Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.055497

ARTICLE

A DDoS Identification Method for Unbalanced Data CVWGG

Haizhen Wang[1,2,*], Na Jia[1,2], Yang He[1] and Pan Tan[1,2]

[1]College of Computer and Control Engineering, Qiqihar University, Qiqihar, 161006, China

[2]Heilongjiang Key Laboratory of Big Data Network Security Detection and Analysis, Qiqihar University, Qiqihar, 161006, China

*Corresponding Author: Haizhen Wang. Email: 01559@qqhru.edu.cn

Received: 28 June 2024 Accepted: 08 October 2024 Published: 19 December 2024

ABSTRACT

As the popularity and dependence on the Internet increase, DDoS (distributed denial of service) attacks seriously threaten network security. By accurately distinguishing between different types of DDoS attacks, targeted defense strategies can be formulated, significantly improving network protection efficiency. DDoS attacks usually manifest as an abnormal increase in network traffic, and their diverse types of attacks, along with a severe data imbalance, make it difficult for traditional classification methods to effectively identify a small number of attack types. To solve this problem, this paper proposes a DDoS recognition method CVWGG (Conditional Variational Autoencoder-Wasserstein Generative Adversarial Network-gradient penalty-Gated Recurrent Unit) for unbalanced data, which generates less noisy data and high data quality compared with existing methods. CVWGG mainly includes unbalanced data processing for CVWG, feature extraction, and classification. CVWGG uses the CVAE (Conditional Variational Autoencoder) to improve the WGAN (Wasserstein Generative Adversarial Network) and introduces a GP (gradient penalty) term to design the loss function to generate balanced data, which enhances the learning ability and stability of the data. Subsequently, the GRU (Gated Recurrent Units) are used to capture the temporal features and patterns of the data. Finally, the logsoftmax function is used to differentiate DDoS attack categories. Using PyCharm and Python 3.10 for programming and evaluating performance with metrics such as accuracy and precision, the results show that the method achieved accuracy rates of 96.0% and 97.3% on two datasets, respectively. Additionally, comparison and ablation experiment results demonstrate that CVWGG effectively mitigates the imbalance between DDoS attack categories, significantly improves the classification accuracy of different types of attacks and provides a valuable reference for network security defense.

KEYWORDS

Conditional variational autoencoder; generating adversarial networks; DDoS attack

1 Introduction

DDoS [1] attacks are one of the most destructive and common threats to network security. An attacker can cause a target system to become unavailable or even crash by sending a large number of requests to a specific network, system, or service that exceeds its processing capacity. DDoS attack causes a target system to become unavailable by overloading its network bandwidth, processor, memory, or other critical resources, resulting in an inability to process legitimate requests, causing

the target system to become unavailable. Currently, DDoS attacks are frequent and growing in size, time, speed and complexity because DDoS attack tools and services are widely available and easy to use, even for less technically qualified attackers. In addition, enterprise data centers, public cloud computing service providers, Internet infrastructure service providers, gaming service providers and Internet infrastructure service providers are the focus of their attacks. DDoS attacks are not easy to detect due to their diverse types, making them even more pervasive.

Distinguishing between the types of DDoS attacks helps to understand more information about the attacks so that more effective countermeasures can be formulated. The disadvantage of existing DDoS attack detection algorithms is that the detection performance is closely related to the selected features and classifiers, which cannot adapt well to the massive data environment. There are rarely studies on DDoS attack type differentiation and the raw traffic data is not balanced. In the proposed unbalanced processing method, the generated data has random noise and instability problems, which affect the DDoS differentiation results.

Based on this, we propose a method for DDoS attack differentiation based on CVWGG, which combines the advantages of CVAE and WGAN, to synthesize the data of the lacking attack types using generative modeling to alleviate the imbalance of the original traffic data and to improve the generalization performance of the model. Not only does it use CVAE as the generator of WGAN to generate high-quality synthetic data, but it also introduces GP to solve the problem of exploding or disappearing gradients that WGAN may encounter during the training process, which makes the training process more stable, thus effectively improving the problem that a few classes will be misclassified and unclassified by the existing detection methods in the face of data imbalance problems. Hence, to differentiate more accurately between DDoS attack types. The main contributions of this paper are as follows:

1. Construct balanced training samples using CVWG. The method uses the CVAE as a generator for WGAN which provides more diverse and higher-quality samples to support adversarial training.

2. The introduction of the GP in the loss function of CVWG significantly alleviates the problems of gradient vanishing and gradient explosion in model training and improves stability.

3. Construct a GRU model to perform feature extraction for DDoS attacks. The model extracts time series information to complete DDoS attack classification and has a low computational cost.

4. Evaluate on the dataset CICDDoS2019 which contains the latest 12 types of DDoS attacks and CICIDS2017 which contains 5 types of DDoS attacks. The experimental results show that this method is compared in terms of accuracy, recall and the experimental results outperform the comparison model.

The rest of the paper is organized as follows. Section 2 briefly describes the related work in this area. In Section 3, the design of the CVWGG model is elaborated. Section 4 provides the performance evaluation of the experiments. Section 5 gives related discussion. Finally, the conclusions of the paper are discussed in Section 6.

2 Related Work

DDoS is the most common type of network disruption attack and one of the most serious problems faced by security managers; researchers have been exploring DDoS detection and defense methods. Bhayo et al. [2] designed a detection module using machine learning, which proved to be effective in detecting DDoS attacks in IoT (Internet of Things) environments. Lu et al. [3] proposed an information entropy and deep learning-based DDoS attack detection method. Suspicious traffic is first

detected by the information entropy of the controller and then packet-based detection is performed using CNN (Convolutional Neural Networks) models to distinguish normal traffic from DDoS attack traffic. Liu et al. [4] pointed out that a large number of DDoS and its variants and data imbalance leads to the fact that most of the methods in DDoS detection are binary classifications, which are not able to effectively detect a small number of attacks and differentiate between the types of DDoS attacks.

It has been shown that most of the research on intrusion detection and network attack detection focuses on the detection rate of the models [5,6] but the data imbalance problem leads to the classification models favoring the categories with more samples in the attack categories and ignoring the categories with fewer samples. To address this problem, researchers have used generative adversarial networks to generate synthetic data samples of a few categories [7,8], which helps classification models achieve better generalization and lower bias.

Table 1 presents relevant literature on intrusion detection based on data imbalance. Kumar et al. [9] proposed the WCGAN (Wasserstein Conditional Generative Adversarial Network) in combination with the XGBoost (Extreme Gradient Boosting) classifier. GP is used along with WCGAN for stable learning of the model, and the final model achieves 96.66% accuracy. Huang et al. [10] designed a solution IGAN-IDS using the GAN (Generative Adversarial Network) to solve the category imbalance problem and to simulate the unknown anomalies. Lee et al. [11] proposed a model that combines GAN with Random Forest classification to solve the problem of data imbalance in intrusion detection systems. Literature [12,13], on the other hand, employed CWGAN (Conditional Wasserstein Generative Adversarial Network) to generate new samples for minority class attacks to alleviate the class imbalance problem and validate the effectiveness of the proposed method through simulation experiments. However, the CWGAN network may have limitations in sample patterns when generating data, resulting in the generated samples often lacking diversity. Therefore, literature [14] used an improved CVAE to learn the intrinsic distribution of traffic data to generate specified attack samples, which increases the diversity of samples and thus improves the detection rate of minority class attacks. To address the problem of insufficient VAE decoder training in VAE (Variational Autoencoder) methods, Chuang et al. [15] applied B-VAE (Batch-Variational Autoencoder) to provide sufficient decoder training by replicating data batches and forming multiple VAE batches to train a decoder for each data. Since VAE is trained by maximizing likelihood estimation, the generated samples are usually of poor quality. Therefore, some researchers have gradually combined the two; Liu et al. [16] combined VAE with CWGAN to generate new samples using the VAE-CWGAN model to solve the problem of class imbalance in the dataset so that the classification model is no longer biased towards the majority of the classes, but it also exposes the model to the problem of unstable training.

Table 1: Research on Intrusion detection based on data imbalance

Literature	Method	Advantages	Limitations
[9]	WCGAN + XGBoost	The model achieves an accuracy of 96.66% using GP optimization.	Training may take a long time and have high computational costs.

(Continued)

Table 1 (continued)

Literature	Method	Advantages	Limitations
[10]	IGAN-IDS	Address category imbalance and handle unknown anomalies.	Did not discuss the performance of detecting minority class attacks in detail.
[11]	GAN + RF	Combined GAN with Random Forest, improving the detection performance of minority classes.	Random forest may perform poorly with high-dimensional data.
[12,13]	CWGAN	Effectively alleviates category imbalance, with its effectiveness validated through simulations.	The generated samples may lack diversity.
[14]	CVAE	Increased sample diversity improves the detection rate of minority class attacks.	May experience performance bottlenecks with large-scale data.
[15]	B-VAE	Provides sufficient decoder training to enhance the diversity of generated samples.	The samples generated by VAE are of low quality.
[16]	VAE + CWGAN	Addresses the category imbalance problem and improves the model's generalization ability.	The training process may be unstable and prone to gradient vanishing issues.

Therefore, current DDoS detection methods are primarily limited to binary classification, distinguishing only between attack traffic and normal traffic. However, further differentiating between different types of DDoS attacks can help implement targeted defense strategies, and accurately identifying the type of attack can optimize the allocation of defense resources, thereby avoiding unnecessary resource waste. Most existing data balancing methods face issues such as insufficient diversity of generated data, unstable training processes, and gradient vanishing problems. To solve the above problem, this paper proposes a method that combines the CVAE, WGAN with the introduction of a gradient penalty term and finally classifies them by GRU to solve the problem of data imbalance between the classes of the original DDoS attack. By using CVAE as the generator of WGAN, the model can effectively generate diverse attack samples by using the attack types as conditions during the training process, which helps to balance the dataset. The discriminator of WGAN takes advantage of the Wasserstein distance, which can more accurately measure the gap between the generated samples and the real data and enhances the training stability of the generator and the generation of the Authenticity. The introduction of the gradient penalty term further improves the training stability of WGAN and avoids the problems of pattern collapse and unstable training in generative adversarial network training. GRU is an efficient recursive neural network structure that can capture the temporal relationship in the sequential data and better extract features.

