mathematics

MDPI

Article

Filter-GAN: Imbalanced Malicious Traffic Classification Based on Generative Adversarial Networks with Filter

Xin Cao [1], Qin Luo [1,*] and Peng Wu [2]

[1] School of Computer Science, Southwest Petroleum University, Chengdu 610100, China
[2] School of Information and Engineering, Sichuan Tourism University, Chengdu 610100, China
* Correspondence: dorothy_lq@163.com

Abstract: In recent years, with the rapid development of Internet services in all walks of life, a large number of malicious acts such as network attacks, data leakage, and information theft have become major challenges for network security. Due to the difficulty of malicious traffic collection and labeling, the distribution of various samples in the existing dataset is seriously imbalanced, resulting in low accuracy of malicious traffic classification based on machine learning and deep learning, and poor model generalization ability. In this paper, a feature image representation method and Adversarial Generative Network with Filter (Filter-GAN) are proposed to solve these problems. First, the feature image representation method divides the original session traffic into three parts. The Markov matrix is extracted from each part to form a three-channel feature image. This method can transform the original session traffic format into a uniform-length matrix and fully characterize the network traffic. Then, Filter-GAN uses the feature images to generate few attack samples. Compared with general methods, Filter-GAN can generate more efficient samples. Experiments were conducted on public datasets. The results show that the feature image representation method can effectively characterize the original session traffic. When the number of samples is sufficient, the classification accuracy can reach 99%. Compared with unbalanced datasets, Filter-GAN has significantly improved the recognition accuracy of small-sample datasets, with a maximum improvement of 6%.

Keywords: malicious network traffic; GAN; imbalanced classification

MSC: 68M25

Citation: Cao, X.; Luo, Q.; Wu, P. Filter-GAN: Imbalanced Malicious Traffic Classification Based on Generative Adversarial Networks with Filter. *Mathematics* **2022**, *10*, 3482. https://doi.org/10.3390/math10193482

Academic Editors: Wen Zhang, Xiaofeng Xu, Jun Wu and Kaijian He

Received: 31 July 2022
Accepted: 19 September 2022
Published: 23 September 2022

1. Introduction

The rapid development of information technology not only brings great convenience to network users but also brings many security threats. Network traffic is an important carrier for network information exchange and transmission. Many network attacks and threats also exist in network traffic, so malicious traffic classification is one of the research focuses of cyberspace security. Due to the wide application of application-layer encryption technology, traditional port matching [1], deep packet inspection (DPI) [2–4], and other technologies cannot accurately identify malicious traffic. Therefore, researchers try to classify malicious traffic using various machine learning algorithms such as SVM, decision tree, naive Bayes, etc. However, machine learning-based methods involve two problems. First, machine learning models always rely on the knowledge and experience of professional security personnel to extract and select traffic features. Second, machine learning models have low accuracy and poor generalization ability in multi-classification tasks. Compared with machine learning, deep learning is a more popular method. As an end-to-end learning method, it can automatically extract data features without human intervention. Although the deep learning network enhances the expressive power, the performance of this classification algorithm decreases in the case of an imbalanced class distribution of the dataset, especially for few samples.

Therefore, overcoming imbalanced class distribution is of great significance for the classification of malicious traffic. To solve the problem of poor generalization of the classification model and the feature representation in the case of a class imbalance dataset, a representation method based on the Markov transition matrix and Adversarial Generative Networks with Filters (Filter-GAN) is proposed in this paper.

The original input to the feature representation method is the traffic session. The method embeds the relevant features into the feature image through the feature image extraction algorithm. The feature images are then used to train a Generative Adversarial Network (GAN) model and data filter based on machine learning algorithms. Finally, the GAN model generates enough new samples to filter out more effective samples through the data filter. These more effective samples serve as a supplement to the original dataset to address the problem of sample imbalance.

Compared with traditional malicious traffic characterization methods and data enhancement methods, this method has a better feature representation effect and stronger data enhancement effect, which can make the classification model have a better generalization and higher classification accuracy. There are two main contributions of this paper:

- A new traffic feature representation method is proposed. The method embeds the header feature and payload feature of each data packet into a feature image, which realizes a unified representation method for traffic files. The method avoids information loss and redundancy during preprocessing because the traffic session file is not sliced or populated. Compared with the traditional grayscale image method, the feature image generated by this method is more friendly to the model.
- A generative adversarial network model with a sample filter is designed. The filter is trained and tested with real samples to screen the generated samples so that the generated samples that pass the screening are closer to the distribution of real samples.

The rest of the paper is organized as follows. The second part introduces the related work of malicious traffic classification. The third part elaborates the whole model framework and training process, including data preprocessing, feature image extraction, model structure, and algorithm. Section 4 describes the experimental details and result evaluation. Section 5 concludes and proposes future work.

2. Related Work

2.1. Classification of Malicious Traffic Based on Deep Learning

Deep learning has been widely used [5–10] in the field of traffic detection and classification. Wang et al. [11] combined deep learning with traffic analysis for the first time, pointing out the similarities between images and TCP traffic. Jia et al. [12] further studied the application of deep learning in traffic analysis, unifying the length of the data packets, so that the length of each data packet reaches 784 bytes. The flow vector is converted into a 28 × 28 byte matrix and fed into a convolutional neural network for malicious traffic classification. Wang et al. [13] converts the data packets of the same five-tuple into fixed-length byte vectors in sequence according to time and then forms feature vectors through data compression. The feature vector is classified using the convolutional neural network with the Gabor function to achieve the purpose of intrusion detection. The two methods inevitably fill or intercept bytes, resulting in loss of information.

Jiarui Man et al. [14] directly used 196 statistical features of malicious traffic to form images, and then used a residual network for intrusion detection. Although the use of a residual network has good advantages in multi-classification, this method does not consider that there is no spatial position information between statistical features. Simply converting a one-dimensional vector into a two-dimensional matrix causes redundancy of information. Huiwen Bai et al. [15] convert network traffic into text, in which the words in the text consist of every byte of the payload. We use the n-gram semantic neural network model to generate continuous domain vectors, and then use the gated recurrent unit (GRU) to obtain feature vectors for final classification. However, with the use of more encryption protocols, the packet payload is randomly encrypted and no longer has specific semantics. This makes

semantic-based malicious traffic detection difficult. Marín [16] takes the network traffic byte stream directly as the input of the convolutional neural network and the recurrent neural network, and evaluates the feature representation effect at the packet and flow level. There are also many related works [17–20] that demonstrate the superiority of deep learning (DL) methods for malicious traffic analysis. We conclude that the DL method application technique consists of three steps: First, converting the data packet or pcap file into the standard input format of DL, then selecting a deep learning model according to the characteristics of the input data, and finally training the DL classifier to automatically extract and classify the characteristics of the traffic.

