Annals of Telecommunications
https://doi.org/10.1007/s12243-021-00876-6

Traffic identification model based on generative adversarial deep convolutional network

Shi Dong[1,2,3] · Yuanjun Xia[1,2] · Tao Peng[1]

Received: 18 August 2020 / Accepted: 2 August 2021

Abstract
With the rapid development of network technology, the Internet has accelerated the generation of network traffic, which has made network security a top priority. In recent years, due to the limitations of deep packet inspection technology and port number-based network traffic identification technology, machine learning-based network traffic identification technology has gradually become the most concerned method in the field of traffic identification with its advantages. As the learning ability of deep learning in machine learning becomes more substantial and more able to adapt to highly complex tasks, deep learning has become more widely used in natural language processing, image identification, and computer vision. Therefore, more and more researchers are applying deep learning to network traffic identification and classification. To address the imbalance of current network traffic, we propose a traffic identification model based on generating adversarial deep convolutional networks (GADCN), which effectively fits and expands traffic images, maintains a balance between classes of the dataset, and enhances the dataset stability. We use the USTC-TFC2016 dataset as training and test samples, and experimental results show that the method based on GADCN has better performance than general deep learning models.

Keywords Traffic identification · Generating adversarial network · Convolutional neural network · Class balance

1 Introduction

The rapid development of the Internet has increased people's dependence on the Internet [1]. At the same time, it has also changed people's lifestyles and social habits. The type and amount of traffic data in cyberspace are also increasing, and more and more network security issues and network service quality issues have brought a lot of influence to users' use. As the basis of network security, network traffic identification technology plays an important role in maintaining information security and ensuring the proper operation of the network. On the one hand, the suspicious data packets can be identified by analyzing the network traffic. On the other hand, administrators can allocate network resources more efficiently and reasonably through network traffic identification technology to provide better network services. In recent years, the most commonly used network traffic identification technologies mainly include port number-based network traffic identification technology [2–4], deep packet inspection technology [5–7], and machine learning-based network identification technology [8–10].

Nowadays, the rapid development of the Internet has led to frequent updates of network applications. More and more new network applications no longer deliberately follow the port number registration rules. Most of the port numbers used are randomly selected, which brings significant challenges to identifying port numbers. Therefore, the reliability of network traffic identification methods based on port numbers is getting lower and lower, and most network traffic can no longer be accurately identified using this method.

Deep packet inspection technology is no longer limited to port numbers but detects load features in network traffic and distinguishes the type of traffic by determining the load feature field and known features in the traffic data. On

✉ Shi Dong
njbsok@163.com

✉ Tao Peng
pt@wtu.edu.cn

1 School of Computer and Artificial Intelligence, Wuhan Textile University, Wuhan, Hubei, China

2 School of Computer Science and Technology, Zhoukou Normal University, Zhoukou, Henan, China

3 State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications), Beijing, China

the one hand, although the identification accuracy of this method is high, simple, and efficient, there are still certain limitations. During the frequent updating and iterative process of network applications, the original feature fields are likely to change, and the feature fields need to be processed and updated in time. Hence, the maintenance cost of this method is relatively high. On the other hand, it is difficult to identify the encrypted traffic with this method because it will lose load features after it is encrypted. Most of the current applications will consider encrypted transmission for security. Therefore, with this method, it is also difficult to meet current network traffic identification requirements.

In the current research on network traffic identification, the most popular method for network traffic identification is based on machine learning. Users can perform large-scale training through computers to learn network traffic features and perform classification.

Although traditional machine learning can achieve good performance in network traffic recognition, it relies on researchers to extract features manually. The classification effect generally depends on the pros and cons of manual feature extraction, and it cannot handle large datasets. The traffic features are outdated quickly and need to be continuously updated, and the selection of manual features limits the robustness [11] and generalization [12] ability of traditional machine learning. Besides, traditional machine learning is a model with a shallow structure, limiting its representation and classification capabilities.

To address the problems mentioned above, related researchers began to devote themselves to deep learning research [13–15]. With the improvement of computer computing power, deep learning is gradually used to solve practical problems, such as face recognition, image classification, and intelligent speech recognition. Deep learning can automatically learn feature representations from complex data without manual intervention compared with traditional machine learning. It is ideal for identifying encrypted traffic and learning the nonlinear mapping relationship between the original input data and the related output. Due to the superiority of deep learning, more and more researchers apply deep learning to the field of network traffic recognition.

The current public and well-known datasets in the field of network security vary significantly in size between classes. When using deep learning for traffic identification and classification, the final results will also be different. Currently, there is relatively little research on solving the problem of dataset imbalance, and the imbalance of dataset is usually adjusted by sampling. There are two main sampling methods: under-sampling [16, 17] and over-sampling [18, 19]. The relative balance of the datasets is achieved by reducing the number of larger samples and increasing the number of smaller samples. However, under-sampling is prone to bias, and over-sampling can easily cause overfitting problems, so these sampling methods have certain limitations in dealing with the issue of unbalanced datasets.

Therefore, we adopted a new dataset reprocessing method. After preprocessing, the unbalanced dataset is processed again through the generative adversarial network (GAN) to generate a new balanced dataset. Make the classifier better learn the traffic features and improve the recognition accuracy. Representative GAN algorithms are as follows: original GAN [20, 21], CGAN [22], DCGAN [23], WGAN [24], etc. However, the original GAN has problems such as difficulty in training, the loss of the generator, and the discriminator cannot indicate the training process and the lack of diversity of generated samples. Although CGAN shows unique advantages in generating specific data, it does not solve the instability of GAN training. Although DCGAN can enrich the diversity of samples, it does not completely solve the problem of training instability. WGAN completely solves a series of problems of unstable training in GAN and basically solves the problem of mode collapse, ensuring the diversity of generated samples. In the literature [24], WGAN uses a multi-layer perceptron (MLP) [25] as a generator and generates high-quality images. The DCGAN uses a fully connected convolutional neural network, and the training time is significantly higher than that of WGAN that uses MLP as the generator. Therefore, this paper uses WGAN to reprocess the dataset.

Considering some of the above issues, we propose a new GADCN learning model, mainly divided into three parts: (1) Preprocessing stage. In this stage, we directly process the raw traffic. We did not extract the traffic features but transformed them into a grayscale image similar to the MNIST handwritten digit set. (2) The optimization processing stage. In this stage, WGAN is used to optimize the grayscale image generated by the preprocessing. After adequate fitting and expansion, the size of each traffic class in the dataset is relatively balanced, making a new IDX file [26]; (3) Training and a classification stage. Through an improved convolutional neural network (ICNN), training and testing can better extract traffic features and improve recognition accuracy.

We chose the public, well-known, and relatively new USTC-TFC2016 [27] dataset to implement the experiment. This dataset has been extensively studied and contains a sufficient number of samples, and the size of the dataset is highly imbalanced.

To evaluate the performance of the GADCN model, we compared it with a series of well-known machine learning models: LeNet-5 [28], long short-term memory network (LSTM) [29], gated recurrent unit (GRU) [30], and improved convolutional neural network (ICNN). Besides, we also compare with the current commonly used methods for processing imbalanced datasets, such as unbalanced dataset, TrafficGAN [31], SMOTE [32], ROS [33], original GAN, and DCGAN. The experimental results prove that our proposed method has more advantages.

In summary, the main contributions of this study are as follows:

(1) In the preprocessing stage of the dataset, the session is used as the representation form of the traffic classification in this paper, which can retain the effective features of the raw traffic to the maximum.
(2) Use WGAN to reprocess the dataset to generate a new dataset that is relatively balanced among traffic classes. When the classification model is trained on the dataset, it can accelerate the learning process of the classification model and effectively improve the accuracy of malicious traffic recognition.
(3) An improved convolutional neural network is used to identify and classify network traffic. A convolutional layer is added after the second convolutional layer of the LeNet-5 model, which can extract traffic features more effectively and improving classification accuracy.
(4) We provided a comprehensive comparison of a series of methods for dealing with imbalanced datasets, including TrafficGAN, SMOTE, ROS, GAN, and DCGAN.

