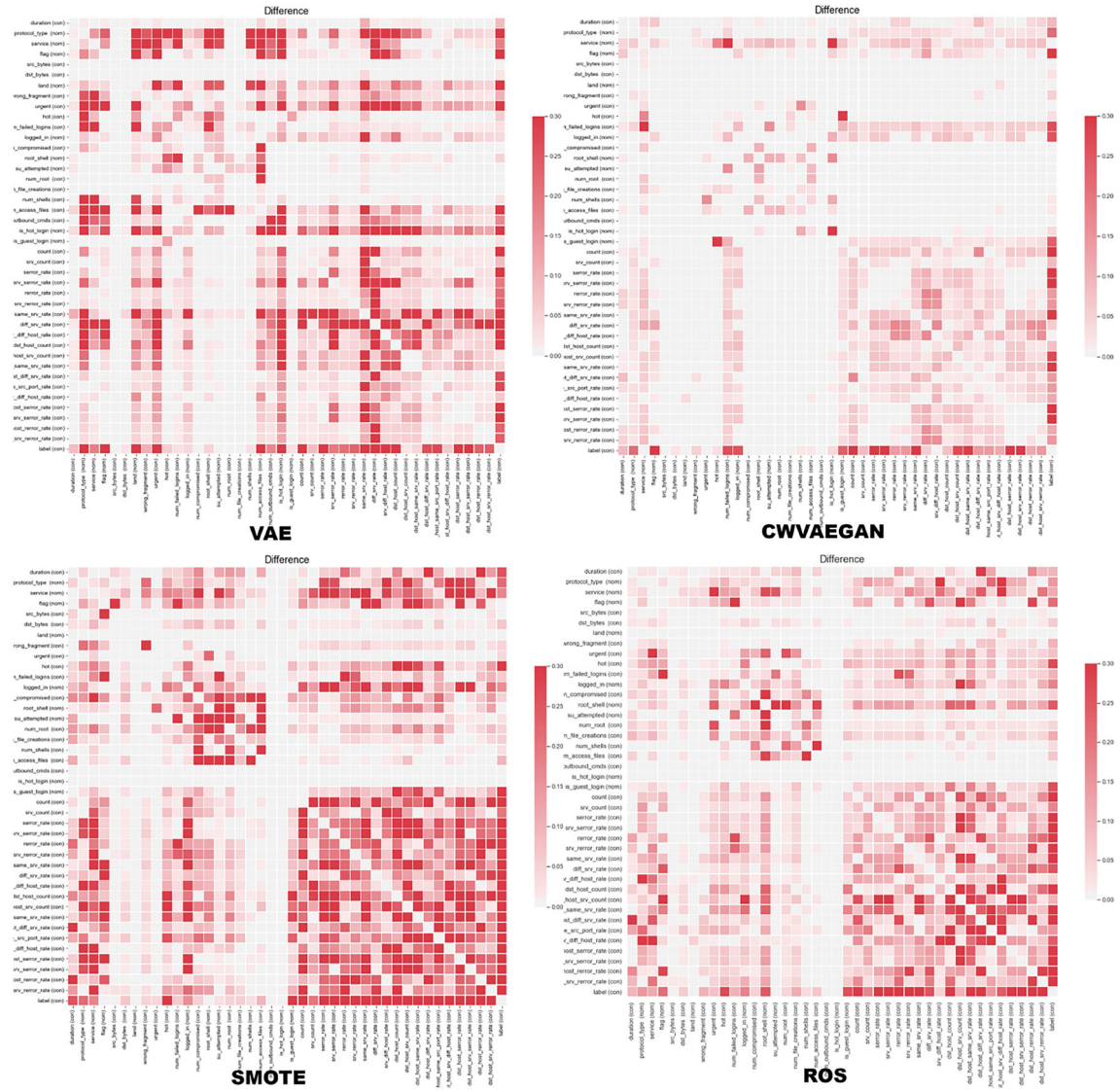


Fig. 16 The absolute difference of associations per column between real dataset and each of the synthesizers

shows the relationship of various class-balancing methods. By drawing the thermodynamic diagram of p-value, we can intuitively see that CWVAEGAN-CNN is comparable with SMOTE-CNN and VAEGAN-CNN, and is significantly different from the other three methods.