2.2. Common Methods for Dealing with Sample Imbalances

In the field of network traffic classification, the class imbalance problem can be expressed as an order of magnitude difference in the number of samples of traffic data in each application category, resulting in the classifier being overwhelmed by the majority class and ignoring the minority class. Misidentifying small categories is often costly. For example, in intrusion detection, the attack class is a small class relative to normal traffic, and misclassifying the attack class may cause network paralysis. There are generally three ways to deal with data imbalance: modify the objective cost function, change the sampling strategy, and generate artificial data as shown in Table 1. The method of modifying the loss function can alleviate the problem of quantity imbalance through different weighting processes according to different sample sizes. This approach gives higher scores to small classes and penalizes large classes for updating the parameters of the classification model.

Methods to address sample imbalance also include random undersampling (RUS) and random oversampling (ROS) strategies [21–23]. Undersampling increases the number of samples in the secondary class by discarding some sample data, thereby reducing the sample size in the main class. Oversampling increases the amount of data for samples with fewer classes by reusing data from classes with fewer classes. However, if the sample size of the minority class is very small, then discarding a large number of samples from the majority class will result in a loss of sample distribution information. While the oversampling method repeats samples leading to severe overfitting problems, which has been the main disadvantage of oversampling.

Table 1. A brief summary of methods for dealing with imbalanced datasets.

Author	Method	Reference
Wang	Random Over-sampling (ROS)	[21]
Vu	Random Under-sampling (RUS)	[22]
Cieslak	RUS + ROS	[23]
Chawla	Synthetic Minority Over-sampling Technique (SMOTE)	[24]
Chen	Adaptive Synthetic Sampling (ADASYN)	[25]
Nguyen	SVM-SMOTE	[26]
Last, F	Kmeans-SMOTE	[27]

As a method of oversampling, SMOTE (Adaptive Synthetic Sampling) [24] is also widely used in the literature to solve the class imbalance problem. However, it relies on interpolation for oversampling, resulting in poor representation of synthetic samples. SVM-SMOTE method [26] use support vector machine (SVM) classifiers to train the support vectors on the original training set. Based on the majority sample density of the nearest neighbors, interpolation or extrapolation techniques are used to combine each minority class support vector with its nearest neighbors to generate new samples. Chen et al. [25] proposed the ADASYN (Adaptive Synthetic Sampling) method for data generation based on the SMOTE method. However, the experimental results show that the distribution of the generated data is quite different from the real data. Last et al. [27] applied the Kmeans algorithm to the SMOTE method, taking inter- and intra-class relationships into account in the generated data.

The classic approach to generating artificial data is based on Generative Adversarial Networks (GAN), which are trained using a few samples as training data, rather than simply replicating it. Related research [28] shows that GANs can efficiently generate high-quality synthetic samples. Vu et al. [29] used the GAN model to generate data to supplement a small number of categories. The SVM, decision tree, and random forest models were trained on mixed data and achieved high classification accuracy. Wang et al. [30] generated multiple minority class data simultaneously through a conditional GAN model. The generated data are then supplemented by the minority class, which effectively improved the accuracy compared to the original dataset. Wang et al. [31] used a GAN model for data generation on an unbalanced encrypted traffic dataset in units of network flows. Although perceptrons have been widely used to solve class imbalance problems, instability problems such as vanishing gradients, mode collapse, etc. have always been shortcomings of multilayer perceptrons. Furthermore, they do not evaluate the data distribution of the generated samples versus the real samples. Therefore, in this paper, we propose to use filters trained on real samples to filter the generated samples, directly avoiding crashes and instability problems in the model. At the same time, the generated samples are evaluated using mathematical statistical methods.

3. The Proposed Method

This section discusses the framework for malicious traffic classification based on the Filter-GAN model, shown in Figure 1. It is divided into three stages: the data processing stage, data enhancement stage, and classification stage. In particular, the difference between the Filter-GAN model-based framework and the general method framework is the data processing stage and the data augmentation stage.

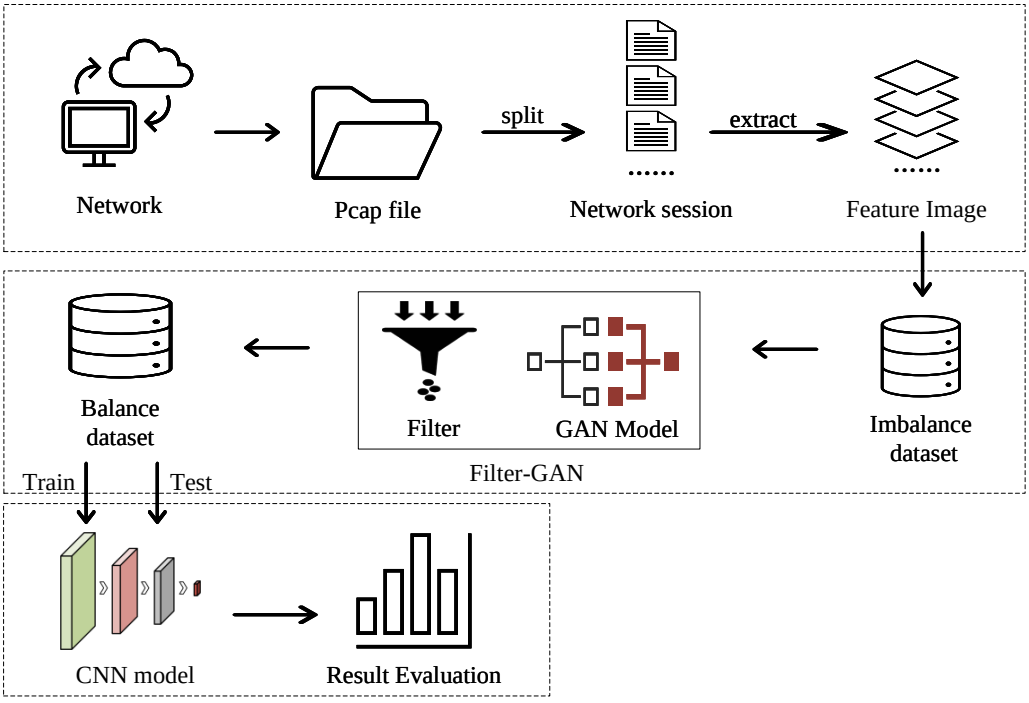


Figure 1. Architecture of Filter-GAN.

In the data processing stage, the original traffic file is divided into network session files according to the network quintuple. The network session file is used as the input of the feature image algorithm. The algorithm generates data matrices of uniform size without filling or intercepting valid fields. Then, these data matrices are converted into feature images to form feature image datasets.

The data augmentation stage is mainly used to generate data to supplement the minority class. Due to the imbalanced distribution of categories in the original traffic dataset,

the feature image set still suffers from imbalanced problem. Therefore, firstly, through the adversarial generative network (GAN), enough new feature images are generated and sent to the filter to screen more effective samples as a supplement to the original feature image set to enhance the diversity of the original feature images. It is worth mentioning that here, the generator generates feature images instead of the original raw traffic files.

In the classification stage, a convolutional neural network model is used as the classification model. The training set and the test set are the mixed feature image dataset and the original feature image dataset, respectively. Finally, the classification effect is evaluated.

