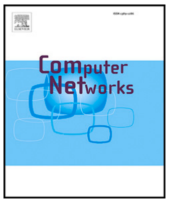
Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



ByteSGAN: A semi-supervised Generative Adversarial Network for encrypted traffic classification in SDN Edge Gateway☆,☆☆

Pan Wang[a,*], Zixuan Wang[a], Feng Ye[b], Xuejiao Chen[c]

[a] School of Modern Posts, Nanjing University of Posts & Telecommunications, Nanjing, China
[b] Department of Electrical & Computer Engineering, University of Dayton, Dayton, OH, USA
[c] Department of Communication, Nanjing College of Information Technology, Nanjing, China

ARTICLE INFO

Keywords:
Encrypted traffic classification
Generative Adversarial Network
Semi-supervised learning
Traffic identification
SDN gateway

ABSTRACT

With the rapid development of communication network technology, the types and quantity of network traffic data are accordingly increasing. Network traffic classification has become a non-trivial research task in the area of network management and security, which not only helps to improve the fine-grained network resource allocation but also enables policy-driven network management. As the closest network element to the subscribers, SDN Edge Gateway can tremendously enhance the user experience with the capability of traffic classification. Deep Learning (DL) performs automatic feature extraction without human intervention, which undoubtedly makes it a highly desirable approach for traffic classification, especially encrypted traffic. However, capturing large labeled datasets is cumbersome and time-consuming manual labor. Semi-Supervised learning is an appropriate technique to overcome this problem. With that in mind, we proposed a Generative Adversarial Network (GAN)-based Semi-Supervised Learning Encrypted Traffic Classification method called *ByteSGAN* embedded in SDN Edge Gateway to achieve the goal of traffic classification in a fine-grained manner to further improve network resource utilization. ByteSGAN can only use a small number of labeled traffic samples and a large number of samples to achieve a good performance of traffic classification by modifying the structure and loss function of the regular GAN discriminator network in a semi-supervised learning way. Based on the public datasets 'ISCX2012 VPN-non VPN' and 'Crossmarket', two experimental results show that the ByteSGAN can efficiently improve the performance of traffic classifiers and outperform the other supervised learning methods like CNN.

1. Introduction

With the rapid development of Communication Networks, the types and quantity of network traffic data are accordingly increasing. Network traffic classification has become a non-trivial research task in the area of network management and security. It is undoubtedly the cornerstone of dynamic access control, network resources scheduling, QoS provisioning, intrusion and malware detection, etc. High efficient and accurate traffic classification is of great practical significance to provide service quality assurance, dynamic access control, and abnormal network behaviors detection. Nonetheless, with the widespread adoption of encryption techniques for the network including 5G and IoT, the growth of a portion of encrypted traffic has dramatically arisen a huge challenge for network management like QoS provisioning. Therefore, accurate and robust encrypted traffic classification not only helps to improve the fine-grained network resource allocation but also enables policy-driven network management.

Software-defined networking(SDN) is envisioned as an emerging and promising networking paradigm with a promise to dramatically improve network resource utilization, simplify network management, reduce operating costs and promote innovation and evolution. Meanwhile, Edge Computing with its inherent nature of low latency, high bandwidth, and good privacy-preserving has attracted a lot of attention from both academics and industry. Intuitively, some researchers started to think of how to push the intelligence to the network edge and exploit the benefit of SDN. Thus, the combination of SDN and Edge Computing gives rise to SDN Edge Gateway (SDN-EGW), which can leverage both SDN at its global visibility of network-wide, programma-

☆ This document is the results of the research project funded by the National Science Foundation.
☆☆ In this work we demonstrate a_b the formation Y_1 of a new type of polariton on the interface between a cuprous oxide slab and a polystyrene micro-sphere placed on the slab.
* Corresponding author.
E-mail addresses: wangpan@njupt.edu.cn (P. Wang), 2020070135@njupt.edu.cn (Z. Wang), fye001@udayton.edu (F. Ye), chenxj@njcit.cn (X. Chen).

https://doi.org/10.1016/j.comnet.2021.108535
Received 25 February 2021; Received in revised form 27 July 2021; Accepted 4 October 2021

List of abbreviations in alphabetical order

FF	Flow Features
TS	Time Series
PP	Packet Payload
TC	Traffic Classification
AE	Auto Encoders
CNN	Convolutional Neural Networks
RF	Random Forest
FC	Fully-Connected

bility for traffic control, and Edge Computing at its low latency and good privacy-preserving.

There are several traffic classifiers using flow statistics for encrypted traffic classification which can achieve good performance, whereas it needs handcrafted features by domain experts which are always laborsome. Deep Learning (DL) including GAN performs automatic feature extraction without human intervention, which undoubtedly makes it a highly desirable approach for traffic classification, especially encrypted traffic. Recent research work has demonstrated the superiority of DL methods in traffic classification [1], such as MLP [2], CNN [3–7], SAE [8], LSTM [9,10].

As above mentioned, previous research works mostly require large quantities of labeled datasets, namely, supervised learning, which has been the center of most research in DL. However, as we all know, capturing large labeled datasets is cumbersome and time-consuming manual labor. Hence, the necessity of creating models capable of learning from less data is increasing faster. With that in mind, semi-supervised learning is an appropriate technique to overcome this problem. Additionally, labeling traffic datasets is a non-trivial and cumbersome process. There are two labeling methods for traffic datasets, one is DPI-based and the other is scripts-based labeling which can mimic human behavior, respectively. Whereas, DPI-based labeling cannot handle traffic encryption, and scripts-based methods are not always accurate. In contrast to these limitations of labeling data, unlabeled data is always abundant and easily obtainable. Hence, How to use semi-supervised learning to leverage the readily available traffic data for accurate traffic classification appears to be particularly important.

In this paper, we proposed a Generative Adversarial Network (GAN)-based Semi-Supervised Learning Encrypted Traffic Classification method called ByteSGAN embedded in SDN Edge Gateway to achieve the goal of traffic classification in a fine-grained manner to further improve network resource utilization. ByteSGAN can use a small number of labeled traffic samples and a large number of unlabeled samples to achieve a good performance of traffic classification by modifying the structure and loss function of the regular GAN discriminator network in a semi-supervised learning way. Based on public datasets 'ISCX2012 VPN-non VPN' and 'Crossmarket', two experimental results show that the ByteSGAN can efficiently improve the performance of traffic classifier and outperform the other supervised learning methods like CNN.

The rest of this paper is organized as follows. Section 2 introduces preliminaries and related works of traffic classification and SDN Edge Gateway . Section 3 describes the principles of GAN and ByteSGAN. Section 4 illustrates the methodology of ByteSGAN. The experimental results are provided and discussed in Section 5. Section 6 concludes our work.

2. Related works

2.1. ML and DL based approach of traffic classification

Classical ML-based classification methods always rely on handcrafted flow features (FF), packet payload (PP) [11] or time-series (TS) [12] based on properties of the first packets of a flow as model input, such as packet length, inter-arrival time and flow duration to circumvent the problems of encrypted content and user's privacy [13]. There are a plethora of works focusing on ML algorithms during the last decades. In general, there are three learning strategies used: first is the supervised methods like decision tree, SVM, and Naive Bayes, second is unsupervised approaches like k-means and PCA [14] and the last one is semi-supervised methods combining the advantages of the first two. Nevertheless, many drawbacks hindered ML-based methods widely applied to traffic classification, such as handcrafted traffic features driven by domain-expert, time-consuming, unsuited to automation, rapidly outdated when compared to the evolution. Unlike most traditional ML algorithms, Deep Learning performs automatic feature extraction without human intervention, which undoubtedly makes it a highly desirable approach for traffic classification, especially encrypted traffic. Recent research work has demonstrated the superiority of DL methods in traffic classification [1–10,15]. The workflow of DL based classification usually consists of three steps, which has been demonstrated in our previous work [16]. First, model inputs are defined and designed according to some principles, such as raw packets or flow statistics features. Second, models and algorithms are elaborately chosen according to the models' characteristics and the aim of the classifier. Finally, the DL classifier is trained to automatically extract the features of traffic.