5 Conclusions

To solve the class imbalance problem in IDS, CWVAEGAN is proposed. We build an IDS based on CWVAEGAN (CWVAEGAN1D-CNN) to solve the problem of class

Table 11 P-value matrix of post-hoc nimenyi test for different class-balancing methods

	CNN	ROS-CNN	SMOTE-CNN	ADSYN-CNN	VAE/GAN-CNN	Ours
CNN	1	0.9	0.364	0.878	0.092	0.013
ROS-CNN	0.9	1	0.878	0.9	0.500	0.155
SMOTE-CNN	0.364	0.878	1	0.9	0.9	0.752
ADSYN-CNN	0.878	0.9	0.9	1	0.626	0.245
VAE/GAN-CNN	0.092	0.500	0.9	0.626	1	0.9
Ours	0.013	0.155	0.752	0.245	0.9	1

imbalance. Compared with the existing IDS, CWVAEGAN is used to generate minority classes samples, which improves the overall performance of the IDS. We apply CWVAEGAN-1DCNN to 2 datasets, and the experimental results shows that our method is superior to the other 18 algorithms, including the most advanced detection technology. But the performance on UNSWNB15 dataset is slightly inferior to other advanced methods, which should be further improved.

Acknowledgements This work is supported by the National Science Foundation of China (61806219, 61703426 and 61876189), by National Science Foundation of Shaanxi Provence (2021JM-226) by Young Talent fund of University and Association for Science and Technology in Shaanxi, China (20190108, 20220106), and by and the Innovation Capability Support Plan of Shaanxi, China (2020KJXX-065).

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

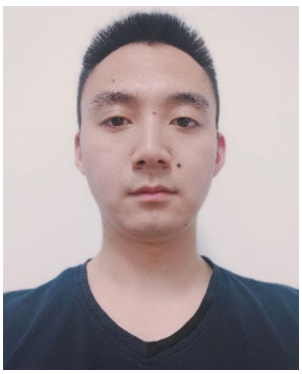
1. Grahn K, Westerlund M, Pulkkis G (2017) Analytics for network security: a survey and taxonomy. In: Alsmadi IM, Karabatis G, Aleroud A (eds) Information fusion for cyber-security analytics. Springer International Publishing, Cham. pp 175–193
2. Panda M, Patra M (20071) Network intrusion detection using naive bayes. p 7
3. Hasan MdA, Nasser M, Pal B, Ahmad S (2014) Support vector machine and random forest modeling for intrusion detection system (IDS). J Intell Learn Syst Appl 06:45–52. https://doi.org/10.4236/jilsa.2014.61005
4. Yang Y, Zheng K, Wu C, Yang Y (2019) Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. Sensors 19:2528. https://doi.org/10.3390/s19112528
5. Srivastava A, Valkov L, Russell C et al (2017) VEEGAN: reducing mode collapse in gans using implicit variational learning
6. Tavallaee M, Bagheri E, Lu W, Ghorbani A (2009) A detailed analysis of the KDD CUP 99 dataset. IEEE Symposium Comput Intell Security Defense Appl, CISDA 2. https://doi.org/10.1109/CISDA.2009.5356528
7. Bishop C (2006) Pattern recognition and machine learning. In: Journal of electronic imaging, pp 140–155
8. Gulrajani I, Ahmed F, Arjovsky M et al (2017) Improved training of wasserstein GANs
9. Larsen A, Sønderby S, Winther O (2015) Autoencoding beyond pixels using a learned similarity metric
10. Wang Y, Wong J, Miner A (2004) Anomaly intrusion detection using one class SVM
11. Deng H (2003) SVM-based intrusion detection system for wireless ad hoc networks
12. Moradi M, ZULKERNINE M (2014) A Neural network based system for intrusion detection and classification of attacks
13. Li Z, Qin Z, Huang K et al (2017) Intrusion detection using convolutional neural networks for representation learning. In: Liu D, Xie S, Li Y (eds) Neural Information Processing. Springer International Publishing, Cham, pp1 858–866
14. Ma T, Wang F, Cheng J et al (2016) A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. Sensors 16:1701. https://doi.org/10.3390/s16101701
15. Yin C, Zhu Y, Fei J, He X (2017) A deep learning approach for intrusion detection using recurrent neural networks. IEEE Acce 5:21954–21961. https://doi.org/10.1109/ACCESS.2017.2762418
16. Adhi Tama B, Comuzzi M, Rhee KH (2019) TSE-IDS: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. IEEE Acc, p 7. https://doi.org/10.1109/ACCESS.2019.2928048
17. Bedi P, Gupta N, Jindal V (2021) I-SiamIDS: an improved Siam-IDS: for handling class imbalance in network-based intrusion detection systems. Appl Intell 51:1133–1151. https://doi.org/10.1007/s10489-020-01886-y
18. Khan F, Gumaei A, Derhab A, Hussain A (2019) A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access:1–10. https://doi.org/10.1109/ACCESS.2019.2899721
19. Yang Y, Zheng K, Wu C, Yang Y (2019) Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. Sensors 19:2528. https://doi.org/10.3390/s19112528
20. Tian Q, Han D, Li K-C, Liu X, Duan L, Castiglione A (2020) An intrusion detection approach based on improved deep belief network. Appl Intell:50, https://doi.org/10.1007/s10489-020-01694-4
21. Lee J, Park K (2019) GAN-based imbalanced data intrusion detection system. Pers Ubiquit Comput, https://doi.org/10.1007/s00779-019-01332-y
22. Yang Y, Zheng K, Wu B et al (2020) Network intrusion detection based on supervised adversarial variational auto-encoder with regularization. IEEE Access 8:42169–42184. https://doi.org/10.1109/ACCESS.2020.2977007
23. Huang S, Lei K (2020) IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. Ad Hoc Netw 105:102177. https://doi.org/10.1016/j.adhoc.2020.102177
24. Puri A, Gupta M (2019) Comparative analysis of resampling techniques under noisy imbalanced datasets
25. Kuang L, Zulkernine M (2008) An anomaly intrusion detection method using the CSI-KNN algorithm
26. Abdulhammed R, Faezipour M, Abuzneid A, Abumallouh A (2019) Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. vol 3:1–4, https://doi.org/10.1109/LSENS.2018.2879990
27. Cieslak D (2006) Combating imbalance in network intrusion datasets
28. Chawla N, Bowyer K, Hall L, Kegelmeyer W (2002) SMOte: synthetic minority over-sampling technique. J Artif Intell Res (JAIR) 16:321–357. https://doi.org/10.1613/jair.953

