Combating Imbalance in Network Traffic Classification Using GAN Based Oversampling

Yu Guo*†, Gang Xiong*†, Zhen Li*†, Junzheng Shi*†, Mingxin Cui*†, Gaopeng Gou*†✉
*Institute of Information Engineering, Chinese Academy of Sciences
†School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
{guoyu,xionggang,lizhen,shijunzheng,cuimingxin,gougaopeng}@iie.ac.cn

***Abstract*—With the proliferation of encrypted traffic, machine learning (ML) based network traffic classification (NTC) has become the mainstream method. However, most studies ignored two issues. On the one hand, Internet traffic presents a natural uneven distribution. On the other hand, machine learning algorithms generally aim to achieve the highest overall accuracy without considering class imbalance. This leads to severe performance degradation of existing ML-based NTC schemes when facing imbalanced scenarios. In this paper, we design a novel Generative Adversarial Network (GAN) architecture to generate traffic samples, in which the addition of the classifier and the pre-training module makes the generation process more stable and effective. We propose an end-to-end framework for imbalanced traffic classification, named ITCGAN, which can generate traffic samples for minority classes to adaptively rebalance the original traffic and simultaneously train the optimal classifier. We evaluate its effectiveness on the public ISCXVPN2016 dataset based on the global metrics and individual metrics. The results show that our method performs well in imbalanced NTC tasks, fully alleviating the performance degradation (a 10.27-percentage-point improvement to the precision of the most minority class). Meanwhile, it surpasses five state-of-the-art oversampling methods.**

***Index Terms*—network traffic classification, machine learning, class imbalance, Generative Adversarial Network**

I. Introduction

In recent years, network traffic classification (NTC) has been extensively studied because of its fundamental role in network management and cyber security [1], [2]. With the full encryption of traffic, port-based and deep packet inspection-based NTC technologies gradually become invalid. Machine learning (ML) technology has become the most effective and mainstream method. A large number of studies on mining the effective traffic feature and exploring the optimal classification network have emerged in the academic community, and have achieved good results [3]–[8]. However, most studies overlooked two important issues. First, Internet traffic presents a natural imbalance distribution [9]. The proportion of traffic generated by different protocols and applications is completely different [10]. Second, most machine learning algorithms are designed to pursue the highest overall accuracy without considering class imbalance, which shifts the training of the classifier to the majority class [11]. Generally, the class with the largest sample size is named the majority class, while the class with a smaller sample size is called the minority class [12]. This results in the performance of existing ML-based NTC schemes being greatly degraded when facing real-world imbalanced traffic classification tasks [3], [6], [13]–[16]. In certain scenarios, such as network censorship, intrusion detection, etc. where high-value traffic tends to occupy only a small proportion, performance degradation on the minority class is catastrophic. Therefore, the imbalance problem in NTC researches should attract enough attention.

The methods that have been proposed to combat imbalance in NTC can be categorized in three main levels: data level, algorithm level and cost-sensitive level [11], [13]. Data level methods address class imbalance by increasing the sample size of minority classes or decreasing the majority classes' sample number, corresponding to oversampling [17] and undersampling [9], respectively. Algorithm level methods employ the algorithms that can award the minority classes and punish the majority while training [18]. Cost-sensitive level methods consider different misclassifying costs for different classes [19]. Among these methods, oversampling are the most commonly used methods for imbalanced traffic classification due to its intuitive principle and good effects [13], [20].

Oversampling refers to increasing the number of samples in the minority class to rebalance the original dataset [17], [20]. Some general oversampling techniques such as Random Oversampling (ROS) [21] and Synthetic Minority Oversampling Technique (SMOTE) [22] are often used to combat the imbalance in traffic. However, these general techniques simply replicate minority samples or generate samples based on Euclidean distance and heuristic rules, without considering the characteristics of traffic, which may easily cause overfitting and introduce noise [23]. In order to better alleviate the imbalance in traffic, recent studies began to use deep learning (DL) for oversampling. Generative Adversarial Network (GAN) [24] is a typical generative network, and its powerful generation capability shows very strong potential [25]–[27]. But the sample generation process is difficult to control, especially for traffic data, a kind of non-visual data. In addition, all studies mentioned above have separated the traffic oversampling and the classifier training into two independent processes, which is not conducive to obtaining the global optimal classifier.

✉ Corresponding author. E-mail address: gougaopeng@iie.ac.cn

ISBN 978-3-903176-39-3© 2021 IFIP

In this article, we first analyze the characteristics of network traffic and point out the inapplicability of the general oversampling methods. Inspired by triple-GAN [28], we designed a novel GAN architecture that trains the classifier while training the generator, so that the generator can flexibly adjust the training direction according to the performance of the current classifier. We propose an end-to-end imbalanced traffic classification framework named ITCGAN, which consists of Traffic Vectorization module, Pre-training module and Formal Training module. The original imbalanced traffic is directly input into the framework and the optimal classifier that is robust to imbalance can be obtained without any manual operation. It also solves the problem that the quality of the synthesized traffic samples are difficult to judge.

The main contributions of our work are briefly summarized as follows:

- We propose an integrated ITCGAN framework for imbalanced traffic classification. Considering the characteristics of traffic, ITCGAN uses GAN to enhance the minority traffic, and combines traffic oversampling and classifier training into a whole.
- The pre-training module can provide more effective information at an early stage and accelerate the convergence of the formal training. In addition, by using the empirical optimal network as the pre-training network, ITCGAN can extend the existing DL-based NTC schemes to be suitable for imbalanced scenarios.
- The constraint design between the classifier and the generator guarantees the quality of the generated traffic samples. According to the constraints, the generator synthesize traffic samples that are most beneficial to the final classification.
- ITCGAN achieves good results on the real-world public ISCXVPN2016 dataset, and outperforms several state-of-the-art methods, proving its superiority in imbalanced traffic classification tasks.

The rest of this paper is organized as follows: Section II discusses the related work. Our proposed framework is introduced in detail in section III. We describe the experimental setup in section IV and analyze the results in section V. Finally, we conclude this paper in section VI.