The remaining sections of this paper are organized as follows. In Section 2, we review related work and explains our work motivation. In Section 3, we introduce the GADCN model, the basic algorithm of the deep learning model we use. In Section 4, we present the dataset we use and the processing method of the dataset. In Section 5, we carried out experiments, compared with the current DL model, and compared the current commonly used methods to solve the imbalance of the dataset. Section 6 summarizes and looks forward to the paper.

2 Related work

In this section, we discuss some representative works using the USTC-TFC2016 dataset and dataset processing methods.

Currently, many workers use the USTC-TFC2016 dataset for simulation experiments of traffic identification and classification: Chen et al. [34] used a new SEEN scheme to achieve unknown traffic detection in network traffic classification. After the first type of traffic, the accuracy rate reached 96.03%. Li et al. [35] proposed a semi-supervised classification scheme based on a deep generative model to classify network traffic. The dataset in the experiment had 20% fewer labels than the original dataset, but the accuracy rate was more than 95%. Wang et al. [36] used a deep layered network for packet-level malware detection traffic, which could learn traffic features from original data packets. Although the accuracy rate reached 99.94%, the model is more complicated and mainly used CNN to extract spatial features and used two GRUs to extract temporal features. Besides, the training prediction time was relatively long, and it was not easy to perform real-time analysis. Ran et al. [37] used 3D CNN for malware traffic detection, and the F1-Score value of Virut traffic reached 93%. Chen et al. [38] used a Mutual Information (MI) and Softmax Variance (SV) Uncertainty based Annotation Error Correction (UAEC) framework, which increased the detection accuracy by 47.92% on average.

Many references dedicate to dataset preprocessing and imbalance research: Liu et al. [31] used a deep learning method of TrafficGAN, which mainly distinguished malware traffic from benign traffic by analyzing traffic sessions, and the F1-Score value reached 96.5%. Tang et al. [39] proposed a novel analysis and identification method based on the repeatability of malware traffic. The experimental dataset selected ten types of traffic from USTC-TFC2016, CTU, and VPN-non VPN to form a new dataset. To solve traffic imbalance problems through under-sampling and under-sampling + Synthetic Minority Over-sampling Technique (SMOTE). The experimental results showed that F1-Score values had reached 93% and 95%, respectively. Hasibi et al. [40] proposed a novel dataset expansion method, which mainly used a long short-term memory network (LSTM) to learn and generate data packets with a small number of classes. Then complete the sequence features by generating random values based on the distribution of a particular feature and using the kernel density estimation (KDE) to copy the probability density function of each class. Finally, put the features generated by these functions into an array to generate a new dataset. Vu et al. [41] used an AC-GAN deep network to generate synthesized data samples to balance secondary and primary classes. Vu et al. [42] studied various methods to solve the imbalance of datasets: under-sampling, random over-sampling (ROS), SMOTE and SMOTE-SVM, etc. Experimental analysis showed that these techniques for solving imbalanced datasets could help machine learning algorithms to achieve better performance. Tang et al. [43] proposed a multi-model coupling method to solve the problem of data imbalance in network traffic classification. Amina et al. [44] used an under-sampling method to solve real-time imbalanced traffic classification. Saber et al. [45] combined over-sampling and under-sampling and performed principal component analysis (PCA) to select the best feature subset before effective traffic classification. Chen et al. [46] developed a simple imbalanced data gravitation classification (S-IDGC) algorithm to deal with unbalanced datasets. Lee et al. [47] used Generative Adversarial Networks (GAN) to generate new data similar to existing data to solve the problem of dataset imbalance.

This paper, inspired by the above work, proposes a traffic recognition model based on generating adversarial deep convolutional networks, mainly using WGAN to process the dataset again and using ICNN to identify and classify benign traffic and malware traffic.

3 Proposed method

This paper focuses on classifying malware traffic by maintaining the balance between the classes of the dataset. In this section, we first introduce CNN and GAN and then present our method, which is essential for understanding our thinking.

3.1 Improved convolutional neural network

We first introduce ICNN, mainly based on the LeNet-5 model. LeNet-5 is a special CNN, used primarily for MNIST handwritten digit set (a kind of IDX file) recognition, mainly divided into five parts: the input layer, convolutional layer, down-sampling layer, fully connected layer, and output layer. The IDX file format is also the currently commonly used dataset format, so our dataset will also use this format. The handwritten digit set consists of 28*28 grayscale images, and LeNet-5 has achieved good results in handwritten digit set recognition and can quickly train and predict. The dataset we finally used is also composed of 28*28 grayscale images, and choosing a simple CNN architecture can highlight the advantages of the dataset reprocessing method. Hence, we select LeNet-5 as the benchmark. In addition, to improve the model's ability to extract traffic features, we improved LeNet-5. Add a convolutional layer after the second convolutional layer, which we call ICNN, as shown in Fig. 1.

(1) Input layer (Input). The dataset reprocessed grayscale image with the composition of 28×28 is selected as input data.

(2) Convolution layer (C1). This layer is the first convolution layer. It extracts different feature information in the matrix by specifying the values of different windows and extracts various features of the data through different convolution kernels. The convolution window size is 5×5, with a total of 32 channels. In addition, the moving step is 1, the filling method is same, and the activation function uses *Relu*. The formula is defined as follows

$$A^{(i)} = g\left(A^{(i-1)} * W^{(i)} + b^{(i)}\right) \tag{1}$$

where $A^{(i)}$ represents the feature map of the $i-th$ layer, $W^{(i)}$ represents the weight vector of the $i-th$ layer, $*$ represents the convolution operation, $b^{(i)}$ represents the $i-th$ layer bias value, and $g(x)$ represents the activation function *Relu*.

(3) Pooling layer (S2). The pooling layer uses Max-Pooling. The pooling layer is also called the down-sampling layer, and its purpose is to reduce the feature map's dimensionality while maintaining the same feature scale. The filter size is 2×2, the moving step is 2, the filling method is same, pooling the features of the previous layer, and the extracted features are $A^{(i+1)}$。

$$A^{(i+1)} = pooling(A^{(i)}) \tag{2}$$

Among them, $pooling(x)$ represents the pooling rules of the pooling layer.

(4) Convolution layer (C3). The convolution window size is 5×5, with a total of 64 channels. In addition, the moving step is 1, the filling method is same, and the activation function uses *Relu*. Other calculation methods are the same as C1.

(5) Convolution layer (C4). The convolution parameters and methods used in this layer are the same as C3.

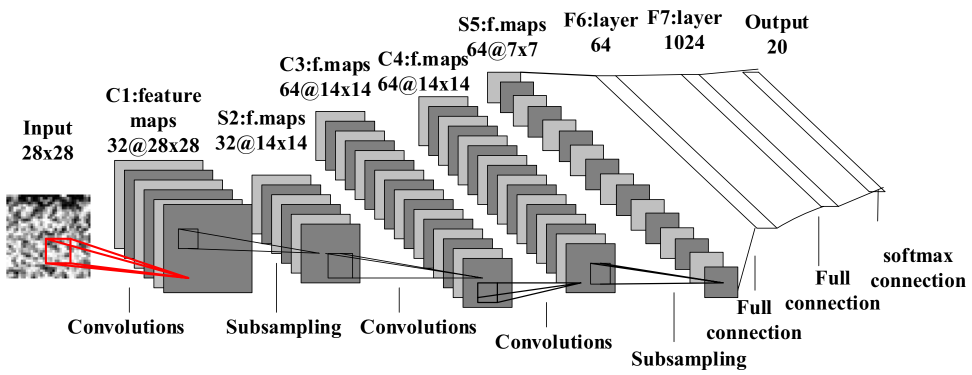


Fig. 1 Network traffic identification model of ICNN

(6) Pooling layer (S5). The pooling parameters and methods used in this layer are the same S2.
(7) Fully connected layers (F6 and F7). Each neuron in the fully connected layer is fully connected to all the neurons in the previous layer. The two-dimensional input features are converted into one-dimensional vectors for final classification.
(8) Output layer (Output). The output value of the last fully connected layer is passed to the output layer and finally classified using *softmax* logistic regression. Assume that the output is $y^{(i)}$, and the final result is Y; the expression is written as formula (3).