3 Proposed Methodology CVWGG

3.1 Modelling Framework

This model framework is shown in Fig. 1.

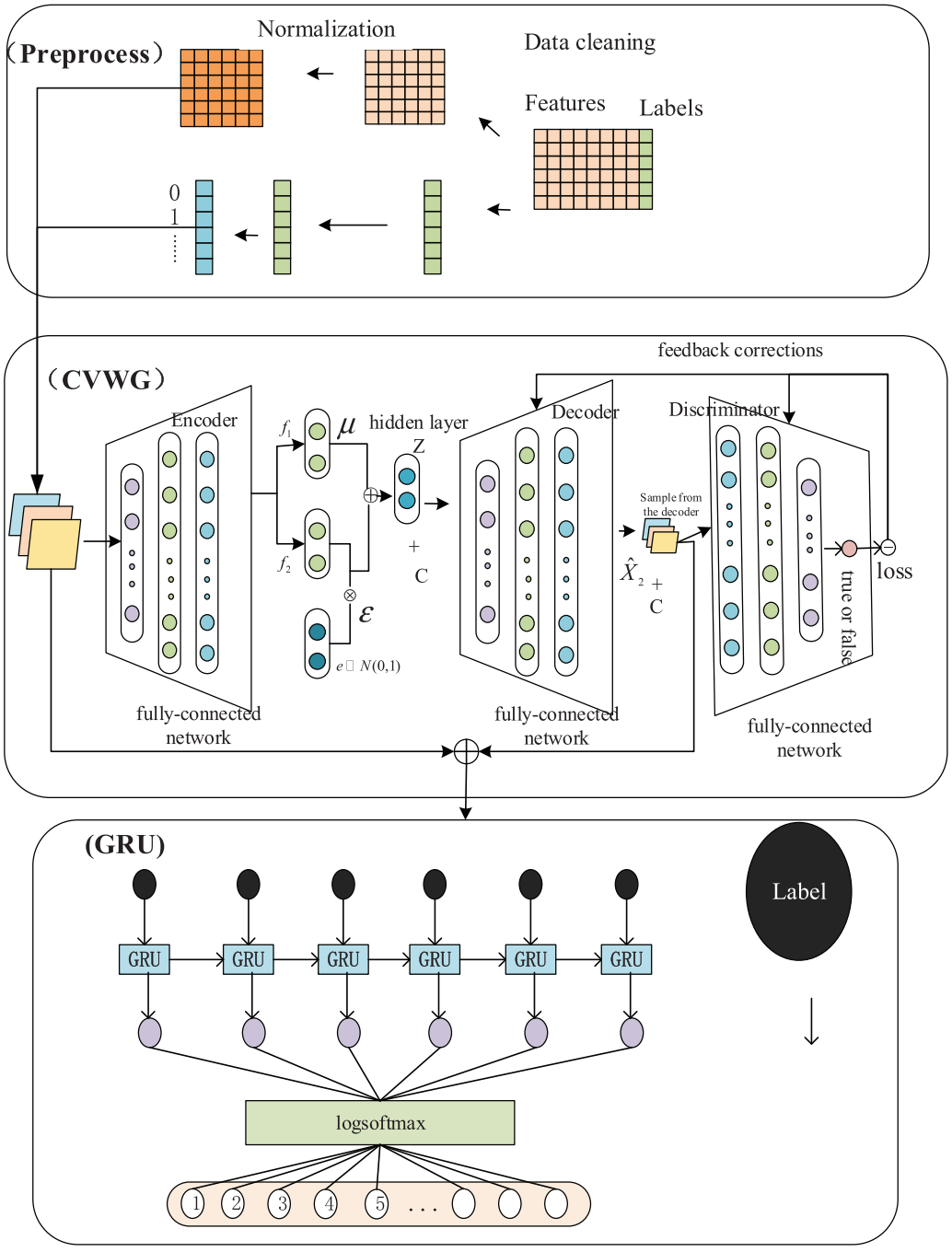


Figure 1: CVWGG model diagram

As shown in Fig. 1, this model contains three main parts: Preprocess, preprocessing operations such as cleaning, coding and normalization of the original data; CVWG (Conditional Variational Autoencoder-Wasserstein Generative Adversarial Network-gradient penalty), data imbalance processing, using CVAE to generate samples of a few categories and balancing the dataset by improving the quality and diversity of the generated samples through WGAN; GRU, passing the balanced data into the GRU and classification, extracting the sequence features and completing the classification.

3.2 Problem Statement

To improve the data quality and stability of existing methods, we propose the CVWGG method, which is a model that introduces the CVAE into the WGAN network. Doing so effectively solves the problem of modeling when generating data for WGAN networks while using gradient penalty techniques to improve the training difficulty of WGAN. Ultimately, we use the GRU network, which has a small number of parameters and a low computational cost, to complete the differentiation of the types of DDoS attacks, thus improving the accuracy and efficiency of detection.

3.3 Unbalanced Data Processing Method-CVWG

In a primitive GAN network, noise is usually randomly sampled from some distribution, which is then fed into a generator that is adversarially trained to generate fake data. However, these noises usually do not contain information about the valid features, so after passing through the generator, the generated data hardly contains specific information. Even with extensive training, the generated fake data is difficult to match the real data. Autoencoders differ from generative adversarial networks in that they can efficiently extract features from the input data and then decode these features to restore the original data as much as possible. VAE take this a step further by improving on the shortcomings of autoencoders. Autoencoders can only restore data but cannot generate new data. To solve this problem, extracts features from the input data, calculates the mean and variance of these features, and then fits them into a Gaussian distribution. It then samples from this distribution and generates new data through the decoder. This allows the network to generate data that is as similar as possible to the real data while maintaining the diversity and variability of the generated data. Therefore, CVWG is designed by combining the CVAE, WGAN and GP techniques.

The following section will provide a detailed introduction to the CVWG method, with the symbols used listed in Table 2.

Table 2: Parameters of this section and their meanings

Parametric	Significance
X	Input attack traffic
c	conditional variable
μ, ε	Mean and variance
$De(\cdot)$	decoder network
$Dis(\cdot)$	discriminator network
λ	Gradient penalty factor
$P_{\hat{x}}$	A distribution between the real data distribution $P_{data(x)}$ and the generated data distribution $P_{De(x)}$

3.3.1 CVWG Network Framework

The network structure involved in the CVWG is shown in Fig. 2 below, which mainly includes an encoder, a decoder, and a discriminator. The following provides a detailed introduction to them.

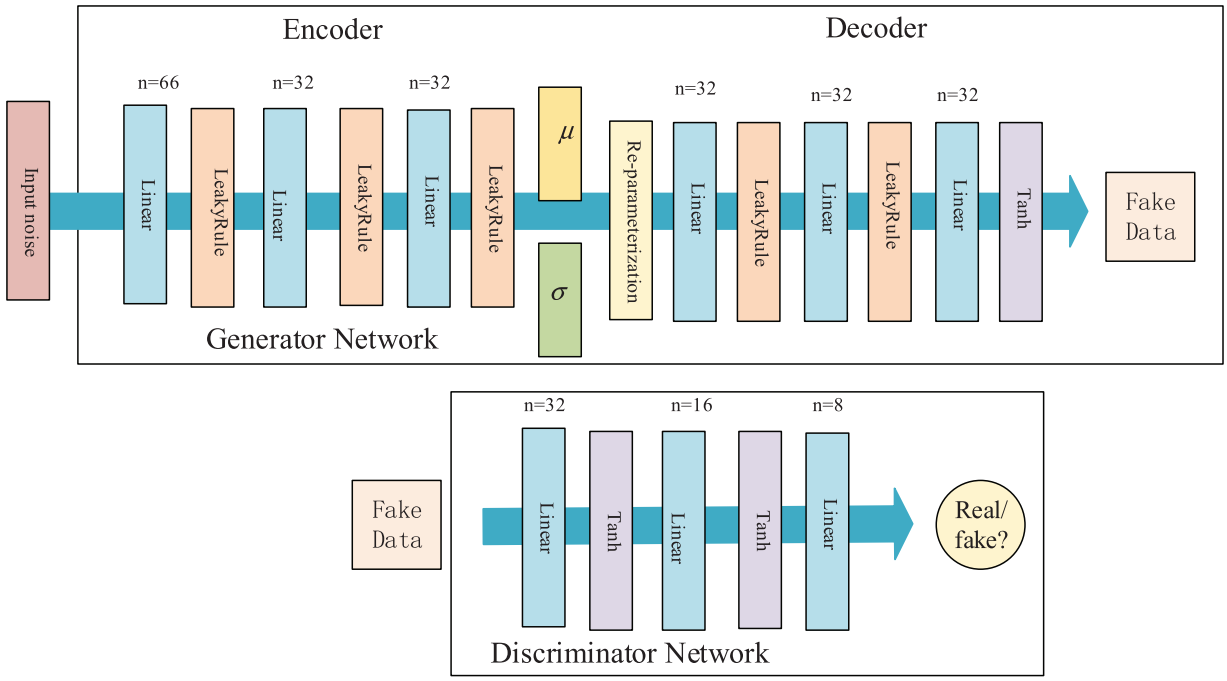


Figure 2: Network structure of CVWG

3.3.2 Encoder

The encoding network transforms the input data X_t (t representing the original dimension of the data) into low latitude mean μ_k and variance ε_k (k representing the encoded dimension) and obtains the intermediate hidden variables z_k through the reparameterization technique. μ_k, ε_k and z_k see Eqs. (1)–(3).

$$\mu_k = f_1(X_t) \tag{1}$$

$$\varepsilon_k = log\varepsilon_k^2 = f_2(X_t) \tag{2}$$

$$z_k = \mu_k + e_k * \exp(\varepsilon_k) \tag{3}$$

where f_1, f_2 represent the forward propagation function of the mean and variance coding network and e_k obey the standard normal distribution $N(0,1)$.

$$p(Z) = \sum_x p(Z|X)\, p(X) = \sum_x N(0,1) \sum_x p(X) = N(0,1) \tag{4}$$

As shown in Eq. (4), $p(Z)$ represents the distribution of the hidden variable space, which is assumed to follow a standard normal distribution to sample the generated data from a known distribution. Losses here are measured using *KLloss* (As shown in Eq. (5)). By introducing the KL divergence, the model is forced to learn a latent variable distribution that is close to a standard normal distribution. This regularization strategy reduces the model's dependence on the training data, thereby

lowering the risk of overfitting.

$$L_{KL} = \frac{1}{2}\sum \mu_k^T \mu_k + sum(\exp(\varepsilon_k) - \varepsilon_k - 1)) \tag{5}$$

3.3.3 Decoder (Generator)

The decoder feeds the hidden layer z_k encoded after the variational operation and the true conditional variable c into the decoder to get the reconstructed completed sample $\widehat{X}_1$ (As shown in Eq. (6)); the vector of the same dimensions z_k^s and the randomly generated natural labels c_g obtained from $N(0,1)$ sampling are fed into the decoder to get the reconstructed sample $\widehat{X}_2$ (As shown in Eq. (7)).

$$\widehat{X}_1 = De(z_k, c) = f_3(z_k, c) \tag{6}$$

$$\widehat{X}_2 = De\left(z_k^s, c_g\right) = f_4(z_k^s, c_g) \tag{7}$$

f_3 represents the forward propagation function from the encoder to the decoder and f_4 represents the forward propagation function for the decoding network.