3.1. Feature Image Extraction Algorithm

Traffic sessions have statistical, timing, and payload-related features. To compress all the features into one image as much as possible and improve the representation ability of the feature image, the feature image extraction algorithm converts the header field, source payload, and destination payload into three matrices, which are compressed into three channels of a picture to form a complete feature image. The specific process is shown in Figure 2.

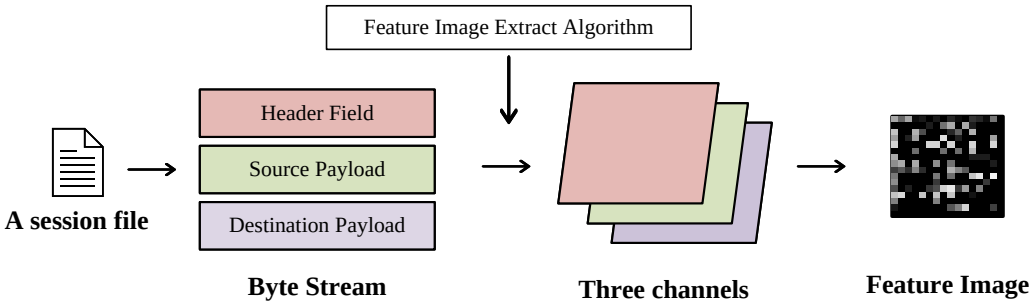


Figure 2. Feature image extraction.

There are data packets in two directions in a network session and the header fields of the information about the data packets themselves. The header field of each data packet needs to remove noise and useless information, such as data link layer information (mac address, frame type, etc.), and truncates the IP address. These data can usually be regarded as a bit stream, and each bit string has different state transition probabilities. This transmission process has the characteristics of a Markov chain. Therefore, the use of Markov models can effectively characterize the spatiotemporal features of session payloads.

We read the header field and payload of each data packet in the session file in binary mode, and then divide the payload into source payload and destination payload. Then , we take every four bits as a value. It can be expressed as:

$$bits = \{b_1, b_2, \ldots, b_{N\times 8}\}, b_i \in (0, 1) \tag{1}$$

$$v = \{s_1, s_2, \ldots, s_{\frac{N\times 8}{4}}\}, s_i \in (0, 1, 2, ..., 15) \tag{2}$$

where N is the byte length and the vector v is the encoded state vector. Consider v as a Markov chain, calculate the probability that two adjacent states s_{i+1} appear after s_i, denoted by $P(s_{i+1}|s_i)$:

$$P_{s_{i+1},s_i} = \frac{P(s_{i+1}, s_i)}{p(s_i)} = \frac{P(s_{i+1}, s_i)}{\sum_{j=0}^{i} P(s_i|s_j))} \tag{3}$$

where $s_i, s_{i+1} \in (0, 1, 2, \ldots, 15)$. All can form a Markov probability transition matrix:

$$M = \begin{bmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,15} \\ P_{1,0} & P_{1,1} & \cdots & P_{1,15} \\ \vdots & \vdots & \ddots & \vdots \\ P_{15,0} & P_{15,1} & \cdots & P_{15,15} \end{bmatrix} \tag{4}$$

Figure 3 is the process of converting binary data into a Markov matrix. Each value of the Markov matrix of malicious traffic is the transmission probability of a fixed-length bit string, not the actual value of the bit string. They represent the distribution characteristics of network session fields. After converting the payload and the header fields into Markov matrices, they are used as three channels of the feature image, respectively. Each value in the matrix is used as a pixel point of the feature image to form a feature image as in feature image extraction Algorithm 1.

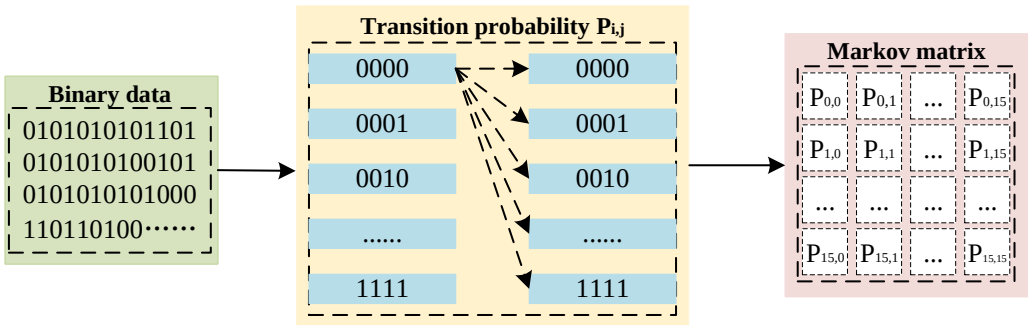


Figure 3. The process of converting binary data into Markov matrix.

Algorithm 1: Feature Image Extract

Input: Network traffic Session(Pcap)
Output: Feature Image
Data: SrcPayloadBits,DstPayloadBits,HeaderBits;
for *Packets* **do**
    Extract header , source payload and destination payload
    SrcPayloadBits = SrcPayloadBits + source payload
    DstPayloadBits = DstPayloadBits + destination payload
    HeaderBits = HeaderBits + header
end
M_1,M_2,M_3 ← Convert Bits to Markov matrix `// using the Markov algorithm`
Encoding M1, M2, M3 matrices as Feature Images;

3.2. Filter-Gan Model Based on Gan with Machine Learning Filter

The structure of Filter-GAN is shown in Figure 4, including generator, discriminator, and sample filter. The first step is GAN network training. Random noise is sent to the generator to generate enough samples, then the generated samples and real samples are sent to the discriminator for backward propagation of the loss function so that the entire model can generate a generated sample that is closer to the real sample image. In the second step, the filter consisting of a machine learning model uses real data for training and testing. When the best classification effect is achieved, it is used to screen the samples generated in the first step. In step 3, the samples generated by the GAN model are fed into the filter to screen more effective samples.

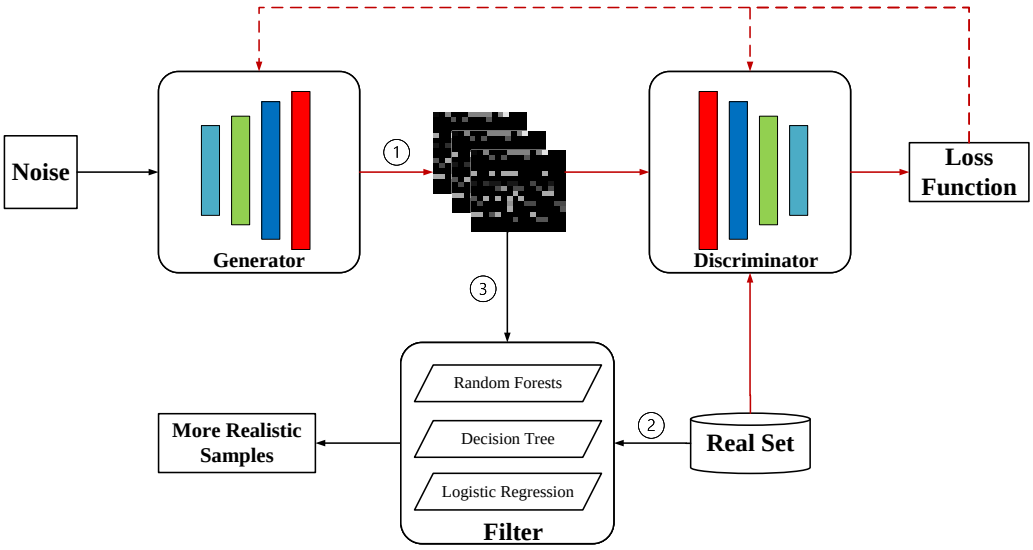


Figure 4. Architecture of Filter-GAN.

