Federated Traffic Synthesizing and Classification Using Generative Adversarial Networks

Chenxin Xu*, Rong Xia*, Yong Xiao*‡,Yingyu Li*, Guangming Shi†‡, and Kwang-cheng Chen§
*School of Electronic Inform. & Commun., Huazhong Univ. of Science & Technology, China
†School of Artificial Intelligence, Xidian University, Xi'an, China
‡Pazhou Lab, Guangzhou, China
§Department of Electrical Engineering, University of South Florida, FL

***Abstract*—With the fast growing demand on new services and applications as well as the increasing awareness of data protection, traditional centralized traffic classification approaches are facing unprecedented challenges. This paper introduces a novel framework, Federated Generative Adversarial Networks and Automatic Classification (FGAN-AC), which integrates decentralized data synthesizing with traffic classification. FGAN-AC is able to synthesize and classify multiple types of service data traffic from decentralized local datasets without requiring a large volume of manually labeled dataset or causing any data leakage. Two types of data synthesizing approaches have been proposed and compared: computation-efficient FGAN (FGAN-I) and communication-efficient FGAN (FGAN-II). The former only implements a single CNN model for processing each local dataset and the later only requires coordination of intermediate model training parameters. An automatic data classification and model updating framework has been proposed to automatically identify unknown traffic from the synthesized data samples and create new pseudo-labels for model training. Numerical results show that our proposed framework has the ability to synthesize highly mixed service data traffic and can significantly improve the traffic classification performance compared to existing solutions.**

I. Introduction

Traffic classification and identification have been considered as fundamental for a wide range of applications such as network anomaly detection and identification, as well as resource slicing for different prioritized services and network access control. Traditional traffic classification [1]–[3] approaches are mostly centralized data processing approaches based on either supervised learning which often relies on a large volume of high-quality labeled datasets to train a model or clustering which divides data samples based on some observable features.

With fast growing demands on innovative services and applications, the next generation telecommunication technology is expected to support a plethora of new services and applications [4], [5]. Traditional traffic classification solutions are facing unprecedented challenges. More specifically, with new services and applications being introduced more frequently than ever, it becomes more difficult to collect a sufficient number of high-quality labeled datasets for each emerging service in its early stage of development. Clustering-based approaches suffer from limited accuracy and often incapable of identifying new unknown services, especially when the total volume of traffic data is relatively low. Furthermore, with the increasing awareness of data protection and user privacy, centralized data processing solutions relying on datasets uploaded to a single computer server will soon be impossible to support future services with stringent privacy and data protection requirements.

Federated learning (FL) is an emerging machine learning framework that enables collaborative model training based on decentralized datasets [6], [7]. It has been considered as one of key enabling technologies for data processing and model construction based on decentralized datasets distributed across a large networking system. In spite of merits, FL suffers from several known limitations. In particular, traditional FL (e.g. FedAvg [7]) is still a supervised learning approach and requires a large number of labeled datasets for training the model. Moreover, FL is known to suffer from slow convergence rate especially when the datasets are mostly unlabeled and heterogeneous (e.g. non-IID).

In this paper, we focus on traffic synthesizing and classification across decentralized datasets distributed throughout a large geographic area. A set of local fog servers, each can access a local dataset (e.g. dataset collected in a local sub-region), have been deployed for training a global traffic classification model without requiring any transferring or data leakage of local datasets. We propose Federated Generative Adversarial Networks and Automatic Classification (FGAN-AC), a novel decentralized traffic synthesizing and classification framework. FGAN-AC seamlessly integrate a federated generative learning model called FGAN into an automatic classification framework. In FGAN, two types of generative learning algorithms have been introduced: computation-efficient FGAN (FGAN-I) and communication-efficient FGAN (FGAN-II). The former algorithm FGAN-I deploys a single model (discriminator) at each fog server to compare the synthetic data generated by the global generator with its local dataset. The later FGAN-II algorithm deploys a complete Generative Adversarial Networks (GANs) [8] consisting of two models, a generator and a discriminator, at each fog server. The fog servers will then coordinate their model training via intermediate parameters exchanging which usually require much less communication overhead compared to exchanging real or fake (synthetic) data. We show that both generative learning algorithms create synthetic samples that capture the mixture of distribution of all the decentralized datasets. Based on these synthetic data samples, an automatic classification and model updating framework has been proposed to automatically identify unknown traffic and assign pseudo-labels for model training. Extensive simulations have been conducted to evaluate the performance of the proposed framework. Our results show that FGAN-AC is able to synthesize highly mixed service data traffic and can significantly improve the traffic classification performance.