$$Y = softmax\left(y^{(i)}\right) \tag{3}$$

In the training process, the loss function adopts the cross-entropy function. Besides, L_2 regular term is used, in which λ is the regularization parameter. The cost function is shown in formula (4).

$$L(\omega, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i ln\hat{y}_i + \left(1 - y_i\right)ln\left(1 - \hat{y}_i\right)\right] + \frac{\lambda}{2m}\|\omega\|_2^2 \tag{4}$$

The training process uses a gradient descent algorithm to continuously update the weight value ω and the bias value b for backpropagation. Let the learning rate is ρ. The definition of the update iteration is as formulas (5) and (6).

$$\omega = \omega - \rho\frac{\partial L}{\partial \omega} \tag{5}$$

$$b = b - \rho\frac{\partial L}{\partial b} \tag{6}$$

3.2 Generative adversarial network

Generative adversarial network (GAN) mainly composed of two networks: generator network (G) and discriminator network (D). GAN's main idea is to let the G and the D continue to play games, and the data distribution is finally learned. The function of G is to receive a random noise z and generate images. The function of D is to judge whether the input data X is real data or generated by G, the formula shown in (7).

$$\underset{G}{min}\ \underset{D}{max}\ V(D, G) = E_{X\sim P_{data}(X)}[\log D(X)] + E_{Z\sim P_z(Z)}\left[\log(1 - D(G(Z)))\right] \tag{7}$$

Among them, $P_{data}(X)$ represents the distribution of real data. In the training process of D, if you want to make $V(D, G)$ as large as possible, you need to make $D(X)$ as large as possible and $D(G(Z))$ as small as possible. In the training process of G, if you want to make $V(D, G)$ as small as possible, $D(G(Z))$ needs to be as large as possible. When $P_{data}(X) = P_z(Z)$, the D output is 0.5. At this time, the D cannot judge the authenticity of the generated samples, and D and G also reach the corresponding Nash equilibrium. The best discriminant is as shown in formula (8).

$$D(X) = \frac{P_{data}(X)}{P_{data}(X) + P_z(Z)} \tag{8}$$

This paper uses Wasserstein GAN (also known as WGAN) to improve the GAN to pursue stability. The structure of WGAN is shown in Fig. 2.

Where x is the real input data, z is the random noise of the input, G is the generating network, and D is the discriminating network.

WGAN completely solves a series of problems of unstable training in GAN and solves the problem of pattern collapse, ensuring the diversity of generated samples. WGAN modified four aspects of GAN:

(1) Remove the sigmoid function in the last line of the D.
(2) The *loss* function of the G and the D does not take a *log* function.
(3) Each time the D parameters are updated, their absolute values are truncated to no more than a fixed parameter c.
(4) Momentum-based optimization algorithms are not suitable for parameter updating, such as momentum and Adam algorithms. Recommended root mean square prop (RMSProp) and stochastic gradient descent algorithm (SGD).

The definition of Wasserstein distance is shown in formula (9).

$$W\left(P_r, P_g\right) = \underset{\gamma \sim \prod(P_r, P_g)}{inf} E_{(x,y)\sim\gamma}\left[\|x - y\|\right] \tag{9}$$

where $\prod(P_r, P_g)$ is the set of all possible joint distributions combined by P_r andP_g. It can also be said that each edge distribution in $\prod(P_r, P_g)$ is P_r and P_g,$\gamma \in \prod(P_r, P_g)$, and for each$\gamma$, you can sample $(x, y) \sim \gamma$ from it and get a real sample

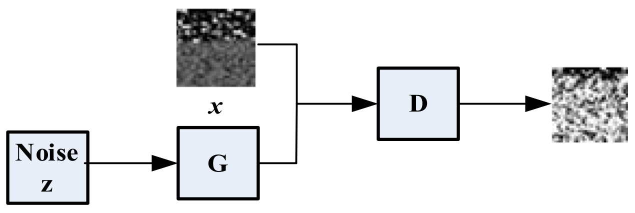


Fig. 2 Schematic diagram of WGAN structure

x and a generated sample y to calculate the distance of the pair of samples$\|x-y\|$, so we can calculate the expected distance of the samples in the joint distribution γ and finally take a lower bound on the expectations in all possible joint distributions because the lower bound ($\underset{\gamma\sim\prod(P_r,P_g)}{inf}$) cannot be solved directly, so the approximate value is obtained through mathematical calculations. The approximate Wasserstein distance is defined as for formula (10).

$$W\left(P_r,P_g\right)\approx\frac{1}{K}\underset{\omega\;:\;\|f\|_{l\le K}}{max}E_{x\sim P_r}\left[f_\omega(x)\right]-E_{x\sim P_g}\left[f_\omega(x)\right]\quad(10)$$

Among them, Lipschitz constant of f is K and f is a discriminator network, among which the parameter $\omega\epsilon[-c,c]$. Besides, the change of K will not affect the direction of gradient change.

3.3 Generate adversarial deep convolutional network

This paper combines GAN's deformed Wasserstein GAN and ICNN to identify network traffic. The identification process is mainly divided into two stages: (1) WGAN is used to refit and expand the defective features after preprocessing the dataset; (2) ICNN identifies the image processed by WGAN. The specific identification model is shown in Fig. 3.

The primary process of the first stage of the identification model is as follows: (1) The trained dataset is converted into a 28 × 28 grayscale image after preprocessing; (2) the random received noise is used to generate images through the G network; (3) D judges whether x is a real image or G generated; and (4) the D and G play multiple games and finally output the image with the best generation effect to make a similar MNIST dataset as the input of the second stage.

The primary process of the second stage is as follows: (1) The grayscale image interacts with different convolution kernels to express the features of the image; (2) dimension reduction of the data through the pooling layer to reduce overfitting and improve the fault tolerance of the model; and (3) the data is converted into one dimension through the fully connected layer. Finally, the traffic classification is performed using *softmax* to obtain the final classification result.

Besides, the related algorithms of related models are described, as shown in algorithms 1, 2, and 3.

Algorithm 1 WGAN

```
Require: $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
Input: $\{x^i\}_{i=1}^m$, a batch from real date; α, the learning rate. C, the clipping parameter.
Output: $\{z^i\}_{i=1}^m$ , a batch of noise-generated data.
 1: while $\theta$ has not converged do
 2:   for t=0, .... , n do
 3:     Sample $\{x^i\}_{i=1}^m \sim P_r$
 4:     Sample $\{z^i\}_{i=1}^m \sim p(z)$
 5:     $g_w \leftarrow \nabla_w\left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
 6:     $w \leftarrow w + \alpha\cdot \mathrm{RMSProp}(w, g_w)$
 7:     $w \leftarrow clip(w,-c,c)$
 8:   end for
 9:   Sample $\{z^i\}_{i=1}^m \sim p(z)$
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:   $\theta \leftarrow \theta - \alpha\cdot \mathrm{RMSProp}(\theta, g_\theta)$
12: end while
```

The default parameters used in this algorithm are as follows: (1) α is the learning rate, and the size is 0.00005; (2) the range of the value of the c discriminator network parameter w is 0.01; (3) m is a training batch with a size of 64; and (4) n is the number of iterations in the game between G and D, with a size of 12,000. There is also a D initialization parameter w and a G parameter θ.

When the G parameter θ has not yet converged, it samples $x^{(i)}$ from the real data and samples $z^{(i)}$ from the automatically generated data. Then, the RMSProp's optimization algorithm is updated with parameters, and the absolute value of w is truncated to no more than a fixed parameter c after the update. When G starts training, sampling $z^{(i)}$ from the data generated by noise before training. The G training is independent of $\frac{1}{m}\sum_{i=1}^m f_w\left(x^{(i)}\right)$, and the G parameter θ is updated at last.

