Distributed Traffic Synthesis and Classification in Edge Networks: A Federated Self-supervised Learning Approach

Journal:	*Transactions on Mobile Computing*
Manuscript ID	TMC-2021-11-0941
Manuscript Type:	Regular
Keywords:	Traffic classification, edge computing, federated learning, self-supervised learning

Distributed Traffic Synthesis and Classification in Edge Networks: A Federated Self-supervised Learning Approach

Yong Xiao, *Senior Member, IEEE*, Rong Xia, Yingyu Li, Guangming Shi, *Fellow, IEEE*, Diep N. Nguyen, Dinh Thai Hoang, Dusit Niyato, *Fellow, IEEE*, and Marwan Krunz, *Fellow, IEEE*

Abstract—With the rising demand for wireless services and increased awareness of the need for data protection, existing network traffic analysis and management architectures are facing unprecedented challenges in classifying and synthesizing the increasingly diverse services and applications. This paper proposes FS-GAN, a federated self-supervised learning framework to support automatic traffic analysis and synthesis over a large number of heterogeneous datasets. FS-GAN is composed of multiple distributed Generative Adversarial Networks (GANs), with a set of generators, each being designed to generate synthesized data samples following the distribution of an individual service traffic, and each discriminator being trained to differentiate the synthesized data samples and the real data samples of a local dataset. A federated learning-based framework is adopted to coordinate local model training processes of different GANs across different datasets. FS-GAN can classify data of unknown type of service and create synthetic samples that capture the traffic distribution of the unknown types. We prove that FS-GAN can minimize the Jensen-Shannon Divergence (JSD) between the distribution of real data across all the datasets and that of the synthesized data samples. FS-GAN also maximizes the JSD among the distributions of data samples created by different generators, resulting in each generator producing synthetic data samples that follow the same distribution as one particular service type. Extensive simulation results show that the classification accuracy of FS-GAN achieves over 20% improvement in average compared to the state-of-the-art clustering-based traffic analysis algorithms. Also FS-GAN has the capability to synthesize highly complex mixtures of traffic types without requiring any labeled data samples.

Index Terms—Traffic classification, self-supervised learning, edge computing, federated learning, generative adversarial networks.

1 Introduction

Telecommunication technologies have evolved tremendously over the past few decades, driven by innovative services and applications in a wide range of verticals [1], [2]. In particular, recently standardized 5G technology introduces three major use cases: eMBB (enhanced Mobile BroadBand), URLLC (Ultra Reliable Low Latency Communications), and mMTC (massive Machine Type Communications), with promises to enable novel applications including IoT [3]–[5], autonomous vehicles [6], and AR/VR [7]. According to recent reports [8], [9], the next-generation mobile technologies, e.g., B5G and 6G, are expected to further extend the application scenarios of 5G by bringing more diverse and innovative services, such as holographic-type communications, Tactile Internet, and others [10].

As more services and applications are introduced and adopted at different times in different regions, network traffic analysis and management face unprecedented challenges. In particular, the diverse demands and requirements of different services significantly increase the dynamics of network traffic, making traffic prediction, network planning, resource allocation and scheduling more challenging than before [11]. Most existing solutions rely on regression or convolutional neural network (CNN)-based supervised learning to fit historical data and accordingly classify the traffic of existing services [12]. These solutions often require a large amount of manually labeled datasets, which may not be practically feasible to obtain [13]. Recent works employ clustering-based unsupervised learning solutions to identify unknown traffic [14]. However, these solutions often suffer from limited accuracy and cannot be applied to predict or keep track of highly mixed and evolving service traffic flows. There is a pressing need to develop intelligent traffic analysis and classification solutions that can automatically identify known services and classify/label unknown services and data samples that emerge in decentralized datasets across various locations.

One promising solution to the above challenges is self-supervised learning, a novel algorithmic framework that does not require any labeled data samples for training machine learning (ML) models. Self-supervised learning combines the advantages of supervised-learning and unsupervised-learning by first relying on some carefully designed pre-tasks to automatically create labels or pseudo-labels for unlabelled data samples. It then employs supervised-learning approaches to construct ML models [15]–[17]. Due to its potential to significantly improve data efficiency and model generality, Yann LeCun, the 2018 Turing award laureate, described self-supervised learning as one of the key technologies of the "next artificial intelligence revolution" [18].

The increasingly stringent requirements on the respon-

Y. Xiao, R. Xia, and Y. Li are with the School of Electronic Information and Communications at the Huazhong University of Science and Technology, Wuhan, China 430074 (e-mail: {yongxiao, rong_xia, liyingyu}@hust.edu.cn).

G. Shi is with School of Artificial Intelligence, Xidian University, Xi'an, Shaanxi 710071, China (e-mail: gmshi@xidian.edu.cn).

D. Nguyen and D. Hoang are with the School of Electrical and Data Engineering, University of Technology Sydney, Australia (e-mail: {hoang.dinh, diep.nguyen}@uts.edu.au).

D. Niyato is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: dniyato@ntu.edu.sg).

M. Krunz is with the Department of Electrical and Computer Engineering, the University of Arizona, Tucson, AZ (e-mail: krunz@arizona.edu).

siveness of smart services and the growing awareness of data protection and network security created new challenges to the traditional cloud-based learning solutions in which every mobile device must upload its locally collected data samples to a cloud server and wait for feedback before making further decisions [19], [20]. These solutions are known to suffer from long communication delay and risk of privacy leakage caused by showing raw data. To address these issues, edge intelligence has been recently introduced, which enables distributed data processing and learning over a large number of low-cost, often decentralized edge devices, e.g., mobile edge servers [21]. It is considered one of the most sought-after functions in B5G/6G. One of the challenges of implementing edge intelligence is to develop simple and effective algorithmic solutions for decentralized data learning and processing over large resource-limited wireless networks [10]. As an emerging distributed AI solution, federated learning (FL) allows collaborative construction of ML models without exposing any raw data across decentralized datasets. However, FL is known to suffer from slow convergence and biased results if the data distribution among datasets are highly heterogeneous and unbalanced. Also, most existing FL solutions are based on supervised learning requiring high-quality labeled samples across all the decentralized datasets [22]. Integrating FL with self-supervised learning will have the potential to significantly reduce communication overhead, enhance data protection, and improve efficiency of model training.

In this paper, we propose the federated self-supervised Generative Adversarial Networks (FS-GAN), a novel framework to support distributed feature learning and traffic synthesis over decentralized datasets of heterogeneous traffic types. FS-GAN is comprised of multiple decentralized GANs, deployed at a set of edge servers that are close to a set of local datasets. Each generator is deployed at an edge server or a virtual machine that offers computational functions for an edge server. In contrast, each discriminator is implemented at a dedicated edge server and has been assigned with an exclusive right to access a local dataset requiring a certain privacy protection for local data samples. FS-GAN learns to create synthetic data samples that capture the statistical features of real data (i.e., its distribution) obtained from unknown service types to support automatic traffic classification. It does that by following three key procedures: (1) local data synthesis, (2) global model coordination and (3) self-supervised learning. In the first procedure, each discriminator is associated with a subset of generators that can learn to produce a mixture of synthetic samples with the same distribution as the local mix of service traffic. In global model coordination, local models trained at edge servers are aggregated via a coordinator following an FL-like procedure. Self-supervised learning is used to obtain the final FS-GAN model, which consists of a set of trained generators with the associated discriminator in which each generator creates synthetic data samples that match the distribution of a specific type of local traffic. By introducing a classifier that automatically creates pseudo-labels specifying which generator each synthetic data sample comes from, FS-GAN is able to automatically construct ML models that can synthesize and classify any future data samples in a self-supervised fashion.

FS-GAN does not require any human-labeled data or exposing any local dataset. We conduct extensive simulations based on real-world traffic associated with ten popular services applications including email, FTP, video and audio chat, etc [23]. Our results show that the classification accuracy of FS-GAN achieves over 20% improvement, on average, compared to the state-of-the-art clustering-based traffic analysis algorithms. Furthermore, FS-GAN's unique ability to synthesize highly complex mixtures of traffic types makes it applicable to a wide range of emerging use cases, such as tracking, assigning network slices in 5G/B5G systems, etc.

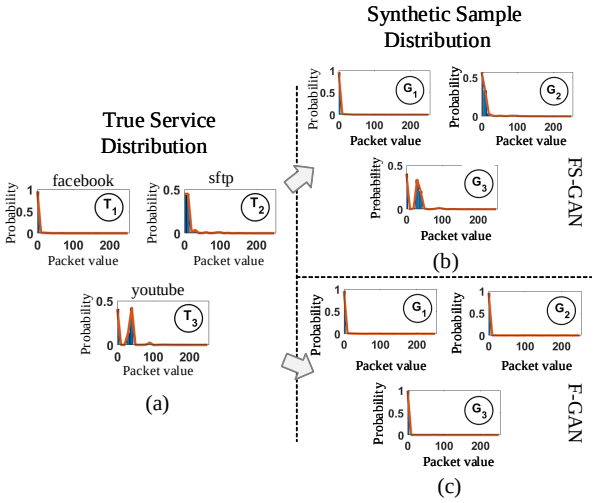

Fig. 1: Comparison between (a) the distribution of three service traffics T_1, T_1, T_3 measured in packet value (converted decimal value of data packet) and (b) the synthesized traffic data produced by FS-GAN and (c) the synthesized traffic produced by F-GAN.

To highlight the advantages of FS-GAN, in Fig. 1, we show the traffic distribution of the data packets from the real traffic data associated with 3 types of (unknown) applications (Fig. 1(a)) and the distribution of synthesized traffic generated by FS-GAN after 50 rounds of training (Fig. 1(b)), compared with the distribution of data produced by a straightforward extension of traditional multi-generator GAN into an FL setting, referred to as F-GAN (Fig. 1(c)). It can be observed that FS-GAN is able to successfully generate synthetic data samples that follow similar distributions as the real service traffic. F-GAN, however, fails to capture the difference in data distributions between various services but can only generate samples with a limited diversity.