To compare the difference between the generated and real samples, the mean square error loss function between the real data X and the reconstructed data $\widehat{X}_1$ was used (As shown in Eq. (8)).

$$L_{recon} = \sum\nolimits_{\widehat{X}_1} (\widehat{X}_1 - X)^2 \tag{8}$$

3.3.4 Discriminators

The discriminator inputs the generated samples and generated labels simultaneously with the true labels and outputs the true and false predicted values (As shown in Eq. (9)).

$$Predict = Dis\left\{\left(X, \widehat{X}_1\right), (C_1, C_2)\right\} \tag{9}$$

where Dis stands for discriminator, X, C_1 for raw data and raw labels, $\widehat{X}_1, C_2$ for generated data and generated labels.

The discriminator uses the loss function of WGAN, and the use of the Wasserstein distance instead of the KL scatter of the original GAN network reduces the occurrence of phenomena such as pattern collapse (As shown in Eq. (10)).

$$L_{wgan} = min_{De} max_{Dis} V(Dis, De) = E_{x\sim p_{data(x)}}[Dis(x)] - E_{\tilde{x}\sim p_{De(x)}}[Dis(\tilde{x})] \tag{10}$$

where L_{wgan} represents the loss function of WGAN, x is the real data, $\tilde{x}$ is the generated data, $E[.]$ represents the expectation function, $p_{data(x)}$ is the original data distribution, $De(x)$ is the data generated by the decoder, $p_{De(x)}$ is the data distribution generated by the decoder, and $Dis(x)$ is the discriminator function.

For the function $Dis(x)$ to satisfy the $1-Lipschitz$ condition, the WGAN network restricts the parameter to the range $[-c, c]$ by weight cropping and sets it to $-c$ or c, once the weights exceed this range in the network update, where c is a human-set parameter.

As the design c depends on experience, too big or too small will make the model unstable; To satisfy the conditions $1-Lipschitz$ while avoiding the problem of restricting the expressive ability of the network to make the model unstable due to weight cropping, this paper adopts the strategy of adding a GP to WGAN. By enforcing the gradient of the discriminator to remain close to 1, the gradient penalty limits the complexity of the discriminator, thereby reducing the risk of overfitting.

This regularization approach helps prevent the discriminator from becoming overly reliant on the training data, improving the model's generalization ability on unseen data. The sampling method of the gradient penalty samples is shown in Fig. 3. $p_{data(x)}$ and $p_{De(x)}$ denote the distributions of real data and generated data, respectively. $\tilde{x}_1, \tilde{x}_2$ are samples of the generated data, x_1 and x_2 are samples of the real data. $p_{\hat{x}}$ implicitly defined as sampling uniformly along a straight line between pairs of points sampled from the real data distribution and the generator distribution.

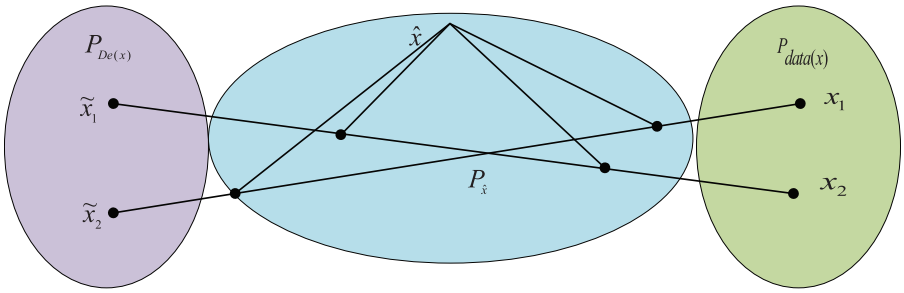


Figure 3: Gradient penalty map

This is done so that a gradient penalty term conforming to $1-Lipschitz$ is added to the objective function, the penalty term enforces the gradient of the discriminator to remain close to 1, thereby preventing overfitting of the discriminator. The loss function of WGAN-GP is show in Eq. (11).

$$L_{wgan-gp} = E_{\tilde{x}\sim p_{De(x)}}\left[Dis\left(\tilde{x}\right)\right] - E_{x\sim p_{data(x)}}\left[Dis\left(x\right)\right] + \lambda E_{\hat{x}\sim P_{\hat{x}}}\left[\left\|\nabla_{\hat{x}} Dis(\widehat{x})\right\|_2 - 1)^2\right] \tag{11}$$

where $\hat{x}$ is the random interpolation between the real data x and the generated data $\tilde{x}$, $\tilde{x}$ satisfied $\hat{x} = x + (1-\varepsilon)\,\tilde{x}, \varepsilon \sim U(0,1)$; $p_{\hat{x}}$ is uniform sampling from a straight line between the real data distribution $p_{data(x)}$ and the sampling points of the generated data distribution $p_{De(x)}$. $E_{\hat{x}\sim p_{\hat{x}}}[(||\nabla_{\hat{x}} Dis\,(x)||_2 - 1)^2]$ is the gradient penalty term.

Integrating all the functions to be optimized, the final objective loss function to be optimized by CVAE-WGAN-GP is show in Eq. (12).

$$L = L_{KL} + L_{recon} + L_{wgan-gp} \tag{12}$$

3.4 GRU and Classification

GRU is a variant of RNN (Recurrent Neural Network) for modeling processing sequence data. GRU is similar to traditional RNN in that it has recurrent connections to capture temporal dependencies in sequences. However, unlike traditional RNN, GRU (see Fig. 4) introduces a gating mechanism to selectively update and pass information through carefully designed gating units. Compared to LSTM (Long Short-Term Memory), GRU has fewer parameters, which means easier training and faster computation. This is useful for processing larger datasets or in resource-limited environments.

We input the data processed by data balancing into the GRU model, the data dimension input into the GRU model is 66 dimensions, meanwhile the model uses a single layer of GRU, through the dropout layer to avoid overfitting, and set the value to 0.5. The time dependence of the traffic data is further processed through the information transfer between the units, and finally the final output is obtained through the logsoftmax layer.

Specifically, GRU introduces two important gating units: reset gate and update gate. The structure of GRU is shown in Fig. 4. The reset gate controls how the past information interacts with the current inputs, while the update gate controls how the new information is retained and updated.

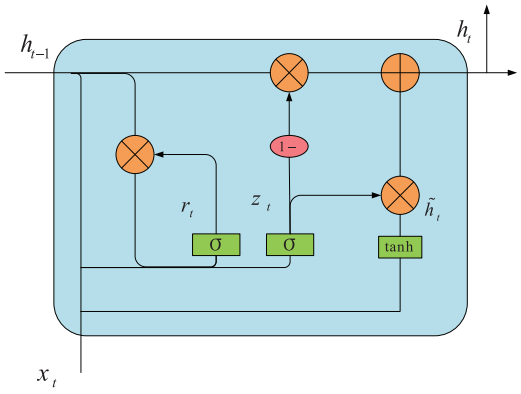


Figure 4: GRU unit structure

Update Gate: The update gate uses the inputs x_t at the current location and the outputs of the hidden layer h_{t-1} at the previous location to select how much of the information from the hidden state is passed on to the current timestep by summing them via a linear change in activation function (As shown in Eq. (13)).

$$Z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{13}$$

where z_t is the output of the update gate, σ is the activation function of *sigmoid*, w_z and b_z are the weight and bias of the update gate, respectively, h_{t-1} is the hidden state of the previous moment, and x_t is the input of the current moment.

Reset Gate: The reset gate uses the input of the current position x_t and the output of the hidden layer at the previous position h_{t-1} to select how much useful information to leave behind by summing them through the activation function after a linear change (As shown in Eq. (14)).

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{14}$$

where r_t is the output of the reset gate, w_r and b_r what are its weight and bias.

Candidate hidden state: A candidate hidden state is computed based on the activation value of the reset gate. This state combines the information from the current input and the previous hidden states. (As shown in Eq. (15)).

$$\widetilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{15}$$

where $\tilde{h}_t$ is a candidate's hidden state. This hidden state contains information from the current input and retains information from previous time steps.

Hidden state update: Based on the activation value of the update gate, the candidate hidden state and the hidden state of the previous time step are fused with the information to get the hidden state of the current time step. This hidden state contains the information from the current input and retains the information from the previous time step (As shown in Eq. (16)).

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{16}$$

where the hidden state h_t is at the current moment, it contains both the information from the current input and retains the information from the previous step.

Output: The hidden state of the last time step after processing the whole sequence is used as a feature representation of the whole sequence, which is fed into the logsoftmax layer for traffic classification

4 Experiments and Results

4.1 Experimental Setup

4.1.1 Experimental Evaluation Indicators

In this paper, widely used evaluation metrics are used to assess the performance of classification models. These metrics are Accuracy, Precision, Recall and F1-score, which are defined by Eqs. (17)–(20).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{17}$$

$$Precision = \frac{(TP)}{(TP + TN + FP + FN)} \tag{18}$$

$$Recall = \frac{(TP)}{(TP + FN)} \tag{19}$$

$$F_1 - score = \frac{(2 * FP)}{(2 * TP + FP + FN)} \tag{20}$$

where TP, TN, FN and FP stand for True Positive, True Negative, False Positive and False Negative, respectively. Accuracy is the ratio of correctly classified samples to the total number of samples, reflecting the overall predictive performance of the model. Recall is the ratio of correctly identified positive samples to the actual number of positive samples, measuring the model's ability to capture positive instances.