II. Related Work

As discussed in the previous section, increasing sample size for minority classes is the most commonly used strategy to counter imbalance in network traffic classification researches. In this section, we introduce the related researches on this from two perspectives according to their different underlying principles, including general oversampling techniques based methods and deep learning based methods.

A. General Oversampling Techniques Based Methods

As the class imbalance problem is widely studied as one of the most challenging problems in machine learning, some general oversampling techniques have been proposed and adopted by imbalanced traffic classification researches. The simplest technique is ROS, which raises the sample size by randomly copying samples from minority classes. [20], [23], [29] employed ROS to rebalance the traffic dataset before training. However, ROS does not introduce any extra information, which will cause overfitting. SMOTE synthesizes new samples through distance-based rules, improving the problem that ROS is prone to overfitting. Seo et al. proposed an approach to find the optimal SMOTE ratio for imbalanced intrusion detection datasets [17]. Cieslak et al. designed a method called Cluster-SMOTE, which first clustered the minority samples and then used SMOTE to generate traffic samples in individual clusters [30]. They claimed that Cluster-SMOTE improved SMOTE's performance. Similarly, Vu et al. used a variant of SMOTE, SMOTE-SVM, to solve the imbalance in SSH traffic identification [23]. In another work, SMOTE and an undersampling technique were combined to mitigate the imbalance in multiple traffic datasets [31]. SMOTE is proven to be an effective method in imbalanced traffic classification, but it does not consider the situation of the samples located around minority classes, which is easy to introduce noise and cause unstable effect of traffic classification. Gomez et al. made a comprehensive review of imbalanced NTC studies and finally concluded several well-performing oversampling methods [13]. However, these general oversampling techniques are completely separated with the classifier training phase, i.e., oversampling must be performed first, then the classifier can be trained on the obtained balanced dataset.

B. Deep Learning Based Methods

Recently, some DL-based methods have been proposed to solve imbalanced NTC problems. Hasibi et al. presented a novel data augmentation approach based on the use of Long Short Term Memory (LSTM) networks for generating traffic flow patterns and Kernel Density Estimation (KDE) for replicating the numerical features of each class [32]. However, this method limits the form of the input traffic features and has poor versatility. Another well-known data generation technique is GAN, which generates new instances by modeling and learning the distribution of real data. Wang et al. used the traditional GAN to perform data augmentation on the minority classes of ISCXVPN2016 traffic dataset for application classification [25]. Their subsequent work adopted conditional GAN (CGAN) to achieve the same purpose [26]. However, it is difficult to judge the quality of the generated samples because traffic data are invisible. Moreover, these studies also split the imbalanced traffic classification problem into two isolated stages.

In addition to confronting imbalance, GAN is also adopted to generate traffic data for enhancing Intrusion Detection System (IDS) [33], [34], circumventing censorship [35], and verifying the availability of the fake traffic data [36], [37]. GAN shows broad application prospects and good performance, which is worth further exploration. Therefore, in this paper, we propose an imbalanced traffic classification framework using GAN for oversampling and verify its superiority.

III. Design of Framework

In this section, we introduce our proposed framework for imbalanced traffic classification with GAN-based oversampling, named ITCGAN.

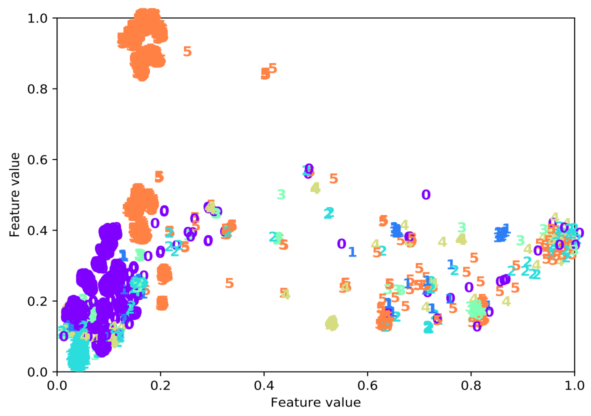

Fig. 1. Visualization of a 6-class imbalanced traffic data by t-SNE.

We first visualize a 6-class real-world traffic data to observe the characteristic of traffic distribution. We use t-SNE to reduce the high-dimensional traffic data to two dimensions and show it in Fig. 1. It can be seen that the six types of traffic have a huge gap in their sample numbers. Moreover, there is much overlap between classes, and the distribution of minority classes is scattered. As mentioned in the previous section, general oversampling algorithms such as SMOTE perform new sample synthesis based on Euclidean distance and specific rules such as interpolation. But the prerequisite for these algorithms to achieve good results is that in the Euclidean space, the samples surrounding the minority samples still belong to the minority class with a high probability. However, traffic data may not meet this premise according to Fig. 1. In fact, traffic samples may consist of statistical features, sequence features, protocol field features, or raw bytes. Euclidean distance and interpolation cannot be simply used to predict the distribution of new samples. Different from the principle of general sampling technologies, GAN generates new samples by fitting the distribution of real data. GAN has powerful learning and generation capabilities, avoiding the drawbacks of general sampling algorithms. Therefore, we decide to use GAN to enhance the minority traffic.

Fig. 2 illustrates the framework of ITCGAN. It consists of three parts, Traffic Vectorization, Pre-training and Formal Training. The Formal Training includes synchronous training of the Generator, the Discriminator and the Classifier. Next, we will introduce each part in detail.

A. Traffic Vectorization

Whether for real-time traffic or stored pcap files, we need to vectorize the raw traffic before proceeding. As we know, traffic is composed of flows, and a flow is often used as a sample. A flow refers to a set of packets with the same 5-tuple (source IP, destination IP, source port, destination port, and transport layer protocol). In order to reduce the interference of irrelevant information, we removed SYN, ACK and other handshake packets, as well as protocol headers. Some irrelevant fingerprints in protocol headers, such as operating system fingerprints, may interfere with classification. We select the first 784 bytes of the application layer data of each flow as its vectorized representation. The part exceeding 784 bytes is truncated, and the insufficient part is filled with 0x00. On the one hand, the use of original datagram reduces the loss of potentially valuable information; on the other hand, only the first 784 bytes are required to identify the traffic, ensuring real-time performance.