3.2.1. Gan Model and Loss Function

GAN [32] includes a generator (G) and a discriminator (D). The structure of the GAN network model constructed in this paper is shown in Tables 2 and 3. In the original imbalanced data, for the minority attack class, the real sample $\mathbf{x}$ is randomly selected as the input of D. $D(\mathbf{x})$ is the output of D, representing the probability that the data distribution of $\mathbf{x}$ belongs to the real data distribution $\mathbb{P}_r$. A noise vector $\mathbf{z}$ is randomly generated in the normal distribution $\mathbb{P}_\mathbf{z}$ as the input to G. G generates synthetic samples $G(\mathbf{z})$. This generated sample is then used as the input of the discriminator, which is used to predict the probability $D(G(\mathbf{z}))$ of $G(\mathbf{z})$ in the generated data distribution $\mathbb{P}_g$.

Table 2. The detailed network structure of the discriminator. The input data dimension is (3, 16, 16).

Layer (Type)	Output
Input	(3, 16, 16)
Flatten	(768, 1)
Dense 1	(512, 1)
Dense 2	(256, 1)
Dense 3	(128, 1)
Dense 4	(64, 1)
Output	(1,)

Table 3. The detailed network structure of the generator, the input data is a random latent vector of length 100.

Layer (Type)	Output
Latent dim	(100, 1)
Dense 1	(128, 1)
Dense 2	(256, 1)
Dense 3	(512, 1)
Dense 4	(1024, 1)
Dense 5	(768, 1)
Reshape and output	(3, 16, 16)

The objective function of GAN is given as follows:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \quad (5)$$

The purpose of GAN is to maximize the discriminator D and minimize the generator G. $\mathbb{P}_r$ and $\mathbb{P}_z$ are the probability distributions of the real data and latent vectors, respectively.

Accordingly, the loss functions of D and G are:

$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{6}$$

$$\mathcal{L}_G = \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{7}$$

3.2.2. Sample Filter Based on Machine Learning

GAN uses the maximum similarity to measure the loss of the entire model training. When the data are distributed in high dimensions, the similarity of the data is difficult to define, which can cause the generated samples to not be close to the real samples. However, since GAN models can generate infinite samples, we use decision trees, random forests, and logistic regression models to compose a sample filter to filter the generated samples. The screening process is shown in Figure 5.

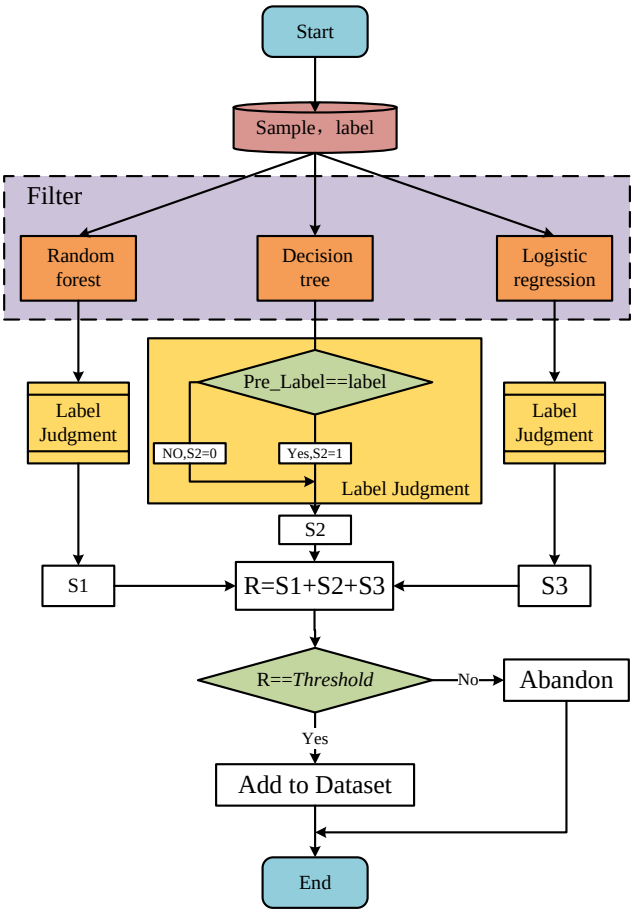


Figure 5. The process of filtering generated samples.

Each generated sample with a label is sent to three machine learning models to obtain predicted labels. The value is 1 if the predicted label is the same as the label of the generated sample, and 0 otherwise. Finally, we add the results of the three models to obtain R, $R \in (0, 1, 2, 3)$. For example, when a sample obtains $R = 2$ through the filter, meaning that the sample is judged to be of this class by the two machine learning models in the filter.

To limit the filtering granularity of the filter, a *Threshold* is set. If R is greater than or equal to the *Threshold*, the generated sample is added to the dataset, otherwise, it is discarded. Algorithm 2 is the training process of Filter-GAN.

Algorithm 2: Filter-GAN Training process.

Input: Real feature images set
Output: Discriminator D, Genarator G of GAN and Filter F
for *each feature image* **do**
    `// Traning GAN`
    Feed the feature image into discriminator D to calculate loss
    Randomly generate Gaussian noise z
    Feed the z Genarator G to generate generated samples X_{fake}
    Feed X_{fake} into discriminator D to Calculate loss
    Compute the training error
    Update weight and bias
    `// Training Filter`
    Convert feature images to a one-dimensional feature data X_{data}
    Feed X_{data} to Random Forests, Decision Trees, Logistic Regression model for training
end
Combining three machine learning models as a filter F

4. Experimental Evaluation

4.1. The Datasets and Evaluation Metrics

4.1.1. Malicious Traffic Dataset

The dataset in the experiments comes from the Malware Capture Facility project [33], which collects malicious traffic over a long period. The datasets released by this project contain malicious traffic generated by various malicious attack methods, such as ransomware, DDoS attacks, and Trojan horse attacks. Many of these malicious programs cannot generate enough network traffic during the attack or propagation process, resulting in an imbalanced number of malicious traffic samples.

Twelve types of malicious traffic were selected. The SplitCap.exe tool [34] was used to split the original traffic file PCAP into network sessions, which were converted to feature images using the feature image extraction algorithm.

In Table 4, it can be seen that there is an extreme imbalance between the samples. For example, MinerTrojan's samples only accounted for 1.23% of the entire dataset, and PUA only accounted for 0.69% of the entire dataset. Therefore, there was a distribution imbalance problem between the categories of this dataset, which can be used to verify the method and model proposed in this paper.

Table 4. Dataset distribution after segmentation and feature image transformation.