II. Related Work

Federated Learning: Federated learning is an emerging distributed machine learning solution enabling collaborative model training based on decentralized datasets [9]–[12]. It has been proved to be an very effective solution for reducing communication overhead [9] and increasing data privacy [10]. Recent results suggest that combining FL and generative learning is possible to create synthetic data samples that capture the mixture of distributions associated with decentralized datasets [13]–[15]. For example, in [14], the authors proposed a framework that could apply multiple discriminators and generators to synthesize dataset with complex features.

Deep Learning Based Traffic Classification: Deep neural network is considered as a promising solution for traffic classification [1]–[3], [16], [17]. In particular, the authors in [17] introduced a novel solution with attention mechanism that extracted features of the payload segments and output the classification results through Softmax classification layer. The authors in [18] proposed an classification model that can identify unknown classes of data and then automatically updating its model. Different from the above works, we integrate federated generative learning into a automatic classification framework to create synthetic samples to assist the automatic traffic classification. To the best of our knowledge, this is the first work that exploits data synthesizing for improving automatic traffic classification performance.

III. System Model And Architecture Overview

A. System Model

We consider a fog computing networking system that can access and process a set of decentralized datasets. In particular, the system consists of the following elements:

Local datasets consist of service data generated by users in different service coverage areas. We assume there are no overlapping among different datasets and each dataset consists of a limited number of types of service traffic.

Local fog servers are deployed at different service areas to process its local datasets. We assume each fog server is associated with a local dataset. Due to the privacy consideration, each dataset can only be accessed by the associated fog server.

Global coordinator is the computing server connected to local fog servers across all service areas. The coordinator cannot access any local dataset, but can communicate with all the fog servers and process their uploaded intermediate results.

B. Architecture Overview

The main objective is to construct a joint training model based on the decentralized datasets to synthesize and classify different service traffic data. The joint training process will be coordinated by the global coordinator. We propose a novel framework called FGAN-AC that can capture the distributions of multiple decentralized datasets and perform global service synthesizing and classification without resulting any data leakage. In the FGAN-AC approach, we propose a three-step framework for automatic data synthesizing and classification described as follows:

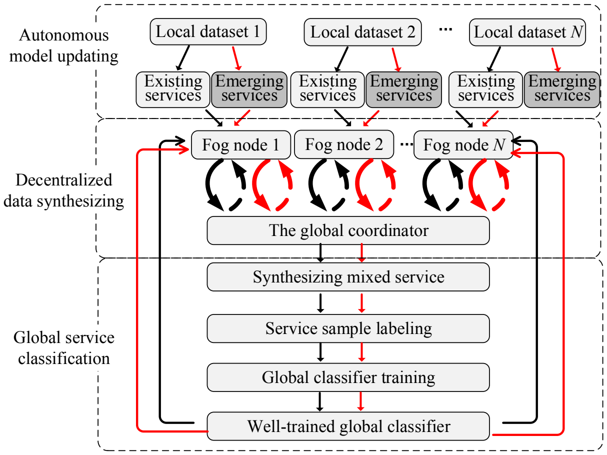
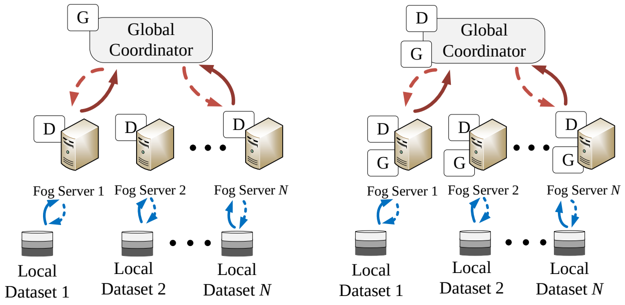


Fig. 1. Architecture overview

Decentralized Service Synthesizing: In this step, a large volume of synthetic service traffic will be created by the global coordinator to capture the mixed distributions of traffic data from all local datasets. We borrow the idea from federated learning to efficiently aggregate the models deployed and trained over different fog servers. Two types of FGAN approaches are proposed to coordinate among fog servers as shown in Fig. 2: computation-efficient FGAN, labeled as FGAN-I, and communication-efficient FGAN, labeled as FGAN-II.



(a) FGAN-I (b) FGAN-II

Fig. 2. FGAN Architecture.