Precision is the ratio of correctly identified positive samples to the number of samples predicted as positive, reflecting the accuracy of the model's positive predictions. F1-score is the harmonic means of precision and recall, considering the balance between the two, and particularly suitable for addressing imbalanced datasets.

4.1.2 Experimental Environment

This experiment was conducted under the Windows 11 operating system, using Python 3.6 for programming and modeling; the hardware and software configuration and hyperparameter configuration are shown in Table 3.

Table 3: Experimental hardware and software configuration

Hardware and software	Configuration information
CPU	AMD Ryzen 7 6800 HS Creator Edition
Memory	16 GB

(Continued)

Table 3 (continued)

Hardware and software	Configuration information
Operating system	Windows 11
Programming environment	Python 3.10
Programming tools	PyCharm

4.2 Experimental Results

4.2.1 Dataset Preprocess

The datasets used in this paper are CICDDoS2019 [17] and CICIDS2017 [18]. The most DrDoS_NTP in the CICDDoS2019 dataset has 120,000 entries, the least has only 51 entries, the most in CICIDS2017 has 44,081 entries, and the least has only 11 entries. This huge gap in the number leads to a few categories of types that are not adequately trained due to the limitation of the amount of data, and the accuracy rate is very low. Before conducting the experiments, given the unbalanced nature of the original data, we processed the dataset numerically and normalized it. The following are the detailed processing steps:

(1) Data cleaning: First, we removed missing and incomplete data. Specific operations included deleting rows containing "nan" entries and replacing numbers containing "infinity" with "−1". The purpose of this step was to remove invalid values from the data in order to improve its accuracy.

(2) Data condensation: Attack flow datasets usually contain a large amount of socket information, such as flowID, target IP, secure IP, etc. To facilitate model training, we condense this information to reduce the complexity of the data.

(3) Randomization of samples: To ensure the randomness of the samples, we randomly deleted records while importing the dataset to complete the streamlining of the data. This step helps to eliminate any bias and interference in the data and ensures the randomness of the data.

(4) Feature selection: We removed some attributes in the dataset because they only contained "0" values and did not contribute to model training.

(5) Normalization: In the final stage of preprocessing, the data are normalized. Use Eq. (21) to scale the data into [0, 1].

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{21}$$

x is the sample data, x' is normalize data, x_{max} is the maximum value of the sample data, and x_{min} is the minimum value of the sample data.

In this paper, to better verify the effect of data generation, we extract data in accordance with the proportion of the original data distribution. In the CICDDoS2019 dataset, we scaled 100,000 data items from the data distribution for the training set and 30,000 data items for the test set. After construct the training set, the data was balanced by expanding the data from a few categories to the same 28,000 entries as the maximum number of categories, DrDoS_NTP. Table 4 demonstrates the specific number of entries in the data distribution.

Table 4: CICDDoS2019 dataset distribution

	Class	Original	Training set	Testing set
DDoS	DrDoS_NTP	121,368	28,000	10,000
	TFTP	98,917	22,930	8189
	Syn	4933	11,440	4087
	UDP	28,510	6609	2360
	MSSQL	14,735	3415	1220
	UDPLag	8927	2069	738
	DrDoS_DNS	3669	850	303
	DrDoS_SNMP	2717	6298	225
	NetBIOS	1242	6217	222
	LDAP	1906	330	158
	Portmap	685	171	57
	WebDDoS	51	15	6
Normal	Benign	97,831	22,670	8099

In the CICIDS2017 dataset, 40,006 entries were first extracted as the training set and 13,860 entries as the test set according to the original data distribution. To balance the training set, the sample size of a few categories was expanded to 20,000 entries. The data distribution is shown in Table 5.

Table 5: CICIDS2017 dataset distribution

	Class	Original	Training set	Testing set
DDoS	DoS hulk	231,073	10,000	4620
	DoS goldeneye	10,293	5000	205
	DoS slow loris	5796	2500	115
	DoS slowhttptest	5499	2500	115
	Heartbleed	11	6	5
Normal	BENIGN	440,031	20,000	8800

In this paper, the lesser data in the original category of CICDDoS2019 is enhanced to the same number as the most DrDoS_NTP. To prove that the distribution of the generated data is similar to the distribution of the original data, the TSNE (T-Distributed Stochastic Neighbor Embedding) technique is used to visualize the generated data with the original data, and the results are shown in Fig. 5, from which it can be seen that the distribution of the original data concentrates between −100 and 150, and the generated data also concentrates between −100 and 150, which conforms to the combined original distribution and proves that the generated data is valid.

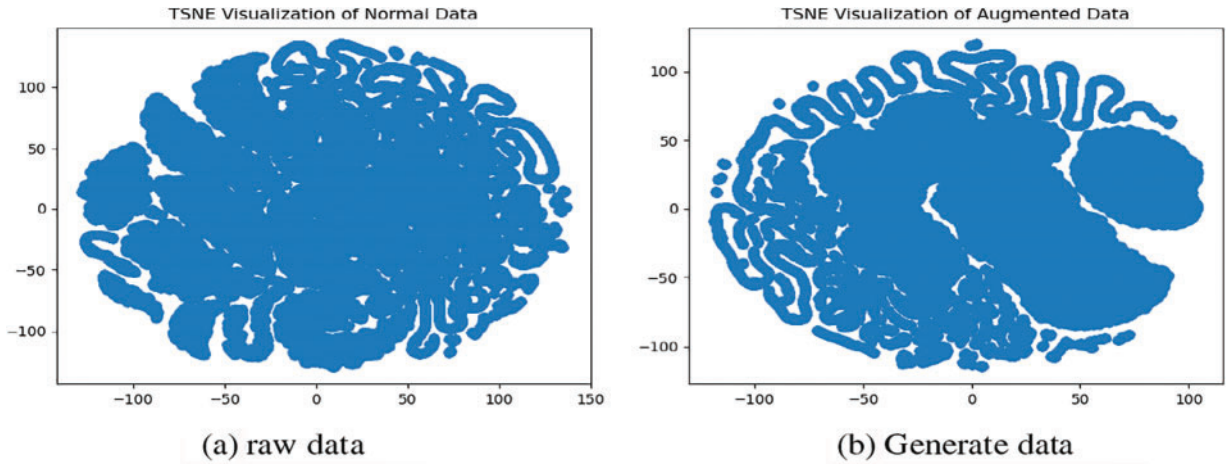


Figure 5: Distribution of CVWG generated data *vs*. original data on CICDDoS2019

4.2.2 Parameter Settings

The tanh activation function is applied to the decoder output, normalizing the output to the range of −1 to 1. After experimental validation, the model uses WGAN-GP's optimizer RMSprop, the learning rate is set to 0.0002, and the batch size is set to 1000 to get the best experimental results, and the training loss diagram of CVWG's model is shown in Fig. 6.

Figure 6: Training loss diagram for CVWG

As shown in Fig. 6, the loss is divided into the discriminator loss and the generator loss, which are in a game with each other, and when they finally meet the threshold value we set at the same time, i.e., when they meet the threshold value we set, then we can use them. Game, and finally, when both meet the threshold value we set, i.e., $|L_{KL} + L_{recon}| \bigwedge |L_{wgan-gp}| \leq y$. Here in this paper, the threshold value y is set to 0.05, which requires both the loss of CVAE and the loss of WGAN-GP to be less than the threshold value y.

4.2.3 Ablation Experiment

This section gives the classification results and confusion matrix of the data before and after balancing the training set, as shown in Table 6 and Fig. 7, which shows that before balancing in CICDDoS2019, due to the huge difference in the number of data categories, the lowest category's indexes are 0, and the precision rate of DrDoS_NTP with the most entries is only 0.42, and the indexes of the classification after balancing are significantly improved, and the precision rate and recall rate of most categories can reach 1. The precision and recall of most categories can reach 1.

Table 6: Comparison of results before and after balancing on CICDDoS2019

Label	Class	Before balancing			After balancing		
		Precision	Recall	F1-score	Precision	Recall	F1-score
0	Benign	0.54	0.37	0.44	1.00	0.97	0.98
1	UDP	0.20	0.00	0.01	1.00	1.00	1.00
2	MSSQL	0.10	0.00	0.00	1.00	1.00	1.00
3	DrDoS_NTP	0.42	0.97	0.59	1.00	1.00	1.00
4	Portmap	0.00	0.00	0.00	0.22	1.00	0.35
5	Syn	0.75	0.00	0.00	1.00	0.89	0.94
6	NetBIOS	0.00	0.00	0.00	0.44	0.95	0.60
7	UDPLag	0.00	0.00	0.00	0.97	0.90	0.93
8	LDAP	0.00	0.00	0.00	0.78	0.96	0.86
9	DrDoS_DNS	0.00	0.00	0.00	0.98	0.97	0.98
10	WebDDoS	0.00	0.00	0.00	0.02	0.83	0.03
11	TFTP	0.83	0.68	0.75	1.00	0.93	0.96
12	DrDoS_SNMP	0.00	0.00	0.00	0.30	1.00	0.46

As shown in Table 7 and Fig. 8, the indicators of each category increased after balancing in CICIDS 2017, which proves that this model has a better effect on the detection effect.

To verify the performance of each part of the CVWG, this section analyses the contribution of different modules of the CVWG model to the overall performance through ablation experiments, and the final classifiers are all selected from GRU, specifically CVAE, WGAN, GP. The ablation experimental results on the two datasets are shown in Tables 8 and 9, Figs. 9 and 10.