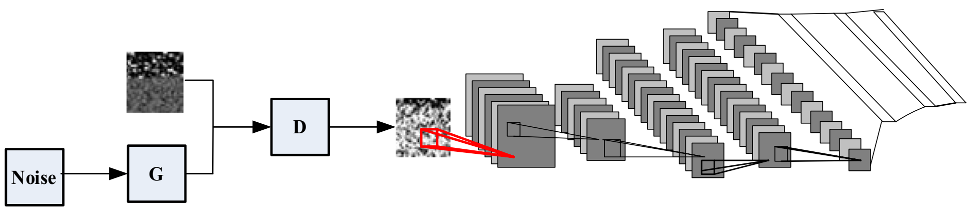


Fig. 3 GADCN-based traffic identification model

Algorithm 2 ICNN

Require: l, the depth of the network, $W^{(i)}$, $i \in \{1,2,\cdots,l\}$, the weight matrix of the model, $b^{(i)}$, $i \in \{1,2,\cdots,l\}$, the bias of the model.

Input: x, a batch of 28x28 picture data. α=0.01, the learning rate.

Output:y, a batch of traffic types and accuracy.

```
1: $h^{(0)} = x$
2: for k = 1,2,⋯,$l$ do
3:   $a^{(k)} = b^{(k)} + W^{(k)}h^{(k-1)}$
4:   $h^{(k)} = Relu\left(a^{(k)}\right)$
5: end for
6: $\hat{y} = h^{(k)}$
7: $\mathrm{J} = \mathrm{L}(\hat{y}, y) + \lambda\Omega(\theta)$
8: $g \leftarrow \nabla_{\hat{y}}J = \nabla_{\hat{y}}L(\hat{y}, y)$
9: for k= $l, l-1,\cdots,1$ do
10:     $g \leftarrow \nabla_{a^{(k)}}J = g \odot Relu'\left(a^{(k)}\right)$
11:     $\nabla_{b^{(k)}}J = g + \lambda\nabla_{b^{(k)}}\Omega(\theta)$
12:     $\nabla_{W^{(k)}}J = gh^{(k-1)\intercal} + \lambda\nabla_{W^{(k)}}\Omega(\theta)$
13:   $g \leftarrow \nabla_{h^{(k-1)}}J = W^{(k)\intercal}g$
14: end for
```

The default parameters used in this algorithm are as follows: (1) α is the learning rate, and the size is 0.001; (2) $W^{(i)}$ is the weight matrix of the model; and (3) $b^{(i)}$ is the bias value. Besides, x is the input of the model, and y is the target output.

The model first performs forward propagation, assuming that the model has l neurons, calculate the $k-th$ feature map $a^{(k)}$ of the control function that the $(k-1)-th$ layer maps to the $k-th$ layer, and use the *Relu* activation function to calculate $h^{(k)}$, the algorithm loops l times to calculate the final loss J, where $\Omega(\theta)$ represents the regular term.

Then proceed to the backpropagation phase of the model. Starting from the output layer and calculating backward to the first hidden layers, these gradients can be regarded as the guidance of adjusting each layer's output to reduce the error. Then gradients for each layer parameter can be obtained based on these gradients, such as $\nabla_{b^{(k)}}J$ and $\nabla_{W^{(k)}}J$, and the gradient on the weight and bias can be used as part of the random gradient update immediately. Finally, the hidden layer propagation gradient of the next lower layer is calculated.

Algorithm 3 GADCN

Require: w_0, initial critic parameters. θ_0, initial generator's parameters.n=20, number of traffic classes.

Input: *pcap* file

Output: a batch of traffic classes and accuracy.

```
1 : while input pcap file do
2 :   for t=0, .... , n do
3:     Table 1 WGAN related algorithms
4:   end for
5:   Generate dataset
6:   Table 2 ICNN related algorithms
7: end while
```

The algorithm is implemented by combining algorithm 1 WGAN and algorithm 2 CNN. Due to the significant difference in size between different traffics, the model is always biased towards traffic with a large amount of data when learning. To solve the problem of decreased accuracy caused by unbalanced traffic, we add optimization processing after preprocessing. The 20 different traffic classes are fully fitted and expanded through 20 cycles to generate a series of traffic images. The batch of traffic images is randomly divided into 3: 7 (3 is the test sample and 7 is the training sample) proportions to make a similar MNIST dataset as the next stage's input. This process maintains a balance between dataset classes, enhances the stability of the dataset, and is more conducive to the next step of identification and classification. Finally, after training, a *txt* file is generated, and various criteria of the classification result are recorded in the file.

4 Dataset

This paper uses the public dataset USTC-TFC2016 as the experimental dataset. This dataset was formed by Wei Wang et al. in 2016 and primarily consists of two parts: (1) 10 classes of malware traffic collected by CTU researchers from real network environments [48] and (2) using network traffic simulation 10 classes of benign traffic collected by equipment (IXIA BPS) [49]. The format of this dataset is *pcap*, which is the same as the traffic format captured by most packet capture software. The size is 3.71 GB, as shown in Table 1.

Table 1 Malware traffic and benign traffic size

Malware	Size (Mb)	Benign	Size (Mb)
Cridex	94.8	BitTorrent	7.33
Geodo	28.9	Facetime	2.4
Htbot	83.6	FTP	60.2
Miuref	16.3	Gmail	9.05
Neris	90.1	MySQL	22.3
Nsis-ay	281.2	Outlook	11.1
Shifu	57.9	Skype	4.22
Tinba	2.55	SMB	1206.8
Virut	109.3	Weibo	1620.4
Zeus	13.4	World of Warcraft	14.9

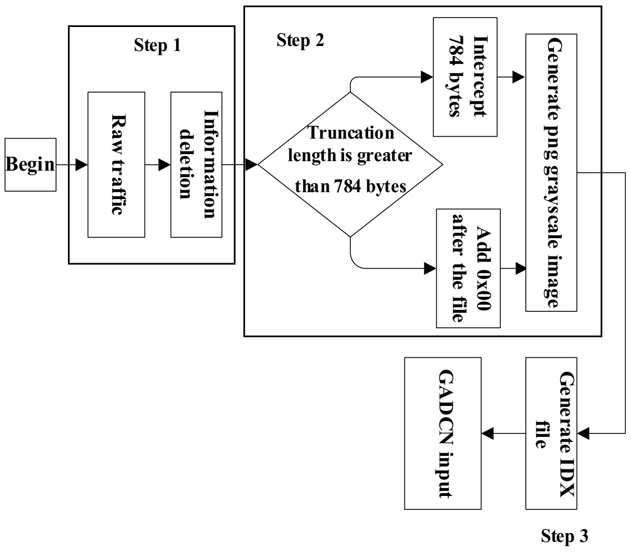


Fig. 4 Dataset preprocessing process

4.1 Dataset preprocessing

From the perspective of traffic granularity, the general network traffic segmentation method can be divided into five types: flow, session, host, TCP connection, and service, and it has been proved in the literature [27] that the session is more suitable for the traffic representation type of the USTC-TFC2016 dataset. A flow usually consists of a five-tuple, source port, destination port, source IP, destination IP, and transport protocol type. The session is also composed of a five-tuple, but the source and address can be interchanged, which is also called a two-way flow. Compared with the flow, the two-way flow contained in the session contains more interactive information, and more effective features can be learned from it. Therefore, we split the continuous raw *pcap* traffic into multiple discrete session units. We unify the aggregation of packet bytes in the entire session unit to 784 bytes instead of data packets in the session unit. Due to the different lengths of different session units, to better extract the effective features of the traffic, we select the first 784 bytes of the session units as the input of deep learning.

Data preprocessing is mainly divided into the following three steps, as shown in Fig. 4.