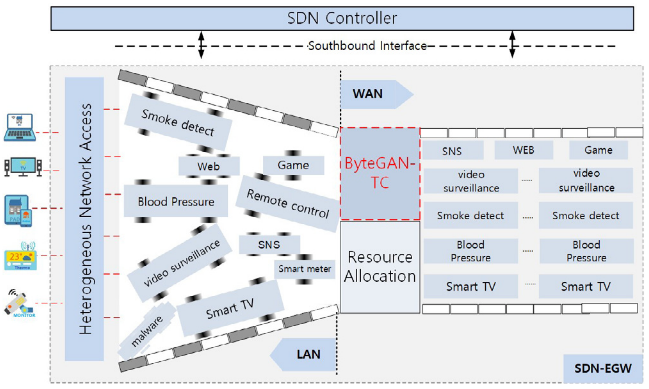


Fig. 1. The scheme of traffic classification over SDN-EGW

As shown in Fig. 1, several kinds of smart devices (Blood pressure monitors/video surveillance/Smart TV, etc.) connect to the SDN-EGW via heterogeneous access technology (WiFi/ZigBee/Bluetooth, etc.). Each smart applications have different QoS requirements, however, it is challenging that no fine-grained traffic classification capability in SDN-EGW, especially encrypted applications.

To overcome the above-mentioned shortcomings of the home network, we proposed a GAN-based Semi-Supervised Learning Encrypted Traffic Classification method called ByteSGAN embedded in SDN Edge Gateway. Our contributions are as follows:

1. ByteSGAN can take full advantage of a small number of labeled traffic samples and a large number of unlabeled ones to achieve a good performance;
2. BytesSGAN can handle not only regular traffic but also encrypted one;
3. BytesSGAN is a lightweight classifier that can only consume lesser computing resources of SDN-EGW;

2.2. Semi-supervised learning based methods of traffic classification

A similar research work about semi-supervised-based TC methods is shown in Table 1. As stated earlier, FF, PP, and TS are three possible input patterns of the classical ML/DL TC model, semi-supervised-based

Table 1
Related works: Semi-supervised learning based traffic classification methods.

Works	Technique	Model input	Dataset	Target classes	Performance	ML/DL
2007 [17]	Clustering : K-Means	FF + TS	Private	29	≈93%	ML
2016 [18]	RF + Hierarchical clustering	TS	Private	7	≥90	ML
2019 [19]	CNN	TS	Public	5	≥84	DL
2020 [20]	DCGAN	TS	Public	15	≥89	GAN
2020 [21]	AE	FF	Private	78	≥89	DL
2020 [22]	CGAN + CNN	PP	Public	15	≥90	GAN

methods are not an exception. Before DL-based methods are becoming popular, clustering-based algorithms are always used on an unlabeled dataset and traditional ML classifiers like RF or SVM on a few labeled ones [17,18]. In [19], the authors pre-train a CNN model on a large unlabeled dataset where the input is the TS features of a few sampled packets. Then the learned weights are transferred to a new model that is re-trained on a small labeled dataset. As an unsupervised model, AE is well suited to learn the latent features and is always used for pretraining TC tasks. In [21], FF are fed into AE for pretraining. Then they retrain this AE model concatenated with a simple FC on a few labeled datasets in a semi-supervised manner. As a generative model, GAN usually plays a generator role no matter in the area of computer vision, image recognition, or NLP. In our previous work [22], GAN was explored to synthesize the traffic samples to mitigate the problem of class imbalance. Besides this, the work of [20] further explored the discriminator to act as a multi-class classifier to achieve the goal of TC.

3. Generative Adversarial Networks (GAN)

GAN provides a promising way to learn deep representation without extensively annotated training data. They achieve this through deriving backpropagation signals through a competitive process involving a pair of networks. GAN can be applied in a variety of applications like image synthesis, style transfer, semantic image editing, which are gaining more and more attention from both academics and industry.

3.1. GAN

A regular GAN consists of two parts, the generator G and the discriminator D. As shown in Fig. 2, The role of the generator is to take random noise as input by learning the characteristic distribution of real data. The discriminator aims at determining whether the data is real or generated by G. The generator G simulates the feature distribution P_g of the real data by the prior distribution $P_z(z)$. The input of the discriminator is the real and generated data, correspondingly, the output $D(x)$ indicates the probability of whether the input data is real or not [23]. During the training process, G and D play a two-player mini–max game until D cannot judge whether the sample data is real, which means that the two networks reach the Nash Equilibrium. The objective function of GAN can be expressed by (1):

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

In Eq. (1), $P_{data}(x)$ represents distribution of the real data. When training D, the goal is to optimize the probability of TRUE $D(G(z))$ as small as possible and the probability of TRUE $D(x)$ of the real data x as much as possible. When training G, the goal is to make $D(G(z))$ as much as possible. From (1), we can calculate the optimal discriminator as (2). As can be seen from (2) below, when $P_{data}(x) = P_z(z)$, it means that D cannot distinguish whether the sample is true or false, D and G reach the Nash Equilibrium, and the discriminator output is 0.5.

$$D(x) = \frac{P_{data}(x)}{P_{data}(x) + P_z(z)} \tag{2}$$

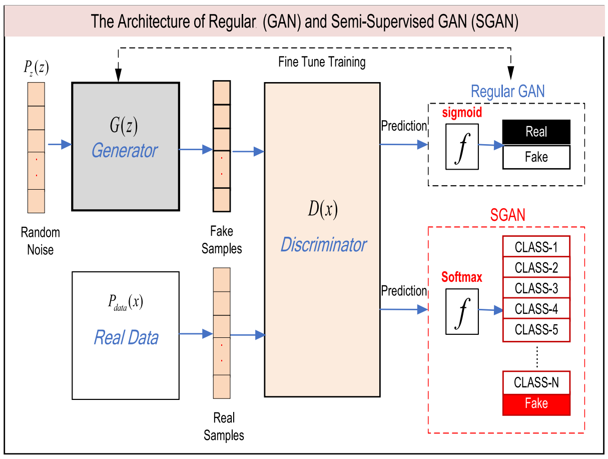


Fig. 2. The Basic Architecture of GAN.

3.2. Semi-Supervised GAN (SGAN)

Semi-Supervised GAN, so-called SGAN is an extension of GAN to the semi-supervised context by forcing the discriminator network to output class labels. A generative model G and a discriminator D are trained on a dataset with inputs belonging to one of N classes. During training time, D is made to predict which of N + 1 classes the input belongs to. Besides this, an extra class referring to REAL or FAKE is added to the outputs of G. The study of [24] showed that this method can be used to create a more data-efficient classifier than a regular GAN.