FGAN-I: In this approach, only a single model is deployed at each fog server and the coordinator. More specifically, the coordinator will be implemented with a generator to create synthetic data to be transmitted to each fog server where a local discriminator is deployed to distinguish between the real traffic data and the synthetic data received from the generator. The generator in the coordinator will be compete with all the discriminators in fog servers to generate a mixture of data samples related to the services in local datasets.

FGAN-II: In this approach, a GANs model consisting of both a generator and a discriminator will be deployed at each fog server as well as the coordinator. In this case, all fog servers will coordinate their intermediate model training parameters with each other via the coordinator. The coordinator will first aggregate all the uploaded parameters and then broadcast the updated ones to each fog server.

Global Service Classification: In this step, a global service classification model will be trained based on the synthesized dataset. In particular, we first adopt a deep clustering approach, named as Deep Embedded Clustering (DEC) [19], to create pseudo-labels for high-quality synthesized service data samples. Different from traditional clustering algorithms which heavily rely on the original dataset, DEC further exploits the latent features of the input data to help separate different data traffic. Secondly, we feed the above pseudo-labeled dataset to a classifier located at the coordinator for supervised model training. Finally, the global coordinator will broadcast the well-trained classification model to all the connected fog servers for local data classification and unknown traffic filtering. Trained with global mixed service samples, the broadcasted classification model can not only distinguish local dataset for each individual fog server, but also identify the potential service data that only available in other datasets.

Autonomous Model Updating: In this step, a self-updating mechanism is adopted to autonomously identify newly arrived unknown service traffics and assign new pseudo-labels for them. And then those newly synthesized service samples will be merged with the original dataset to re-train the classifier. After the training process is completed, the global coordinator will broadcast the updated classifier again for future traffic identification.

IV. FGAN-AC Architecture

In this section, we present more detailed descriptions of the three main steps of FGAN-AC.

A. Decentralized Service Synthesizing

As mentioned earlier, one main objective of this paper is to synthesize sufficient service traffic samples that can capture the distribution of the real service datasets located in multiple fog servers. In particular, we utilize a popular deep generative neural network, GANs, to produce high-quality samples. GANs consist of two neural networks referred to as generator G and discriminator D that competitively play a two player min-max game. The generator tries to fool the discriminator with synthetic samples mapped from Gaussian noises, and the discriminator aims to differentiate the real data samples from the fake ones. The payoff value of GANs can be formulated as:

$$\begin{aligned}\min_G \max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim P_{data}}[\log D(\boldsymbol{x})] \\ +\mathbb{E}_{\boldsymbol{z}\sim P_z}[\log(1-D(G(\boldsymbol{z})))],\end{aligned} \tag{1}$$

where $\boldsymbol{x}$ denotes data samples drawn from real dataset, P_{data} is the data distribution of real dataset, and $\boldsymbol{z}$ is drawn from Gaussian space P_z. In this step, we train a decentralized GANs model over multiple datasets to efficiently capture the distribution of mixed service traffic collected by different fog servers. In particular, we introduce two types of FGAN approaches: computation-efficient FGAN-I and communication-efficient FGAN-II.

1) FGAN-I: As shown in Fig. 2(a), in FGAN-I, a global generator G located at the coordinator is competing with N decentralized discriminator $\{D_1, D_2, \ldots, D_N\}$, each associated with an exclusive dataset d_n. We can write the pay-off value of FGAN-I as follows:

$$\min_G V(G) = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\boldsymbol{z}\sim P_z}[\log(1-D_n(G(\boldsymbol{z})))], \tag{2a}$$

$$\begin{aligned}\max_{D_n} V(D_n) = \mathbb{E}_{\boldsymbol{x}\sim P_{d_n}}[\log D_n(\boldsymbol{x})] \\ +\mathbb{E}_{\boldsymbol{z}\sim P_z}[\log(1-D_n(G(\boldsymbol{z})))],\end{aligned} \tag{2b}$$