(1) Traffic split and deletion of Mac address and IP address.
   Read the raw *pcap* traffic data, split the continuous raw traffic into multiple discrete session units according to the traffic granularity. In addition, delete the Mac address of the data link layer and the IP address of the IP layer. Generally, the Mac address and IP address are deleted by discarding the relevant strings in the data packet because their existence will cause the model to overfit.
(2) Uniform size and generated grayscale images.
   The file after the above processing is uniformly intercepted to a length of 784 bytes. If the file length is less than 784 bytes, 0×00 should be added at the end of the file; if it is greater than 784 bytes, it is intercepted to 784 bytes. Then the file of uniform length is converted into grayscale images in binary form, and the grayscale image format is png, and the size is 28×28.
(3) Converting the generated grayscale images to the IDX files.
   Because IDX files are a standard file input format commonly used in the field of deep learning, grayscale images in png format should be converted into IDX files.

For example, as shown in Fig. 5, we randomly select one of the 28×28 grayscale images generated after step 2.

4.2 Dataset reprocessing

The IDX files of 20 classes of traffic generated by data preprocessing are fitted and expanded by WGAN to generate new

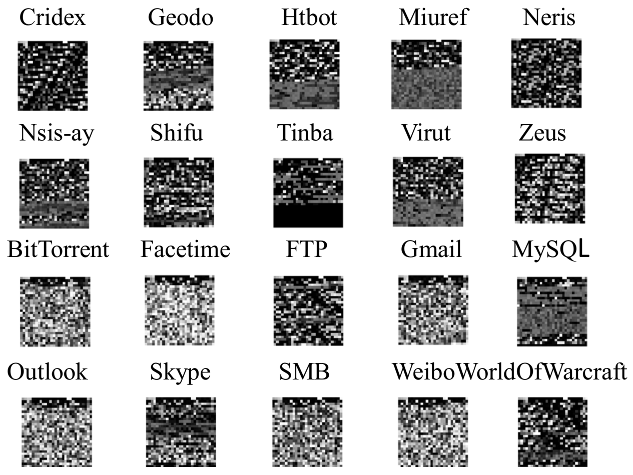


Fig. 5 Gray image after preprocessing

28*28 grayscale images. Each class of traffic generates 12,000 grayscale images and deletes the first 5000 inadequate fittings. The remaining 7000 images are randomly divided into training samples and test samples at a ratio of 7:3 to maintain the balance between the classes. Then these grayscale images are processed through data preprocessing step (3). After the process, these grayscale images are transformed to generate a new IDX file, which is finally used as the input of ICNN.

After fitting and expanding the various classes of 28 * 28 grayscale images, one of each class is randomly selected, as shown in Fig. 6.

From Fig. 5, we can see that the images of Cridex and Neris, Facetime and Weibo, Htbot and Miuref, Gmail and Weibo, Nsis-ay, and Shifu have little difference. In Fig. 6, the difference between the two can be clearly seen. When the difference between the grayscale images of each class is significant, the model can identify and classify more accurately.

5 Experiment

The experimental environment of this experiment is shown in Table 2.

5.1 Experimental evaluation criteria

Evaluation in machine learning is generally measured by Accuracy, Precision, Recall, Comprehensive Evaluation (F1-Score), etc. [50], such as formulas (11), (12), (13), and (14).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

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

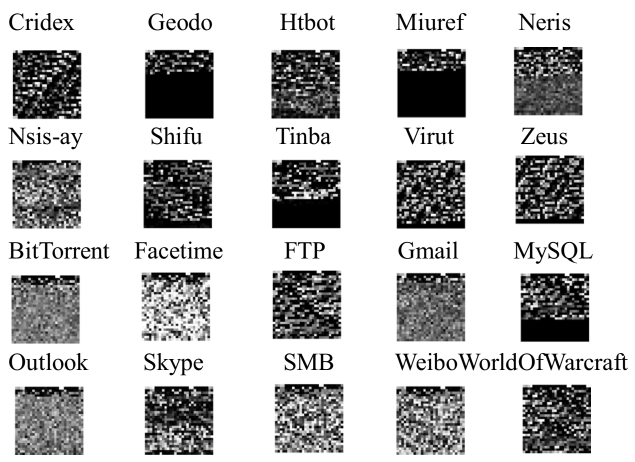


Fig. 6 Gray image after fitting and expansion

Table 2 Experimental environment and configuration

Experiment environment	Environment configuration
Computer brand	DELL
Operating system	Windows10
Processor	Intel(R)Core(TM) i5-6200U
CPU	2.40 GHz
RAM	8 GB
Language	Python 3.7
Learning framework	Tensorflow 1.13.1

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

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

where *TP* is the number of correctly labeled samples in the target sample, *FP* is the number of incorrectly labeled samples in the non-target sample, *TN* is the number of samples incorrectly labeled as other types in the target sample, and *FN* is the number of samples labeled in the non-target sample non-sample number.

5.2 Analysis of results

In this paper, our primary purpose is to prove the advantages of GADCN in dealing with imbalanced datasets. To verify the feasibility of the GADCN algorithm and evaluate its performance, we conducted a series of experiments. First, we compare with popular deep learning models. The experiments are carried out on the same device, and the number of training was 40,000 times and compared with the LeNet-5, GRU, LSTM, and ICNN methods. Among them, we analyze the training process of each method, as shown in Fig. 7.

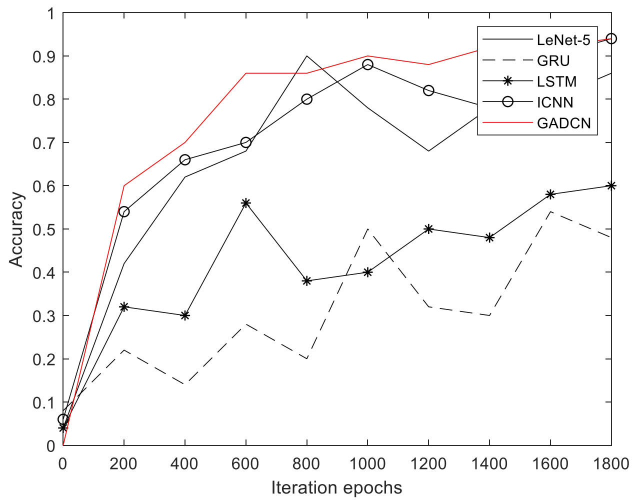
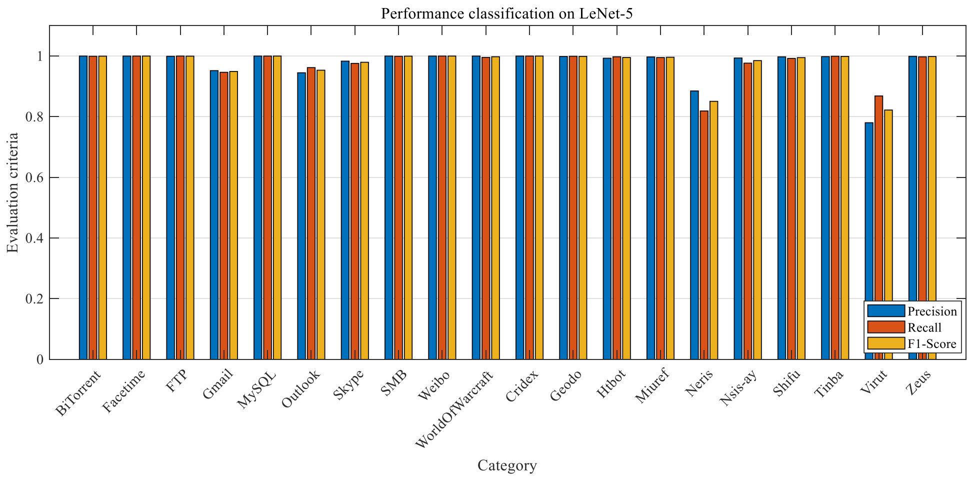
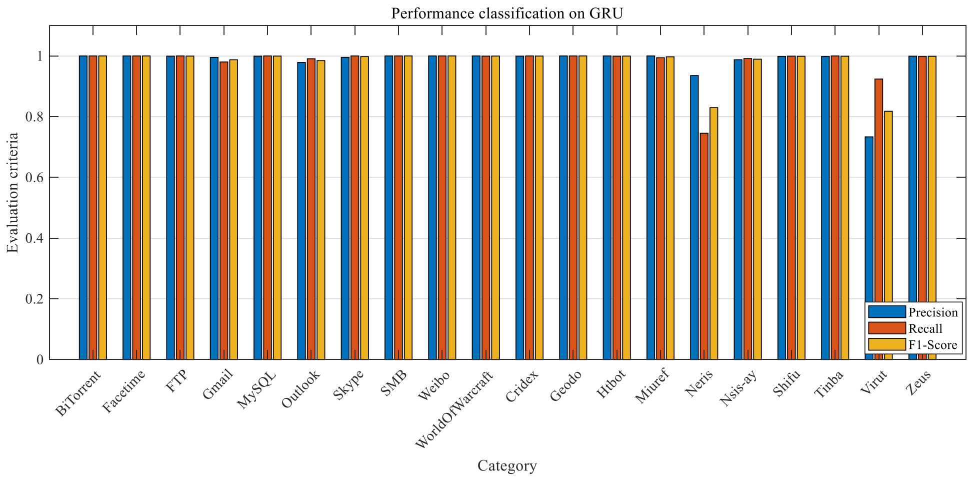