After the vectorization, we will get an imbalanced traffic dataset X consisting of n classes and N samples in total, which will be used as the training set. Label the n traffic classes from 1 to n according to each class' sample size in ascending order, i.e., the class with the smallest sample size N_1 is encoded as class 1 and its traffic sample set is X_1, while the class with the largest sample size N_n is encoded as class n and its traffic sample set is X_n. N_i and X_i respectively represent the number and collection of all traffic samples of the i-th class, where $i \in L = \{1, 2, \cdots n\}$, L being the set of possible class labels. We aim to build an end-to-end model containing an effective generator that can synthesize traffic samples for minority classes to eliminate the imbalance, and a synchronously well-trained classifier, which can predict the correct label L_i for any subsequent traffic sample $x^{(i)}$.

B. Pre-training

The Pre-training process refers to training an empirically superior network called Net on the original imbalanced X and saving the pre-trained architecture and parameters as the initial state of the Classifier in the subsequent Formal Training. Numerous existing NTC researches draw on the achievements in the field of computer vision, using the classic Convolutional Neural Networks (CNN) as the traffic classification network. Therefore, we depict Net as the architecture shown in Fig. 2, which is exactly the network architecture employed in the following experiments. When faced with different NTC tasks, Net can be replaced with a more suitable network architecture.

In addition, the focal loss [38] is involved as the loss function in Pre-training process to help alleviate class imbalance, avoiding the model collapse on the most minority class. The focal loss for multi-classification can be calculated as follows:

$$\begin{aligned} FL\left(y_{pred}\right) = -\alpha\left(1-\operatorname{softmax}\left(y_{pred}\right)\right)^{\gamma} \\ \cdot \log\left(\operatorname{softmax}\left(y_{pred}\right)\right) \end{aligned} \tag{1}$$

where y_{pred} is the predicted output of Net on the true label and $\operatorname{softmax}\left(y_{pred}\right)$ is the corresponding probability defined by (2). α is the balancing factor and γ is the focusing parameter.

$$\operatorname{softmax}\left(y_{pred}\right) = \frac{\exp\left(y_{pred}\right)}{\sum_{i=1}^{n}\exp\left(y_i\right)} \tag{2}$$

There are two reasons for setting the Pre-training module. Firstly, it can accelerate the convergence of the Generator and

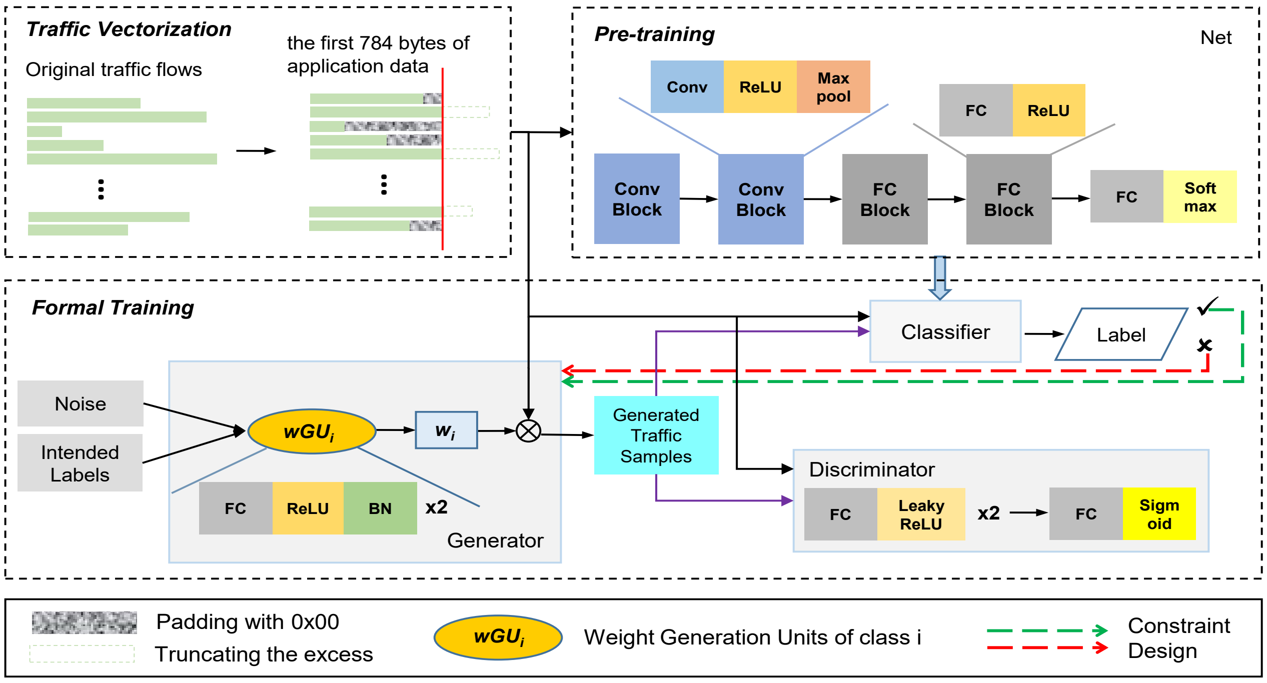

Fig. 2. The framework of ITCGAN. The feedback provided by Discriminator to Generator is omitted.

Discriminator. Compared to a random initialization, using the pre-trained model as the initial for the Classifier will provide the Generator with a much more meaningful signal at the early stage. Secondly, it makes ITCGAN be able to serve as an extension and improvement of the existing DL-based NTC schemes. As mentioned before, existing NTC studies have proposed many excellent methods, including traffic feature extraction and network design. But they ignored the imbalance of traffic data, resulting in the performance degradation on imbalanced NTC tasks. By adopting their proposed superior network architectures as that of the Pre-training Net, ITCGAN could easily adapt the existing studies to imbalanced network traffic tasks.

C. *Formal Training*

1) Generator: The Generator is one of the most critical parts in ITCGAN. The quality of the traffic samples generated by it will directly determine the final performance of the Classifier. The generator in the original GAN proposed by [24] is trained to learn a mapping from a low-dimension latent space, which is always a uniform random distribution, to the target data distribution. Considering that our goal is to eliminate the imbalance in traffic data by synthesizing the minority samples with the Generator, the synthesized samples should well fit the original distribution and be as realistic as possible, but in the meanwhile, be different from the existing data.