where P_{d_n} is the data distribution of real dataset d_n. In each joint training round, G first generates two batches of fake samples with batch size b, denoted as $\boldsymbol{x}^{(d)}$ and $\boldsymbol{x}^{(g)}$, and sends them to all the connected fog servers. $\boldsymbol{x}^{(d)}$ is used for training the discriminator, and $\boldsymbol{x}^{(g)}$ is used for the discriminator to calculate loss value of the generator. Secondly, each discriminator D_n deployed on a fog server draws a batch of samples $\boldsymbol{x}^{(r)}$ from its local dataset d_n and performs a E-step updating by ascending the gradient of loss value L^n_{disc}. Thirdly, each updated discriminator D_n will utilize $\boldsymbol{x}^{(g)}$ to calculate a loss value L^n_{gen} for the generator. After receiving all the feedbacks from N fog servers, the global generator G will also update by descending the gradient of an aggregated loss value L_{gen}. We summarize the training process of FGAN-I in Algorithm 1. The above mentioned loss functions for model updating are formulated as follows:

$$L^n_{disc} = \frac{1}{b}\sum_{x\in\boldsymbol{x}^{(r)}}\log(D_n(x)) + \frac{1}{b}\sum_{x\in\boldsymbol{x}^{(d)}}\log(1-D_n(x)), \tag{3a}$$

$$L^n_{gen} = \frac{1}{b}\sum_{x\in\boldsymbol{x}^{(g)}}\log(1-D_n(x)), \tag{3b}$$

$$L_{gen} = \frac{1}{N}\sum_{n=1}^{N}L^n_{gen}. \tag{3c}$$

Algorithm 1 FGAN-I

```
Input: maximum global training round $I$, local training epoch $E$;
Output: global generator $G$ ;
for global training round $i \leq I$ do
    Global generator broadcasts two batches of synthesized samples $\boldsymbol{x}^{(g)}$ and $\boldsymbol{x}^{(d)}$;
    for each local discriminator parallel do
        $L^n_{gen} \leftarrow LocalUpdate(\boldsymbol{x}^{(g)}, \boldsymbol{x}^{(d)})$
    end for
    Aggregate local loss values by Equation (3c)
    Update generator by descending the gradient of the aggregated loss;
end for
Function $LocalUpdate$ $(\boldsymbol{x}^{(g)}, \boldsymbol{x}^{(d)})$
for local training round $e \leq E$ do
    1) Sample a batch of real data $\boldsymbol{x}^{(r)}$;
    2) Update discriminator by ascending the gradient of Equation (3a);
    3) Calculate the loss value of global generator by Equation (3b);
    4) Upload the loss value to global generator;
end for
return loss value of global generator
```

In FGAN-I, only a single component of GANs is deployed at each fog server and coordinator, which significantly reduces the computation complexity. However, a large amount of synthesized data samples need to be transmitted between the global coordinator and each fog server in the training process. When the batch size is large or the data sample is complex which is common in practice, the communication overhead of FGAN-I will be extremely heavy.

2) FGAN-II: In order to avoid redundant data sample transmission, in this approach, both the generator and the discriminator are deployed at each fog server as well as the global coordinator as shown in Fig. 2(b). Different from FGAN-I, we borrow the idea from federated learning [7] to efficiently aggregate information obtained from multiple decentralized datasets. Firstly, in each global training round, the global GANs model located at the coordinator broadcasts its latest model parameters, denoted as w for generator and θ for discriminator, to each fog server. Secondly, local GANs models located at different fog servers update their parameters according to the received w and θ, and then performs an E-step local model updating by ascending the gradient of loss function L_D^n and descending the gradient of loss function L_G^n. After the local training process is completed, local parameters denoted as w_n and θ_n will be report to the global coordinator. At last, the global coordinator aggregates all the received local parameters by:

Algorithm 2 FGAN-II

```
Input: local GANs with parameters $w_n$ and $\theta_n$, global GANs with parameters $w$ and $\theta$, global training round $I$, number of local training epoch $E$;
Output: $w$ and $\theta$;
for global training round $i \leq I$ do
    for each local GANs parallel do
        $(w_n, \theta_n) \leftarrow LocalUpdate(w, \theta)$;
    end for
    Aggregate received parameters by Equation (4);
end for
return $w$,$\theta$
Function LocalUpdate ($w$, $\theta$)
$w_n \leftarrow w$, $\theta_n \leftarrow \theta$;
for i=1 → E do
    Update $\theta_n$ by ascending the gradient of Equation (5a);
    Update $w_n$ by descending the gradient of Equation (5b);
end for
return $w_n, \theta_n$
```

$$w = \frac{1}{N}\sum_{n=1}^{N} w_n, \quad \theta = \frac{1}{N}\sum_{n=1}^{N} \theta_n. \tag{4}$$