The main contributions of this paper are as follows:

1) We introduce a novel framework called FS-GAN that exploits concepts from GAN, FL, and self-supervised learning to enable automatic learning, synthesis, and classification of heterogeneous traffic over decentralized datasets. Compared to regression-based clustering solutions, which separate data samples based on a single or a limited number of pre-selected features, FS-GAN autonomously creates synthetic samples with pseudo-labels to capture the distributions of local mixtures of traffic types and then applies supervised-learning-like solutions to further improve the classification performance. To the best of our knowledge, FS-GAN is the first network traffic synthesis and classification solution that uses self-supervised learning.

2) We prove that the model trained by FS-GAN minimizes the difference between the distribution of the real traffic and that of the synthetic data samples created by the FS-GAN generators. The unique ability of FS-GAN to learn and create synthetic samples that capture the distribution of a heavily mixed but unknown traffic across decentralized datasets allows it to be applied to a range of novel applications, well beyond traditional traffic classification. We discuss some potential applications and pointed out the limitations of FS-GAN under different scenarios.
3) We conduct extensive simulations using real traffic datasets. Our results show that FS-GAN achieves significant improvement in both traffic classification and data synthesis.

The rest of the paper is organized as follows. In section 2, we first review recent works that are relevant to FS-GAN. We then introduce the architecture and discuss its application scenarios in Section 3. Details of the proposed architecture, algorithms, as well as the main theoretical results are presented in Section 4. We conduct extensive simulations and evaluate the performance of FS-GAN in Section 5. The paper is concluded in Section 6.

2 Related Work

The solution proposed in this paper is closely related to recent progress in deep-learning-based traffic classification and FL-based data analysis and synthesis. Most related works are reviewed in this section.

2.1 Deep-learning-based Traffic Classification

Deep neural network (DNNs), are one of the promising solutions for network traffic classification [12], [14], [24]–[31]. For example, Wang *et al.* proposed a hybrid neural network that combines a recurrent neural network (RNN) and a CNN to learn dynamic features of mobile data for traffic classification [12]. Liang *et al.* introduced a deep reinforcement learning-based approach, called NeuroCuts, for packet classification [27]. Zhang *et al.* proposed a deep learning-based traffic clustering solution that can classify packets associated with unknown classes and automatically build a new training dataset to update the classifier of unknown traffic [14]. Liu *et al.* applied an RNN-based flow sequence network approach to classify encrypted traffic flows [25].

2.2 FL-based Network Data Analysis

FL and its applications in network data analysis have recently attracted significant interest [32]–[44]. FL has the potential to protect data privacy during distributed training and coordination across decentralized datasets. Wen *et al.* proposed a two-step FL framework to achieve privacy preserving model training with high communication efficiency. Wang *et al.* proposed an empirically driven solution to optimize the selection of client devices that participate in model updating [35]. Recently, FL was extended to data analysis in resource-limited networking systems. For example, Wang *et al.* investigated the convergence of gradient descent-based FL solutions and optimized the tradeoff between local update and global model aggregation under a given resource constraint [45]. Luo *et al.* introduced a low-cost sampling-based algorithm to learn the convergence-related parameters and minimize the cost of learning time and energy [46]. Xiao *et al.* proposed a federated edge intelligence framework for implementing FL in edge computing-assisted IoT networks with communication and computational resource constraints [47].

2.3 Self-supervised learning-based Data Analysis

Recently, self-supervised learning was introduced as a way to improve data efficiency and model generalization [15]–[17], [48]–[51]. In particular, Araslanov *et al.* proposed a domain adaptation approach for semantic segmentation based on predictions produced by pseudo-supervision targets [17]. Li *et al.* proposed a two-stage framework for anomaly detection in which a self-supervised deep representation model was learned to build a generative classifier [48]. Sun *et al.* introduced a novel model training algorithm for Graph Convolutional Networks (GCNs). Their model reduces the number of labeled samples and yet improves model generalization performance [49].

Our proposed FS-GAN differs from traditional clustering-based data analysis techniques in that it adopts a self-supervised learning-based solution to first create synthetic samples with pseudo-labels that statistically match a mixture of real traffic types and then train an ML model using the pseudo-labeled datasets to classify and identify each individual service traffic.

3 Architecture Overview and Application Scenarios

3.1 Architecture Overview

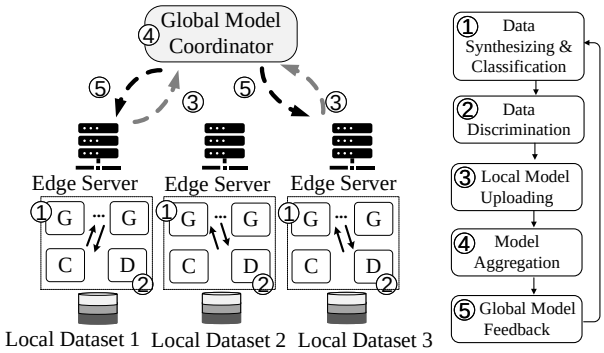

Fig. 2: FS-GAN architecture, including components generators (G), discriminators (D), classifiers (C), and a global model coordinator (left figure); and the main algorithmic procedures (right figure).

The main architecture, including the key components and algorithmic procedures of FS-GAN, is presented in Fig. 2. FS-GAN consists of: (1) multiple generators that create synthetic samples whose statistical distribution is similar to that of real traffic in local datasets, and (2) multiple discriminators, each of which discriminator tries to differentiate the real traffic from the synthetic samples produced by the generators. Our proposed FS-GAN framework employs the following three procedures:

(1) Local Data Synthesis: One discriminator is deployed in each edge server with exclusive right access to a local dataset. Each discriminator is jointly trained with several associated generators. Although GANs, in general, have high flexibility in generating synthetic samples [52]–[54], they suffer from the so-called *model collapsing problem*, where the generator learns to produce synthetic samples with extremely limited diversity, e.g., samples that only capture the data distribution of a single traffic type or a fixed mixture of a given set of traffic types. To tackle this problem, we borrow the idea of multi-generator GANs [55]–[57] and assign a pseudo-label to each generated sample to indicate which generator the sample came from. As will be proven later, a generator that is locally trained with a discriminator is able to create synthetic data samples that match the data distribution of the real traffic in the local dataset.

(2) Global Model Coordination: The local models trained by the discriminators and generators will be periodically uploaded to a global model coordinator. The coordinator can be deployed at one of the edge servers or at the cloud data center. One straightforward implementation of model coordination is to adopt FL solutions [40], [41], in which the local model parameters (of generators and/or discriminators) are aggregated, e.g., averaged in FedAvg algorithm [58], to form an updated global model that is then broadcasted to edge servers. Note that the number of model parameters from local edge servers will affect the communication efficiency. In this paper, we consider two model coordination schemes:

1) Coordination Scheme I (C-I): Both locally trained discriminators and the associated generators are uploaded to the global coordinator.

2) Coordination Scheme II (C-II): Only discriminators are uploaded and aggregated by the coordinator.

C-II requires less model-related data to be exchanged between edge servers and the coordinator, resulting in a higher communication efficiency. However, as will be shown later in this paper, C-I has faster convergence than C-II, which in some cases compensates for the extra communication overhead.

In the local data synthesis procedure, each discriminator only determines whether the data samples belong to a real traffic or are synthetic data produced by generators. After the global model coordination, all the discriminators should be able to differentiate real data in *any* local dataset from synthetic data generated by *any* generator. Similarly, the generators will be able to capture richer features across different local datasets. To address the model collapsing problem, we introduce a classifier in each local datatset to classify samples from different generators. As shown in Section 4, this classifier increases the divergence of the synthetic data distributions of different generators, hence alleviating the model collapsing problem of traditional GANs.

(3) Self-Supervised Learning: The aforementioned trained and updated classifier creates pseudo-labels for synthetic samples generated by different generators. As the generators become more capable of producing close-to-real synthetic data, the classifier associated with each discriminator will naturally support data classification and self-labeling of real traffic that arrive in the future. We show that the classifier actually shares the same model parameters as the locally trained discriminator, and therefore does not require any extra effort for model training.

We present a theoretical analysis that proves the effectiveness of the FS-GAN framework. In particular, we prove that FS-GAN possesses two important properties: (1) the difference between the distributions of the mixture of the synthetic data generated by generators and that of the real traffic data distribution is minimized, and (2) the JSD divergence of the distributions of synthetic data samples generated by different generators is maximized. Based on these properties, we prove the main result of this paper, namely, the synthetic data samples generated by each individual generator captures the real traffic distribution of each individual traffic type, and that the classifier can differentiate all the synthetic samples coming from different generators, if the distributions of the traffic data associated with different services are separable.

3.2 Examples of Application Scenarios

Compared to traditional traffic classification solutions, FS-GAN adopts a fundamentally different approach by first generating synthetic samples that capture the distribution of real traffic and then applying the parameters of the already trained local models for classification of real traffic. This uniqueness of FS-GAN makes it applicable to a range of new application scenarios and use cases, including the following:

1) Network Slicing in 5G/B5G: 5G NR release 16 [59] introduced the concept of network slicing, which focuses on isolating and preserving partitions of network resources and functions, commonly referred to as *slices*. These slices are tailored and orchestrated to support applications with diverse QoS requirements. One of the key challenges in network slicing is now to keep track of the traffic demands of potentially unknown applications, so the appropriate amount of resources can be reserved while meeting the needs of these applications. FS-GAN can be directly applied to classify and keep track of the needs of different services from a highly mixed traffic. It can also assist the network slicing manager in identifying newly observed service traffic types that do not have a sufficient amount of pre-labeled data.

2) Attack Detection: An important application scenario for traffic classification is attack detection. Existing solutions mostly focus on detecting known attacks. Detecting unknown attacks is a notoriously difficult problem [60]. FS-GAN has the potential to identify unknown attacks and simulate various attack scenarios as well as their mixtures.

3) Traffic Prediction-based Network Planning: Because the final model obtained by FS-GAN is capable of generating synthetic data samples that are statistically similar to the real data of various types of emerging traffic, FS-GAN can be applied to predict future traffic trends and assist in planning the network infrastructure to better cope with anticipated needs.