Inspired by [39], the generator can be designed to generate a set of weights for each minority class rather than directly generate the data itself. Specifically, our Generator consists of a series of Weight Generation Units (wGU). Each wGU_i corresponds to a minority class, whose task is to learn a conditional mapping g_i from the latent space to a vector $w_i = g_i(\mathbf{z}|i)$ of N_i weights, where $\mathbf{z}$ is the input noise. Hence, the sample generated by the Generator for class i is $G(\mathbf{z}|i) = w_i^T X_i$, that is, in each feature dimension, every sample of class i contributes the value of the corresponding weight, and the synthesized sample value in this dimension is obtained by summarizing these contributions. In addition, the Generator is designed to automatically compensate for the imbalance of different minority classes. The sample set it generates for the i-th class $(i < n)$ is X_i' and its size is $N_n - N_i$.

The objective optimization function for the Generator is presented as follows:

$$\min_G V(G) = \sum_{i \in L} (V_{i1} - V_{i2} - V_{i3}) \tag{3}$$

$$\text{where} \quad V_{i1} = \frac{N_n - N_i}{N} \mathbb{E}_{G(\mathbf{z}|i)\sim p_i^g}[\log(1 - D(G(\mathbf{z}|i)))],$$

$$V_{i2} = \frac{N_n - N_i}{N} \mathbb{E}_{G(\mathbf{z}|i)\sim p_i^g}\left[\log C_i(G(\mathbf{z}|i))\right],$$

$$V_{i3} = \sum_{j \in L\setminus\{i\}} \frac{N_n - N_j}{N} \mathbb{E}_{G(\mathbf{z}|j)\sim p_j^g}\left[\log\left(1 - C_i(G(\mathbf{z}|j))\right)\right],$$

p_i^d and p_i^g separately indicate the real and generated class conditional probability distributions of class i. Minimizing V_{i1} aims to fool the Discriminator with the synthesized fake samples, while maximizing $V_{i2} + V_{i3}$ means to make the generated samples be predicted as the correct labels, a constraint from the Classifier.

2) Discriminator: The function of the Discriminator in ITCGAN is the same as that in traditional GAN, which is designed for distinguishing whether the input sample is real or generated by the generator.

The Discriminator's objective function is expressed by (4):

$$\max_{D} V(D) = \sum_{i \in L} (V_{i1} + V_{i4}) \tag{4}$$

$$\text{where} \quad V_{i4} = \frac{N_i}{N} \, \mathbb{E}_{\mathbf{x} \sim p_i^d}[\log D(\mathbf{x})]$$

3) Classifier: The initial Classifier is obtained through Pre-training, and then trained synchronously with the Generator and Discriminator. Its input includes two parts, the real and the synthesized traffic samples. For each minority class, the number of traffic samples generated by the Generator is the difference between its original sample size and N_n. Therefore, the oversampling ratio is automatically adapted without manual designation.

There are three roles playing by the Classifier in the whole framework. Firstly, it is the ultimate goal of ITCGAN, that is, a target classifier suitable for the imbalanced traffic classification task. The Formal Training finishes when the Classifier's best performance is achieved. Secondly, it provides guidance and constraints for the Generator to generate high-quality traffic data. The samples synthesized by the Generator have to go through the Classifier, which will feed back the classification results to the Generator in the form of gradients to guide its training. As shown in Fig. 2, when the Classifier predicts a generated sample correctly, it will provide a positive feedback to the Generator (i.e., the loss is zero), while when the predicted label is incorrect, a negative feedback is delivered. Thirdly, the Classifier also serves as an indicator. The quality of the generated samples can be inferred from the change of the metric scores instead of paying attention to whether the Generator and Discriminator are converged, which is usually difficult to judge, especially for non-image data.

The principle above is equivalent to the following optimization problem:

$$\max_{C} V(C) = \sum_{i \in L} (V_{i1} + V_{i2} + V_{i5} + V_{i6}) \tag{5}$$

$$\text{where} \quad V_{i5} = \frac{N_i}{N} \, \mathbb{E}_{\mathbf{x} \sim p_i^d} \left[\log C_i(\mathbf{x})\right],$$

$$V_{i6} = \sum_{j \in L \setminus \{i\}} \frac{N_j}{N} \, \mathbb{E}_{\mathbf{x} \sim p_j^d} \left[\log \left(1 - C_i(\mathbf{x})\right)\right]$$

IV. Experimental setup

A. Dataset

The datasets used to evaluate ITCGAN come from the real-world public ISCXVPN2016 dataset [40]. There are two considerations about choosing ISCXVPN2016. Firstly, it provides 28G real-world traffic captured from ISCX, assuring the richness in diversity and quantity. Secondly, Secondly, it is publicly available which aids in making our evaluation more convincing and credible. ISCXVPN2016 mainly contains 7 types of traffic based on different user behaviors and applications. Each type includes a regular encrypted traffic session and a VPN-encapsulated traffic session. We choose the VPN-encapsulated traffic as our experimental dataset for its greater classification difficulty due to the full encryption.