Class Name	Size	Session Number	Rate
CoinMiner(CM)	54 M	10,385	7.11%
WebCompanion(WebC)	227 M	6442	4.41%
Trickbot(Tbot)	33 M	30,052	20.58%
Sathurbot(Sbot)	244 M	11,453	7.84%
Ursnif	150M	10,558	7.23%
Artemis Trojan(AT)	773 M	16,719	11.45%
HPEMOTET(HPE)	70 M	13,737	9.41%
Wannacry Ransomware(WR)	6.8 M	10,829	7.41%
Necurse	20 M	2633	1.81%
Magic Hound(M-H)	20 M	30,431	20.84%
MinerTrojan(MinerT)	309 M	1795	1.23%
PUA	54 M	1008	0.69%
Total	1.91G	146,042	100%

4.1.2. Evaluation Metrics

To evaluate the performance of our method on imbalanced datasets from multiple perspectives, we used accuracy, precision, recall, and $F1$:

$$ACCURACY = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$PRECISION = \frac{TP}{TP + FP} \tag{9}$$

$$RECALL = \frac{TP}{TP + FN} \tag{10}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{11}$$

4.1.3. Experimental Platform And Configuration

The training and testing of the model have been carried out under the Windows 10 operating system. The deep learning model was implemented using the Python language based on the Torch framework. The parameters are listed in Table 5.

Table 5. Generator structure parameters.

Category	Version
GPU	Nvidia GPU(GeForce GTX 2080TI)
Operating System	Windows 10, 64 bit
Deep Learining Platform	Torch
DeepLearning Backend	Torch-gpu
Cuda version	11.6
CuDNN version	7.1.4

4.2. Experimental Results and Analysis

The experiment was carried out from two aspects, the first is to prove whether the feature extraction algorithm can effectively characterize malicious traffic. The second is to prove the data generation model and data filter proposed in this paper, which can effectively supplement the samples with an unbalanced number of samples.

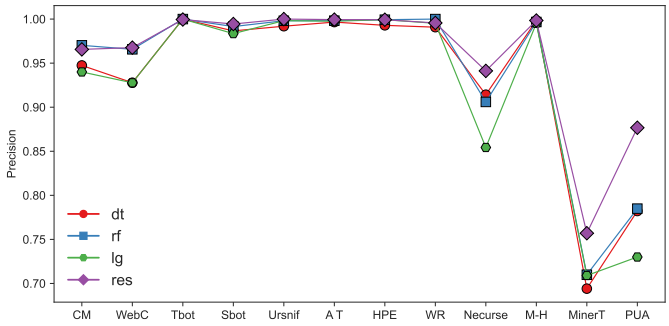
4.2.1. Feature Image Representation Ability Experiment

To verify that the feature extraction algorithm proposed in this paper can effectively characterize malicious network traffic, the feature image sets obtained after data processing are respectively sent to the machine learning model and the neural network model for multi-classification experiments. Machine learning models include random forest (rf), decision trees (dt), and logistic regression (lr) models.

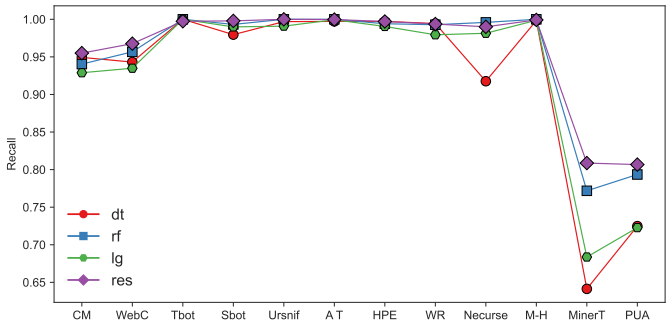
In the machine learning model, each pixel of the three-dimensional feature image (dimension (3, 16, 16)) is regarded as a feature, converted into a one-dimensional vector with a feature number of 768, and normalized as a machine input to the learning model. The neural network model includes the residual convolutional neural network (res, ResNet), which directly uses the three-dimensional feature image as the input of the model. The training parameters are shown in Table 6.

The results of Precision, Recall, and F1 for each category in the classification results are shown in Figure 6. the classification effects of various categories are the ResNet model, Random Forest, Decision Tree, and Logistic Regression. The ResNet model performs significantly better than the machine learning model, because the feature images generated by the feature image extraction algorithm are essentially three-channel images. Each pixel on the image is the transition probability of a fixed-length bit string in the traffic field,

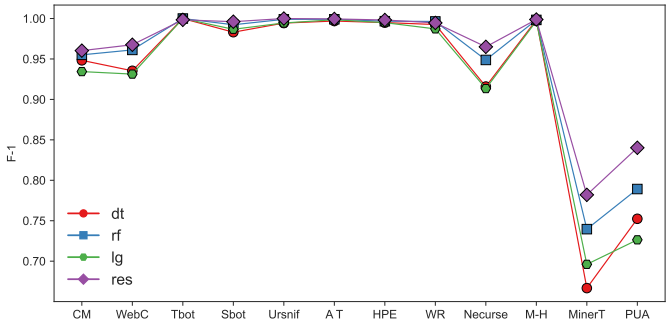
which represents the feature distribution of the field. In the field of image classification, ResNet has a very large advantage, so the results of the ResNet model are better than the classification results of the machine learning model. At the same time, it also shows that the feature image extraction algorithm proposed can effectively extract malicious traffic session features when there are enough samples in the dataset.



(**a**)



(**b**)



(**c**)

Figure 6. Experimental results of random forest (rf), decision trees (dt), logistic regression (lr), and residual networks (res). (**a**) Precision. (**b**) Recall. (**c**) *F*1.

In an addition, Figure 6 shows the detection accuracy of each category using the ResNet model, which shows results above 0.95 for most classes, except Necurse, MinerTrojan, and PUA. In particular, the number of samples in these three categories accounts for less than 2% of the total number of samples, and the number of training samples is less than 1000, which is very small as the amount of pre-training data for deep learning models. Therefore, the reason for the low classification accuracy of these three categories is largely due to the insufficient number of samples, resulting in the imbalance of sample classes.

Table 6. The training parameters in ResNet model.

Training Parameters	Optimizer	Learning Rate	Loss Function	Epochs	Mini_Batch
Value	SGD	0.001	crossentropy	100	128

To solve this problem, in the following experiments, according to the distribution of the sample size of the original dataset, Filter-GAN was used to perform data enhancement for the categories with less than 10,000 samples.

4.2.2. Data Generation and Dataset Balancing

The network session was transformed into feature images through feature extraction algorithms to form feature image datasets. The number of various feature images in the feature image set also inherits this shortcoming due to the unbalanced number of web sessions per class. According to the results of the above experiments, it can be seen that the imbalance in the amount of data between samples leads to poor classification results.

Therefore, for categories with less than 10,000 feature images in the dataset, including WebCompanion, Necurse, MinerTrojan, and PUA, the Filter-GAN model was used for data augmentation. These feature image datasets were trained as input to the GAN model in Filter-GAN. The featured image was then converted into a 1D vector (one feature per pixel) and fed to the filter for training. Finally, the trained GAN model was used for sample generation, and the generated samples were screened by the filter. The samples generated by screening were closer to the data distribution of the real samples.