Following the notations in FGAN-I, $\boldsymbol{x}^{(r)}$ is a batch of real service samples, $\boldsymbol{x}^{(g)}$ and $\boldsymbol{x}^{(d)}$ denotes a batch of synthesized samples used for training generator and discriminator, respectively. The loss functions of the discriminator and the generator at each fog server n can be formulated as:

$$L_D^n = \frac{1}{b}\sum_{x \in \boldsymbol{x}^{(r)}} \log(D_n(x)) + \frac{1}{b}\sum_{x \in \boldsymbol{x}^{(d)}} \log(1 - D_n(x)), \tag{5a}$$

$$L_G^n = \frac{1}{b}\sum_{x \in \boldsymbol{x}^{(g)}} \log(1 - D_n(x)). \tag{5b}$$

We present a more detailed description of FGAN-II in Algorithm 2.

In FGAN-II, instead of transmitting the synthesized data samples, we only perform coordination on the intermediate training parameters that are independent from the data dimension and training batch size. Unfortunately, both fog servers and the coordinator need to allocate more computation resources for the joint training process.

B. Global Service Classification

In this step, we establish a global service classifier based on the synthetic samples produced by the generator at the global coordinator. Trained with high-quality synthesized global mixed service traffic data across all the decentralized local datasets, this classifier is able to identify different services at each fog server. Two procedures are needed to generate high accuracy global service classification: service sample labeling and global classifier training. We present a step-by-step description in Algorithm 3.

1) Service Sample Labeling: In this procedure, we adopt a deep clustering method, referred to as DEC [19], to create a pseudo-label for each sample in the synthesized dataset, denoted as $T = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{n_T}\}$. In particular, we optimize a pre-trained encoder and cluster centroid by minimizing the KL divergence between a target distribution p_{ij} and a soft assignment distribution q_{ij} as follows:

$$L = KL(P||Q) = \sum\nolimits_i \sum\nolimits_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{6}$$

Let Z_i be the latent feature of $\hat{x}_i$ extracted by the above mentioned encoder, and μ_j be the centroid of cluster j. Then, q_{ij} can be formulated as:

$$q_{ij} = \frac{1 + (||Z_i - \mu_j||^2)^{-1}}{\sum\limits_{j=1}^{k} (1 + (||Z_i - \mu_j||^2))^{-1}}, \tag{7}$$

where k is number of clusters. In order to improve the cluster purity, the target distribution p_{ij} is formulated as first squaring the soft assignment distribution and then normalizing it

$$p_{ij} = \frac{{q_{ij}}^2 \big/ f_j}{\sum_j {q_{ij}}^2 \big/ f_j}, \tag{8}$$

where $f_j = \sum_i q_{ij}$ is the soft clustering probability. In this way, data points $\{Z_1, Z_2, \cdot, Z_{n_T}\}$ can be assigned with a higher confidence. After the training process is completed, all the samples $\hat{x}_i$ in the synthesized dataset T will be labeled via the optimal solution of maximizing q_{ij}.

To find the optimal number of clusters, Bayesian information criterion (BIC) is also adopted in our algorithm. For each possible cluster number k, we calculate a BIC value for clustering performance evaluation by:

$$BIC_k = n_T \times \ln(\frac{R}{n_T}) + k \times \ln(n_T), \tag{9}$$

$$R = \sum_{j=1}^{k} \sum_{Z_i \in \mu_j} \sqrt{(Z_i - \mu_j)^2}, \tag{10}$$

where R is the sum of all the Euclidean distances between latent features Z_i and its corresponding cluster centroid μ_j, n_T is the number of samples in the synthesized dataset T and optimal number of cluster, denoted as k^*, will be found by first calculating BIC values with all possible $k \in \{1, 2, ..., K_{max}\}$, then choosing the k whose BIC value decreasing the most.

Algorithm 3 Autonomous Service Classification

```
Input: synthesized dataset $T = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{n_T}\}$; maximum number of clustering $K_{max}$; stopping threshold $\delta$; maximum iterations $I_{max}$;
Output: global classifier $C$;
for $k = \{1, 2, ..., K_{max}\}$ do
    for iteration round $i \leq I_{max}$ do
        1) initialize encoder and cluster centroid $\mu_j$;
        2) mapping $\hat{x}_i$ to $Z_i$ with the encoder;
        3) compute the distance between $Z_i$ and $\mu_j$ by equation (7);
        4) label each $Z_i$ with the closest cluster index;
        5) save labeled dataset as:
            $T_{new} = \{(\hat{x}_1, y_1), (\hat{x}_2, y_2), \ldots, (\hat{x}_{n_T}, y_{n_T})\}$;
        6) optimize encoder and $\mu$ according the clustering loss in equation (6);
        7) re-calculate $y$ with the updated clustering model;
        8) if sum$(y_{old} \neq y)/n_T \leq \delta$ then
            Stop training;
    end for
    Calculate $BIC_k$ value by equation (9);
    $\Delta BIC$=$BIC_k$-$BIC_{k-1}$;
end for
Find the optimal clustering model with maximum $\Delta BIC$ value;
Calculate assignment probability $q_{ij}$;
Label $\hat{x}_i$ with $\arg\max_j q_{ij}$;
Train service classifier $C$ with $T_{new}$;
Broadcast $C$ to all the fog servers;
```