To get the labeled training set and test set for the experiments, we discard the "Web Browsing" category that is difficult to label. After Traffic Vectorization, we get the dataset, namely VPN_ISCX , as described in Table I. The class labels for each dataset are encoded in ascending order according to their sample sizes, as described in section III. '#Train' and '#Test' denote the number of samples in each class in the training set and test set respectively. Moreover, we use imbalance ratio per label (IR) defined by (6) to measure the imbalance level of each dataset. IR is the ratio between the sample size of the majority class and that of class i. It is equal to 1 for the majority class and the fewer samples a minority class has, the larger its IR will be. As can be seen, VPN_ISCX has a maximum IR of 11.07.

$$IR_i = \frac{N_{\text{majority class}}}{N_i} \tag{6}$$

TABLE I
THE OVERVIEW OF VPN_ISCX DATASET

Traffic type	**Label**	**#Train**	**IR**	**#Test**
Email	0	976	11.07	108
P2P	1	1410	7.66	156
Streaming	2	1530	7.06	170
File transfer	3	3300	3.27	360
Chat	4	10800	1	1200
VoIP	5	10800	1	1200

B. Performance Metrics

Metric selection should be very cautious in imbalanced NTC tasks. Since the designs of some traditional metrics do not consider the class imbalance. Overall accuracy, a popular metric often used by NTC studies to measure the classifier's performance, is actually vulnerable to the class distribution skew [13]. In this paper, we adopt metrics from two levels: Global metrics and Individual metrics. Global metrics can measure the classifier on the entire dataset, while Individual metrics assess it more meticulously, providing us with a clear observation of how the given method influences each class.

a) Individual metrics: For individual classes, Precision and Recall from the field of information retrieval are good choices. Although Precision and Recall are first proposed for binary classification, they can be extended to multi-classification tasks with One-versus-All principle, which are defined by (7) and (8) respectively. For class i, TP_i represents the number of samples correctly classified as class i, FP_i is the number of samples misclassified as class i, TN_i is the number of samples correctly predicted as non-class i,

TABLE II
THE COMPARISON METHODS USED IN OUR EVALUATIONS

	Algorithm Description
Random OverSampling (ROS)	The samples from the minority class are randomly selected, duplicated and added to the training dataset. This is the simplest oversampling technique, that is also proven to be robust [41].
Synthetic Minority Oversampling TEchnique (SMOTE)	SMOTE generates new synthetic samples for the minority class. Selecting k nearest minority neighbors for each minority sample and choosing one of those k neighbors. New sample is generated at the line between the current minority sample and its chosen neighbor [22].
ADAptive SYNthetic algorithm (ADASYN)	ADASYN builds on SMOTE by shifting the importance of the classification boundary to those minority classes which are difficult. It uses a weighted distribution for different minority samples according to their level of difficulty in learning, so that more synthetic samples are generated for minority samples that are harder to learn [42].
SMOTE+Support Vector Machine (SMOTE-SVM)	SMOTE-SVM focuses on generating new minority samples using SMOTE near borderlines with SVM, so as to generate minority samples with greater values [43].
SMOTE+Tomek Links (SMOTE-TL)	SMOTE-TL is a hybrid sampling method that combining oversampling and undersampling techniques. It firstly oversamples minority samples using SMOTE and, afterwards, removes the TL links [44].
Conditional generative adversarial network (CGAN)	CGAN is an extension of the GAN where a conditional setting is applied. Both the generator and discriminator are conditioned on some sort of auxiliary information such as class labels or data from other modalities [45].

and FN_i is the number of samples that are misclassified as non-class i. Besides, F_1 score and Area Under Precision-Recall Curve (AUC-PR), which can take Precision and Recall into consideration at the same time, are also employed as our Individual metrics. F_1 score is the harmonic mean of Precision and Recall, as shown in (9). AUC-PR is a rank metric calculated from the area under the Precision-Recall curve.

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{7}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{8}$$

$$F_{1^i} = 2 \cdot \frac{Precision_i \cdot Recall_i}{Precision_i + Recall_i} \tag{9}$$

b) Global metrics: G-mean (GM) and MAUC-PR (MAUC) are chosen as Global metrics in this paper. GM is the geometric mean of all individual Recalls. MAUC is the macro average of all AUC-PRs. Both Global metrics treat all categories equally, meaning that the majority and minority classes have the same effect on the final score. They are separately defined by (10) and (11).

$$GM = \sqrt[n]{\prod_{i=1}^{n} Recall_i} \tag{10}$$

$$MAUC = \frac{1}{n}\sum_{i=1}^{n} AUC\text{-}PR_i \tag{11}$$

C. Comparison methods

We train a classifier directly on the original traffic dataset without any operations for solving imbalance as the **Baseline**. As for the comparison methods, we collect the best 5 traditional oversampling methods and 1 GAN-based data augmentation method that are frequently used to combat the imbalance in traffic classification: **ROS**, **ADASYN**, **SMOTE**, **SMOTE-SVM**, **SMOTE-TL** and **CGAN**. Table II contains a brief description of each algorithm.

Additionally, the 1D-CNN network proposed in [15] proves to have the best performance on the ISCXVPN2016 dataset. Therefore, it is adopted by our baseline, all comparison methods, and the Pre-training Net in ITCGAN. Table III lists its specific network architecture.

TABLE III
THE MAIN PARAMETERS OF THE 1D-CNN NETWORK USED IN OUR EVALUATIONS

Layer	**Operation**	**Input**	**Filter**	**Stride**	**Pad**	**Output**
1	Conv+ReLU	784*1	25*1	1	same	784*32
2	1D Max Pool	784*32	3*1	3	same	262*32
3	Conv+ReLU	262*32	25*1	1	same	262*64
4	1D Max Pool	262*64	3*1	3	same	88*64
5	FC+ReLU	88*64	–	–	none	1024
6	FC	1024	–	–	none	6
7	softmax	6	–	–	none	6

D. Setting of the ITCGAN

In Pre-training, the focal loss is employed as the loss function, where α and γ are respectively set as 0.25 and 2. The model is trained up to 300 epochs with a 512-batch-size. In Formal Training, the batch sizes of the Generator, Discriminator and Classifier are all set to 512, and the maximum training step is set to 40000. One step here represents a parameters-update of the entire ITCGAN, which occurs twice within a batch. The Classifier uses the parameters obtained through pre-training to continue being trained, where the loss function is replaced by Cross Entropy, and the learning rate is set as 0.0003 with a decay of 1×10^{-6}. Both the Generator and Discriminator are models consisting of several fully-connected layers. Their loss functions are Cross Entropy and the learning rate is 0.001 with a decay of 0.0001. Besides, all models are optimized by the Adam optimizer. Our model ITCGAN is implemented with Keras.

V. Results and Analysis

A. Results of Comparison Experiments

The comparison experiment results are shown in Table IV and Fig. 3. In order to show the effects of each method more intuitively, we mark the results with different colors. The red number indicates an improvement compared to the Baseline, while the blue number implies that the Baseline is weakened. The bold red highlights the highest score of all methods under the current metric.