Fig. 7 Trend of accuracy rate during training

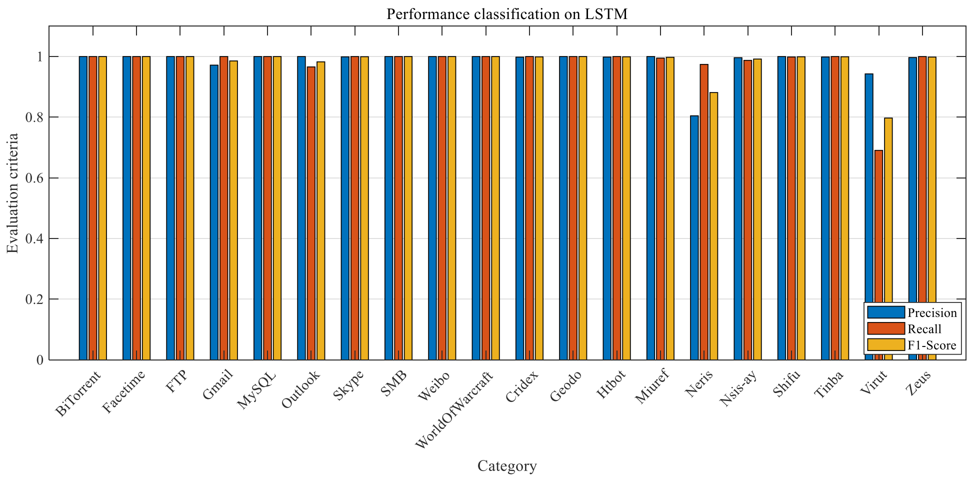
Fig. 8 Performance indicators of the deep learning model. **a** LeNet-5. **b** GRU. **c** LSTM. **d** ICNN. **e** GADCN



(a) LeNet-5



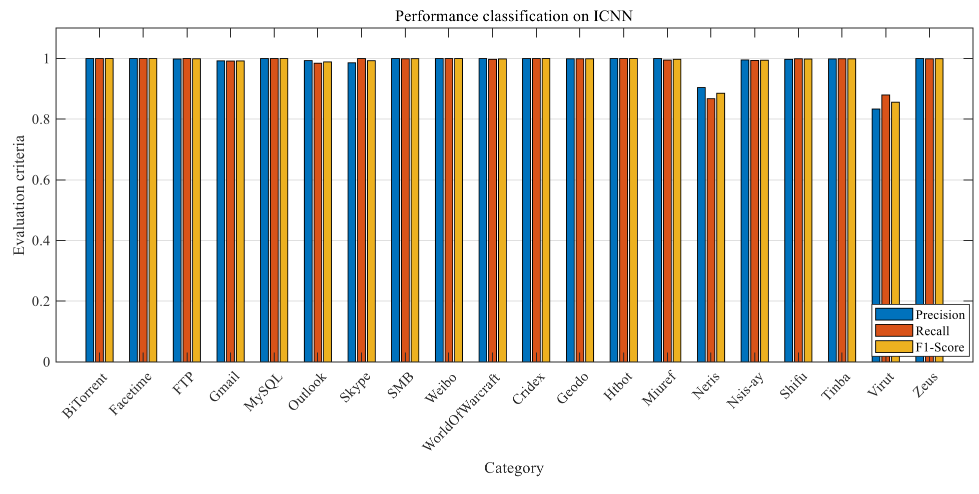
(b) GRU



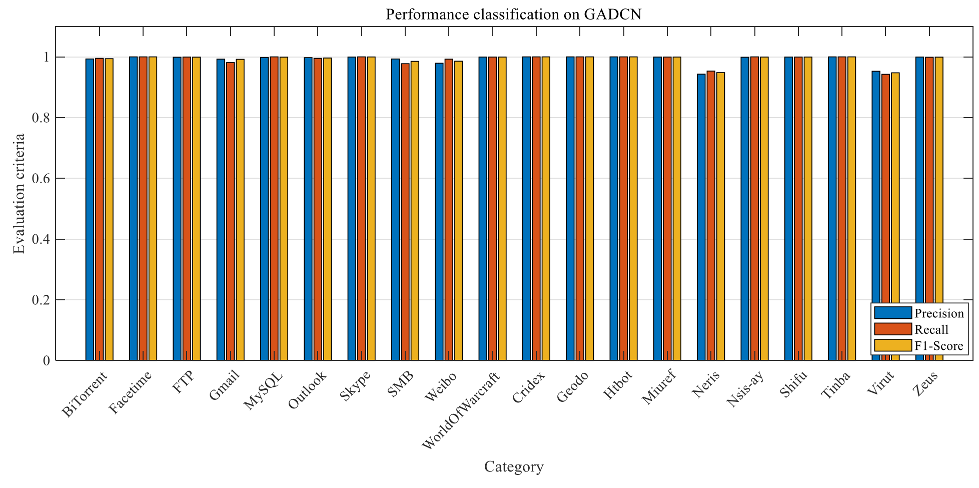
(c) LSTM

The above figure shows the accuracy rate change trend during the training process (select 0 to 1800 epochs). As can be seen from the figure, the GADCN model's training accuracy is higher and relatively stable, while the LeNet-5, LSTM, and GRU model training accuracy are relatively unstable. When the number of iteration epochs is 1800, the

Fig. 8 (continued)



(d)ICNN



(e)GADCN

training accuracy of LeNet-5, ICNN, and GADCN models exceeds 85%, while GRU and LSTM are low. By analyzing the model training process, the GAGDN model has a faster convergence rate than LeNet-5, ICNN, GRU, and LSTM and improves recognition accuracy. It shows that the GADCN model in this paper can better deal with the task of traffic identification in complex networks to a certain extent.

To further illustrate the effectiveness of our method, we use the scoring criteria in Section 5.1 to evaluate the above methods. The histogram is shown in Fig. 8, the overall evaluation results are shown in Table 3, and the average classification performance statistics table is shown in Table 4.

Analyzing the experimental results, we can find that the GADCN method in this paper has a good effect on the classification of benign traffic and malware traffic. Compared with LeNet-5, LSTM, and GRU, the ICNN proposed in this paper further improves the accuracy rate, and GADCN has the highest accuracy rate. Besides, GADCN has a better effect on malware traffic identification. Compared with the optimal model ICNN, the F1-Score of Neris and Virut traffics increased by about 6% and 9%, respectively. From the classification performance statistics in Table 4, it can be seen that the GADCN model has a significant improvement in accuracy, precision, recall, and comprehensive evaluation (F1-Score) compared to other models. It shows that we can use GADCN to reprocess the dataset to fit and expand the grayscale image fully, balance the dataset classes, avoid the classifier over learning a large number of classes, and enhance the stability of the dataset. It also proves that the GADCN model we proposed has certain advantages.