4 FS-GAN Design and Theoretical Analysis

In this section, we present the algorithmic details of the main procedures of FS-GAN (both C-I and C-II): local data synthesis, global model coordination, and self-supervised learning. The detailed architecture is shown in Fig. 3. Theoretical analysis is presented at the end of this section.

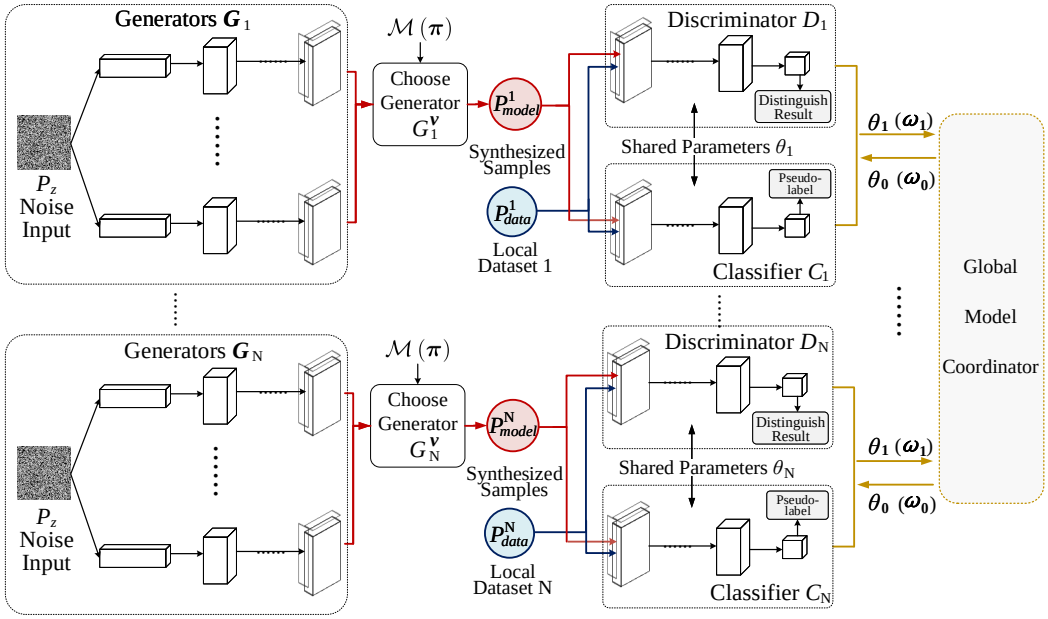

Fig. 3: Detailed architectural components of FS-GAN

4.1 Local Data Synthesis

First, we consider local-model training of a single generator at the dth local dataset, $d = 1, ..., N$. As mentioned before, the main objective of this procedure is to train a model to generate synthetic data samples that captures the distribution of real traffic of a local dataset. Let G_d and discriminator D_d be, respectively, the generator and discriminator associated with the dth dataset. Training GANs can be formulated as a minimax game between G_d and D_d [52]. The generator and the discriminator are often DNNs. Let P_{data} be the distribution of the real dataset and suppose that $\boldsymbol{x}$ is drawn from P_{data}, i.e., we write as $\boldsymbol{x} \sim P_{data}$. The generator G_d aims to train a multi-layer neural network to map its input variables. Initially, a random noise $\boldsymbol{z}$ is generated to synthesize data samples whose distribution is denoted as $P_{\boldsymbol{z}}$. Without loss of generality, we abuse the notation and use $G_d(\boldsymbol{z})$ to denote the trained generator's output given training input $\boldsymbol{z}$. Meanwhile, the discriminator D_d tries to distinguish real data samples from synthetic samples produced by the generator. We use $D_d(\cdot)$ to denote the output of discriminator, which represents the probability that the discriminator decides the given input sample comes from the real dataset rather than being synthesized by the generator. More formally, we can express the adversarial training process between a single generator and a single discriminator via the following minimax problem:

$$\min_{G_d} \max_{D_d} \mathbb{E}_{\boldsymbol{x} \sim P_{data}}[\log D_d(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim P_z}[\log(1 - D_d(G_d(\boldsymbol{z})))]. \tag{1}$$

The key idea of FS-GAN is to use multiple generators $\boldsymbol{G}_d = \{G_d^1, G_d^2, ..., G_d^M\}$, rather than a single one to jointly train a model that can capture the distribution of a mixture of different traffic types in a given local dataset d. To address the issue of model collapse, we introduce a classifier network C_d that classifies synthetic data produced by the generators. More specifically, the classifier first generates a pseudo-label for each synthetic data sample, indicating which generator it came from. It then minimizes its loss to encourage generators to create samples that are different from each other. As we prove later in this section, by employing C_d, each of the generators will synthesize data samples that represent a specific type of traffic, provided that the differences between the distributions of different traffic types are sufficiently large.

Now we focus on the training process for the multi-generator local model. We use the subscript to denote the index of the local dataset and the superscript for the generator index. Suppose a set $\boldsymbol{G}_d = \{G_d^1, G_d^2, ..., G_d^M\}$ of M generators has already been assigned to the discriminator D_d, where $d \in \{1, 2, ..., N\}$ represents the discriminator index. Each discriminator D_d also interacts with a local dataset X_d, and each generator $G_d^m, m \in \{1, 2, ..., M\}$, maps the input noise $\boldsymbol{z}$ to $\hat{\boldsymbol{x}} = G_d^m(\boldsymbol{z})$. The classifier C_d, is co-trained with $\boldsymbol{G}_d$ and D_d. Let $P_{G_d^m}$ be the distribution of the synthetic samples produced by G_d^m. We now describe the rule for each generator to interact with the discriminator. We set an index vector $\boldsymbol{v}$, randomly drawn from a predefined multinomial distribution $\boldsymbol{v} \sim \mathcal{M}(\boldsymbol{\pi})$ where $\boldsymbol{\pi} = \langle \pi_{\boldsymbol{v}} \rangle_{\boldsymbol{v} \in \boldsymbol{G}_d}$ is the weighting coefficient of the distribution mixture. By setting $G_d^{\boldsymbol{v}} \triangleq \{G_d^i : i \in \boldsymbol{v}\}$ as the subset of $\boldsymbol{G}_d$, we can convert the multi-generator GANs into a standard GAN with a generator vector $G_d^{\boldsymbol{v}}$ and discriminator D_d. We use P_{data}^d to denote the real distribution of dataset X_d. We can then formulate the model of local data synthesis with a multi-generator as the following minimax game:

$$\begin{aligned} \min_{\boldsymbol{G}_d, C_d} \max_{D_d} \quad & \mathbb{E}_{\boldsymbol{x} \sim P_{data}^d}[\log D_d(\boldsymbol{x})] \\ & + \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{z} \sim P_z}[\log(1 - D_d(G_d^m(\boldsymbol{z})))] \\ & - \lambda \sum_{m=1}^{M} \left(\pi_m \mathbb{E}_{\hat{\boldsymbol{x}} \sim P_{G_d^m}}[\log C_d^m(\hat{\boldsymbol{x}})] \right) \end{aligned} \tag{2}$$

where $C_d^m(\hat{\boldsymbol{x}})$ is the output of classifier C_d, specifying the

probability that data sample $\hat{\boldsymbol{x}}$ is generated by G_d^m. The last term in (2) is a standard softmax loss in a multi-classification setting [61]. In other words, minimizing C_d according to (2) under fixed D_d minimizes the entropy for C_d, which leads to increased divergence among generators. We introduce a diversity hyper-parameter $\lambda > 0$ to specify the degree that the classifier can influence the generators. We compare the model performance under different λ in Section 5.

Let ω_d be the model parameters of the generators $\boldsymbol{G}_d$ associated with discriminator D_d, and let ω_d^m be the model parameters of the generator $G_d^m, m \in \{1, ..., M\}$. Classifier C_d and discriminator D_d share the same model parameters except for the last layer. Let θ_d be the model parameters shared between the classifier and the discriminator D_d. Note that the last layer of C_d outputs an automatically learned pseudo-label for each input data sample, while the last layer of D_d outputs a binary result. Similar to a standard GAN, we alternatively learn ω_d and θ_d using the stochastic gradient descend (SGD) method. The local data synthesis framework is illustrated in Fig. 3.

Let B be the selected mini-batch size during the training process. At the beginning of each local training iteration, each discriminator D_d will sample a mini-batch of B data points $(\boldsymbol{x}_d^{(1)}, \boldsymbol{x}_d^{(2)}, ..., \boldsymbol{x}_d^{(B)})$ from the real data distribution P_{data}^d. A mini-batch of B synthetic data samples $(\hat{\boldsymbol{x}}_d^{(1)}, \hat{\boldsymbol{x}}_d^{(2)}, ..., \hat{\boldsymbol{x}}_d^{(B)})$ will also be created by a mixture of generators G_d^v in $\boldsymbol{G}_d$. This mixture can be generated according to a predefined sampling probability π_m. During each training iteration, we update C_d, D_d, and $G_d^{\boldsymbol{v}}$ along the gradients of their loss functions. According to (2), we can write the loss functions under the aforementioned settings as follows:

$$\mathcal{L}(C_d, \hat{\boldsymbol{x}}_d) = -\frac{1}{B}\sum_{b=1}^{B} \log C_d^m\left(\hat{\boldsymbol{x}}_d^{(b)}\right) \tag{3a}$$

$$\mathcal{L}(D_d, \boldsymbol{x}_d, \hat{\boldsymbol{x}}_d) = -\frac{1}{B}\sum_{b=1}^{B}\Big(\log D_d\left(\boldsymbol{x}_d^{(b)}\right) + \log\left(1 - D_d\left(\hat{\boldsymbol{x}}_d^{(b)}\right)\right)\Big) \tag{3b}$$

$$\mathcal{L}(G_d^{\boldsymbol{v}}, \hat{\boldsymbol{x}}_d) = -\frac{1}{B}\sum_{b=1}^{B}\left(\log D_d\left(\hat{\boldsymbol{x}}_d^{(b)}\right) + \lambda \log C_d^m\left(\hat{\boldsymbol{x}}_d^{(b)}\right)\right) \tag{3c}$$

We summarize the detailed procedure in Algorithm 1.