As can be seen, ITCGAN is the only method that enhances Baseline on all metrics and outperforms other methods on most metrics. In terms of global metrics, ITCGAN improves GM and MAUC by 4.30 and 2.27 percentage points, respectively, indicating a good enhancement on overall effect of the classifier. SMOTE-SVM and SMOTE perform second, which shows that SMOTE is a good oversampling algorithm and its advanced variant also works well. However, SMOTE-TL performs worse than Baseline. The reason may be that it introduces the undersampling method TL. Massive removal of the majority samples leads to the loss of potentially valuable information. Both ROS and ADASYN have a higher GM score and a lower MAUC score than the Baseline, suggesting that these two methods help improve the Recall rate of the classifier, but fail to perform well in Precision, resulting in low MAUC values. ITCGAN's sample generation principle is different from the above-mentioned oversampling techniques, that is, it integrates the information of all samples within a class and generates new traffic samples based on the distribution, leading to its superiority. Besides, the performance of CGAN is also quite poor, because it cannot accurately give a signal about the quality of the generated samples. We have no way of knowing when the generated samples are most suitable for traffic augmentation, so that we can only take the samples synthesized when the losses of the generator and the discriminator both converge. Nevertheless, GAN's convergence is not equivalent to the fact that the synthesized samples can provide good augmentation effect for minority classes, especially for traffic data, a kind of non-image data. The Formal training of ITCGAN solves this problem well, so that the generated traffic samples can make the classifier achieve the best performance.

As for individual metrics, from Fig. 3(a), ITCGAN has the greatest degree of improvement in F_1 scores of all categories, especially for class 0, class 1 and class 2, which are improved by 7.30, 4.53, and 6.62 percentage points, respectively, and their corresponding IRs are 11.07, 7.66 and 7.06, demonstrating that ITCGAN is remarkably effective in enhancing the minority classes. The improvements of $Precision_0$ and $Recall_1$ as high as 10.27 and 7.26 percentage points also prove this. For class 4 and class 5 with the largest sample size, ITCGAN's Precision and Recall scores also increase, while the scores of other comparison methods have almost declined. This proves the robustness of ITCGAN, that is, it guarantees the performance of the majority classes while enhancing the minority classes. Although ITCGAN did not get the highest score on a few individual metrics, as shown in Fig. 3(b) that the AUC score of ROS on class 0 is a little higher than that of ITCGAN, all metric scores of ROS on other classes are far inferior to ITCGAN. In other words, ITCGAN has the best overall performance, taking every class into consideration. This is why the F_1 scores of ITCGAN are completely higher than other methods.

B. Further Analysis on ITCGAN

We conduct further analysis on ITCGAN and a few more experiments are performed:

- **ITCGAN-NP (ITCGAN-No Pre-training)**: We take off the Pre-training in ITCGAN and directly use the Classifier with random initialization parameters to participate in the Formal Training.
- **ITCGAN-NCC (ITCGAN-No Classifier Constraint)**: We remove the constraint provided by the Classifier to the Generator. Only the adversarial interaction from the Discriminator is maintained.
- **ITCGAN-conv**: We change the network of the Generator and Discriminator from the fully connect network to the convolutional network in Table III to see if a more complex network can further enhance ITCGAN's performance.

The results are shown in Table V. We can draw the following conclusions:

1. The Pre-training can indeed accelerate the convergence and make it achieve better performance. ITCGAN needs to train 27660 steps to achieve the best results. Without the Pre-training, ITCGAN-NP requires 46710 steps training to reach its best performance, and the best results is still inferior to ITCGAN.

2. The constraint design can better guide the Generator's training process, as well as the sample generation. Viewing ITCGAN-NCC's performance, the metric scores on class 0, class 1 and class 2 are relatively low, indicating its unsatisfying ability for data augmentation. Adding the constraint from the Classifier to the Generator can improve the quality of the generated traffic samples to better eliminate imbalance.

3. Finally, the results of ITCGAN-conv show that using convolutional networks in the Generator and Discriminator does not improve further, but increases training difficulty and time consumption. The fully-connected network can fully meet the sample generation requirements on the current dataset.

Furthermore, we explore the influence of input noise dimension on ITCGAN's performance. Different dimensions of input noise (25, 50, 75, 100, 125, 150, 175, 200) are set and the results are shown in Figure 4. Obviously, ITCGAN achieves the best performance when the noise dimension is 125. Too small value may cause the diversity of generated data to become worse, while large value will increase the calculation burden. Therefore, 125 is a compromise and the best choice in our experiments. When employing ITCGAN in

TABLE IV
GLOBAL AND INDIVIDUAL METRICS OBTAINED BY DIFFERENT METHODS. THE RESULTS ARE EXPRESSED IN %

Methods	Global metric		Individual metric											
	GM	MAUC	P_0	R_0	P_1	R_1	P_2	R_2	P_3	R_3	P_4	R_4	P_5	R_5
Baseline	86.89	91.90	84.96	89.74	96.22	88.74	82.10	68.41	88.01	87.52	94.86	95.50	96.95	94.53
ROS	90.06	91.00	88.66	**95.30**	94.89	90.74	79.25	75.47	84.71	90.80	94.05	95.00	97.12	94.83
ADASYNC	90.82	91.31	93.41	93.41	96.63	94.31	68.39	79.00	81.35	90.80	97.44	93.5	95.97	95.00
SMOTE	89.75	93.17	91.86	**95.30**	97.65	92.59	83.00	73.12	85.98	90.26	94.99	94.83	95.35	94.67
SMOTE-SVM	90.08	93.08	86.27	87.89	90.46	95.15	78.26	**79.71**	87.24	89.16	95.82	95.50	96.44	94.17
SMOTETL	86.24	91.80	93.34	91.59	95.05	92.59	**87.52**	62.53	86.22	84.79	91.96	95.73	94.70	95.50
CGAN	86.84	91.52	95.08	89.74	**99.00**	87.46	84.71	69.59	83.82	87.52	92.11	94.83	95.31	94.67
ITCGAN	**91.19**	**94.17**	**95.23**	93.94	97.72	**96.00**	86.67	76.47	**88.30**	**90.90**	**97.65**	**95.83**	**97.46**	**95.67**