Figure 7: Confusion matrix before and after balancing on CICDDoS2019

Table 7: Comparison of results before and after balancing on CICIDS2017

Label	Class	Before balancing			After balancing		
		Precision	Recall	F1-score	Precision	Recall	F1-score
0	BENIGN	0.79	0.96	0.87	0.99	0.95	0.97
1	DoS hulk	0.92	0.78	0.85	0.93	0.99	0.96
2	DoSgoldeneye	0.99	0.78	0.88	0.99	0.99	0.99
3	DoS slow loris	0.60	0.81	0.69	0.95	0.99	0.97
4	DoSslowhttptest	0.85	0.46	0.60	0.98	0.99	0.98
5	Heartbleed	0.00	0.00	0.00	0.40	0.67	0.50

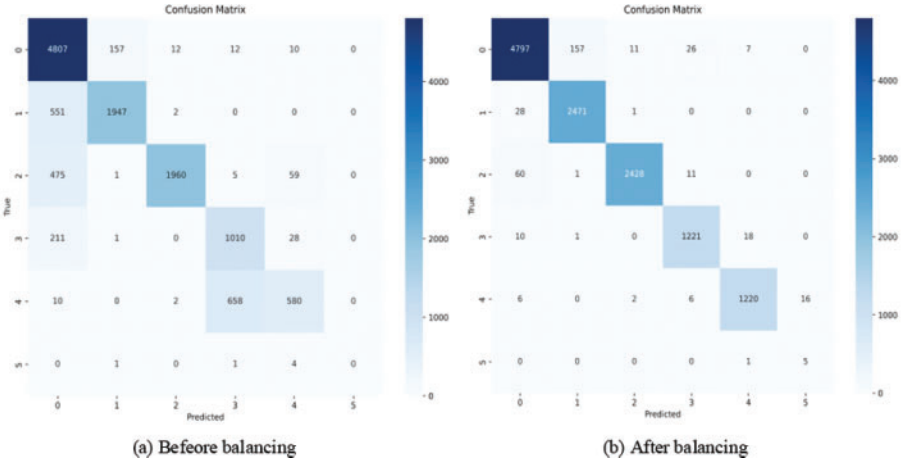


Figure 8: Confusion matrix before and after balancing on CICIDS2017

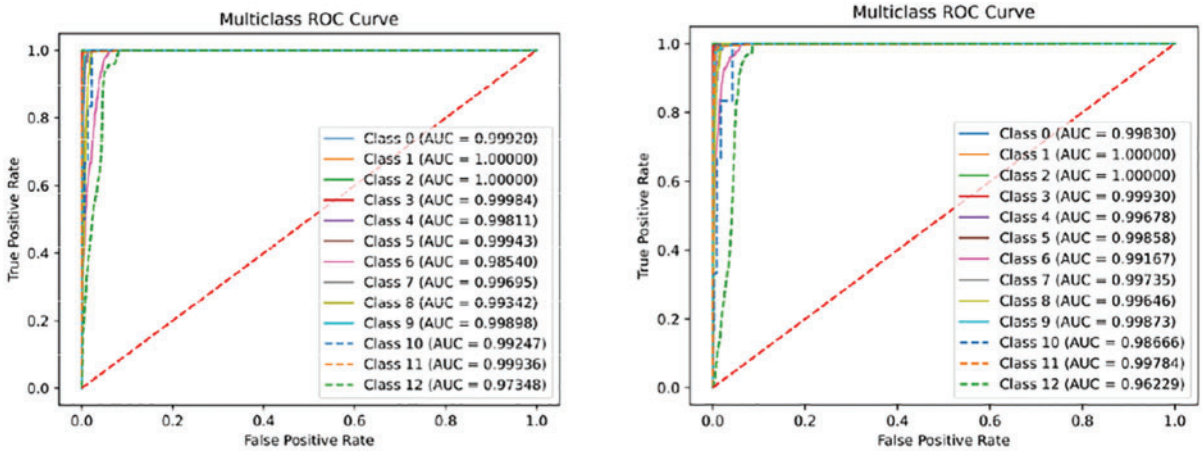
Table 8: Ablation experiment results on CICDDoS2019

Model	Recall	F1-score	Accuracy	Precision
CVAE-GRU	0.862	0.9068	0.862	0.974
WGAN-GRU	0.872	0.913	0.872	0.974
WGAN-GP-GRU	0.933	0.952	0.933	0.952
CVWGG	0.960	0.988	0.960	0.971

Table 9: Ablation experiment results on CICIDS2017

Model	Recall	F1-score	Accuracy	Precision
CVAE-GRU	0.884	0.880	0.884	0.892
WGAN-GRU	0.920	0.920	0.920	0.927
WGAN-GP-GRU	0.958	0.958	0.958	0.962
CVWGG	0.973	0.973	0.973	0.974

As can be seen, on CICDDoS2019, the classification accuracy was 0.862 when using only CVAE and 0.872 when using only WGAN. on CICIDS2017, the classification accuracy was 0.884 when using only CVAE and 0.920 when using only WGAN. After incorporating the CVWG model, all index values of the model have improved, indicating that the combination of the CVAE, the WGAN and the GP techniques significantly enhances the model's accuracy. In this paper, we combine the WGAN and GP techniques to address data imbalance, demonstrating that this approach is the most effective.



(a) CVAE-GRU (b) WGAN-GRU

Figure 9: (Continued)

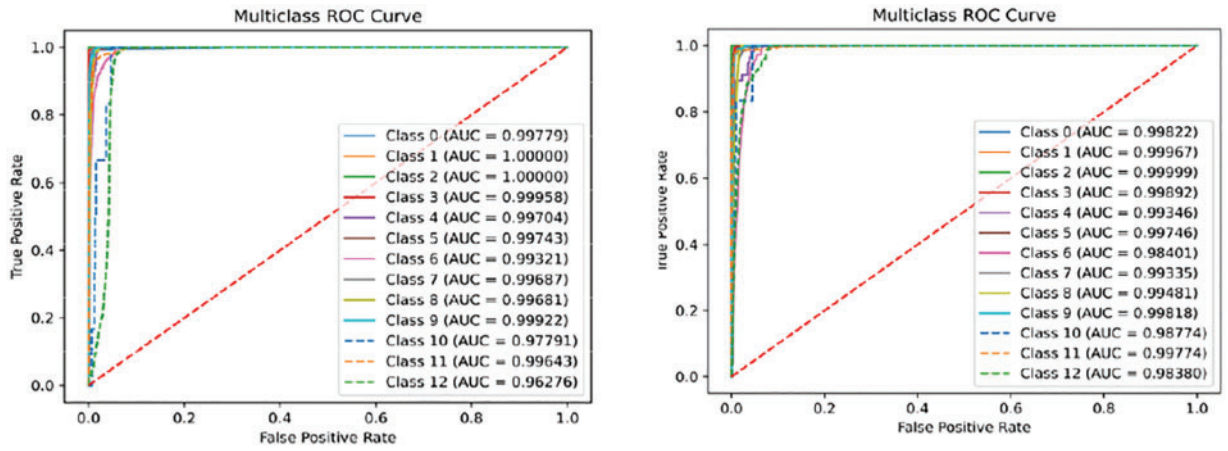


Figure 9: Ablation experiment result on CICDDoS2019

4.2.4 Comparison Experiments

The impact of the CVWG-based data enhancement algorithm CVWGG with B-GAN [19], LCVAEGAN [20], **WGAN-DIV-XGBoost [21]**, **WCGAN-GP-XGBoost [22]** and ADASYN [23] data enhancement methods on the model performance of DDoS attack detection is compared. The GRU model is trained using the balanced training set of the above data enhancement algorithms, and the detection effect of the model is shown in Tables 10 and 11, Figs. 11 and 12.

From the results of CICDDoS2019 experiments, the model CVWGG, enhanced using CVWG model data, achieves better results in terms of accuracy, precision, F1-score, and recall, with an accuracy rate that is eight percentage points higher than that of using BGAN, 14 percentage points higher than that of LCVAEGAN, 12 percentage points higher than that of WGAN-DIV-XGBoost,11.1 percentage points higher than that of WCGAN-GP-XGBoost and 7.9 percentage points higher than that of ADASYN.

It can be obtained from the experimental results of CICIDS2017 in Table 11 and Fig. 12 the model CVWGG, which was enhanced using the CVWG model data, achieved better results in terms of accuracy, precision, F1 score and recall, with an accuracy rate that was 4.6 percentage points higher than that of using BGAN, 1.4 percentage points higher than that of LCVAEGAN, 1.2 percentage points higher than that of WGAN-DIV-XGBoost, 1.1 percentage points higher than that of WCGAN-GP-XGBoost, and 7.1 percentage points higher than that of ADASYN. It can be concluded that this model can achieve good results in the differentiation of DDoS on the two datasets.

4.2.5 Model Accuracy and Loss Experiments

This model training was conducted on CICDDoS2019 and CICIDS2017, the loss and accuracy are shown in Figs. 13 and 14. It can be seen that the loss is less than 0.05, the accuracy can reach 0.96 on CICDDoS2019, and the loss is less than 0.1, the accuracy can reach 0.973 on CICIDS2017.