Samples that can pass the filter were selected and those that did not pass were discarded directly. Because a GAN model can use random noise as input, it can keep generating samples until enough valid samples are generated. Figure 7 shows the comparison of the amount of data before and after the four small sample balances in the dataset so that the entire dataset achieves class balance.

To highlight the effect of the filter, we use the following statistical methods to compare the effectiveness of generated samples and real data. Considering the real data samples $\mathbf{A} = \{\mathbf{a}_i\}_{i=1}^{M} \sim \mathbb{P}_r$, and the generated data samples $\mathbf{B} = \{\mathbf{b}_j\}_{j=1}^{N} \sim \mathbb{P}_g$, where $\mathbf{a}_i, \mathbf{b}_j \in \mathbb{R}^D$. The average of the real and generated data samples is computed as $\mu_A = \frac{1}{M}\sum_{i=1}^{M} \mathbf{a}_i$ and $\mu_B = \frac{1}{N}\sum_{j=1}^{N} \mathbf{b}_j$, respectively.

1. Euclidean distance: Euclidean distance (ED) is used to evaluate the distance of two samples in Euclidean space. As shown in Equation (12), the lower ED indicates that the real sample and the generated sample are more similar.

$$ED(\mathbf{A}, \mathbf{B}) = \|\mu_{\mathbf{A}} - \mu_{\mathbf{B}}\|^2 = \sum_{d=1}^{D} (\mu_{\mathbf{A},d} - \mu_{\mathbf{B},d})^2 \tag{12}$$

2. Correlation Coefficient: The Pearson correlation coefficient ($CC \in [-1, 1]$) assesses the correlation between two samples, as shown in Equation (13). The higher the correlation between the two samples, the closer the CC is to 1.

$$\begin{aligned} CC(\mathbf{A}, \mathbf{B}) &= \frac{D\sum_d \mu_{\mathbf{A},d}\mu_{\mathbf{B},d} - \sum_d \mu_{\mathbf{A},d}\mu_{\mathbf{B},d}}{c_1 \times c_2} \\ c_1 &= \sqrt{D\sum_d \mu_{\mathbf{A},d}^2 - (\sum_d \mu_{\mathbf{A},d})^2} \\ c_2 &= \sqrt{D\sum_d \mu_{\mathbf{B},d}^2 - (\sum_d \mu_{\mathbf{B},d})^2} \end{aligned} \tag{13}$$

3. Fréchet distance: The Fréchet distance (FD) is given by Equation (14). Evaluating the distance of two samples in metric space is a robust measure relative to Euclidean distance. A lower FD indicates that the two samples are more similar.

$$
\begin{aligned}
FD(\mathbf{A},\mathbf{B}) &= \|\mu_{\mathbf{A}} - \mu_{\mathbf{B}}\|^2 + Tr(\Sigma'_{\mathbf{AB}}) \\
\Sigma'_{\mathbf{AB}} &= \Sigma_{\mathbf{A}} + \Sigma_{\mathbf{B}} - 2\sqrt{\Sigma_{\mathbf{A}}\Sigma_{\mathbf{B}}} \\
\Sigma_{\mathbf{A}} &= \frac{1}{M-1}\sum_{i=1}^{M}(\mathbf{a}_i - \mu_{\mathbf{A}})(\mathbf{a}_i - \mu_{\mathbf{A}})^T \\
\Sigma_{\mathbf{B}} &= \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{b}_i - \mu_{\mathbf{B}})(\mathbf{b}_i - \mu_{\mathbf{B}})^T
\end{aligned} \tag{14}
$$

In order to evaluate the effect of the filter, we set different *thresholds*, including 1, 2, and 3, to generate sample screening. For example, a threshold of 2 indicates that the generated samples pass both machine learning models in the filter. The generated samples of each class are screened, and the similarity estimates are calculated with the real samples, including ED, CC, and FD. The results are shown in Table 7.

For the result of each class of samples and each threhold, the result that generated samples pass the three machine learning models in the filter is the best; that is, the ED and FD are the smallest, and the CC value is the largest. This shows that the filter built by machine learning is able to filter out generated samples that are more similar to the real samples. In other words, compared with the general GAN model, the samples that Filter-GAN generate are closer to the real samples.

Table 7. Statistical evaluation of data augmentation methods.

Class	Filter Threshold	ED	CC	FD
PUA	1	1.4233	0.7396	0.2764
	2	1.5597	0.74910	0.2380
	3	**0.7429**	**0.8763**	**0.1475**
WebCompanion	1	3.3419	0.2437	0.7057
	2	1.4356	0.6712	0.34327
	3	**0.7512**	**0.8776**	**0.2236**
Necure	1	1.8990	0.5266	0.3572
	2	1.1253	0.7132	0.2105
	3	**1.0797**	**0.7808**	**0.1730**
MinerTrojan	1	1.0582	0.7802	0.1510
	2	0.8240	0.8421	0.1332
	3	**0.6757**	**0.8895**	**0.1061**

Note: Bold represents best values.

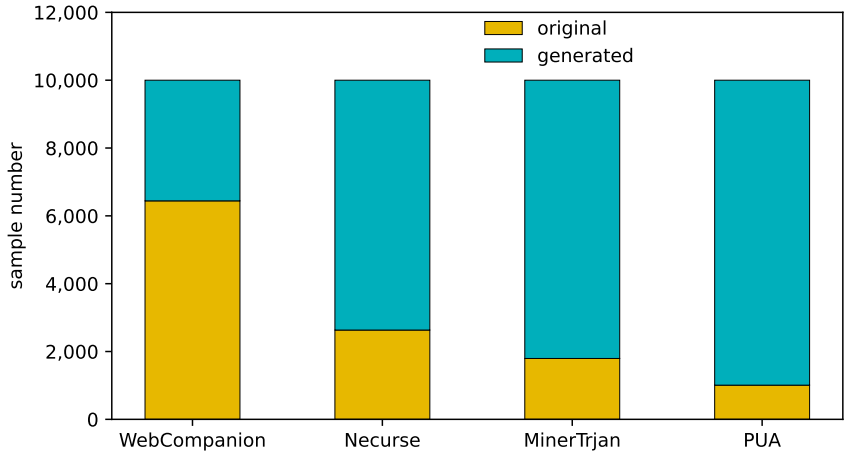


Figure 7. Data distribution after balancing subclasses.

4.2.3. Experimental Results on Balanced Datasets

To compare the filtering effect of Filter on the generated samples. Experiments are carried out on imbalanced datasets and balanced datasets with different filter thresholds. As can be seen in Section 3.2.2, when the threshold = 1 is set, the sample passes only one machine learning model in the filter. As can be seen from Table 8, when only the imbalance feature image set is used as the training set for classification, the accuracy of the small sample categories such as Necurse, MinerTrojan, and PUA is very low.

Table 8. Comparative experimental results based on balanced and unbalanced datasets.