29. Qazi N, Raza K (2012) Effect of feature selection. smote and under sampling on class imbalance classification. https://doi.org/10. 1109/UKSim.2012.116
30. Tesfahun A, Bhaskari L (2013) Intrusion detection using random forests classifier with smote and feature reduction
31. Kingma D, Welling M (2014) Auto-Encoding Variational Bayes
32. Goodfellow IJ, Pouget-Abadie J, Mirza M et al (2014) Generative adversarial networks. arXiv:14062661 [cs, stat]
33. Mirza M, Osindero S (2014) Conditional generative adversarial nets
34. Arjovsky M (2017) Wasserste GAN, Bottou, L
35. Xiang Q, Wang X, Song Y, Lei L, Li R, Lai J (2020) One-dimensional convolutional neural networks for high-resolution range profile recognition via adaptively feature recalibrating and automatically channel pruning. Int J Intell Syst, p 36. https://doi.org/10.1002/int.22302
36. Misra D (2019) Mish: a self regularized non-monotonic neural activation function
37. Moustafa N, Slay J (2016) The evaluation of Network Anomaly Detection Systems: statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset, pp 1–14. https://doi.org/10.1080/19393555.2015.1125974
38. He H, Bai Y, Garcia E, Li S (2008) Adaptive synthetic sampling approach for imbalanced learning, ADASYN
39. Adhi Tama B, Comuzzi M, Rhee KH (2019) TSE-IDS: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. IEEE Acc vol 7, https://doi.org/10.1109/ACCESS.2019.2928048
40. Li Z, Gonzalez Rios A, Xu G, Trajkovic L (2019) Machine learning techniques for classifying network anomalies and intrusions
41. Park N, Mohammadi M, Gorde K, Jajodia S, Park H, Kim Y (2018) Data synthesis based on generative adversarial networks. Proc VLDB Endow 11:1071–1083. https://doi.org/10.14778/3231751.3231757
42. Benavoli A, Corani G, Mangili F (2015) Should we really use post-hoc tests based on mean-ranks?. arXive:1505.02288 Accessed 07 07 2022

Jiaxing He received the M.S. degree from Air Force Engineering University, Xi'an, China, in 2021. He is currently working toward the Ph.D. degree in with the College of Air and Missile Defense, Air Force Engineering University. His research interest covers cyber intrusion detection and generative adversarial networks.

Xiaodan Wang received the Ph.D. degree in computer science from Northwestern Polytechnical University, Xi'an, China. She is currently a Professor and a Ph.D. Advisor with the College of Air and Missile Defense, Air Force Engineering University. Her research interests include pattern recognition, machine learning, and artificial intelligence.

Yafei Song received the Ph.D. degree in computer science from Air Force Engineering University, Xi'an, China. He is currently an Associate Professor and a M.S. Advisor with the College of Air and Missile Defense, Air Force Engineering University. His research interests include pattern recognition, intelligent information processing, and evidential reasoning.



Qian Xiang received the M.S. degree from Air Force Engineering University, Xi'an, China, in 2020. He is currently working toward the Ph.D. degree in with the College of Air and Missile Defense, Air Force Engineering University. His research interests include pattern recognition, few-shot learning and intelligent information processing.

Chen Chen received the bachelor's degree from National University of Defense Technology, Changsha, China, in 2012. He is currently working toward the M.S. degree in with the College of Air and Missile Defense, Air Force Engineering University. His research interest includes network operation and maintenance, network security protection and network security situation assessment.