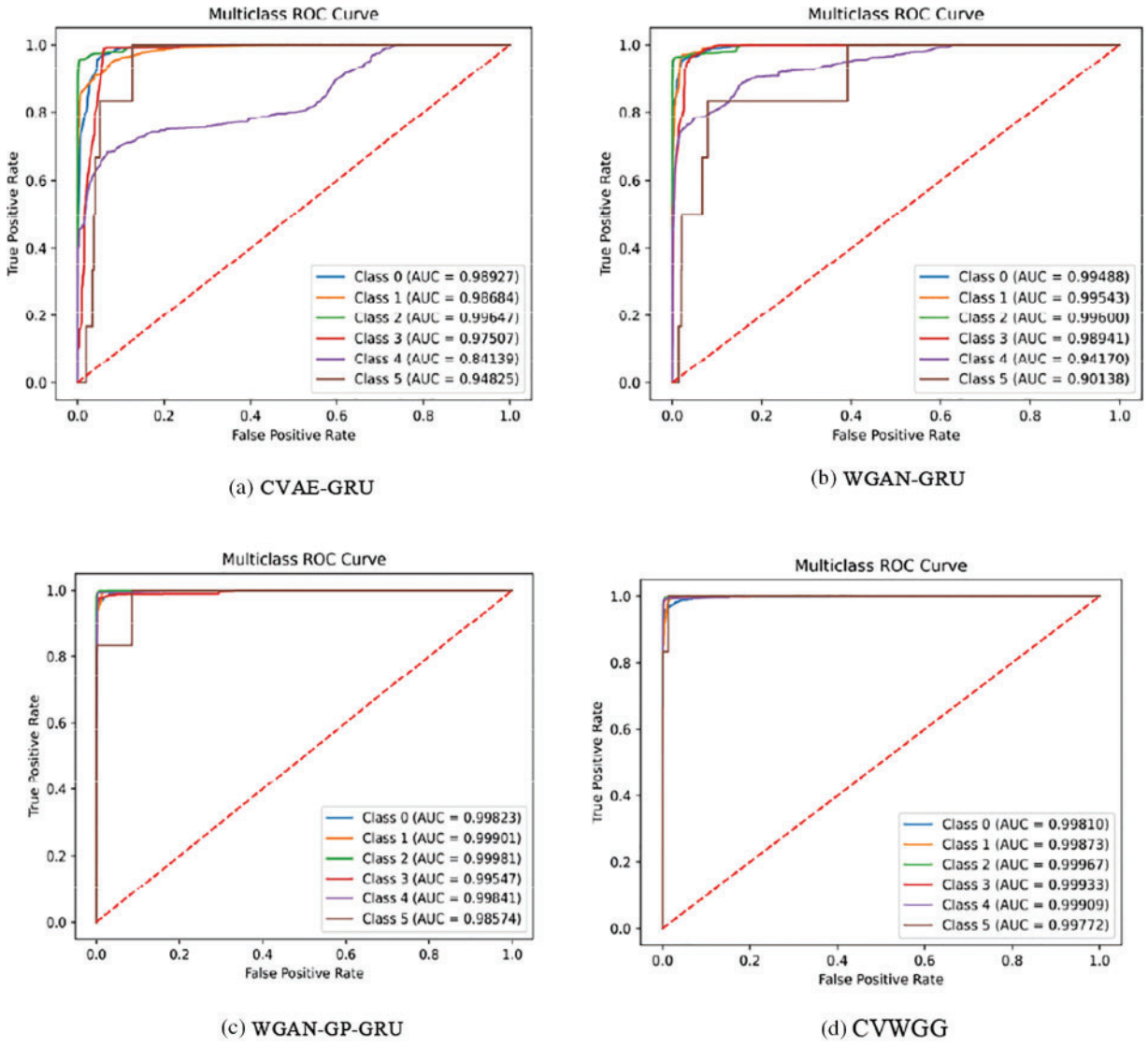


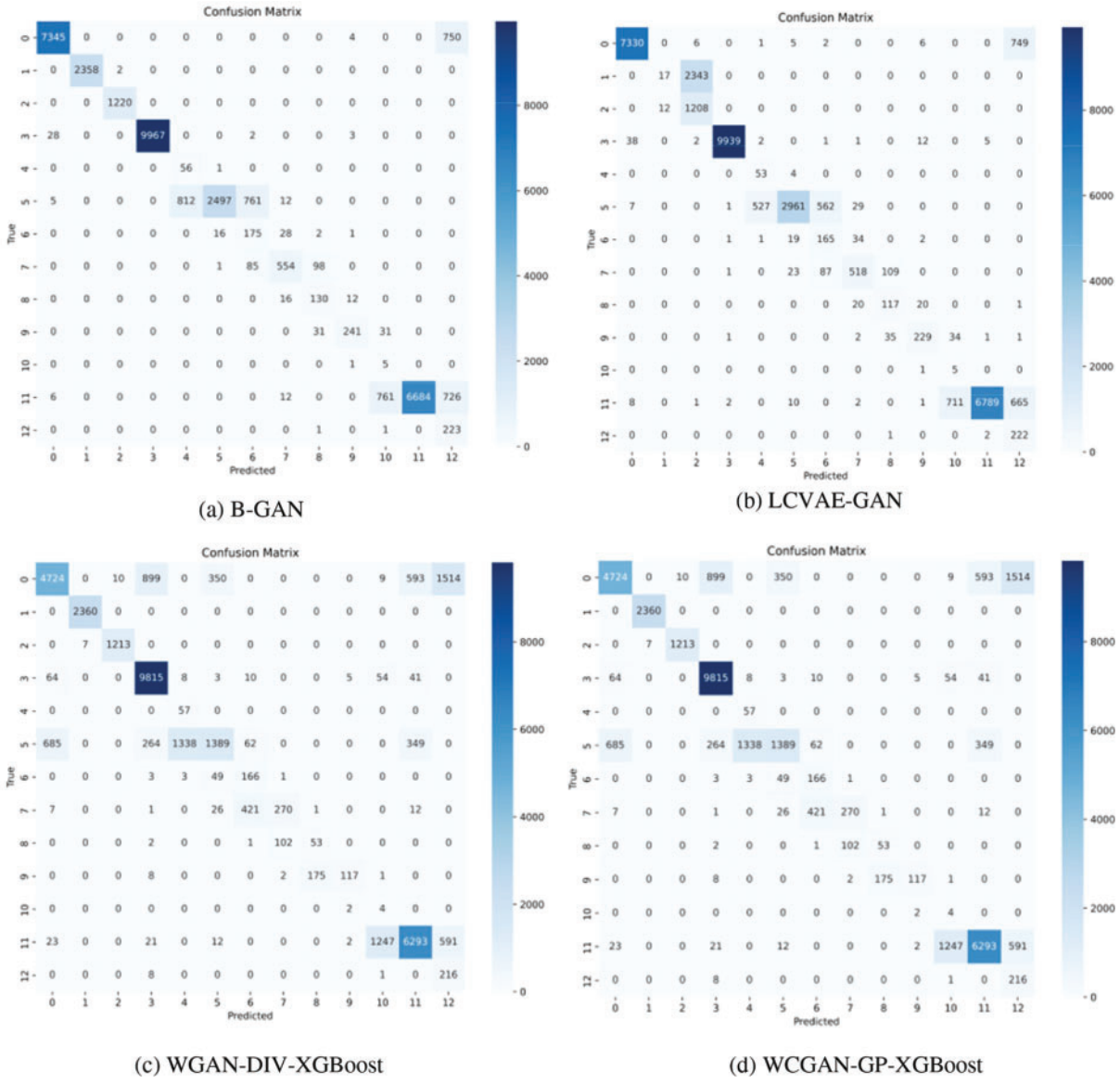
Figure 10: Ablation experiment result on CICIDS2017

Table 10: Comparison experiment results on CICDDoS2019

Model	Accuracy	Precision	Recall	F1-score
B-GAN	0.88	0.882	0.868	0.856
LCVAEGAN	0.82	0.85	0.78	0.821
WGAN-DIV-XGBoost	0.84	0.97	0.849	0.897
WCGAN-GP-XGBoost	0.849	0.849	0.849	0.849
ADASYN	0.881	0.980	0.881	0.918
CVWGG	0.960	0.988	0.960	0.971

Table 11: Comparison experiment results on CICIDS2017

Model	Accuracy	Precision	Recall	F1-score
B-GAN	0.927	0.935	0.928	0.935
LCVAEGAN	0.959	0.962	0.959	0.962
WGAN-DIV-XGBoost	0.961	0.962	0.961	0.961
WCGAN-GP-XGBoost	0.962	0.964	0.962	0.962
ADASYN	0.902	0.914	0.900	0.902
CVWGG	**0.973**	**0.973**	**0.973**	**0.974**