Traffic Class Name	**Imbalance Dataset**			***Threshold* = 1**			***Threshold* = 2**			***Threshold* = 3**		
	Pre	**Rec**	**F1**	**Pre**	**Rec**	**F1**	**Pre**	**Rec**	**F1**	**Pre**	**Rec**	**F1**
CoinMiner	0.9656	0.955	0.9603	0.9671	0.9523	0.9596	0.9654	0.9574	0.9614	0.9753	0.9599	0.9675
WebCompanion	0.9676	0.9676	0.9676	0.9627	0.962	0.9623	0.9697	0.9711	0.9704	0.9607	0.9699	0.9653
Trickbot1	0.9995	0.9974	0.9984	0.9968	0.9967	0.9967	0.9975	0.9988	0.9981	0.9983	0.997	0.9976
Sathurbot	0.9944	0.9979	0.9961	0.9916	0.9947	0.9931	0.9948	0.9965	0.9956	0.9907	0.996	0.9933
Ursnif	1.0	1.0	1.0	0.9977	0.9986	0.9981	0.9995	0.999	0.9992	0.9995	0.9981	0.9988
Artemis Trojan	0.9994	0.9997	0.9995	0.9982	0.9994	0.9988	0.9997	0.9994	0.9995	0.9994	1.0	0.9997
HPEMOTET	0.9993	0.9971	0.9982	0.9982	0.9994	0.9988	0.9971	0.9971	0.9971	0.9993	0.9946	0.9969
Wannacry-R	0.9956	0.9937	0.9946	0.9977	0.9926	0.9951	0.9937	0.9914	0.9925	0.9981	0.9963	0.9972
Necurse	0.9412	0.99	0.965	0.9452	0.9848	0.9646	0.955	0.9903	0.9723	0.9303	0.9848	0.9568
Magic Hound	0.9984	0.9992	0.9988	0.9987	0.9998	0.9992	0.9987	0.9992	0.9989	0.9984	0.9995	0.9989
MinerTrojan	0.757	0.8087	0.782	0.7303	0.767	0.7482	0.7872	0.7521	0.7692	**0.8187**	0.821	0.8198
PUA	0.8767	0.8067	0.8402	0.8497	0.8283	0.8389	0.8394	0.8592	0.8492	0.8535	0.8622	0.8578
Avarage	0.9578	0.9594	0.9584	0.9528	0.9563	0.9545	0.9581	0.9593	0.9586	0.9602	0.9649	0.9625

Note: Bold represents best values.

When using a balanced dataset with the filter, $Threshold = 1$. We use generated feature image samples as a complement to the few-shot class training set. Compared with using only the imbalanced dataset, the classification accuracy of this case is not significantly improved, which is large because the data distribution of the samples generated by the GAN network is not close to the real samples. That is to say, the sample size of these four types of samples used to train the GAN network model is too small so that the GAN network model cannot fully learn the data distribution of the small sample category so that the generated samples of effective data distribution cannot be accurately generated. Therefore, it is necessary to use filters to screen the generated samples.

When the $Threshold = 2$ or 3, the generated samples as supplementary four-type small samples pass the classification of at least one machine learning model in the filter. The results show that the classification results of MinerTrojan, WebCompanion, and Necurse are improved. The F-1 indicators all rose above 0.97. Necurse's Precision and F-1 metrics improved by 1%. In particular, the MinerTrojan category achieved an accuracy of 0.8187, an improvement of 6%. This shows that through the filtering of the filter, the data distribution of the generated samples is closer to the real samples, which effectively solves the problem of class imbalance in the real dataset. Filter-GAN considers the diversity and accuracy of the generated samples, making the classification effect of the samples more significant.

4.2.4. Comparative Experiment

We implement the commonly used methods for dealing with data imbalance as a comparative experiment for Filter-GAN. The first method is *the Random Oversampling (ROS)* [22], which aims to improve the sampling rate of small classes. ROS generates some copies of the secondary class examples. The method is simple to implement and requires little computation. The second way is the *synthetic minority over-sampling technique (SMOTE)* [24]. The SMOTE method generates samples by evaluating the feature space of the sample and its k-nearest neighbors, where k is determined by the number of minority samples. Specifically,

first let the d_i vector be different from the feature vector of the minority sample x_i, that is, the k nearest neighbors, let $d_i^{'} = d_i \times r$ where $r \in [0, 1]$. Then, the new sample $x_i^{'} = x_i + d_i^{'}$. Further, some derivatives of SMOTE are also implemented, such as SVM-SMOTE [26], ADASYN [25].

Table 9 shows the results of the comparison experiments. The table shows the overall accuracy for the classification task and the classification accuracy for the four minority classes. The overall accuracy of our method is higher than other methods. Secondly, the accuracy rates of CM, WebC, MinerT, and PUA are also higher than other research methods. Therefore, the Filter-GAN method outperforms other compared methods, which shows that our method has advantages in malicious traffic characterization and data augmentation.

Table 9. The result of comparative experiment.

Method	Overall Accuracy	CM	WebC	Necurse	MinerT	PUA
ROS	0.9867	0.9623	0.9599	0.9392	0.7058	0.8462
RUS	0.9691	0.8907	0.8586	0.8734	0.7808	0.7972
SMOTE	0.9841	0.9578	0.9465	0.9392	0.7519	0.7168
SVM-SMOTE	0.9856	0.951	0.9502	0.9516	0.7365	0.8112
ADASYN	0.987	0.9639	0.9647	0.9417	0.7038	0.8112
GAN without Filter	0.9852	0.9603	0.9627	0.9452	0.7303	0.8497
Filter-GAN	0.9907	0.9753	0.9607	0.9303	0.8187	0.8535

5. Conclusions

The class imbalance of malicious traffic datasets leads to the low classification accuracy of deep learning and weak generalization ability. It is solved by two aspects in this paper: on the one hand, the features of the network session are mapped to the feature image, which makes the low-rank feature space of the image represent the diversity of the original traffic session. This method can generate images of uniform size and realize the uniform expression of features in different sessions. On the other hand, to balance the original traffic session dataset, the Filter-GAN model continuously generates new feature images through the GAN model and uses filter to filter the generated samples, so that the generated samples are closer to the distribution of real data. Experimental results show that the feature image representation method can effectively classify malicious traffic families. When there are enough class samples, the detection accuracy of the samples can reach 99%. In the case of unbalanced samples of malicious traffic families, after using the Filter-GAN model to enhance the small sample data, the detection and classification accuracy is also significantly improved up to 6%.

Author Contributions: Conceptualization, X.C. and Q.L.; methodology, X.C.; software, P.W.; resources, X.C.; data curation, X.C.; writing—original draft preparation, X.C.; writing—review and editing, X.C.; visualization, P.W.; supervision, Q.L.; project administration, P.W.; funding acquisition, Q.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 61902328 and Key R&D projects of Sichuan Science and technology plan (2022YFG0323).

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to lab privacy.