As shown in Fig. 2, the traditional discriminator network D always outputs an estimated probability that the input samples is fake. This process is typically implemented with a feed-forward network ending in a sigmoid unit, however, if we implement this with a softmax output layer with one unit for each of the classes [REAL, FAKE], apparently D could have N + 1 output units corresponding to [CLASS-1, CLASS-2, …CLASS-N, FAKE]. In this case, D can also act as a classifier, namely C. We can call this network D/C. While training an SGAN, it is similar to a GAN except that D/C is trained to minimize the negative log-likelihood with regard to the given labels and G is trained to maximize it.

4. The methodology of ByteSGAN encrypted traffic classification

As we all know, obtaining large labeled datasets is cumbersome and time-consuming manual labor. In contrast to these limitations, unlabeled data is always abundant and easily obtainable. It is particularly important that how to use semi-supervised learning to leverage the readily available unlabeled traffic data for accurate traffic classification. As we all know, GAN provides a promising way to learn deep representation without extensively annotated training data. Intuitively, we attempt to explore the capability of the GAN-based encrypted traffic classification in a semi-supervised manner.

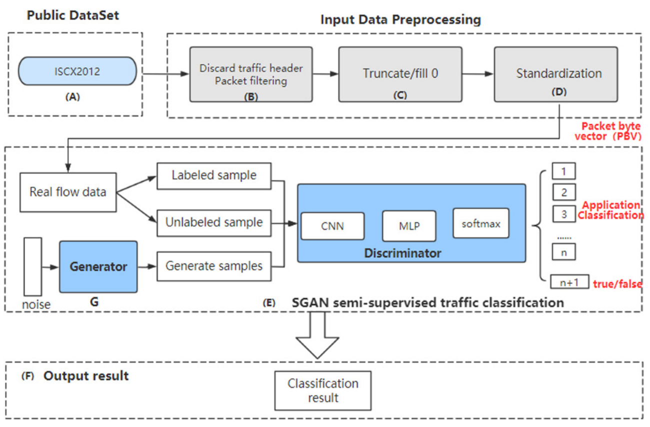


Fig. 3. The Workflow of ByteSGAN.

4.1. The workflow of ByteSGAN

As shown in Fig. 3, ByteSGAN mainly includes two steps: Input data pre-processing and ByteSGAN based traffic classification. First, we have to filter some packet data in the dataset like ARP/DHCP/ICMP which is not relevant to specific applications. In addition, packet data will be truncated or padding with zero to normalize to a Packet Byte Vector (PBV) [2]. Then the standardized PBV will be sent to ByteSGAN, namely SGAN semi-supervised traffic classification model as input. It is worth noting that the label and data in the PBV are both fed into the discriminator network D as real data. Then the generator network G and discriminator network D are alternately trained and iteratively updated to converge and output the classification results.

In ByteSGAN, multiple traffic samples can be classified and trained at the same time, which effectively solves the problem that a regular GAN has to be trained separately for each application. Moreover, we can easily overcome the limitation of the labeled data using Semi-Supervised Learning.

4.2. Loss function

The discriminator D of ByteSGAN is an N + 1 dimensional classifier. Its input is a packet data sample and output is an N + 1 dimensional vector, which can be represented as class probability, where formula (3) means the x is false, instead, formula (4) means the x is true as well as it belongs to the class i.

$$P_{\text{model}}\ (y = N+1 \mid x) = \frac{\exp\left(c_{N+1}\right)}{\sum_{j=1}^{N+1} \exp\left(c_j\right)} \tag{3}$$

$$P_{\text{model}}\ (y = i \mid x, i < N+1) = \frac{\exp\left(c_i\right)}{\sum_{j=1}^{N+1} \exp\left(c_j\right)} \tag{4}$$

4.3. Training process

The training of ByteSGAN uses a combination of supervised and unsupervised ways, which can significantly improve the learning ability of the model. Specifically, during the semi-supervised training, the discriminator D and the generator G are trained alternately and the parameters are updated iteratively. On one hand, while training the generator, feature learning will be used to make the generated traffic as close to the real data distribution as possible. On the other hand, while training the discriminator, the network parameters are iteratively updated to minimize the cross-entropy of the labeled traffic sample and the predicted probability distribution. The network model parameters of samples and generated samples are updated by adversarial training. As shown in Alg. 1, the training steps are as follows:

1. Generate a random vector with Gaussian noise z and input it to the generator network G, then get the generated traffic $G(z)$;
2. Input both the generated traffic $G(z)$, labeled and real traffic x into the discriminator network D in batches, then output the sum of probability of $D(x)$ and $D[G(z)]$ by the activation function softmax;
3. Fix the parameters of the generator network G. If the input flow is generated flow (i.e. fake data), we can use $E_{x\sim G} \log \left[p_{\text{model}}\ (y = N+1 \mid x)\right]$ as the loss function; if the real flow x has been labeled, we can use L_{labeled} as the loss function; if the real flow x is not labeled, we can use L as the loss function. Adjust the parameters of the discriminator D by Adam Gradient Descent method;
4. Fix the parameters of the discriminator network D, perform feature matching operations on the real traffic data x and the generated traffic $G(z)$, then select the output of the fully connected layer as the middle layer and use feature matching to adjust the parameters of the generator network G;
5. Repeat steps (1)–(4) until the number of iterations is completed.

4.4. Network structure of ByteSGAN

We referred to the network structure of DCGAN [25] and the definition of the semi-supervised model in the paper of Tim Salimans et al. [26] to adjust the input and output, the network structure of generator and discriminator of ByteSGAN to make model training more stable. The detailed adjustments are as follows:

Algorithm 1 The model training algorithm of ByteSGAN

```
Require: Real labeled sample $X$ after data preprocessing
Ensure: Accuracy of traffic classification
 1: for number of training iterations do
 2:   for $k$ steps do
 3:     Take a minibatch from the $m$ noise samples $\left\{z^{(1)}, \ldots, z^{(m)}\right\}$ whose noise prior distribution is $p_g(z)$ .
 4:     Take a minibatch from the $m$ training samples $\left\{x^{(1)}, \ldots, x^{(m)}\right\}$ with data distribution $p_{\text{data}}(x)$.
 5:     If the real traffic is not marked, use $L$ as a loss function:
 6:     Update the discriminator through Adam gradient descent.
 7:     If the real traffic $x$ has been marked, use $L_{\text{labeled}}$ as a loss function:
 8:     Update the discriminator through Adam gradient descent:
 9:     $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[-\log\left[p_{\text{model}}\ \left(y = i \mid x^{(i)}, i < N+1\right)\right]\right]$
10:     If the input flow is generated flow, use $L_g$ as a loss function:
11:     Update the discriminator through Adam gradient descent:
12:     $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left[p_{\text{model}}\ \left(y = N+1 \mid x^{(i)}\right)\right]\right]$
13:   end for
14:   Take a minibatch from $m$ noise $p_z(z)$ samples whose prior distribution of noise is $\left\{z^{(1)}, \ldots, z^{(m)}\right\}$ .
15:   Update generator.
16: end for
```

1. The convolutional layer with strides is used instead of the pooling layer.
2. The discriminator network D adopts convolution instead of the pooling layer and the generator network G model employs deconvolution instead of the pooling layer.
3. LeakyReLU [27] is used for activation in both the generator G and the discriminator D. In addition, the generator G's output layer adopts the tanh as the activation function according to the paper [26].
4. The output layer of the discriminator D is built as a stacked model with shared weights. First, build a supervised model by using k classes of outputs and a softmax activation function. Then define the unsupervised model, which accepts the output of

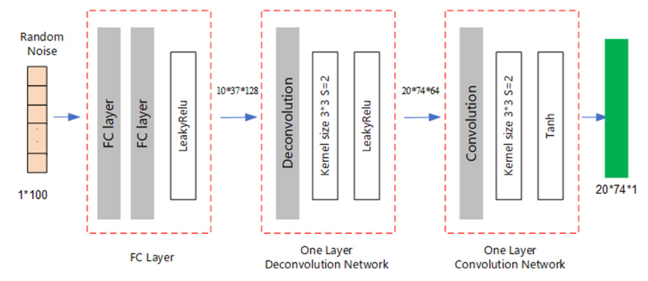


Fig. 4. The Structure of Generative Network Model.