4.2 Global Model Coordination

In the previous section, the generators associated with a local discriminator produce synthetic data samples that capture the distributions of different traffic types in the local dataset. However, different local datasets may have different mixtures of traffic. It is therefore desirable to train a global model that can capture the diverse traffic across all local datasets. We adopt the FL-based framework to coordinate the local model training procedures without requiring any exchange of local datasets. One of the key ideas in FL is to coordinate different local model training procures via their model parameters. In one of the popular FL solutions, called FedAvg, each edge server periodically uploads its locally trained models to a global coordinator. The uploaded models are then averaged to form an updated global model, which will be broadcasted to all edge servers.

Algorithm 1 Local Data Synthesis

Input: Local dataset X_d; noise distribution $P_{\boldsymbol{z}}$; sampling probability π_m;
Output: Updated local parameter ω_d generators $\boldsymbol{G}_d$; updated local parameter θ_d for discriminator D_d and classifier C_d;
Initialize: $\lambda > 0$; mini-batch size B; network parameters ω_d and θ_d; maximum number of Iterations I;
for each iteration $i \leq I$

Generator $\boldsymbol{G}_d$ **do**:
1) Select generators $G_d^{\boldsymbol{v}}$ according to π_m;
2) Each generator $G_m^d \in G_d^{\boldsymbol{v}}$ simultaneously **do**:
   - Sample a noise input variable $\boldsymbol{z}$ from $P_{\boldsymbol{z}}$;
   - Synthesize a batch of data sample $\hat{\boldsymbol{x}}_d^{(b)}, b \in \{1, 2, ..., B\}$;
3) Send the synthesize data sample $\hat{\boldsymbol{x}}_d^{(b)}$ to the discriminator D_d and classifier C_d;
4) Update ω_d by ascending along its gradient of loss function: $\nabla_{\omega_d}\mathcal{L}(G_d^{\boldsymbol{v}})$;

Discriminator D_d **do**:
1) Sample a batch of data $\boldsymbol{x}_d^{(b)}, b \in \{1, 2, ..., B\}$ from local dataset X_d;
2) Distinguish $\boldsymbol{x}_d^{(b)}$ and $\hat{\boldsymbol{x}}_d^{(b)}$;
3) Update θ_d by descending along their gradients of loss functions: $\nabla_{\theta_d}\left(\mathcal{L}(C_d) + \mathcal{L}(D_d)\right)$;

Classifier C_d **do**:
1) Classify the synthesized samples $\hat{\boldsymbol{x}}_d^{(b)}$ with respect to different generators;
2) Update θ_d by descending along their gradients of loss functions: $\nabla_{\theta_d}\left(\mathcal{L}(C_d) + \mathcal{L}(D_d)\right)$;

$i = i + 1$;

Return locally trained parameters

Formally, let Ω' be the subset of discriminators in coordination scheme C-II (or discriminator-and-generator pairs in scheme C-I) that has been selected for uploading the local models. Note that $\Omega' \subseteq \Omega$. Let ω_0 and θ_0 be the corresponding global model parameters, and let p_d be the weight of the dth local dataset such that $p_d \geq 0$ and $\sum_{d=1}^{N} p_d = 1$. We can write the global model coordination procedure as follows:

$$\begin{cases} \omega_0 = \sum\limits_{d\in\Omega'} p_d\omega_d \text{ and } \theta_0 = \sum\limits_{d\in\Omega'} p_d\theta_d, & \text{if Scheme C-I,} \\ \theta_0 = \sum\limits_{d\in\Omega'} p_d\theta_d, & \text{if Scheme C-II.} \end{cases} \tag{4}$$

4.3 Self-Supervised Learning

As mentioned in the previous procedures, the classifier shares the same network parameters with the local discriminator. Therefore, this classifier can be updated in the global model coordination procedure to classify and create pseudo-labels for the synthetic samples according to their corresponding generators. This classifier will also be able to classify the real data associated with all traffic types in all datasets. Since this classifier is a part of the training

Algorithm 2 Global Model Coordination (Scheme C-I)

Input: Locally trained parameter θ_d of D_d and C_d; locally trained parameter ω_d of $\boldsymbol{G}_d$;
Output: Aggregated parameter θ_0 for the global discriminator D_0 and classifier C_0; aggregated parameter ω_0 for the global generators $\boldsymbol{G}_0$
Initialize: The number of local datasets N; maximum number of communication rounds J;
for the number of communication rounds $j \leq J$

The global model coordinator **do**:
1) Calculate the aggregated parameter θ_0 and ω_0 according to equation (4):
$\theta_0, \omega_0 \leftarrow$ *Local Data Synthesis*$(X_d, \theta_d, \omega_d)$
2) Broadcast θ_0 to and ω_0 to local discriminator, classifier and generator, respectively;

Each local model **do**:
Initialize local parameter θ_d and ω_0 by:
$\theta_d \leftarrow \theta_0; \omega_d \leftarrow \omega_0$;
$j = j + 1$;

end for

Algorithm 3 Global Model Coordination (Scheme C-II)

Input: Locally trained parameter θ_d of D_d and C_d;
Output: Aggregated parameter θ_0 for the global discriminator D_0 and classifier C_0;
Initialize: The number of local datasets N; maximum number of communication rounds J;
for the number of communication rounds $j \leq J$

The global model coordinator **do**:
1) Calculate the aggregated parameter θ_0 according to equation (4b):
$\theta_0 \leftarrow$ *Local Data Synthesis*(X_d, θ_d)
2) Broadcast θ_0 to local discriminator and classifier;

Each local model **do**:
Initialize local parameter θ_d by:
$\theta_d \leftarrow \theta_0$;
$j = j + 1$;

end for

process of FS-GAN and is trained based on the pseudo-labeled synthetic datasets, it does not introduce any extra training efforts.

4.4 Theoretical Analysis

We analyze FS-GAN from the following two aspects: data synthesis ability and classification accuracy across local models. Consistent with our previous notation, we use P^d_{data} to denote the mixture distribution of real data samples in dataset X_d, and use P^d_{model} to denote the distribution of the combination of synthetic data samples generated by generators $G^1_d, G^2_d, ..., G^M_d$. First, we provide the optimal solution for classifier C_d in Equation (2).

Proposition 1. For a set of generators $\boldsymbol{G}$ and for a given sampling probability π_m, the optimal distribution learned by classifier ${C_d}^*(\boldsymbol{x}; \theta_d)$ has the following form:

$$C^{m*}_d(\boldsymbol{x}; \theta_d) = \frac{\pi_m P_{G^m_d}(\boldsymbol{x})}{\sum_{i=1}^{M} \pi_i P_{G^i_d}(\boldsymbol{x})}, m \in \{1, 2, ..., M\} \tag{5}$$

Proof: See Appendix A. □

The result of the output of the optimal classifier ${C_d}^*(\boldsymbol{x}; \theta_d)$ can be seen as a weighted normalization of different synthesized sample distributions.

We follow a commonly adopted setting and use Jensen Shannon Divergence (JSD) [62] to quantify the difference between two distributions. For the optimal generator ${\boldsymbol{G}_d}^*(\boldsymbol{x}; \omega_d)$, we have the following proposition:

Proposition 2. Given the optimal discriminator and classifier, the objective of generators is to minimize:

$$\begin{aligned} {\boldsymbol{G}_d}^*(\boldsymbol{x}; \omega_d) \quad = \quad & \arg\min_{G}(2 \cdot \mathrm{JSD}(P^d_{data} \| P^d_{model}) \\ & -\lambda \cdot \mathrm{JSD}_{\pi_1, \pi_2, ..., \pi_M}(P_{G^1_d}, P_{G^1_d}, ..., P_{G^M_d}). \end{aligned} \tag{6}$$

Proof: See Appendix B. □

It can be observed that the two terms on the right-hand-side of (6) correspond, respectively, to the difference between distributions P^d_{data} and P^d_{model}, and the inverse of the difference among the synthetic data generated by different generators. In other words, minimizing these two terms directly results in minimizing the difference between P^d_{data} and P^d_{model} while maximizing the differences among the generators.

Before presenting the main theorem, we make the following assumption:

Assumption 1. The real data distribution P^d_{data} of dataset X_d can be written in the form:

$$P^d_{data}(\boldsymbol{x}) \quad = \quad \sum_{m=1}^{M} \pi_m p^m_d(\boldsymbol{x}), \tag{7}$$

where for any given $\boldsymbol{x}$, if $p^m_d(\boldsymbol{x}) > 0$ for $m \in \{1, 2, ..., M\}$, then $m' \neq m$, $p^{m'}_d(\boldsymbol{x}) = 0$.

Equation (7) means that the data distribution of data samples is a mixture of M separable distributions given by $p^m_d(x)$ for $m = 1, 2, ..., M$. We can prove the following theorem:

Theorem 1. Suppose that Assumption 1 holds. At the equilibrium point of the minimax game defined in equation (2), the optimal classifier and generators satisfy the following equations:

$$C^{m*}_d(\boldsymbol{x}) \rightarrow \begin{cases} 1, \text{if } x \sim P_{G^m_d}, \\ 0, \text{if } x \sim P_{G^{m'}_d}, \text{ for } m' \neq m, \end{cases} \tag{8a}$$

$$P^*_{G^m_d}(\boldsymbol{x}) = p^m_d(\boldsymbol{x}), \ \forall m = 1, 2, ..., M, \tag{8b}$$

$$P^d_{model}(\boldsymbol{x}) = P^d_{data}(\boldsymbol{x}). \tag{8c}$$

Proof: See Appendix C. □

We can observe from (8a) that the classifier can correctly differentiate synthetic samples produced by different generators, i.e., the probability for the classifier to identify the correct generator G^m_d that produces each given synthetic samples x approaches one. As shown in (8b) and

(8c), synthetic samples produced by optimal generator G_d^{m*} follow the same distribution as p_d^m. Also, all the generators weighted by π_m together synthesize samples that match the same data distribution of that of the real dataset.