Acknowledgments: The authors wish to thank the editor and the anonymous reviewers for their valuable suggestions on improving this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cotton, M.; Eggert, L.; Touch, D.J.D.; Westerlund, M.; Cheshire, S. Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335, 2011. Available online: https://www.rfc-editor.org/info/rfc6335 (accessed on 24 October 2020).
2. Khalife, J.; Hajjar, A.; Diaz-Verdejo, J. A multilevel taxonomy and requirements for an optimal traffic-classification model. *Int. J. Netw. Manag.* **2014**, *24*, 101–120. [CrossRef]
3. Park, B.C.; Won, Y.J.; Kim, M.S.; Hong, J.W. Towards automated application signature generation for traffic identification. In Proceedings of the NOMS 2008—2008 IEEE Network Operations and Management Symposium, Bahia, Brazil, 7–11 April 2008; pp. 160–167.
4. Sherry, J.; Lan, C.; Popa, R.A.; Ratnasamy, S. Blindbox: Deep packet inspection over encrypted traffic. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, London, UK, 17–21 August 2015; pp. 213–226.
5. Zhang, K.; Wang, Z.; Chen, G.; Zhang, L.; Yang, Y.; Yao, C.; Wang, J.; Yao, J. Training effective deep reinforcement learning agents for real-time life-cycle production optimization. *J. Pet. Sci. Eng.* **2022**, *208*, 109766. [CrossRef]
6. Yin, F.; Xue, X.; Zhang, C.; Zhang, K.; Han, J.; Liu, B.; Wang, J.; Yao, J. Multifidelity genetic transfer: An efficient framework for production optimization. *SPE J.* **2021**, *26*, 1614–1635. [CrossRef]
7. Zhang, K.; Zhang, J.; Ma, X.; Yao, C.; Zhang, L.; Yang, Y.; Wang, J.; Yao, J.; Zhao, H. History matching of naturally fractured reservoirs using a deep sparse autoencoder. *SPE J.* **2021**, *26*, 1700–1721. [CrossRef]
8. Ma, X.; Zhang, K.; Zhang, L.; Yao, C.; Yao, J.; Wang, H.; Jian, W.; Yan, Y. Data-driven niching differential evolution with adaptive parameters control for history matching and uncertainty quantification. *SPE J.* **2021**, *26*, 993–1010. [CrossRef]
9. Xu, X.; Wang, C.; Zhou, P. GVRP considered oil-gas recovery in refined oil distribution: From an environmental perspective. *Int. J. Prod. Econ.* **2021**, *235*, 108078. [CrossRef]
10. Xu, X.; Hao, J.; Zheng, Y. Multi-objective artificial bee colony algorithm for multi-stage resource leveling problem in sharing logistics network. *Comput. Ind. Eng.* **2020**, *142*, 106338. [CrossRef]
11. Wang, Z. The applications of deep learning on traffic identification. *BlackHat USA* **2015**, *24*, 1–10.
12. Jia, W.; Liu, Y.; Liu, Y.; Wang, J. Detection Mechanism Against DDoS Attacks based on Convolutional Neural Network in SINET. In Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 12–14 June 2020; Volume 1, pp. 1144–1148.
13. Wang, Y.; Jiang, Y.; Lan, J. Fcnn: An efficient intrusion detection method based on raw network traffic. *Secur. Commun. Netw.* **2021**, *2021*, 5533269. [CrossRef]
14. Man, J.; Sun, G. A residual learning-based network intrusion detection system. *Secur. Commun. Netw.* **2021**, *2021*, 5593435 . [CrossRef]
15. Bai, H.; Liu, G.; Liu, W.; Quan, Y.; Huang, S. N-gram, semantic-based neural network for mobile malware network traffic detection. *Secur. Commun. Netw.* **2021**, *2021*, 5599556. [CrossRef]
16. Marín, G.; Caasas, P.; Capdehourat, G. Deepmal-deep learning models for malware traffic detection and classification. In *Data Science–Analytics and Applications*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 105–112.
17. Hwang, R.H.; Peng, M.C.; Huang, C.W.; Lin, P.C.; Nguyen, V.L. An unsupervised deep learning model for early network traffic anomaly detection. *IEEE Access* **2020**, *8*, 30387–30399. [CrossRef]
18. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; pp. 43–48.
19. Zhang, W.; Wang, J.; Chen, S.; Qi, H.; Li, K. A framework for resource-aware online traffic classification using cnn. In Proceedings of the 14th International Conference on Future Internet Technologies, Phuket, Thailand, 7–9 August 2019; pp. 1–6.
20. Lotfollahi, M.; Jafari Siavoshani, M.; Shirali Hossein Zade, R.; Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2020**, *24*, 1999–2012. [CrossRef]
21. Wang, P.; Chen, X. SAE-based encrypted traffic identification method. *Comput. Eng.* **2018**, *44*, 140–147.
22. Vu, L.; Van Tra, D.; Nguyen, Q.U. Learning from imbalanced data for encrypted traffic identification problem. In Proceedings of the Seventh Symposium on Information and Communication Technology, Ho Chi Minh, Vietnam, 8–9 December 2016; pp. 147–152.
23. Cieslak, D.A.; Chawla, N.V.; Striegel, A. Combating imbalance in network intrusion datasets. In Proceedings of the GrC, Citeseer, Atlanta, GA, USA, 10–12 May 2006; pp. 732–737.
24. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
25. Chen, Z.; Zhou, L.; Yu, W. ADASYN- Random Forest Based Intrusion Detection Model. In Proceedings of the 2021 4th International Conference on Signal Processing and Machine Learning, Beijing, China, 18–20 August 2021; pp. 152–159.
26. Nguyen, H.M.; Cooper, E.W.; Kamei, K. Borderline over-sampling for imbalanced data classification. In Proceedings of the Proceedings: Fifth International Workshop on Computational Intelligence & Applications, IEEE SMC Hiroshima Chapter, Hiroshima City, Japan, 10–12 November 2009; Volume 2009, pp. 24–29.
27. Last, F.; Douzas, G.; Bacao, F. Oversampling for imbalanced learning based on k-means and smote. *arXiv* **2017**, arXiv:1711.00837.
28. Hu, W.; Tan, Y. Generating adversarial malware examples for black-box attacks based on GAN. *arXiv* **2017**, arXiv:1702.05983.

29. Vu, L.; Bui, C.T.; Nguyen, Q.U. A deep learning based method for handling imbalanced problem in network traffic classification. In Proceedings of the Eighth International Symposium on Information and Communication Technology, Nha Trang, Vietnam, 7–8 December 2017; pp. 333–339.
30. Wang, P.; Li, S.; Ye, F.; Wang, Z.; Zhang, M. PacketCGAN: Exploratory study of class imbalance for encrypted traffic classification using CGAN. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–7.
31. Wang, Z.; Wang, P.; Zhou, X.; Li, S.; Zhang, M. FLOWGAN: Unbalanced network encrypted traffic identification method based on GAN. In Proceedings of the 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 16–18 December 2019; pp. 975–983.
32. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Commun. ACM* **2014**, *27*, 139–144.
33. The CTU Dataset from Malware Capture Facility Project. Available online: https://www.stratosphereips.org/datasets-malware (accessed on 15 March 2021).
34. SplitCap.exe Tool. Available online: https://www.netresec.com/?page=SplitCap (accessed on 24 October 2020).