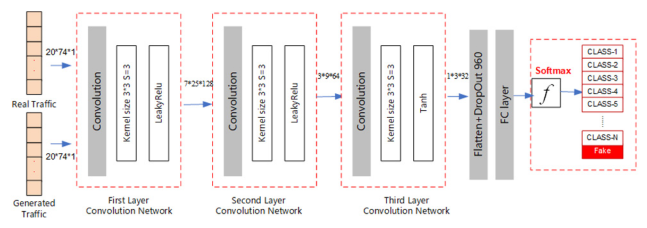


Fig. 5. The Structure of Discriminator Network Model.

the supervised model before the softmax activation function, and then calculates the normalized sum of the exponential output.

$$D(x) = \frac{Z(x)}{Z(x)+1}, \text{ where } Z(x) = \sum_{k=1}^{K} \exp\left[l_k(x)\right] \quad (5)$$

The output of the unsupervised model before the softmax activation function are all very small positive or negative values. When using formula (5), we will get the output of (0.0-1.0), which means that the model is encouraged to output strong predictions for real samples and small predictions or low activations for fake samples.

4.5. *The generator model structure*

The generator G is shown in Fig. 4. G is constructed by a One Layer Deconvolution Network and a One Layer Convolution Network. Firstly, we will input a 100-dimensional random noise z with the Gaussian distribution into the FC network and make it into a three-dimensional tensor by dimensional transformation (i.e. reshape). Then we feed this tensor into the deconvolution layer with a kernel size of 3*3 and stride of 2, which is activated by LeakyReLU. Finally, the output of the deconvolution layer will be fed into a convolution layer with a kernel size of 3*3 and stride of 2, using Tanh activation to generate a traffic sample tensor of (20*74*1).

4.6. *The discriminator model structure*

As shown in Fig. 5, the discriminator D is composed of 3 convolutional layers and 2 FC layers. The real traffic sample of PBV with a length of 1480 after input data preprocessing is transformed to a three-dimensional tensor (20*74*1) by dimensional transformation (i.e. reshape), then sent to a 3-layers convolution kernel with a size of 3* 3. LeakyReLU is used for activation after each convolution. LeakyReLU can retain a small slope in the negative half axis (in this paper it is set to 0.2). Compared with the ReLU activation function, LeakyReLU can avoid the problem of the gradient vanish during training. After flattening through the Flatten layer, it is input to the FC network which is built as a stacked network with shared weights and adopts Lambda as activation function and then employs softmax to output normalized categories probabilities.

Table 2
Description of the chosen datasets from ISCX.

Application	Security	Dataset samples	
	Protocol	Quantity	Percentage
AIM	HTTPS	4869	2.356%
Email-Client	SSL	4417	2.137%
Facebook	HTTPS	5527	2.674%
Gmail	HTTPS	7329	3.546%
Hangout	HTTPS	7587	3.671%
ICQ	HTTPS	4243	2.053%
Netflix	HTTPS	51932	25.126%
SCP	SSH	15390	7.446%
SFTP	SSH	4729	2.287%
Skype	proprietary	4607	2.229%
Spotify	proprietary	14442	6.987%
torTwitter	proprietary	14654	7.089%
Vimeo	HTTPS	18755	9.074%
voipbuster	proprietary	35469	17.161%
Youtube	HTTPS	12738	6.163%
TOTAL		**206688**	**100%**

5. Evaluation and experimental results

5.1. *Experimental settings*

5.1.1. *Dataset for evaluation*

The experiments in this chapter are based on two public datasets, "ISCX VPN-nonVPN 2012" [28] and "crossmarket" [1], both of which are composed of PCAP format. The former contains many encryption applications or protocols such as HTTPS, SFTP, Facebook, Hangouts, etc. In addition to these applications encapsulated in regular sessions, some VPN tunnel applications were also captured in this dataset. 15 applications were chosen to build the original dataset as shown in Table 2. The latter dataset contains the 100 most popular apps in mobile app markets of a combination of three regions (US, China, India) and mobile OS (iOS, Android) on August 2017. All the traffic was collected by manually interacting with a given app for five minutes and test all the main features. As shown in Table 3, We selected 20 common iOS and Android apps from the Chinese region of the dataset, including social software, music, video, instant messaging, etc. In order to keep the balance of the dataset, we randomly select 30,000 flow records from the PCAP files of each application as a subset of the original dataset for subsequent experiments.

5.1.2. *Experimental tasks*

Based on the two datasets, we carried out two experiments as follows:

1. In the first experiment, we use the same number of labeled samples and different numbers of unlabeled samples to perform classification tasks on two selected datasets to verify the impact of the number of samples on the ByteSGAN and AE classification results.
2. In the second experiment, we use a different number of labeled samples to perform classification tasks on two datasets by ByteSGAN and CNN respectively, to verify the improvement of classification accuracy of ByteSGAN with a small number of labeled samples.

5.1.3. *Configurations of the computing platform*

The experimental environmental parameters are shown in Table 4. The performance evaluations are conducted using a Dell R730 server with an Intel I7-7600U CPU 2.8 GHz, 16 GB RAM, and an external GPU (Nvidia GeForce GTX 1050TI). The software platform for deep learning is built on Keras library with Tensorflow (GPU-based version 1.13.1) as the back-end support.

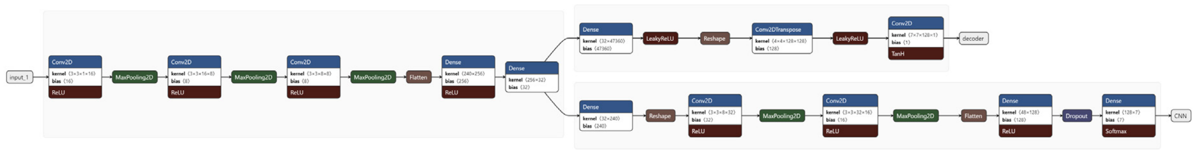


Fig. 6. AE based traffic classification model structure.

Table 3
Description of the chosen datasets from Cross-market.

Application name	Platform	Application type	Selected quantity
Meiyanxiangji	IOS	Camera	30000
Qiuqiudazuozhanbob	IOS	Game	30000
QQ	IOS	Instant messaging	30000
YY	IOS	Instant messaging	30000
QQkongjian	IOS	Social software	30000
Baidutieba	IOS	Social software	30000
Weibo	IOS	Social software	30000
Tengxunshipin	IOS	Video	30000
Aiqiyi	IOS	Video	30000
Kuaishou	IOS	Video	30000
Total			300000
Tencent	Android	Instant messaging	30000
Xiami	Android	Music	30000
Tingshu	Android	Music	30000
Sohu	Android	News	30000
Qzone	Android	Social software	30000
Duowan	Android	Social software	30000
Tongcheng	Android	Social software	30000
QQlive	Android	Video	30000
Youku	Android	Video	30000
Letv	Android	Video	30000
Total			300000

Table 4
Experimental environment parameters.