In cases where Assumption 1 is not satisfied, Theorem 1 can be considered as an upper-bound for the local data synthesis and classification performance. Existing works as well as our own experiments show that in many practical scenarios, even if Assumption 1 does not hold, FS-GAN can still create high-quality synthesized data with high classification accuracy performance. We will come back to this issue in Section 5.

So far, we have analyzed the optimal solution of the local model. Next, we focus on the convergence of the global model. In the global model coordination procedure, we use Federated Averaging (*FedAvg*) [58], a widely popular algorithm in federated learning, to periodically aggregate local models. More specifically, suppose that each participating local model performs I local updates after receiving the latest global model. Let T be the total number of local iterations performed by each client. Let B be the mini-batch size. Let $\mathcal{L}_d(\boldsymbol{\varepsilon}, \xi_t^d), d \in \{1, 2, ..., N\}$, be the loss function of local model in tth iteration with network parameters $\boldsymbol{\varepsilon}$ and a mini-batch of B samples ξ_t^d. We define $\mathcal{L}_d(\boldsymbol{\varepsilon})$ as the mean value of $\mathcal{L}_d(\boldsymbol{\varepsilon}, \xi_t^d), t = 1, ..., T$, i.e., we have $\mathcal{L}_d(\boldsymbol{\varepsilon}) = \mathbb{E}[\mathcal{L}_d(\boldsymbol{\varepsilon}, \xi_t^d)]$. Consider the following optimization model:

$$\min_{\boldsymbol{\varepsilon}} \{\mathcal{L}(\boldsymbol{\varepsilon}) \triangleq \sum_{d=1}^{N} p_d \mathcal{L}_d(\boldsymbol{\varepsilon})\}. \tag{9}$$

Theoretical guarantees of (9) can be obtained by following the same approach as [43], [63], [64].

5 Performance Evaluation

5.1 Simulation Setup

To evaluate the performance of FS-GAN, we consider real-world dataset, "VPN-nonVPN dataset" (ISCXVPN2016) [23], to evaluate a highly mixed traffic data flow consisting of multiple types of unknown services. Within this dataset, we select data samples associated with 10 popular services, summarized in Table 1. As our focus is on classification of valid information, we remove the PHY and MAC headers from the packets and limite the length of each payload to the same size of 2500 bytes so as to achieve a proper trade-off between the training efficiency and classification accuracy.

TABLE 1: Dataset Used for Evaluation of FS-GAN

Service	# of Packets	Service	# of Packets
Email	32,566	Youtube	12,738
ftps (upload)	47,795	sftps (upload)	107,234
SCP (download)	85,018	Skype Video	140,569
Facebook Audio	91,815	Skype Chat	53,996
Tor-Youtube	54,294	VPN-Vimeo	215,102

We conduct our experiments on a workstation with an Intel(R) Core(TM) i9-9900K CPU@3.60GHz, 64.0 GB RAM@2133 MHz, 2 TB HD and two NVIDIA Corporation GP102 [TITAN X] GPUs. Each edge server ia simuylated with a TITAN X GPU running on Ubuntu 16.04, Python 3.6, CUDA 10.0 and PyTorch 1.3.1. The training process between edge servers is simulated using the PySyft framework [65]. We combine the packet of all selected traffic types into one trace and equally divide the data into six local datasets, each being assigned to a discriminator (an edge server). The network architecture and parameter setting for the generator, discriminator, and classifier are summarized in Table 2.

TABLE 2: Simulation Setup

Architecture Setup	Network Type		
	Generator	Discriminator	Classifier
Input Size	100×1	2500×1	2500×1
Output Size	2500×1	2×1	# of Generator
Activation Function	LeakyReLU	Sigmoid	LeakyReLU
# of Layers	8	6	
Optimizer	Adam with $\beta_1 = 0.5$ and $\beta_2 = 0.9$		

TABLE 3: Clustering Performance of FS-GAN and State-of-the-art Schemes

Scheme	Training Loss	Training Method	RI	NMI	ACC
K-means++	Mean Square Error	Iterative Method	0.71	0.30	0.32
DEC	Reconstruction and Cluster Assignment	Pretraining and Fine-tuning	0.82	0.39	0.39
DCN	Reconstruction and K-means Loss	Pretraining and Joint Training	0.81	0.31	0.35
IDEC	Reconstruction and Cluster Assignment	Pretraining and Joint Training	0.85	0.42	0.40
FS-GAN	Adversarial and Classification Loss	Jointly Adversarial Training	**0.89**	**0.62**	**0.58**

5.2 Classification Performance

As mention earlier, compared to existing clustering solutions, FS-GAN introduces a novel clustering solution for model training by autonomously creating synthetic data with pseudo-labels. In this subsection, we compare the clustering performance of FS-GAN with existing clustering solutions. We consider three commonly adopted performance metrics:

(1) Rand Index (RI): measures the fraction of samples pairs being correctly clustered, including similar samples being classified into the same category and different samples being classified into different categories. Formally, RI is defined as follows:

$$\text{RI} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}, \tag{10}$$

where TP is the number of two same-type packets being assigned to the same traffic. TN refers to the number of different-type packet pairs being assigned to different service types. FN is the number of same-type packets pairs being assigned to different service types. Finally, FP is the number of different-type packet pairs being assigned to one service type. RI evaluates the clustering performance by judging whether samples of the same type fall into the same category. In particular, the value of RI must be grater than or equal to 0 and it approaches 1 if the values of FN and FP approach zero.

(2) Normalized Mutual Information (NMI): measures the uncertainty that can be reduced about the ground truth clustering result when the clustering solution is given. Let U be the ground truth class and V be the resulting label of the proposed clustering algorithm. NMI is defined as:

$$\text{NMI}(U, V) = \frac{2\text{MI}(U, V)}{H(U) + H(V)}, \tag{11}$$

TABLE 4: Model Training Time and Clustering Performance under Different Schemes

N	n	E	B	Time per-round (sec) C-I	Time per-round (sec) C-II	RI C-I	RI C-II	NMI C-I	NMI C-II	ACC C-I	ACC C-II
1	12,000	4	100	31.19		0.91		0.64		0.63	
6	2,000	4	100	34.32	39.16	0.89	0.89	0.62	0.43	0.58	0.57
3	2,000	4	100	34.72	35.43	0.89	0.88	0.61	0.42	0.60	0.58
6	2,000	4	100	34.32	39.16	0.89	0.89	0.62	0.43	0.58	0.57
9	2,000	4	100	38.03	37.21	0.89	0.88	0.60	0.40	0.63	0.57
6	500	4	100	10.14	11.36	0.89	0.88	0.62	0.41	0.60	0.51
6	1000	4	100	18.33	18.14	0.89	0.89	0.60	0.47	0.56	0.55
6	2000	4	100	34.32	39.16	0.89	0.89	0.62	0.43	0.58	0.57
6	2000	2	100	20.32	19.73	0.89	0.88	0.62	0.43	0.60	0.53
6	2000	4	100	34.32	39.16	0.89	0.89	0.62	0.43	0.58	0.57
6	2000	8	100	68.15	75.32	0.89	0.90	0.64	0.47	0.61	0.54
6	2000	4	50	61.17	62.73	0.90	0.89	0.63	0.46	0.57	0.60
6	2000	4	100	34.32	39.16	0.89	0.89	0.62	0.43	0.58	0.57
6	2000	4	200	21.50	24.13	0.89	0.89	0.57	0.43	0.57	0.57

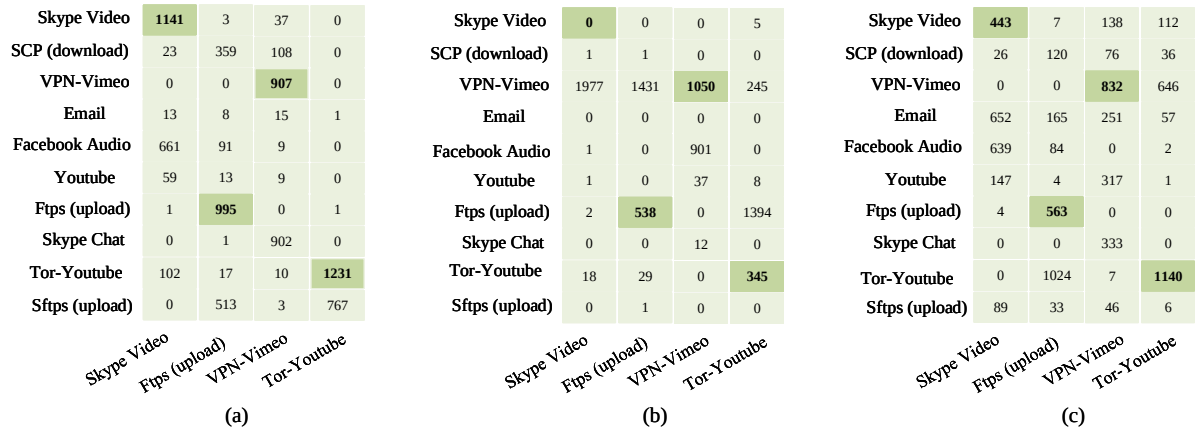

Fig. 4: Clustering results of 10 service traffic data using: (a) FS-GAN, (b) K-means++, and (c) IDEC.