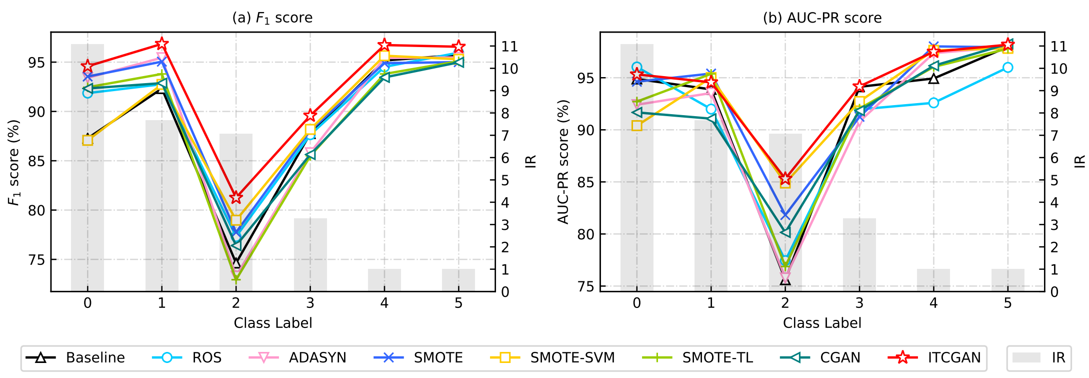

Fig. 3. The individual F_1 score and AUC-PR score of different methods.

TABLE V
RESULTS OF THE FURTHER EXPERIMENTS FOR ITCGAN. THE RESULTS ARE EXPRESSED IN %

	Global metric		Individual metric						Step
	GM	MAUC	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	
ITCGAN-NP	90.11	93.05	95.41	96.29	80.63	87.43	96.10	95.81	46710
ITCGAN-NCC	90.07	92.57	92.73	93.88	80.87	87.69	96.78	96.05	27630
ITCGAN-conv	90.98	92.99	94.44	94.67	82.63	86.43	96.37	96.66	34590
ITCGAN	91.19	94.17	94.58	96.85	81.25	89.58	96.73	96.55	27660

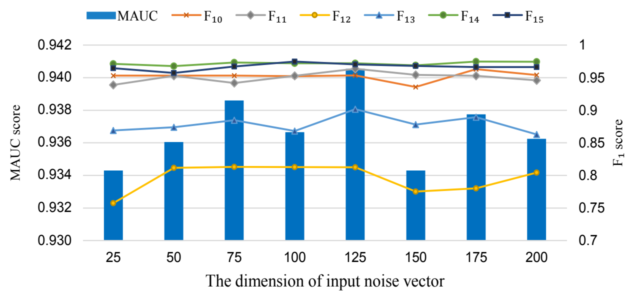

Fig. 4. Results of ITCGAN with different dimensions of input noise.

other NTC tasks, the dimension of input noise should be set according to the actual demand.

VI. CONCLUSION AND FUTURE WORK

Due to the neglect of traffic imbalance, most ML-based NTC schemes may face performance degradation in actual application. In this paper, we propose an end-to-end GAN-based imbalanced traffic classification framework named ITC-GAN. It employs GAN to oversample the minority traffic to rebalance the imbalanced traffic data. Adding the Classifier to the traditional GAN makes the integration of oversampling and training become possible, which helps the classification results achieve the global optimal. The constraint from the Classifier to the Generator ensures the quality of synthesized traffic samples. Besides, Pre-training module can easily expand the existing DL-based NTC methods to adapt to imbalanced tasks. Comprehensive experiments on the public real-world dataset ISCXVPN2016 prove that ITCGAN can effectively alleviate the performance degradation caused by traffic imbalance, and outperforms other state-of-the-art comparison methods.

In the future, we will verify the effectiveness of our proposed method on more real-world traffic datasets, and carry out further research on the data-level solution of imbalanced network traffic classification. In addition, it is a promising direction to solve the problem of imbalanced NTC directly from the algorithm level without any data preprocessing. We will also carry out in-depth researches on that.

Acknowledgment

This work is supported by The National Key Research and Development Program of China (No.2020YFE0200500, No.2020YFB1006100 and No.2018YFB1800200) and Key research and Development Program for Guangdong Province under grant No. 2019B010137003.

References

[1] Anderson B, Paul S, McGrew D. Deciphering malware's use of TLS (without decryption)[J]. Journal of Computer Virology and Hacking Techniques, 2018, 14(3): 195-211.

[2] Barradas D, Santos N, Rodrigues L. Effective detection of multimedia protocol tunneling using machine learning[C]//27th USENIX Security Symposium (USENIX Security 18). 2018: 169-185.

[3] Liu J, Fu Y, Ming J, et al. Effective and real-time in-app activity analysis in encrypted internet traffic streams[C]//Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017: 335-344.

[4] Velan P, Čermák M, Čeleda P, et al. A survey of methods for encrypted traffic classification and analysis[J]. International Journal of Network Management, 2015, 25(5): 355-374.

[5] Shi H, Li H, Zhang D, et al. An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification[J]. Computer Networks, 2018, 132: 81-98.

[6] Taylor V F, Spolaor R, Conti M, et al. Robust smartphone app identification via encrypted network traffic analysis[J]. IEEE Transactions on Information Forensics and Security, 2017, 13(1): 63-78.

[7] Liu C, He L, Xiong G, et al. Fs-net: A flow sequence network for encrypted traffic classification[C]//IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019: 1171-1179.

[8] Zhang Z, Kang C, Xiong G, et al. Deep Forest with LRRS Feature for Fine-grained Website Fingerprinting with Encrypted SSL/TLS[C]//Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019: 851-860.

[9] Amina S I S M, Abdolkhalegh B, Khoa N K, et al. Featuring Real-Time imbalanced network traffic classification[C]//2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data. IEEE, 2018: 840-846.