Besides, our proposed GADCN is also compared with dataset imbalance processing methods in related literature reviews, including unbalanced datasets (ICNN), TrafficGAN, SMOTE, GAN, ROS, and DCGAN. In addition to the experimental results of TrafficGAN from the literature [31], other methods all use ICNN as the classifier; moreover, the literature [31] does not mention the accuracy of TrafficGAN. The experimental results are shown in Table 5. Analyzing the

Table 3 Description of experimental results of each deep learning method

Model Traffic class	LeNet-5			GRU			LSTM			ICNN			GADCN		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
BitTorrent	1.0	0.9993	0.9996	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9931	0.9950	0.9940
Facetime	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FTP	0.9992	1.0	0.9996	0.9993	1.0	0.9996	1.0	1.0	1.0	0.9985	1.0	0.9992	0.9988	0.9994	0.9991
Gmail	0.9519	0.9464	0.9491	0.9947	0.9802	0.9873	0.9717	1.0	0.9856	0.9924	0.9918	0.9921	0.9925	0.9812	0.9919
MySQL	1.0	1.0	1.0	0.9993	1.0	0.9996	1.0	1.0	1.0	1.0	1.0	1.0	0.9981	1.0	0.9991
Outlook	0.9449	0.9619	0.9533	0.9782	0.9906	0.9843	1.0	0.9659	0.9826	0.9932	0.9846	0.9889	0.9975	0.9950	0.9962
Skype	0.9832	0.9758	0.9795	0.9950	1.0	0.9975	0.9991	1.0	0.9995	0.9860	1.0	0.9929	0.9994	1.0	0.9997
SMB	1.0	0.9992	0.9996	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9992	0.9996	0.9930	0.9775	0.9852
Weibo	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9790	0.9925	0.9857
World of Warcraft	1.0	0.9955	0.9977	1.0	0.9994	0.9997	1.0	1.0	1.0	1.0	0.9974	0.9987	0.9994	0.9994	0.9994
Cridex	1.0	1.0	1.0	0.9994	1.0	0.9997	0.9981	1.0	0.9990	1.0	1.0	1.0	1.0	1.0	1.0
Geodo	0.9985	0.9993	0.9989	1.0	1.0	1.0	1.0	1.0	1.0	0.9992	0.9992	0.9992	1.0	1.0	1.0
Htbot	0.9929	0.9976	0.9952	1.0	0.9992	0.9996	0.9984	1.0	0.9992	1.0	1.0	1.0	1.0	1.0	1.0
Miuref	0.9970	0.9950	0.9960	1.0	0.9940	0.9970	1.0	0.9950	0.9975	1.0	0.9950	0.9975	0.9994	0.9994	0.9994
Neris	0.8850	0.8189	0.8507	0.9351	0.7454	0.8295	0.8047	0.9741	0.8813	0.9044	0.8677	0.8856	0.9431	0.9532	0.9481
Nsis-ay	0.9935	0.9766	0.9850	0.9874	0.9910	0.9892	0.9963	0.9874	0.9918	0.9954	0.9937	0.9945	0.9987	1.0	0.9994
Shifu	0.9974	0.9923	0.9948	0.9981	0.9993	0.9987	1.0	0.9987	0.9993	0.9974	0.9993	0.9984	0.9994	0.9994	0.9994
Tinba	0.9981	0.9993	0.9987	0.9981	1.0	0.9990	0.9987	1.0	0.9993	0.9987	0.9993	0.9990	1.0	1.0	1.0
Virut	0.7802	0.8685	0.8220	0.7334	0.9240	0.8177	0.9429	0.6907	0.7973	0.8336	0.8799	0.8561	0.9526	0.9425	0.9475
Zeus	0.9991	0.9975	0.9983	0.9991	0.9983	0.9987	0.9967	1.0	0.9983	1.0	0.9991	0.9995	0.9994	0.9987	0.9991
Average	0.9756	0.9761	0.9759	0.9808	0.9811	0.9800	0.9854	0.9805	0.9815	0.9848	0.9853	0.9850	0.9919	0.9917	0.9922

Table 4 Average classification performance statistics

Model	Accuracy	Precision	Recall	F1-Score
LeNet-5	0.9736	0.9756	0.9761	0.9759
GRU	0.9779	0.9808	0.9811	0.9800
LSTM	0.9807	0.9854	0.9805	0.9815
ICNN	0.9839	0.9848	0.9853	0.9850
GADCN	0.9922	0.9919	0.9917	0.9922

Table 5 Performance evaluation of dataset processing methods

Dataset processing method	Accuracy	Precision	Recall	F1-Score
Unbalanced dataset	0.9839	0.9848	0.9853	0.9850
TrafficGAN	\	0.9640	0.9670	0.9650
SMOTE	0.9829	0.9815	0.9842	0.9829
ROS	0.9883	0.9876	0.9878	0.9877
GAN	0.9845	0.9845	0.9846	0.9846
DCGAN	0.9936	0.9936	0.9928	0.9932
GADCN	0.9922	0.9919	0.9917	0.9922

experimental results, we can find that the F1-Score value of TrafficGAN is the lowest. It is due to the generation of new variants, which adds a certain degree of difficulty to the classifier, so the result is slightly lower. In an unbalanced dataset, there is a large difference in the number of classes, which brings certain difficulties to the learning of the model, so the result is low. SMOTE and ROS are only on the original dataset to a certain extent, increase or decrease the number of raw samples, so the effect is general. Although GAN generates new samples based on the original dataset, due to the lack of diversity in the samples generated by the original GAN, the effect is not significant. Compared with GAN, DCGAN and GADCN have enriched various samples, and all criteria have been improved. DCGAN uses a fully connected CNN as G and D, and WGAN uses MLP as G. Since the image generation ability of a fully connected CNN is better than MLP, the result of DCGAN is slightly better than GADCN. However, both G and D in DCGAN use the fully connected CNN structure, and the training time is significantly higher than that of GADCN, which requires more computer resources. On the whole, GADCN has more advantages than DCGAN.

6 Conclusion

In this paper, a GADCN-based traffic identification model is proposed to identify benign traffic and malware traffic. We use the current deep learning model as a comparative experiment. The experimental results show that the model solves the problems of low grayscale image quality and imbalanced dataset classes and improves the recognition accuracy of malware traffic. Finally, we compare with the currently commonly used dataset imbalance processing methods. Experimental results show that our method achieves good performance. At present, the application of GAN is becoming more and more extensive, and this method still has a lot of room for improvement. The next step is to apply this method to encrypted traffic and identify and classify malware encrypted traffic more efficiently.

Funding This paper is supported by Project supported by Key Scientific and Technological Research Projects in Henan Province (Grand No. 192102210125), Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) (KLNST-2020–2-01), Hubei Provincial Department of Education Youth Project (Q201316), and Hubei Provincial Department of Education Research Program Key Project (D20191708). In addition, the authors also will thank the anonymous reviewers for their comments and suggestions.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Berberyan A (2021) Impact of internet dependence on the life meaning system of personality[C]//E3S Web of Conferences. EDP Sciences 258:07061
2. Schneider P (1996) Tcp/ip traffic classification based on port numbers[J]. Division Of Applied Sciences, Cambridge, MA, 2138(5):1–6
3. Yoon SH, Park JW, Park JS et al (2009) Internet application traffic classification using fixed IP-port[C]//Asia-Pacific Network Operations and Management Symposium. Springer, Berlin, Heidelberg, pp 21–30
4. Zander S (2006) Misclassification of game traffic based on port numbers: a case study using enemy territory[J]. Technical Report 060410D, CAIA. http://caia.swin.edu.au/reports/060410D/CAIA-TR-060410D.pdf
5. El-Maghraby R T, Abd Elazim N M, Bahaa-Eldin A M (2017) A survey on deep packet inspection[C]//2017 12th International Conference on Computer Engineering and Systems (ICCES), IEEE, pp 188-197
6. Sherry J, Lan C, Popa RA et al (2015) Blindbox: Deep packet inspection over encrypted traffic[C]//Proceedings of the. ACM Conference on Special Interest Group on Data Communication 2015:213–226
7. Bujlow T, Carela-Español V, Barlet-Ros P (2014) Extended Independent Comparison of Popular Deep Packet Inspection (DPI) Tools for Traffic Classification[J]. Universitat Politècnica de Catalunya. https://www.ac.upc.edu/app/research-reports/html/research_center_index-CBA-2014,en.html
8. Bakker J, Ng B, Seah W K, Pekar A (2019) Traffic classification with machine learning in a live network[C]//2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, pp 488-493
9. Thupae R, Isong B, Gasela N, Abu-Mahfouz A M (2018) Machine learning techniques for traffic identification and classifiacation in SDWSN: A survey[C]//IECON 2018-44th Annual