Category	Parameters
GPU	Nvidia GPU(GeForce GTX 1050Ti)
Operating system	Win 10
Deep learning platform	TensorFlow 1.13.1 + Keras 1.0.7
CUDA version	9.0
CuDNN version	7.6.0

Table 5
The first experimental results on ISCX dataset.

	Number labeled	Number unlabeled	Accuracy	precision	recall	f1-score
SGAN	1000	4000	98.79%	98.79%	98.77%	98.56%
	1000	6000	99.04%	99.03%	99.03%	98.95%
	1000	8000	99.14%	99.16%	99.15%	99.06%
AE	1000	4000	97.24%	97.27%	97.24%	97.10%
	1000	6000	97.57%	97.57%	97.57%	97.41%
	1000	8000	98.18%	98.20%	98.18%	98.02%

Table 6
The first experimental results on Cross-market dataset.

	Number labeled	Number unlabeled	Accuracy	Precision	Recall	f1-score
SGAN	100	4000	89.32%	89.46%	89.49%	89.39%
	100	8000	93.57%	93.61%	93.57%	93.52%
	100	12000	94.93%	94.91%	94.96%	94.91%
	1000	4000	98.00%	98.03%	98.01%	98.01%
	1000	8000	98.12%	98.18%	98.13%	98.14%
	1000	12000	98.60%	98.60%	98.62%	98.61%
AE	100	4000	83.70%	83.77%	83.70%	83.67%
	100	8000	90.30%	90.31%	90.30%	90.28%
	100	12000	93.00%	93.04%	93.00%	92.97%
	1000	4000	97.28%	97.28%	97.28%	97.28%
	1000	8000	98.08%	97.28%	97.28%	97.28%
	1000	12000	98.45%	98.45%	98.45%	98.45%

5.2. The first experiment

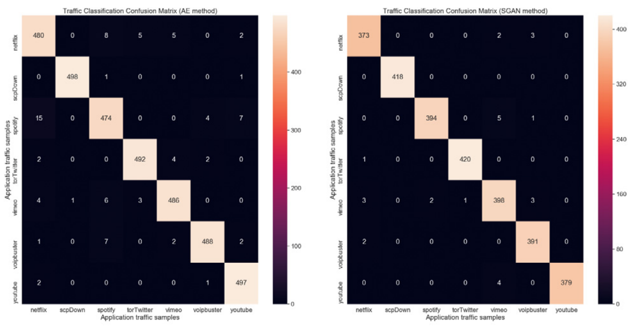
To compare with the proposed SGAN, we designed and implemented a semi-supervised classification model based on AE. The specific structure is shown in Fig. 6. The model includes two parts: autoencoder and classification model. The autoencoder consists of two main components, the encoder and the decoder. The encoder is used as a feature extractor, and as the data goes to deeper layers, it detects features from low-level to high-level. The extracted features are stored in latent vectors. The decoder takes these features and converts them back to the original shape, enters the flow data, and tries to reconstruct the original flow. We use a CNN network as a classifier to receive features from the encoder for classification. The autoencoder is an unsupervised learning model, and we combine it with the CNN classifier to form a semi-supervised learning model.

1. The input layer data adopts the Packet Byte Vector (PBV) with (20, 74, 1) as input.
2. The encoder uses a three-layer convolutional neural network. Each layer has a convolution kernel with the shape of (3, 3). The encoder receives PBV and outputs the feature vector with the shape of (32,1)
3. The decoder is designed to have the same structure as the SGAN generator, using the feature vector of shape (32, 1) to restore the original traffic
4. The encoder is also connected to a classifier. The classifier has two layers, the shape of the convolution kernel is (3, 3), and the softmax activation function is used to output the classification results.
5. Use unlabeled samples to train the encoder and decoder and set the loss function to MAE. Use labeled samples to train encoders and classifiers, and use categorical cross-entropy as the loss function

On the selected data set, 1000 labeled samples are used at the same time, and a different number of unlabeled samples are used for three experiments. The stochastic gradient descent method is used for training, and the batch parameter is 256. The noise is randomly sampled from the uniform distribution of [−1, 1], and the sample size is 100. The learning rate parameter is 0.0002, and the Adam optimizer is used to optimize the loss function of the two models. The results of the three experiments are shown in Table 5:

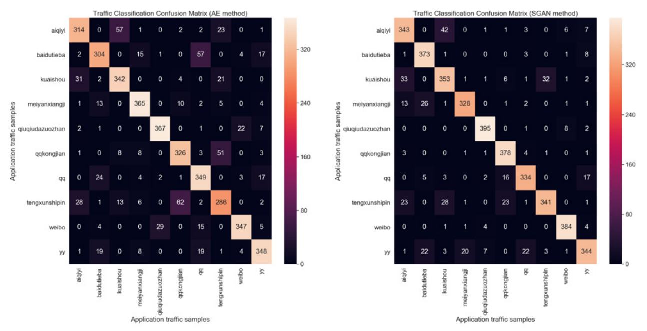
As shown in Table 5, It can be found that both models can use a small amount of labeled data to achieve high accuracy. Although there is little difference, the three indicators of ByteGAN are both 1% higher than AE under the same conditions. We can also found that increasing the number of samples can improve the classification accuracy of ByteSGAN when the number of labeled samples is constant as shown in Fig. 7.

As shown in Table 6, it can be found that in the case of less unlabeled data, especially when the number of labeled samples is 100 and the number of unlabeled samples is 4000, ByteSGAN performs better on the Cross-market dataset than AE Up to about 6%. Through the confusion matrix, it is also obvious that AE has a large error in the identification of video application (kuaishou/aiqiyi/tengxunshipin) traffic, while ByteGAN has significantly improved as shown in Fig. 8.



(a) AE method (b) ByteSGAN method

Fig. 7. Comparison of Confusion Matrix on ISCX dataset (1000 labeled sample and 6000 unlabeled sample).



(a) AE method (b) ByteSGAN method

Fig. 8. Comparison of Confusion Matrix on Cross-market (IOS) dataset (100 labeled sample and 4000 unlabeled sample).

5.3. The second experiment

5.3.1. Experimental description

To compare with the classification experiment, the data set as shown in Tables 2 and 3 was used to conduct experiments on both the CNN-based and the ByteSGAN-based classification model. Before the experiment, select a specified number of labeled data from the dataset and the rest as unlabeled data for the experiment. We adopt the stochastic gradient descent method for training and the batch parameter is 256. The noise is randomly sampled from the uniform distribution of [−1, 1], and the sample size is 100. The learning rate parameter is 0.0002, and the Adam optimizer is used to optimize the loss function of the generator and the discriminator. For the sake of comparison, a simple CNN-based classification model is designed and implemented and its network structure is shown in Fig. 9.

The CNN-based encrypted traffic classification model is composed of an input layer, three convolution and pooling connection layers, and an output layer. The input layer is packet data after the preprocessing. The specific parameters of the convolution kernel pooling layer are shown in the figure. The output layer adopts softmax classification. The neurons of the output layer represent 15 different applications. The detailed model structure is as follows:

1. The input layer data adopts the Packet Byte Vector (PBV) with (1480, 1) as input.
2. The convolution layer in the first layer performs convolution operation uses 128 filters with a size of 5 and a step size of 1. The output result is calculated by the activation function ReLU (Shaped Linear Unit).
3. Max-pooling is adopted with a size of 5 and stride of 1.