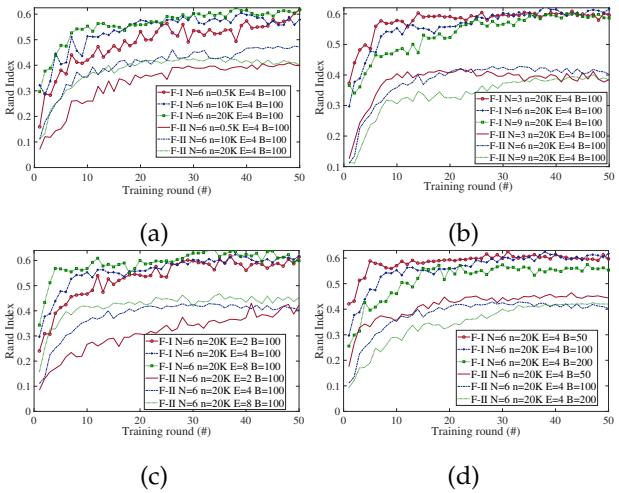

Fig. 5: Clustering performance of FS-GAN-I and FS-GAN-II based on real-world traffic.

where $\mathrm{MI}(U, V)$ is the mutual information between U and V, and $H(*)$ is the entropy. We have

(3) Unsupervised Accuracy (ACC): measures the best one-to-one relationship between the resulting label of the proposed clustering solution and the ground-truth classes. ACC is defined as:

$$\mathrm{ACC} = \frac{1}{n} \max_{\sigma} \sum_{i=1}^{n} 1_{y_i = \sigma(x_i)} \tag{12}$$

where x_i and y_i refer respectively, to the labels of the clustering solution and the ground-truth class for each data sample i. σ represents all the possible one-to-one permutations between any two class labels.

Note that each of the above three metrics takes values between 0 and 1. The larger the better is the clustering performance.

In Table 3, we compare FS-GAN with four clustering solutions: K-means++, DEC [66], IDEC [67], and DCN [68]. It can be observed that FS-GAN achieves the best performance of all tested solutions in all three performance metrics. In particular, DEC, IDEC, and DCN use deep neural networks to recover different latent representations for clustering. Compared to K-means++, which directly searches for cluster centers or centroids of the data samples, deep-learning-based solutions often achieve better performance. FS-GAN takes a novel approach by first generating synthetic data samples with pseudo-labels and then training a classifier based on the labeled dataset. In some senses, this approach addresses the poor performance caused by the lack of labelled dataset and can result in improved performance when the quality of the pseudo-labeled synthetic data samples is high.

To compare the impact of model training parameters on the computational loads of FS-GAN under different setups, we present in Table 4 the running time per round of computation for FS-GAN in a fixed hardware and software environment under various performance metrics for 50 rounds of model training. We mainly evaluate the impact of four key parameters, including the number of local datasets, size of each dataset, the number of local training steps (N, n, E) between two consequent global model coordination, and the mini-batch size B. The learning results under one centralized dataset is also presented. We can see that both the learning time and performance of FS-GAN are slightly degraded compared with a centralized setting, because the model aggregation procedure is no longer needed. It can be observed that the computational load is most sensitive to the size of the dataset. Compared to scheme C-I, Scheme C-II always results in slightly more computational load for training. This is because, in scheme C-I, both generators and discriminators are coordinated, which accelerates convergence of the model training and leads to reduced overall computational time. The improved performance offered by Scheme C-I over C-II can be further shown in the NMI and ACC metrics. More specifically, C-I always exhibits a more accurate clustering performance than C-II when the number of training rounds is fixed. C-I and C-II offer similar performance in terms of RI. This is because RI measures the combined performance of the correct solutions over all the clustering results, instead of the highest performance among individual classes as NMI and ACC. Therefore, the performance difference in RI achieved by different schemes is always smaller than that of the other two performance metrics.

In Fig. 4, we present the clustering results when FS-GAN is used to classify the 10 services in Table 1. Once again, FS-GAN delivers superior performance to IDEC and K-means++. Note that since K-means++ directly separates data samples based on the centroid, it tends to cluster most of the samples into a single or a limited number of clusters.

In Fig. 5, we compare the convergence performance based on NMI under different model parameters. We can observe that C-I always offers better convergence performance than C-II. This result is consistent with the previous observation. When the size of local dataset is the same, the convergence performance is closely correlated with this size. Specifically, the larger the dataset size, the faster is the convergence of the model training process. Note that the fluctuations in NMI is due to the interaction between generators and discriminators in the local data synthesis as well as the global model coordination.

As mentioned earlier, the classifier helps increasing the diversity of the synthesized samples produced by generators. At the same time, it interferes with the adversarial training process between the generator and discriminator. We investigate the impact of the classifier on the data clustering performance by adjusting the diversity hyperparameter λ in Equation (2). Intuitively, when the penalty brought by the classifier becomes larger than that of the discriminator, generators will be encouraged to produce more diversified samples even if these samples may not follow the same distribution as the real data. In Table 5, we present the clustering performance of FS-GAN for different values of λ. We can observe that a very large or small value of λ will not typically offer the best performance The optimal λ varies with the model setup. Finding the optimal λ that achieves the right balance between classification and discrimination will be left for future work.

TABLE 5: Performance under Different Classifier Level λ

Diversity Parameter		$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 5$
FS-GAN (C-I)	RI	0.89	0.89	**0.90**
	NMI	0.60	0.60	**0.64**
	ACC	0.57	0.58	**0.60**
FS-GAN (C-II)	RI	0.88	**0.89**	0.88
	NMI	0.43	**0.47**	0.40
	ACC	0.52	**0.57**	0.51

5.3 Performance Evaluation of Traffic Data Synthesis

As mentioned earlier, FS-GAN is more than just a traffic classification approach. It can also learn from the decentralized datasets and produce synthetic data samples that capture the distribution of real data associated with unknown services. In this subsection, we evaluate this aspect of FS-GAN.

We compare the data synthesis capability of FS-GAN to various extensions of multi-generator GANs with FL, which we refer to as F-GAN. In particular, we compare the following 4 different schemes:

1) F-GAN (three-G): 3 generators for each dataset without the classifier.
2) FS-GAN (three-G): 3 generators for each dataset.
3) F-GAN (six-G): 6 generators for each dataset without the classifier.
4) FS-GAN (six-G): 6 generators for each dataset.

As mentioned earlier, one of the key challenges of GAN-based solutions is the possibility of triggering a model collapse problem where, in this case, different generators will only produce samples with limited variety regardless of the input. In Fig. 6, we evaluate the diversity of synthesized data samples produced by different schemes. We compare the default numbers of synthetic samples produced by each scheme that are associated with different services when the same input is employed. It can be observed that different generators tend to produce different numbers of synthetic samples. For example, among all six types of services, SCP (download) is the least popular service to be synthesized with the least number of synthesized data samples, e.g., less than 0.02% of total synthesized samples produced by F-GAN(C-I, six-G) falls into this service category. We, however, observe that FS-GAN, especially C-I scheme with six generators, produces relatively uniform number of samples for different services. This means that FS-GAN offers the best performance in terms of balancing synthesized data samples.

To quantify the diversity of the synthesized samples, we use statistical distance metrics, which measure the difference between the distribution of the synthetic data samples and that of the real service traffic data. We consider three distance metrics: Kullback-Leibler (KL) divergence, Jensen Shannon (JS) divergence [62], and Wasserstein (W) distance [69], consider two random variables, p and q, with respective distributions $p(x)$ and $q(x)$. Table 6 provides the distance

TABLE 6: Quality of Synthesized Under Different Scenarios

Metric	F-GAN(C-I)		F-GAN(C-II)		FS-GAN(C-I)		FS-GAN(C-II)	
	Three-G	Six-G	Three-G	Six-G	Three-G	Six-G	Three-G	Six-G
KL Divergence	5.643	0.360	0.226	0.224	0.091	**0.043**	0.188	0.099
JS Divergence	0.086	0.062	0.054	0.051	0.022	**0.011**	0.041	0.025
Wasserstein Distance	0.647	0.619	1.013	0.578	0.353	**0.286**	0.484	0.387

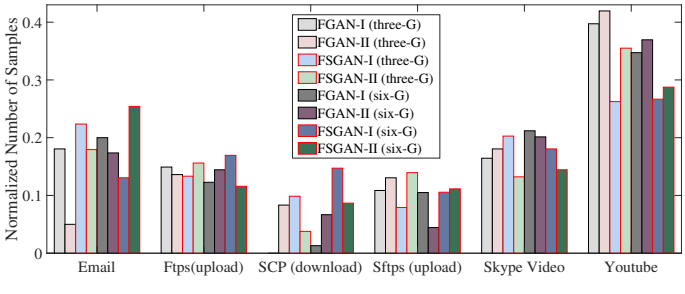

Fig. 6: Distribution of the assigned labels under different schemes.

results under both FS-GAN and F-GAN. FS-GAN achieves the smallest distance between synthetic and real data, where the JS divergence under FS-GAN-I is shown to be as low as 0.011, almost one fifth of the lowest distance result achieved by F-GAN. This means that the proposed FS-GAN is ideal to classify and synthesize highly heterogeneous traffic flows with mixed services.

6 Conclusions

In this paper, we proposed FS-GAN, a federated self-supervised learning framework to automatically recognize, classify, and synthesize different types of traffic over a large number of decentralized datasets. FS-GAN consists of three components: local data synthesis, global model coordination, and self-supervised learning. We adopted an FL-like approach and proved that our jointly trained global model can simultaneously minimize the JSD between the distribution of real data across all the datasets and that of the synthesized data samples.It also maximizes the JSD among the distributions of data samples created by different generators. Simulation results show that FS-GAN can achieve significant performance improvement over state-of-art data clustering solutions, and almost five times improvement over the federated GAN solutions in terms of data synthesis diversity .

References

[1] K. Pentikousis, Y. Wang, and W. Hu, "Mobileflow: Toward software-defined mobile networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 44–53, Jul. 2013.

[2] V. Yazıcı, U. C. Kozat, and M. O. Sunay, "A new control plane for 5G network architecture with a case study on unified handoff, mobility, and routing management," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 76–85, Nov. 2014.

[3] M. Agiwal, A. Roy, and N. Saxena, "Next Generation 5G Wireless Networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, Feb. 2016.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, Jun. 2015.

[5] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[6] T. Luettel, M. Himmelsbach, and H. Wuensche, "Autonomous ground vehicles—concepts and a path to the future," *Proceedings of the IEEE*, vol. 100, pp. 1831–1839, Apr. 2012.