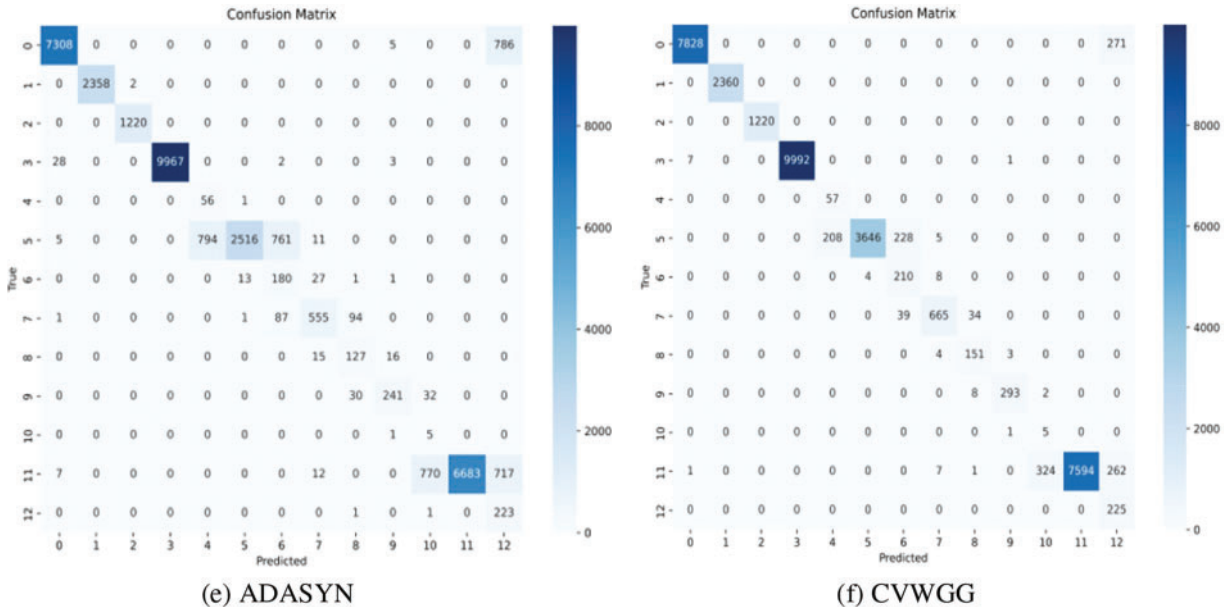
(a) B-GAN

(b) LCVAE-GAN

(c) WGAN-DIV-XGBoost

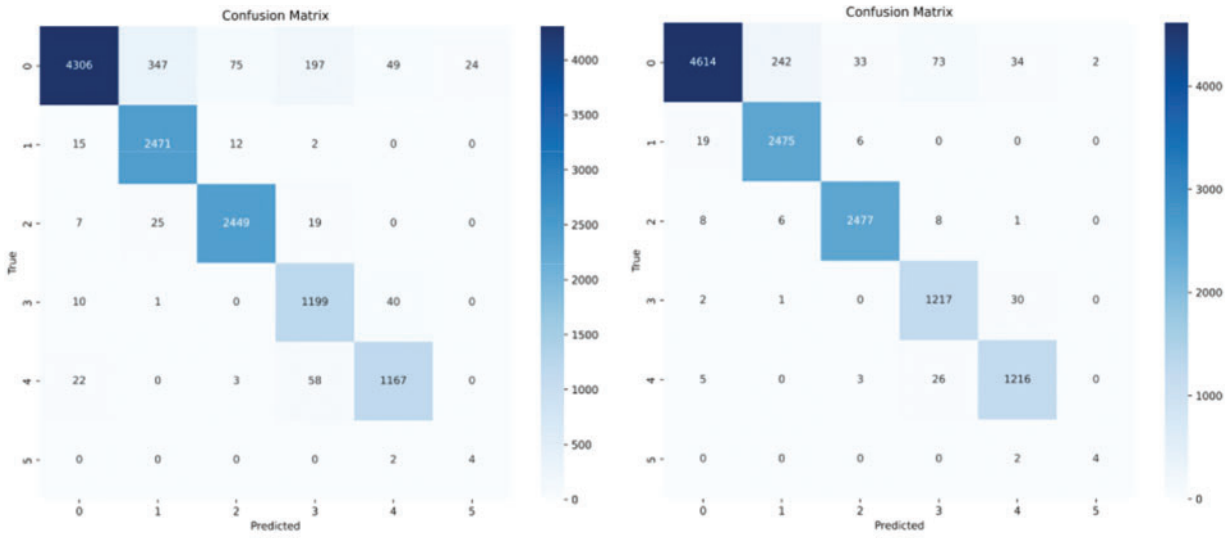
(d) WCGAN-GP-XGBoost

Figure 11: (Continued)



(e) ADASYN (f) CVWGG

Figure 11: Comparison experiment results on CICDDoS2019



(a) B-GAN (b) LCVAEGAN

Figure 12: (Continued)

Figure 12: Confusion matrix for comparison experiments on CICIDS2017

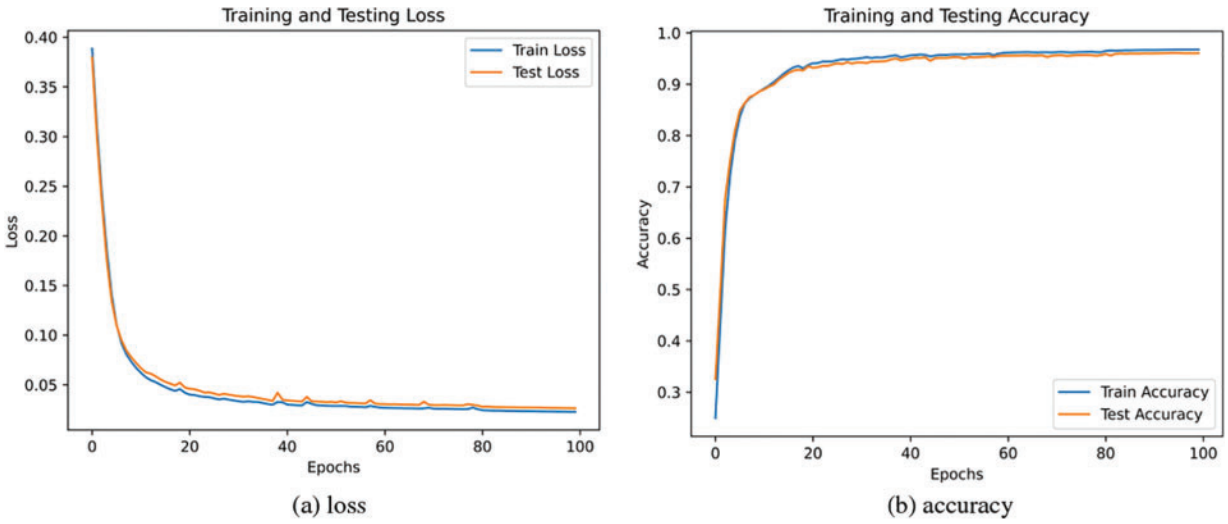


Figure 13: Loss and accuracy on CICDDoS2019

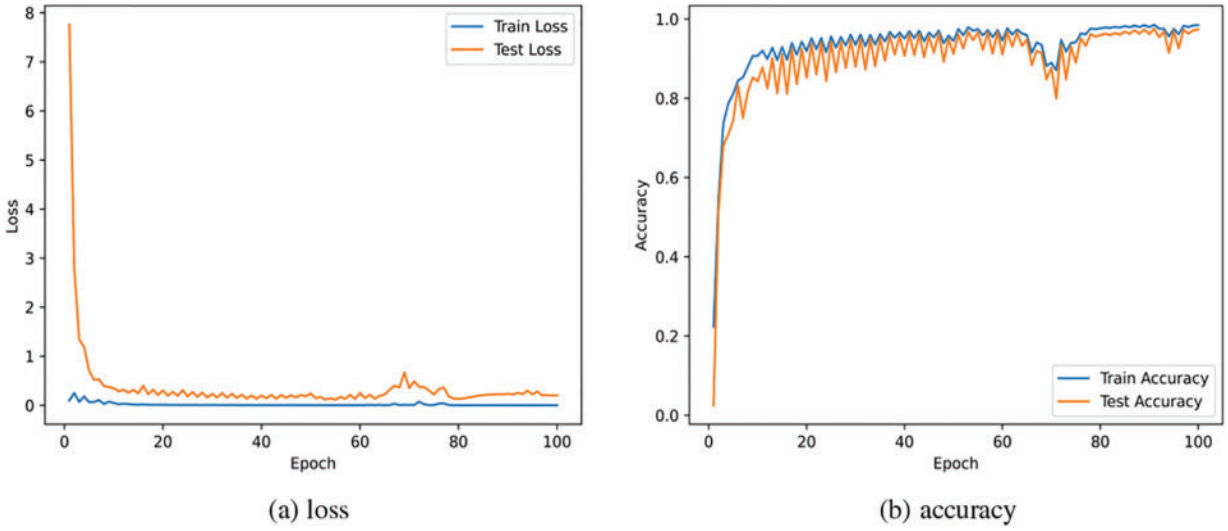


Figure 14: Loss and accuracy on CICIDS2017

5 Discussion

5.1 Complexity Discussion

The complexity of our model mainly depends on the CVWG method. Therefore, we discuss the complexity according to the specific network structures, parameter counts, and time complexities for the encoder, decoder, and discriminator in Fig. 2. As shown in Table 12, the total time complexity of the model is $O(7940)$, indicating a relatively low computational cost. This allows the model to demonstrate good scalability when handling larger datasets, making it easy to increase the sample size without significantly increasing the computational burden. Furthermore, the moderate number of parameters enhances the model's learning capacity while reducing the risk of overfitting, thereby strengthening its generalization ability in real-world scenarios.

Table 12: Computational complexity analysis

	Network architecture	Number of parameters	Time complexity
Encoder	Input Layer: $n_{11} = 66$ Hidden Layer 1: $m_{11} = 32$ Hidden Layer 2: $m_{12} = 32$	3168	$O(n_{11} * m_{11} + m_{11} * m_{12})$
Decoder	Input Layer: $n_{21} = 32$ Hidden Layer 1: $m_{21} = 32$ Hidden Layer 2: $m_{22} = 32$	4224	$O(n_{21} * m_{21} + m_{21} * m_{22})$
Discriminator	Input Layer: $n_{31} = 32$ Hidden Layer 1: $m_{31} = 16$ Hidden Layer 2: $m_{32} = 8$	664	$O(n_{31} * m_{31} + m_{31} * m_{32})$
Total		**8056**	***O*** **(7940)**

5.2 Decision-Making Discussion

To show the decision-making process of the model, this paper quantifies the contribution of each feature to the model's output by calculating SHAP (SHapley Additive exPlanations) values, thereby helping to identify which features are most important for prediction results. SHAP values provide a game-theoretic measure for each feature, illustrating its role and positive or negative influence in specific predictions. The Fig. 15 shows the top 20 features that most significantly impact the output for each category on the CICIDS2017 dataset. These features reveal how the model utilizes them to make classification decisions through the magnitude and direction of their SHAP values.

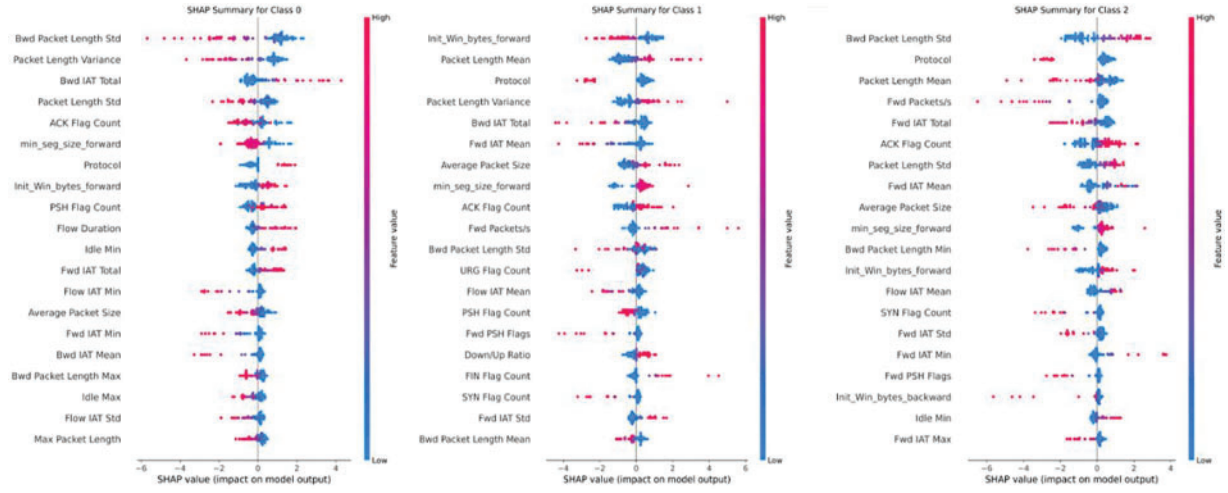


Figure 15: (Continued)