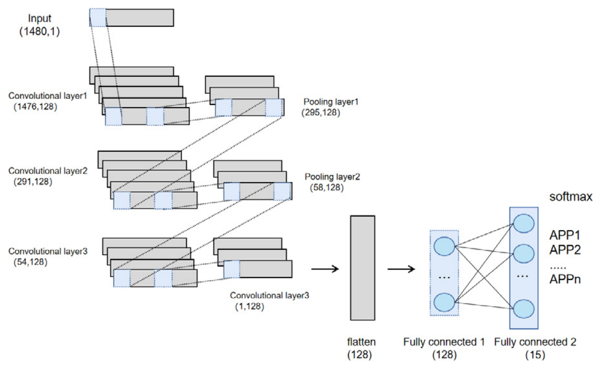


Fig. 9. CNN based traffic classification model structure.

4. The convolution layer in the second layer of convolution and pooling layer uses 128 filters with a size of 5 and a stride of 1. The output result is calculated by ReLU.
5. The parameters used by the pooling layer in the second layer are the same as those in step (3).
6. The convolution layer in the third layer uses 128 filters with a size of 5 and a stride of 1. The output result is calculated by ReLU.
7. The pooling layer in the third layer employs maxi-pooling, with a size of 35 and a stride of 1.
8. Through two fully connected layers, the network is transformed into a one-dimensional vector for subsequent classification.
9. The output layer uses softmax for classification with 15 output units corresponding to 15 different encrypted traffic applications

On the selected dataset, a small batch stochastic gradient descent method is used for training, and the batch parameter is 128. Use Rmsprop optimizer for the cross-entropy loss function. In this paper, we use ByteSGAN to perform classification experiments under the condition that there are only 1000, 2000, 3000, and 4000 labeled data in each class sample and compares the classification results with CNN under the same conditions to prove the superiority of ByteSGAN in semi-supervised encrypted traffic classification.

5.3.2. Performance evaluation

Fig. 10 shows the trend of the adversarial loss on the ByteSGAN discriminator and generator. The adversarial loss of the discriminator gradually decreases, and that of the generator gradually rises, and both of them keep stable around 2000 epochs.

To better demonstrate the superiority of ByteSGAN, we selected four different DL-based methods to make a comparison of performance over ISCX datasets. There are three supervised-based methods and two semi-supervised-based methods. As shown in Table 9, all the three supervised-based methods slightly perform better than our method. Meanwhile, we can see that ByteSGAN can outperform the DCGAN-based method [20]. It is worth noting that we selected 1000 labeled samples and 8000 unlabeled ones from the dataset, respectively. From the comparison results, we can see that the performance of ByteSGAN is very close to the supervised-based ones.

We conducted an evaluation experiment for the computing resource consumption, throughput, and training speed of the model. The experiment environment is show in Table 4, we used the above-mentioned CNN, SAE, and ByteSGAN model to classify 10,000 packets. The performance parameters of the experiment are shown in Table 7. It can be seen that the proposed ByteSGAN has advantages in model parameters, bandwidth, and computing resource consumption. At the same time, the use of CPU and memory can be maintained at the same level as the lightweight CNN model.

Table 7
Performance evaluation.

	CNN	SemiAE	ByteSGAN
Modelsize	1.55 MB	22.1 MB	3.87 MB
Parameters	199050	1,925,691	334,858
Per-step speed	600 us/step	530 us/step	380 us/step
Throughput	97 Mbps	81 Mbps	130 Mbps
cpu usage	9%~11%	10%~12%	10%~11%
Memory usage	4194.40 MB	5570.56 MB	4259.84 MB

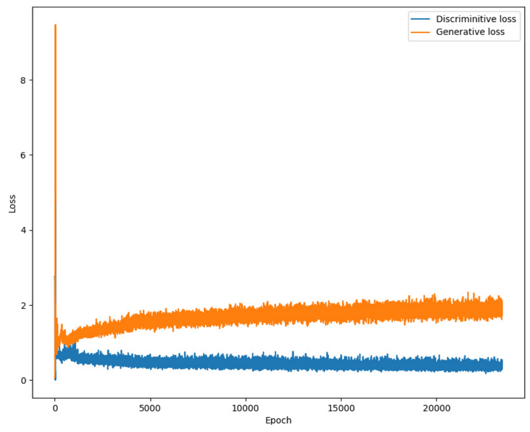


Fig. 10. ByteSGAN adversarial training loss.

Table 8
Results on ISCX dataset.

	Number of labeled	ACC	Precision	Recall	f1-score
CNN	1000	88.25%	88.87%	88.24%	88.42%
	2000	89.60%	89.91%	89.51%	89.54%
	3000	92.40%	92.43%	92.42%	92.36%
	4000	93.33%	93.73%	93.29%	93.41%
SGAN	1000	92.15%	92.13%	92.16%	92.07%
	2000	92.92%	93.62%	92.82%	92.95%
	3000	93.10%	94.26%	93.08%	93.21%
	4000	93.18%	93.49%	93.12%	93.18%

Table 9
Related DL-based Methods Comparison Results on ISCX.

Methods	Algorithm	ACC	Precision	Recall	F1-Score
1D-CNN [4]	CNN, supervised	99.34%	99.36%	99.35%	99.34%
DeepPacket [3]	SAE, supervised	99.01%	99.06%	99.05%	99.03%
DataNet [2]	MLP, supervised	99.68%	99.66%	99.65%	99.66%
DCGAN [20]	GAN, semi-supervised	93.02%	90.17%	91.51%	92.16%
ByteSGAN	GAN, semi-supervised	99.14%	99.16%	99.15%	99.06%

Table 10
Results on crossplatform Andorid dataset.

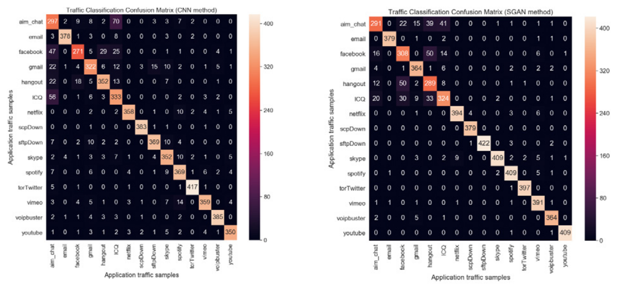
	Number of labeled	ACC	precision	recall	f1-score
CNN	1000	94.63%	94.62%	94.57%	94.57%
	2000	96.33%	96.38%	96.30%	96.32%
	3000	96.83%	96.81%	96.88%	96.83%
	4000	97.78%	97.79%	97.81%	97.79%
SGAN	1000	98.45%	98.48%	98.44%	98.46%
	2000	98.52%	98.54%	98.53%	98.53%
	3000	98.68%	98.70%	98.67%	98.68%
	4000	98.85%	98.86%	98.84%	98.84%

5.3.3. Classification result

We employ ByteSGAN and CNN to perform classification experiments when the number of labeled samples is 1000, 2000, 3000, 4000, and the classification results are as follows:

Table 11
Results on crossplatform IOS dataset.