[7] C. Wang, J. Liu, Y. Chen, H. Liu, L. Xie, W. Wang, B. He, and S. Lu, "Multi - touch in the air: Device-free finger tracking and gesture recognition via cots rfid," in *IEEE INFOCOM*, Honolulu, USA, Apr. 2018, pp. 1691–1699.

[8] "Itu 2019 network 2030," Geneva, Jul. 2018. [Online]. Available: https://www.itu.int/en/ITU-T/focusgroups/net2030/Pages/default.aspx

[9] I. FG-NET2030, "New services and capabilities for network 2030: description, technical gap and performance target analysis," Oct. 2019.

[10] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, "Towards self-learning edge intelligence in 6G," *IEEE Communications Magazine*, vol. 58, no. 12, Dec. 2020.

[11] M. Reiter and R. Steinberg, "Forward contracts for complementary segments of a communication network," in *IEEE INFOCOM*, San Diego, USA, Mar. 2010, pp. 1–9.

[12] X. Wang, S. Chen, and J. Su, "App-net: A hybrid neural network for encrypted mobile traffic classification," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 424–429.

[13] M. H. Almannaa, M. Elhenawy, and H. A. Rakha, "A novel supervised clustering algorithm for transportation system applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 222–232, Jan. 2020.

[14] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for dl-based traffic classifier update," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 397–405.

[15] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[16] X. Wang, S. Zhang, Z. Qing, Y. Shao, C. Gao, and N. Sang, "Self-supervised learning for semi-supervised temporal action proposal," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021.

[17] N. Araslanov and S. Roth, "Self-supervised augmentation consistency for adapting semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021.

[18] Y. LeCun, "Self-supervised learning." Keynote Presentation at AAAI 2020 conference, Feb. 2020.

[19] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of things for smart healthcare: Technologies, challenges, and opportunities," *IEEE Access*, vol. 5, pp. 26 521–26 544, 2017.

[20] D. Perepelkin and M. Ivanchikova, "Problem of network traffic classification in multiprovider cloud infrastructures based on machine learning methods," Budva, Jun. 2021.

[21] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[22] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[23] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," in *ICISSP*, Portugal, Feb. 2016, pp. 407–414.

[24] P. Tang, Y. Dong, and S. Mao, "Online traffic classification using granules," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 1135–1140.

[25] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *IEEE INFOCOM*, Paris, France, Apr. 2019, pp. 1171–1179.
[26] E. Areström and N. Carlsson, "Early online classification of encrypted traffic streams using multi-fractal features," in *IEEE INFOCOM*, Paris, France, Apr. 2019, pp. 84–89.
[27] E. Liang, H. Zhu, X. Jin, and I. Stoica, "Neural packet classification," *Proceedings of the ACM Sigcomm*, Beijing, China, Aug. 2019.
[28] R. Li, X. Xiao, S. Ni, H. Zheng, and S. Xia, "Byte segment neural network for network traffic classification," in *IEEE ACM IWQoS*, Alberta, Canada, Jun. 2018, pp. 1–10.
[29] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "Xai meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, 2021.
[30] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1962–1976, 2021.
[31] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, p. 102985, 2021.
[32] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, Jan. 2019.
[33] H. Wen, Y. Wu, C. Yang, H. Duan, and S. Yu, "A unified federated learning framework for wireless communications: towards privacy, efficiency, and security," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 653–658.
[34] X. Zhang, F. Li, Z. Zhang, Q. Li, C. Wang, and J. Wu, "Enabling execution assurance of federated learning at untrusted participants," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 1877–1886.
[35] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 1698–1707.
[36] T. Huang, B. Ye, Z. Qu, B. Tang, L. Xie, and S. Lu, "Physical-layer arithmetic for federated learning in uplink mu-mimo enabled wireless networks," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 1221–1230.
[37] M. H. ur Rehman, K. Salah, E. Damiani, and D. Svetinovic, "Towards blockchain-based reputation-aware federated learning," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 183–188.
[38] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM*, Paris, France, Apr. 2019, pp. 2512–2520.
[39] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM*, Paris, France, Apr. 2019, pp. 1387–1395.
[40] Q. Li, Z. Wen, and B. He, "Federated learning systems: Vision, hype and reality for data privacy and protection," *ArXiv: 1907.09693*, Jul. 2019.
[41] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020.
[42] J. Shi, H. Zhao, M. Wang, and Q. Tian, "Signal recognition based on federated learning," in *IEEE INFOCOM*, Virtual conference, Jul. 2020, pp. 1105–1110.
[43] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *ICLR*, New Orleans, Apr. 2019.
[44] X. Zhu, N. Shu, H. Wang, and T. Wu, "A distributed traffic classification model based on federated learning," in *IEEE International Conference on Big Data Computing and Communications (BigCom)*, 2021, pp. 75–81.
[45] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, June 2019.
[46] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *IEEE INFOCOM*, Virtual conference, 2021.
[47] Y. Xiao, Y. Li, G. Shi, and H. V. Poor, "Optimizing resource efficiency for federated edge intelligence in iot networks," in *International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, China, Oct. 2020.
[48] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "Cutpaste: Self-supervised learning for anomaly detection and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021.
[49] K. Sun, Z. Lin, and Z. Zhu, "Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, New York, USA, Feb. 2020.
[50] A. Saeed, F. D. Salim, T. Ozcelebi, and J. Lukkien, "Federated self-supervised learning of multisensor representations for embedded intelligence," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1030–1040, 2021.
[51] J. Z. Bengar, J. van de Weijer, B. Twardowski, and B. Raducanu, "Reducing label effort: Self-supervised meets active learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1631–1639.
[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *NIPS*, Montreal, Canada, Dec. 2014, pp. 2672–2680.
[53] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient," in *AAAI conference on artificial intelligence*, San Francisco, California USA, Feb. 2017.
[54] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," in *NIPS*, Barcelona, Spain, Dec. 2016, pp. 2172–2180.
[55] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, "MGAN: Training Generative Adversarial Nets with Multiple Generators," in *ICLR*, Vancouver, Canada, May 2018.
[56] A. Ghosh, V. Kulharia, V. P. Namboodiri, P. H. Torr, and P. K. Dokania, "Multi-agent Diverse Generative Adversarial Networks," in *ICPR*, Beijing, China, Aug. 2018, pp. 8513–8521.
[57] H. Zhang, S. Xu, J. Jiao, P. Xie, R. Salakhutdinov, and E. P. Xing, "Stackelberg GAN: Towards Provable Minimax Equilibrium via Multi-Generator Architectures," Nov. 2018.
[58] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *AISTATS*, vol. 54, Ft. Lauderdale, USA, Apr. 2017, pp. 1273–1282.
[59] "Release 16," Jul. 2020. [Online]. Available: https://www.3gpp.org/release-16
[60] N. Mangrulkar, A. Bhagat Patil, and A. Pande, "Network attacks and their detection mechanisms: A review," *International Journal of Computer Applications*, vol. 90, Feb. 2014.
[61] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, "Multi-category classification by soft-max combination of binary classifier," in *Multiple Classifier Systems*, Berlin, Heidelberg, Jun. 2003, pp. 125–134.
[62] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, Sep. 1991.
[63] M. Rasouli, T. Sun, and R. Rajagopal, "FedGAN: Federated Generative Adversarial Networks for Distributed Data," *arXiv preprint arXiv: 2006.07228*, Jun. 2020.
[64] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *ArXiv preprint arXiv: 1806.00582*, Jun. 2018.
[65] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017*, 2018.
[66] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *ICML*, NY, USA, Jun. 2016, pp. 478–487.
[67] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation." in *IJCAI*, Melbourne, Australia, Aug. 2017, pp. 1753–1759.
[68] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *ICML*, Sydney, Australia, Aug. 2017, pp. 3861–3870.
[69] S. Vallender, "Calculation of the wasserstein distance between probability distributions on the line," *Theory of Probability & Its Applications*, vol. 18, no. 4, pp. 784–786, Feb. 1973.
[70] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Trans. Inf. Theor.*, vol. 37, no. 1, Jan. 1991.

APPENDIX A
PROOF OF PROPOSITION 1

Proof: According to the multi-player zero-sum game in Equation (2), classifier C_d tries to maximize the softmax quantity $V(C)$, where

$$\begin{aligned} V(C) &= \int_{\boldsymbol{x}} \sum_{m=1}^{M} \pi_m P_{G_d^m}(\boldsymbol{x}) \log C_d^m(\boldsymbol{x}) dx \\ &= \int_{\boldsymbol{x}} \pi_1 P_{G_d^1}(\boldsymbol{x}) \log(1 - \sum_{m=2}^{M} C_d^m(\boldsymbol{x})) dx \\ &\quad + \int_{\boldsymbol{x}} \sum_{m=2}^{M} \pi_m P_{G_d^m}(\boldsymbol{x}) \log C_d^m(\boldsymbol{x}) dx. \end{aligned} \tag{13}$$

In order to get the optimal classifier, we first calculate the functional derivative w.r.t. $C_d^m, m = 2, 3, ..., M$, we get:

$$\frac{\partial V(C)}{\partial C_d^m(\boldsymbol{x})} = \frac{\pi_m P_{G_d^m}(\boldsymbol{x})}{C_d^m(\boldsymbol{x})} - \frac{\pi_1 P_{G_d^1}(\boldsymbol{x})}{C_d^m(\boldsymbol{x})}. \tag{14}$$

Then, by setting the above equation to zero for $m \in \{2, 3, ..., M\}$, we obtain:

$$\frac{\pi_1 P_{G_d^1}(\boldsymbol{x})}{C_d^1(\boldsymbol{x})} = \frac{\pi_m P_{G_d^m}(\boldsymbol{x})}{C_d^m(\boldsymbol{x})}, \forall m \in \{2, 3, ..., M\}. \tag{15}$$