[10] Chen Z, Yan Q, Han H, et al. Machine learning based mobile malware detection using highly imbalanced network traffic[J]. Information Sciences, 2018, 433: 346-364.

[11] Dong Q, Gong S, Zhu X. Imbalanced deep learning by minority class incremental rectification[J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 41(6): 1367-1381.

[12] Japkowicz N, Stephen S. The class imbalance problem: A systematic study[J]. Intelligent data analysis, 2002, 6(5): 429-449.

[13] Gómez S E, Hernández-Callejo L, Martínez B C, et al. Exploratory study on class imbalance and solutions for network traffic classification[J]. Neurocomputing, 2019, 343: 100-119.

[14] Anderson B, McGrew D. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity[C]//Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017: 1723-1732.

[15] Wang W, Zhu M, Wang J, et al. End-to-end encrypted traffic classification with one-dimensional convolution neural networks[C]//2017 IEEE International Conference on ISI. IEEE, 2017: 43-48.

[16] Lotfollahi M, Siavoshani M J, Zade R S H, et al. Deep packet: A novel approach for encrypted traffic classification using deep learning[J]. Soft Computing, 2020, 24(3): 1999-2012.

[17] Seo J H, Kim Y H. Machine-learning approach to optimize smote ratio in class imbalance dataset for intrusion detection[J]. Computational Intelligence and Neuroscience, 2018, 2018.

[18] Chawla N V, Lazarevic A, Hall L O, et al. SMOTEBoost: Improving prediction of the minority class in boosting[C]//European conference on principles of data mining and knowledge discovery. Springer, Berlin, Heidelberg, 2003: 107-119.

[19] Peng L, Zhang H, Yang B, et al. A new approach for imbalanced data classification based on data gravitation[J]. Information Sciences, 2014, 288: 347-373.

[20] Liu Q, Liu Z. A comparison of improving multi-class imbalance for internet traffic classification[J]. Information Systems Frontiers, 2014, 16(3): 509-521.

[21] Batista G E, Prati R C, Monard M C. A study of the behavior of several methods for balancing machine learning training data[J]. ACM SIGKDD explorations newsletter, 2004, 6(1): 20-29.

[22] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: synthetic minority over-sampling technique[J]. Journal of artificial intelligence research, 2002, 16: 321-357.

[23] Vu L, Van Tra D, Nguyen Q U. Learning from imbalanced data for encrypted traffic identification problem[C]//Proceedings of the 7th Symposium on ICT. 2016: 147-152.

[24] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.

[25] Wang Z X, Wang P, Zhou X, et al. FLOWGAN: Unbalanced Network Encrypted Traffic Identification Method Based on GAN[C]//2019 IEEE ISPA/BDCloud/SocialCom/SustainCom. IEEE, 2019: 975-983.

[26] Wang P, Li S, Ye F, et al. PacketCGAN: Exploratory Study of Class Imbalance for Encrypted Traffic Classification Using CGAN[J]. arXiv preprint arXiv:1911.12046, 2019.

[27] Vu L, Bui C T, Nguyen Q U. A deep learning based method for handling imbalanced problem in network traffic classification[C]//Proceedings of the 8th International Symposium on ICT. 2017: 333-339.

[28] Chongxuan L I, Xu T, Zhu J, et al. Triple generative adversarial nets[C]//Advances in neural information processing systems. 2017: 4088-4098.

[29] Zhong W, Raahemi B, Liu J. Learning on class imbalanced data to classify peer-to-peer applications in IP traffic using resampling techniques[C]//2009 IJCNN. IEEE, 2009: 3548-3554.

[30] Cieslak D A, Chawla N V, Striegel A. Combating imbalance in network intrusion datasets[C]//GrC. 2006: 732-737.

[31] Oeung P, Shen F. Imbalanced Internet Traffic Classification Using Ensemble Framework[C]//2019 ICOIN. IEEE, 2019: 37-42.

[32] Hasibi R, Shokri M, Dehghan M. Augmentation scheme for dealing with imbalanced network traffic classification using deep learning[J]. arXiv preprint arXiv:1901.00204, 2019.

[33] Charlier J, Singh A, Ormazabal G, et al. SynGAN: Towards Generating Synthetic Network Attacks using GANs[J]. arXiv preprint arXiv:1908.09899, 2019.

[34] Lin Z, Shi Y, Xue Z. Idsgan: Generative adversarial networks for attack generation against intrusion detection[J]. arXiv preprint arXiv:1809.02077, 2018.

[35] Li J, Zhou L, Li H, et al. Dynamic Traffic Feature Camouflaging via Generative Adversarial Networks[C]//2019 IEEE Conference on Communications and Network Security (CNS). IEEE, 2019: 268-276.

[36] Cheng A. PAC-GAN: Packet Generation of Network Traffic using Generative Adversarial Networks[C]//2019 IEEE 10th Annual IEMCON. IEEE, 2019: 0728-0734.

[37] Ring M, Schlör D, Landes D, et al. Flow-based network traffic generation using generative adversarial networks[J]. Computers & Security, 2019, 82: 156-172.

[38] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2980-2988.

[39] Mullick S S, Datta S, Das S. Generative adversarial minority oversampling[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 1695-1704.

[40] Draper-Gil G, Lashkari A H, Mamun M S I, et al. Characterization of encrypted and vpn traffic using time-related[C]//Proceedings of the 2nd ICISSP. 2016: 407-414.

[41] Ling C X, Li C. Data mining for direct marketing: Problems and solutions[C]//Kdd. 1998, 98: 73-79.

[42] He H, Bai Y, Garcia E A, et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning[C]//2008 IEEE international joint conference on neural networks. IEEE, 2008: 1322-1328.

[43] Nguyen H M, Cooper E W, Kamei K. Borderline over-sampling for imbalanced data classification[J]. International Journal of Knowledge Engineering and Soft Data Paradigms, 2011, 3(1): 4-21.

[44] Batista G E, Bazzan A L C, Monard M C. Balancing Training Data for Automated Annotation of Keywords: a Case Study[C]//WOB. 2003: 10-18.

[45] Mirza M, Osindero S. Conditional generative adversarial nets[J]. arXiv preprint arXiv:1411.1784, 2014.