	Number of labeled	ACC	Precision	Recall	f1-score
CNN	1000	96.03%	96.08%	96.03%	96.04%
	2000	97.38%	97.38%	97.37%	97.36%
	3000	98.12%	98.14%	98.15%	98.14%
	4000	98.85%	98.86%	98.85%	98.85%
SGAN	1000	98.25%	98.25%	98.23%	98.23%
	2000	98.42%	98.87%	98.44%	98.63%
	3000	98.80%	98.80%	98.81%	98.80%
	4000	99.00%	99.00%	99.00%	99.00%



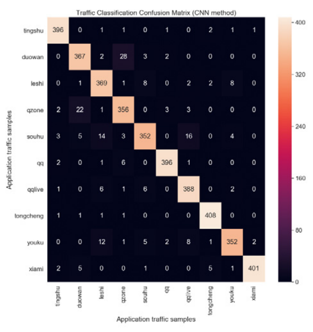
(a) CNN method (b) ByteSGAN method

Fig. 11. Comparison of Confusion Matrix on ISCX dataset (1000 labeled sample).

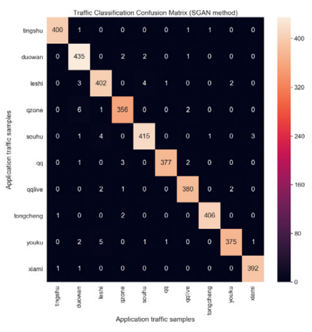
Through Table 8, the experimental result shows that when the number of labeled samples is 1000, the classification accuracy of ByteSGAN is improved by about 4% compared to CNN. When the number of labeled samples is 2000, the classification accuracy of ByteSGAN is improved by about 3%. And when the number is 3000, ByteSGAN's classification accuracy is improved by less than 1%. When the number of labeled samples is 4000, the classification accuracy is the same as that of CNN. It can be seen that when the number of labeled samples is insufficient, ByteSGAN can effectively improve the classification accuracy by using unlabeled samples and samples generated by ByteSGAN to perform semi-supervised learning. The accuracy is still not a good reflection of the classification performance of ByteSGAN. Below we use the evaluation metrics to analyze the improvement of each application more comprehensively. As shown in Fig. 11, except Facebook and hangout, ByteSGAN's accuracy is better than that of CNN in 1000, 2000, and 3000 samples. Among them, the improvement of aim chat is higher and the improvement is close to 20%. The increasing rate of email, Gmail, and ICQ is close to 5% and other applications such as Netflix, scpdown, skype, Spotify, torTwitter, voipbuster, youtube also have a small increase.

Through Tables 10 and 11, Experiments show that ByteSGAN can show good classification accuracy in the face of both Android applications and IOS application traffic, especially in IOS application classification, using 4000 labeled samples can achieve 99% classification accuracy. When there are few labeled data, such as only 1000 labeled samples, ByteSGAN's precision, recall, and F1 score are about 4% higher than CNN's, which proves that semi-supervised learning is helpful to improve the classification accuracy. Compared with the results on the ISCX dataset, ByteSGAN can effectively improve the classification accuracy of the classification model in the case of insufficient labeled samples.

As shown in Figs. 12 and 13, it can be found that since DuoWan and Qzone are both social software, Kuaishou and aiqiyi are both video software, the traffic of application transmission is similar. Due to the small number of labeled samples, when CNN performs supervised learning, misrecognition occurs, and the misrecognition rate reaches 5% However, ByteSGAN takes advantage of unlabeled samples, and the false recognition rate of these two types is less than 1%.

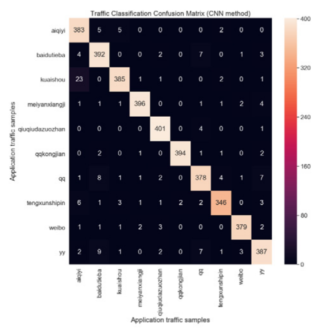


(a) CNN method

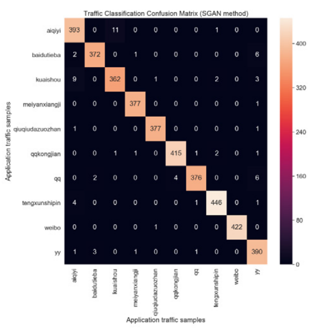


(b) ByteSGAN method

Fig. 12. Comparison of Confusion Matrix on Cross-market (Android) dataset (1000 labeled sample).



(a) CNN method



(b) ByteSGAN method

Fig. 13. Comparison of Confusion Matrix on Cross-market (IOS) dataset (1000 labeled sample).

6. Conclusion and future work

As we all know, capturing large labeled datasets is cumbersome and time-consuming manual labor. Semi-Supervised learning is a desirable learning way to alleviate this problem. Motivated by this idea, we proposed a Generative Adversarial Network (GAN)-based Semi-Supervised Learning Encrypted Traffic Classification method called *ByteSGAN* embedded in SDN Edge Gateway to achieve the goal of traffic classification in a fine-grained manner to further improve network resource utilization. ByteSGAN can use a small number of labeled traffic samples and a large number of ones to achieve a good performance of traffic classification by modifying the structure and loss function of the regular GAN discriminator network in a semi-supervised learning way. Based on the public dataset 'ISCX2012 VPN-non VPN' and 'Cross-market', two experimental results show that the ByteSGAN can efficiently improve the performance of traffic classifier and outperform the other supervised learning methods like CNN. In the future, we will further explore the potential of the generative capability of GAN to improve the data imbalance problem for traffic classification.

However, mode collapse and instability of GAN training still perplex our works, we will improve them through the GAN training tricks. Besides, cross-validation as the traditional evaluation method is not the best choice for streaming data like real-time network traffic, we will try prequential evaluation technology to tackle the streaming traffic data, especially in the scenario of online learning.

CRediT authorship contribution statement

Pan Wang: Conceptualization, Methodology, Software. **Zixuan Wang:** Methodology, Software. **Feng Ye:** Data curation, Writing – original draft. **Xuejiao Chen:** Methodology, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The paper is sponsored by National Natural Science Foundation (General Program) Grant 61972211, China, and National Key Research and Development Project Grant 2020YFB1804700, China.

References

[1] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Mobile encrypted traffic classification using deep learning, in: 2018 Network Traffic Measurement and Analysis Conference, TMA, 2018, pp. 1–8, http://dx.doi.org/10.23919/TMA.2018.8506558.

[2] P. Wang, F. Ye, X. Chen, Y. Qian, Datanet: Deep learning based encrypted network traffic classification in sdn home gateway, IEEE Access 6 (2018) 55380–55391, http://dx.doi.org/10.1109/ACCESS.2018.2872430.

[3] Lotfollahi, Deep packet: A novel approach for encrypted traffic classification using deep learning, 2017, Available from http://www.arxiv.org.

[4] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: 2017 IEEE International Conference on Intelligence and Security Informatics, ISI, 2017, pp. 43–48.

[5] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware traffic classification using convolutional neural network for representation learning, in: 2017 International Conference on Information Networking, ICOIN, 2017, pp. 712–717, http://dx.doi.org/10.1109/ICOIN.2017.7899588.

[6] Z. Chen, K. He, J. Li, Y. Geng, Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks, in: 2017 IEEE International Conference on Big Data, Big Data, 2017, pp. 1271–1276, http://dx.doi.org/10.1109/BigData.2017.8258054.

[7] X. Chen, J. Yu, F. Ye, P. Wang, A hierarchical approach to encrypted data packet classification in smart home gateways, in: 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTech, 2018, pp. 41–45, http://dx.doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00022.