Given the fact that all the training samples used by classifier definitely come from generators, meaning $\sum_{m=1}^{M} C_d^m(\boldsymbol{x}) = 1$, we can get the optimal solution of classifier C_d^m, that is:

$$C_d^{m*}(\boldsymbol{x}; \theta_d) = \frac{\pi_m P_{G_d^m}(\boldsymbol{x})}{\sum_{i=1}^{M} \pi_i P_{G_d^i}(\boldsymbol{x})}, m \in \{1, 2, ..., M\}. \tag{16}$$

This concludes the proof. □

APPENDIX B
PROOF OF PROPOSITION 2

Proof: Let the optimal discriminator be ${D_d}^*$. Here, we directly use the conclusion in [52] that ${D_d}^* = \frac{P_{data}^d(\boldsymbol{x})}{P_{data}^d(\boldsymbol{x}) + P_{model}^d(\boldsymbol{x})}$. Denote the quantity function of generators as $V(\boldsymbol{G})$. According to the multi-player zero-sum game in (2), given the optimal discriminator ${D_d}^*$ above and classifier ${C_d}^*$ in proposition 1, generators try to minimize:

$$\begin{aligned} V(\boldsymbol{G}(\boldsymbol{x}; \omega_d)) =& \mathbb{E}_{\boldsymbol{x} \sim P_{data}^d} [\log \frac{P_{data}^d(\boldsymbol{x})}{P_{data}^d(\boldsymbol{x}) + P_{model}^d(\boldsymbol{x})}] \\ &+ \mathbb{E}_{\boldsymbol{x} \sim P_{model}^d} [\log \frac{P_{model}^d(\boldsymbol{x})}{P_{data}^d(\boldsymbol{x}) + P_{model}^d(\boldsymbol{x})}] \\ &- \lambda \{ \sum_{m=1}^{M} \pi_m \mathbb{E}_{\boldsymbol{x} \sim P_{G_d^m}} [\log \frac{\pi_m P_{G_d^m}(\boldsymbol{x})}{\sum_{i=1}^{M} \pi_i P_{G_d^i}(\boldsymbol{x})}] \}. \end{aligned} \tag{17}$$

The first two terms in the right-hand of (17) is same as in the standard GANs [52], which equals to $2 \cdot \text{JSD}(P_{data}^d \| P_{model}^d) - \log 4$. Here, we focus on the last term of this equation to clarify the effect of the auxiliary classifier.

In light of the conclusion in standard GANs, we can formulate (17) as follows:

$$\begin{aligned} V(\boldsymbol{G}(\boldsymbol{x}; \omega_d)) =& 2 \cdot \text{JSD}(P_{data}^d \| P_{model}^d) - \log 4 \\ &- \lambda \{ \sum_{m=1}^{M} \pi_m \mathbb{E}_{\boldsymbol{x} \sim P_{G_d^m}} [\log \frac{\pi_m P_{G_d^m}(\boldsymbol{x})}{\sum_{i=1}^{M} \pi_i P_{G_d^i}(\boldsymbol{x})}] \} \end{aligned} \tag{18}$$

In order to convey the proof more clearly, we use $-\lambda\{\hat{V}\}$ to denote the last term in (18), and let $\hat{P_G}(\boldsymbol{x}) = \sum_{m=1}^{M} \pi_m P_{G_d^m}(\boldsymbol{x})$. Then, We have:

$$\hat{V} = \sum_{m=1}^{M} \pi_m \mathbb{E}_{\boldsymbol{x} \sim P_{G_d^m}} [\log \frac{\pi_m P_{G_d^m}(\boldsymbol{x})}{\sum_{i=1}^{M} \pi_i P_{G_d^i}(\boldsymbol{x})}] \tag{19a}$$

$$= \sum_{m=1}^{M} \pi_m \mathbb{E}_{\boldsymbol{x} \sim P_{G_d^m}} [\log \frac{P_{G_d^m}(\boldsymbol{x})}{\hat{P_G}(\boldsymbol{x})}] + \sum_{m=1}^{M} \pi_m \log \pi_m \tag{19b}$$

$$\begin{aligned} &= \pi_1 \mathbb{E}_{\boldsymbol{x} \sim P_{G_d^1}} [\log \frac{P_{G_d^1}(\boldsymbol{x})}{\hat{P_G}(\boldsymbol{x})}] + ... \\ &\quad + \pi_M \mathbb{E}_{\boldsymbol{x} \sim P_{G_d^M}} [\log \frac{P_{G_d^M}(\boldsymbol{x})}{\hat{P_G}(\boldsymbol{x})}] + \sum_{m=1}^{M} \pi_m \log \pi_m. \end{aligned} \tag{19c}$$

Using the standard definition of KL divergence, the above formula can be expressed by the sum of multiple KL divergences and the negative entropy of $\boldsymbol{\pi}$, that is:

$$\begin{aligned} \hat{V} &= \pi_1 \cdot \text{KL}(P_{G_d^1}(\boldsymbol{x}) \| \hat{P_G}) + ... \\ &\quad + \pi_M \cdot \text{KL}(P_{G_d^M}(\boldsymbol{x}) \| \hat{P_G}) + \sum_{m=1}^{M} \pi_m \log \pi_m \end{aligned} \tag{20a}$$

$$= \sum_{m=1}^{M} \pi_m \text{KL}(P_{G_d^m}(\boldsymbol{x}) \| \hat{P_G}) + \sum_{m=1}^{M} \pi_m \log \pi_m \tag{20b}$$

$$= \text{JSD}_{\pi_1, \pi_2, ..., \pi_M}(P_{G_d^1}, P_{G_d^1}, ..., P_{G_d^M}) + \sum_{m=1}^{M} \pi_m \log \pi_m, \tag{20c}$$

where $\text{JSD}_{\pi_1, \pi_2, ..., \pi_M}(P_{G_d^1}, P_{G_d^1}, ..., P_{G_d^M})$ is the generalized Jensen-Shannon divergence [70]. Therefore, we can rewrite the quantity function of generators in equation (17) as:

$$V(\boldsymbol{G}(\boldsymbol{x}; \omega_d)) = 2 \cdot \text{JSD}(P_{data}^d \| P_{model}^d) - \log 4 - \lambda\{\hat{V}\} \tag{21a}$$

$$\begin{aligned} =& 2 \cdot \text{JSD}(P_{data}^d \| P_{model}^d) \\ &- \lambda \cdot \text{JSD}_{\pi_1, \pi_2, ..., \pi_M}(P_{G_d^1}, P_{G_d^2}, ..., P_{G_d^M}) \\ &- \lambda \sum_{m=1}^{M} \pi_m \log \pi_m - \log 4. \end{aligned} \tag{21b}$$

Ignoring the last two constant terms, we can finally get the objective function of generators in equation (6). Therefore, optimizing the generators is equivalent to minimizing the JD divergence between P_{data}^d and P_{model}^d while maximizing the differences among those generators. This concludes our proof. □

APPENDIX C
PROOF OF THEOREM 1

Proof: In order to further simplify the objective function of generators in Proposition 2, we firstly recast equation of $\hat{V}$ in (19):

$$\hat{V} = \sum_{m=1}^{M} \pi_m \mathbb{E}_{\boldsymbol{x}\sim P_{G_d^m}}[\log \frac{\pi_m P_{G_d^m}(\boldsymbol{x})}{\sum_{i=1}^{M} \pi_i P_{G_d^i}(\boldsymbol{x})}]. \tag{22}$$

It can be observed that $\hat{V} \leq 0$ and the equality occurs only if $\frac{\pi_m P_{G_d^m}(\boldsymbol{x})}{\sum_{i=1}^{M} \pi_i P_{G_d^i}(\boldsymbol{x})} = 1$. Therefore, we have $P_{G_d^m}(\boldsymbol{x}) \neq 0 \rightarrow \sum_{i\neq m} \pi_i P_{G_d^i}(\boldsymbol{x}) = 0 \rightarrow \forall i \neq m,\ P_{G_d^i}(\boldsymbol{x}) = 0$. In this case, different generators are well-separated without overlapping. Further, we can rewrite the objective function of generators in Proposition 2 under $\hat{V} = 0$:

$$\boldsymbol{G}_d{}^*(\boldsymbol{x}; \omega_d) = \arg\min_{\boldsymbol{G}} 2 \cdot \text{JSD}(P_{data}^d \| P_{model}^d). \tag{23}$$

This is a much simpler formulation and we can get its optimal solution directly, that is: $P_{data}^d = P_{model}^d$. When Assumption 1 holds, which constraints the real data distribution to be a mixture of M components $p_d^m(\boldsymbol{x}), m = 1, 2, ..., M$, we can easily get the results in (8b) and (8c), that is:

$$P_{G_d^m}^*(\boldsymbol{x}) = p_d^m(\boldsymbol{x}),\ \forall m = 1, 2, ..., M, \tag{24a}$$

$$P_{model}^d(\boldsymbol{x}) = \sum_{m=1}^{M} \pi_m P_{G_d^m}^*(\boldsymbol{x}) = P_{data}^d(\boldsymbol{x}). \tag{24b}$$

Up to now, we have proved that at the equilibrium point of the multi-player zero-sum game, generators can synthesize a mixture of traffic types, with the same distirbution as real in a local dataset. In order to evaluate the performance of the classifier, we substitute the well-separated optimal solution $P_{G_d^m}$ into Proposition 1. It can be seen that C_d^{m*} equals to 1 when $\boldsymbol{x}$ is drawn from $P_{G_d^m}(\boldsymbol{x})$; otherwise, C_d^{m*} is 0. Therefore, the optimal classifier can differentiate all the samples produced by different generators correctly. Formally, we have:

$$C_d^{m*}(\boldsymbol{x}) = \begin{cases} 1, & x \sim P_{G_d^w},\ w = m \\ 0, & x \sim P_{G_d^w},\ w \neq m \end{cases} \quad \forall m = 1, 2, \ldots, M, \tag{25}$$

Besides, when $P_{data}^d = P_{model}^d$, meaning generators are synthesizing high-quality samples, the classifier will also be able to classify real samples from the local dataset, e.g., labeling different local services. This concludes our proof. □