Conference of the IEEE Industrial Electronics Society, IEEE, pp 4645-4650

10. Shafiq M, Yu X, Bashir AK et al (2018) A machine learning approach for feature selection traffic classification using security analysis[J]. J Supercomput 74(10):4867–4892
11. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks[C]//2017 ieee symposium on security and privacy (sp). IEEE, 39–57
12. Doshi-Velez F, Kim B (2018) Considerations for evaluation and generalization in interpretable machine learning[M]//Explainable and interpretable models in computer vision and machine learning. Springer, Cham, pp 3–17
13. Aldweesh A, Derhab A, Emam A Z (2020)Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues[J]. Knowledge-Based Systems 189: 105-124
14. Pouyanfar S, Sadiq S, Yan Y et al (2018) A survey on deep learning: algorithms, techniques, and applications[J]. ACM Computing Surveys (CSUR) 51(5):1–36
15. Dong S, Wang P, Abbas K (2021) A survey on deep learning and its applications[J]. Computer Science Review 40: 100379
16. Peng M, Zhang Q, Xing X, et al (2019) Trainable undersampling for class-imbalance learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence, pp 4707-4714
17. Sun B, Chen H, Wang J et al (2018) Evolutionary undersampling based bagging ensemble method for imbalanced data classification[J]. Front Comp Sci 12(2):331–350
18. Chawla NV, Bowyer KW, Hall LO et al (2002) SMOTE: synthetic minority over-sampling technique[J]. Journal of artificial intelligence research 16:321–357
19. Gu X, Angelov PP, Soares EA (2020) A self-adaptive synthetic over-sampling technique for imbalanced classification[J]. Int J Intell Syst 35(6):923–943
20. Goodfellow I, Pouget-Abadie J, Mirza M, et al. (2014) Generative adversarial nets[J]. Advances in neural information processing systems 2014: 2672–2680
21. Ring M, Schlör D, Landes D, et al (2019) Flow-based network traffic generation using generative adversarial networks[J]. Computers & Security 82: 156-172
22. Mirza M, Osindero S (2014) Conditional generative adversarial nets[J]. arXiv preprint arXiv 1411:1784
23. Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks[J]. arXiv preprint arXiv 1511:06434
24. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein gan arXiv preprint arXiv 1701:07875
25. Heidari A A, Faris H, Mirjalili S et al (2020) Ant lion optimizer: theory, literature review, and application in multi-layer perceptron neural networks[J]. Nature-Inspired Optimizers 2020:23–46
26. IDX File Format Specifications, Behaviour and Example, (2016) http://www.fon.hum.uva.nl/praat/manual/IDX_file_format.html
27. Wang W, Zhu M, Zeng X, Ye X, Sheng Y (2017) Malware traffic classification using convolutional neural network for representation learning[C]//2017 International Conference on Information Networking (ICOIN), IEEE, pp 712-717
28. El-Sawy A, Hazem E B, Loey M (2016) CNN for handwritten arabic digits recognition based on LeNet-5[C]//International conference on advanced intelligent systems and informatics, Springer, Cham, pp 566-575
29. Geng Z, Chen GuoFei, Han Y, Gang Lu (2020) FangLi: Semantic relation extraction using sequential and tree-structured LSTM with attention. Inf Sci 509:183–192
30. Htet Myet Lynn (2019) Sung Bum Pan, Pankoo Kim: A deep bidirectional GRU network model for biometric electrocardiogram classification based on recurrent neural networks. IEEE Access 7:145395–145405
31. Liu Z, Li S, Zhang Y, Yun X, Cheng Z (2020) Efficient Malware Originated Traffic Classification by Using Generative Adversarial Networks[C]//2020 IEEE Symposium on Computers and Communications(ISCC), IEEE, pp 1-7
32. Fernández A, Garcia S, Herrera F et al (2018) SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary[J]. J Artific Intell Res 61:863–905
33. Zhang J, Chen L (2019) Clustering-based undersampling with random over sampling examples and support vector machine for imbalanced classification of breast cancer diagnosis[J]. Comput Assisted Surg 24(sup2):62–72
34. Chen Y, Li Z, Shi J, et al. (2020) Not Afraid of the Unseen: a Siamese Network based Scheme for Unknown Traffic Discovery[C]//2020 IEEE Symposium on Computers and Communications (ISCC), IEEE, pp1-735
35. Li T, Chen S, Yao Z, et al. (2018) Semi-supervised network traffic classification using deep generative models[C]//2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). IEEE, 1282–1288.
36. Wang B, Su Y, Zhang M, et al. (2020) A deep hierarchical network for packet-level malicious traffic detection[J]. IEEE Access
37. Ran J, Chen Y, Li S (2018) Three-dimensional convolutional neural network based traffic classification for wireless communications[C]//2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP). IEEE: 624–627
38. Chen W, Li H, Zeng Y, et al. (2019) Model uncertainty for annotation error correction in deep learning based intrusion detection system[C]//2019 IEEE International Conference on Smart Cloud (SmartCloud). IEEE: 137–142
39. Tang ZZ, Zeng X, Guo Z et al (2020) Malware traffic classification based on recurrence quantification analysis[J]. IJ Network Security 22(3):449–459
40. Hasibi, Ramin, Matin Shokri, and Mehdi Dehghan (2019) Augmentation scheme for dealing with imbalanced network traffic classification using deep learning. arXiv preprint arXiv:1901.00204
41. Vu L, Bui C T, Nguyen Q U (2017) A deep learning based method for handling imbalanced problem in network traffic classification[C]//Proceedings of the Eighth International Symposium on Information and Communication Technology, pp 333-339.
42. Vu L, Van Tra D, Nguyen Q U (2016) Learning from imbalanced data for encrypted traffic identification problem[C]//Proceedings of the Seventh Symposium on Information and Communication Technology, pp 147-152
43. Tang Z, Zeng X, Chen J (2020) Multi-model coupling method for imbalanced network traffic classification based on clustering[J]. Int J High Perform Comput Networking 16(1):26–35
44. Amina S I S M, Abdolkhalegh B, Khoa N K, Mohamed C (2018) Featuring Real-Time imbalanced network traffic classification[C]//2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, pp 840-846
45. Saber A, Fergani B, Abbas M (2018) Encrypted traffic classification: combining over-and under-sampling through a PCA-SVM[C]//2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS). IEEE: 1–5

46. Chen Z, Yan Q, Han H, et al. B (2018) Machine learning based mobile malware detection using highly imbalanced network traffic[J]. Information Sciences 433: 346–364
47. Lee J H, Park K H. GAN-based imbalanced data intrusion detection system[J]. Personal and Ubiquitous Computing, 2019: 1–8.
48. CTU University (2016) The Stratosphere IPS Project Dataset[DB/OL], https://stratosphereips.org/creategory/dataset.html (2016)
49. Ixia Corporation (2016) Ixia Breakpoint Overview and Specifications[CP/OL], https://www.ixiacom.com/products/breakpoints (2016)
50. Chicco D, Jurman G (2020) The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation[J]. BMC Genomics 21(1):1–13