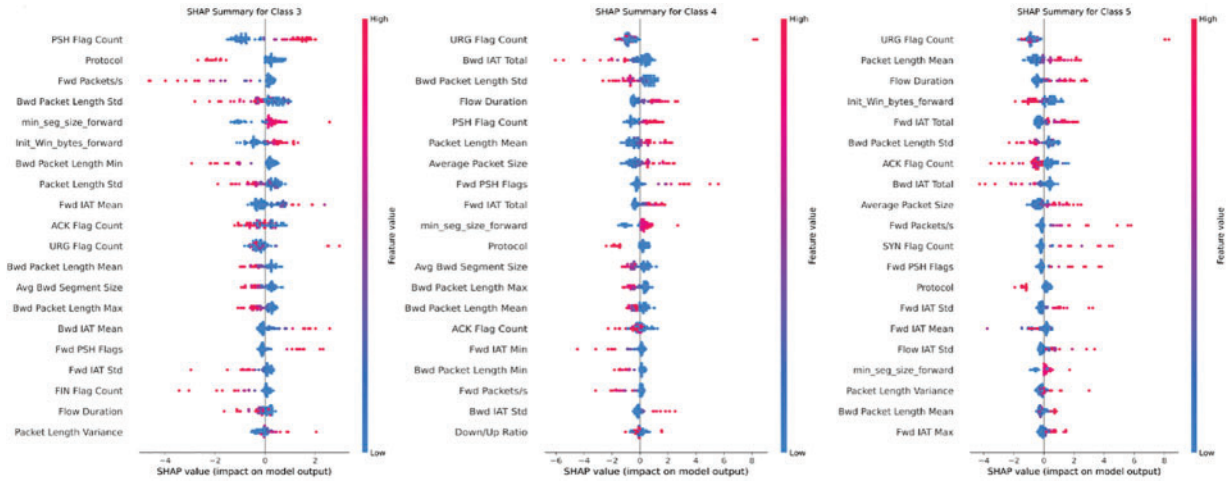


Figure 15: Feature importance map for each class on CICIDS2017

5.3 Scalability Discussion

To verify the applicability of the proposed model on large-scale datasets, we trained the model using 25%, 50%, and 75% of the original data, with results shown in the Table 13. As the data volume increased, the model may not have reached optimal performance due to computer memory limitations. Although there was a slight decline in performance with the increase in data volume, the decrease was not significant. This indicates that the model maintains relatively stable performance when handling large-scale data, demonstrating a certain degree of robustness.

Table 13: Scalability analysis results

	Accuracy	Precision	Recall	F1-score
25%	0.960	0.988	0.960	0.971
50%	0.955	0.989	0.955	0.970
75%	0.935	0.9575	0.9366	0.947

6 Conclusion

In this paper, we have proposed a data imbalance handling method, CVWGG; CVAE can handle data imbalance problems by introducing conditions (e.g., attack types) in training; It can learn and generate various attack samples that help to balance the dataset. Combined with WGAN, the generator can be better trained to produce more realistic traffic data, making that data more representative of real network traffic. It makes the generated data closer to the actual data and solves the problem of category imbalance that exists in the original dataset. The experimental results show that the differentiation of DDoS attack types can be achieved after the expansion of a few categories. Using PyCharm and Python 3.10, multiple metrics were evaluated on the CICDDoS2019 and CICIDS2017 datasets, including accuracy, precision, recall, F1- score, ROC curve and confusion matrix. The results show that the CVWGG method outperforms the comparison methods B-GAN, LCVAEGAN and ADASYN in DDoS detection accuracy, achieving accuracy rates of 96.0% and 97.3%, respectively. Therefore, the

proposed method can more effectively distinguish between different types of DDoS attacks, reduce the likelihood of false positives and false negatives, and assist network security personnel in implementing targeted defense measures, thereby preventing economic losses caused by network service disruptions, data loss and other related issues.

Despite the significant achievements of the CVWGG method in addressing data imbalance and DDoS attack detection, some limitations remain. Firstly, the model has a high training complexity, requiring considerable training time and computational resources, especially when dealing with large-scale datasets. Future research will focus on exploring more efficient generative model architectures to enhance the quality of sample generation and training efficiency, thereby better addressing these challenges.

Acknowledgement: None.

Funding Statement: The financial support from the Fundamental Research Funds for Higher Education Institutions of Heilongjiang Province (Grant No. 145209126) and the Research and Innovation Platform Project (Grant No. 145309314) is acknowledged.

Author Contributions: The authors confirm their contribution to the paper as follows: Study conception and design: Haizhen Wang; Data collection: Na Jia, Yang He; Analysis and interpretation of results: Pan Tan; Draft manuscript preparation: Na Jia. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and codes that support the findings of this study are available from the corresponding authors upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in IoT: A survey," *J. Supercomput.*, vol. 76, no. 8, pp. 5320–5363, 2019. doi: 10.1007/s11227-019-02945-z.

[2] J. Bhayo, S. A. Shah, S. Hameed, A. Ahmed, J. Nasir and D. Draheim, "Towards a machine learning-based framework for DDoS attack detection in software-defined IoT (SD-IoT) networks," *Eng. Appl. Artif. Intell.*, vol. 123, 2023, Art. no. 106432. doi: 10.1016/j.engappai.2023.106432.

[3] W. Lu and Y. Liu, "A DDoS attack detection method based on information entropy and deep learning in SDN," presented at IEEE 4th Inf. Technol. Netw. Electr. Autom. Control Conf. (ITNEC), Chongqing, China, 2020, pp. 1084–1088A.

[4] G. Liu *et al.*, "A CGAN-based DDoS attack detection method in SDN," in *2021 Int. Wirel. Commun. Mobile Comput. (IWCMC)*, Harbin, China: IEEE Press, 2021, pp. 1030–1034.

[5] Y. Qing, X. Liu, and Y. Du, "Mitigating data imbalance to improve the generalizability in IoT DDoS detection tasks," *J. Supercomput.*, vol. 80, pp. 9935–9960, 2024. doi: 10.1007/s11227-023-05829-5.

[6] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença Jr., "Adversarial deep learning approach detection and defense against DDoS attacks in SDN environments," *Future Gener. Comput. Syst.*, vol. 125, no. 1, pp. 156–167, 2021. doi: 10.1016/j.future.2021.06.047.

[7] T. K. Boppana and P. Bagade, "GAN-AE: An unsupervised intrusion detection system for MQTT networks," *Eng. Appl. Artif. Intell.*, vol. 119, 2023, Art. no. 105805. doi: 10.1016/j.engappai.2022.105805.

[8] S. Balaji and S. S. Narayanan, "Dynamic distributed generative adversarial network for intrusion detection system over internet of things," *Wirel. Netw.*, vol. 29, pp. 1949–1967, 2023. doi: 10.1007/s11276-022-03182-8.
[9] V. Kumar and D. Sinha, "Synthetic attack data generation model applying generative adversarial network for intrusion detection," *Comput. Secur.*, vol. 125, 2022, Art. no 103054. doi: 10.1016/j.cose.2022.103054.
[10] S. Huang and K. Lei, "IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks," *Ad Hoc Netw.*, vol. 105, 2020. doi: 10.1016/j.adhoc.2020.102177.
[11] J. H. Lee and K. H. Park, "GAN-based imbalanced data intrusion detection system," *Pers. Ubiquitous Comput.*, vol. 25, pp. 121–128, 2021. doi: 10.1007/s00779-019-01332-y.
[12] G. Zhang, X. Wang, R. Li, Y. Song, J. He and J. Lai, "Network intrusion detection based on conditional Wasserstein generative adversarial network and cost-sensitive stacked autoencoder," *IEEE Access*, vol. 8, pp. 190431–190447, 2020. doi: 10.1109/ACCESS.2020.3031892.
[13] Z. Ma, J. Li, Y. Song, X. Wu, and C. Chen, "Network intrusion detection method based on FCWGAN and BiLSTM," *Comput. Intell. Neurosci.*, vol. 2022, 2022, Art. no. 6591140. doi: 10.1155/2022/6591140.
[14] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors*, vol. 19, 2019, Art. no. 2528. doi: 10.3390/s19112528.
[15] P. Chuang and P. Huang, "B-VAE: A new dataset balancing approach using batched variational AutoEncoders to enhance network intrusion detection," *J. Supercomput.*, vol. 79, pp. 13262–13286, 2023. doi: 10.1007/s11227-023-05171-w.
[16] T. Liu, Y. Fu, K. Wang, and X. Duan, "Network intrusion detection method based on VAE-CWGAN and fusion of statistical importance of feature," (in Chinese), *J. Commun.*, vol. 45, no. 2, pp. 54–67, Feb. 2024. doi: 10.11959/j.issn.1000-436x.2024013.
[17] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. of the IEEE 53rd Int. Carnahan Conf. Secur. Technol.*, Chennai, India, 2019. doi: 10.1109/CCST.2019.8888419.
[18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *4th Int. Conf. Inform. Syst. Secur. Privacy (ICISSP)*, Portugal, Jan. 2018. doi: 10.5220/0006639801080116.
[19] L. Yuan, S. Yu, Z. Yang, M. Duan, and K. Li, "A data balancing approach based on generative adversarial network," *Future Gener. Comput. Syst.*, vol. 141, pp. 768–776, 2023. doi: 10.1016/j.future.2022.12.024.
[20] G. Chen, "Research on smart home network intrusion detection algorithm based on data enhancement," M.S. thesis, Beijing Jiaotong Univ., China, 2023. doi: 10.26944/d.cnki.gbfju.2022.003568.
[21] D. Li, D. Kotani, and Y. Okabe, "Improving attack detection performance in NIDS using GAN," in *2020 IEEE 44th Ann. Comput., Soft., Appl. Conf. (COMPSAC)*, Madrid, Spain, 2020, pp. 817–825. doi: 10.1109/COMPSAC48688.2020.0-162.
[22] A. Srivastava, D. Sinha, and V. Kumar, "WCGAN-GP based synthetic attack data generation with GA based feature selection for IDS," *Comput. Secur.*, vol. 134, no. 4, 2023, Art. no. 103432. doi: 10.1016/j.cose.2023.103432.
[23] H. Chen, C. Jiang, H. Jin, C. Wu, and J. Lu, "An intrusion detection model incorporating improved stacked encoder and multilayer BiLSTM," (in Chinese), *Comput. Eng. Appl.*, pp. 1–12, 2024. doi: 10.3778/j.issn.1002-8331.2312-0017.