[8] Z. Wang, The application of deep learning on traffic identification, 2015, Available from http://www.blackhat.com.

[9] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Network traffic classifier with convolutional and recurrent neural networks for internet of things, IEEE Access 5 (2017) 18042–18050, http://dx.doi.org/10.1109/ACCESS.2017.2747560.

[10] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, M. Zhu, Hast-ids: Learning hierarchical spatial–temporal features using deep neural networks to improve intrusion detection, IEEE Access 6 (2018) 1792–1806, http://dx.doi.org/10.1109/ACCESS.2017.2780250.

[11] A.W. Moore, K. Papagiannaki, Toward the accurate identification of network applications, in: Proceedings of the 6th International Conference on Passive and Active Network Measurement, Springer-Verlag, Berlin, Heidelberg, 2005b, pp. 41–54, http://dx.doi.org/10.1007/978-3-540-31966-5_4.

[12] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, K. Salamatian, Traffic classification on the fly, SIGCOMM Comput. Commun. Rev 36 (2006) 23–26, http://dx.doi.org/10.1145/1129582.1129589.

[13] D.C. Sicker, P. Ohm, D. Grunwald, Legal issues surrounding monitoring during network research, in: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, ACM, New York, NY, USA, 2007, http://dx.doi.org/10.1145/1298306.1298307, URL http://doi.acm.org/10.1145/1298306.1298307.

[14] T.T.T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tutor. 10 (2008) 56–76, http://dx.doi.org/10.1109/SURV.2008.080406.

[15] D. Li, Y. Zhu, W. Lin, Traffic identification of mobile apps based on variational autoencoder network, in: 2017 13th International Conference on Computational Intelligence and Security, CIS, 2017, pp. 287–291, http://dx.doi.org/10.1109/CIS.2017.00069.

[16] P. Wang, X. Chen, F. Ye, Z. Sun, A survey of techniques for mobile service encrypted traffic classification using deep learning, IEEE Access 7 (2019) 54024–54033.

[17] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, C. Williamson, Offline/realtime traffic classification using semi-supervised learning, Perform. Eval. 64 (2007) 1194–1213.

[18] M. Conti, L.V. Mancini, R. Spolaor, N.V. Verde, Analyzing android encrypted network traffic to identify user actions, IEEE Trans. Inf. Forensics Secur. 11 (2016) 114–125, http://dx.doi.org/10.1109/TIFS.2015.2478741.

[19] S. Rezaei, X. Liu, How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets, 2019, URL arXiv:abs/1812.09761.

[20] A.S. Iliyasu, H. Deng, Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks, IEEE Access 8 (2020b) 118–126, http://dx.doi.org/10.1109/ACCESS.2019.2962106.

[21] O. Aouedi, K. Piamrat, D. Bagadthey, A semi-supervised stacked autoencoder approach for network traffic classification, 2020, URL https://hal.archives-ouvertes.fr/hal-02933689. accepted at IEEE ICNP HDR-Nets workshop 2020.

[22] P. Wang, S. Li, F. Ye, Z. Wang, M. Zhang, Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan, in: ICC 2020-2020 IEEE International Conference on Communications, ICC, 2020, pp. 1–7, http://dx.doi.org/10.1109/ICC40277.2020.9148946.

[23] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, 2014, arXiv e-prints http://arxiv.org/abs/1406.2661.

[24] A. Odena, Semi-supervised learning with generative adversarial networks, 2016, arXiv:1606.01583.

[25] R. A, L. M, S. C, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015, arXiv:/abs/1511.06434.

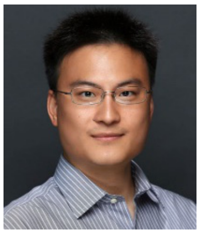
[26] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, 2016, arXiv:/abs/1606.03498.

[27] X. B, W. N, T. Chen, e.a., Empirical evaluation of rectified activations in convolutional network, 2015, arXiv:/abs/1505.00853.

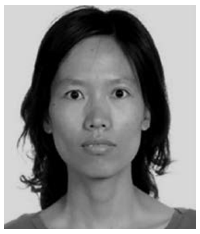
[28] A. Habibi Lashkari, G. Draper Gil, M. Mamun, A. Ghorbani, Characterization of encrypted and vpn traffic using time-related features, 2016.

Pan Wang (M'18) received the BS degree from the Department of Communication Engineering, Nanjing University of Posts & Telecommunications, Nanjing, China, in 2001, and the PhD degree in Electrical & Computer Engineering from Nanjing University of Posts&Telecommunications, Nanjing, China, in 2013. He is currently an Associate Professor in the School of Modern Posts, Nanjing University of Posts & Telecommunications, Nanjing, China. His research interests include cyber security and communication network security, network measurements, Quality of Service, Deep Packet Inspection, SDN, big data analytics and applications. From 2017 to 2018, he was a visiting scholar of University of Dayton (UD) in the Department of Electrical and Computer Engineering.

ZiXuan Wang was born in Nanjing, Jiangsu, China, in 1994. He obtained a master's degree in logistics engineering at Nanjing University of Posts and Telecommunications in 2020, He is currently pursuing a Ph.D. degree at Nanjing University of Posts and Telecommunications. His research interests include encrypted traffic identification and data balancing.



Feng Ye (S'12–M'15) received the BS degree from the Department of Electronics Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2011, and the PhD degree in Electrical & Computer Engineering from the University of Nebraska-Lincoln (UNL), NE, USA, in 2015. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, University of Dayton (UD), Dayton, OH, USA. Prior to joining UD, he was with the Department of ECE, UNL as an instructor and a researcher from 2015 to 2016. His research interests include cyber security and communication network security, wireless communications and networks, green ICT, smart grid communications and energy optimization, big data analytics and applications. He serves as the secretary of the IEEE Technical Committee on Green Communications and Computing (TCGCC). He is currently an Associate Editor of Security and Privacy (Wiley), and China Communications. He served as the Co-Chair of ICNC'19 Signal Processing for Communications Symposium; the Publicity Co-Chair of IEEE CBDCom 2018; the Co-Chair of Cognitive Radio and Networking Symposium, IEEE ICC 2018. He also serves as a TPC member for numerous international conferences, including INFOCOM, GLOBECOM, VTC, ICC, etc. He is also a reviewer for several IEEE journals, including IEEE Transactions on Big Data, IEEE Transactions on Green Communications and Networking, IEEE Transactions on Smart Grid, IEEE Transactions on Vehicular Technology, IEEE Transactions on Wireless Communications, etc. He is the recipient of the 2015 Top Reviewer of the IEEE Vehicular Technology Society.



Xuejiao Chen (M'18) received the BS degree from the Department of Communication Engineering, Nanjing University of Posts & Telecommunications, Nanjing, China, in 2001, and the Master degree in Electrical & Computer Engineering from Nanjing University of Posts & Telecommunications, Nanjing, China, in 2001, and the Master degree in Electrical & Computer Engineering from Nanjing University of Posts & Telecommunications, Nanjing, China, in 2006. She is currently an Assistant Professor in the Department of Communication Engineering, Nanjing College of Information Technology, Nanjing, China. Her research interests include wireless communications and networks, cyber security and communication network security, network measurements, Quality of Service, Deep Packet Inspection. From 2017 till now, she is a visiting scholar of University of Dayton (UD) in the Department of Electrical and Computer Engineering.