2) Global Classifier Training: After autonomous service sample labeling, the original synthesized dataset T will update to $T_{new} = \{(\hat{x}_1, y_1), (\hat{x}_2, y_2), \ldots, (\hat{x}_{n_T}, y_{n_T})\}$, where y_i is the pseudo-label of $\hat{x}_i$. Then, this labeled dataset will be utilized for training a global service classifier C. When the model converges, this well-trained classifier will be broadcasted to all the connected fog servers for local service identification.

C. Autonomous Model Updating

In this step, we introduce a self-updating scheme to automatically update both the synthesizing and the classification model when new types of service traffic arrive. The proposed self-updating scheme consists of two procedures described as follows:

1) Unknown Service Filtering: For a given service traffic sample, the well-trained service classifier C can output a vector indicating the probabilities of it belongs to the known service classes. We can monitor whether a new type of service emerges by periodically feeding a bulk of real service traffic data to the classifier C. Let $\mathbf{o}$ be the multi-dimensional output of classifier C when feeding with a real service sample s, and we use $o^* = max(\mathbf{o})$ to denote the confidence score of s. When s belong to a newly appeared service type for all the connected fog servers, o^* will be relatively small, indicating the well-trained classifier C cannot assign this sample to any known service type with a high probability. Thus, we introduce a threshold value α to determine whether a traffic sample belongs to a known service type or not. That is, samples with $o^* < \alpha$ will be treated as unknown service types.

2) Data Synthesizing and Classifier Updating: Once the new type of service is identified in the above procedure, FGAN will be trained again with the updated datasets, which contain both the unknown service samples and the original dataset. In this way, a new mixture of traffic samples will be synthesized which capturing the updated services data distribution. Similarly, the newly synthesized dataset will first be labeled with pseudo-labels, and then used for classifier training. Finally, the re-trained classifier which is able to distinguish more diversified services will be broadcasted to each fog server for service classification again.

V. Performance Evaluation

A. Dataset Description and Environmental Setup

To evaluate the performance of our proposed framework on both service data synthesizing and classification, we conduct extensive simulations based on a real-world dataset "ISCX VPN-nonVPN dataset" (ISCXVPN2016) which consisting of rich diversities of traffic samples including Emails, Facebook, SCP, etc. We consider data samples associated with 10 servers among which 8 services are considered as the known traffic classes and 2 services are assumed to be unknown services as shown in Table I. In our simulations, data samples of both known and unknown services are uniformly randomly allocated to fog servers.

All the simulations are conducted on a workstation with Intel(R) Core(TM) i5-8500 CPU@3.00GHz, 16.0 GB RAM@2133 MHz, 2 TB HD and four NVIDIA Corporation GP102 [TITAN X] GPUs and implemented with Pytorch library.

TABLE I
Detailed Information of the Dataset for Evaluation

	Application	
Existing classes	$Email$	$Facebook$
	$Netflix$	$SFTP$
	$Skype$	$Vimeo$
	SCP	$Twitter$
Unknown classes	$Youtube^*$	$VOIPbuster^*$

B. Evaluation Results

To evaluate and compare the data synthesizing performance of FGAN-I and FGAN-II, we measure their communication overheads using the numbers of transmitted data bits and computational loads using the running time per global training round in Table II . We also use Maximum Mean Discrepancy (MMD) to evaluate the quality of samples synthesized by two FGAN methods which can be calculated by:

$$MMD^2 (P_r, P_g) = \underset{x_r, x_r' \sim P_r, x_g, x_g' \sim P_g}{\mathbb{E}} \left[g\left(x_r, x_r'\right) - 2g\left(x_r, x_g\right) + g\left(x_g, x_g'\right) \right], \tag{11}$$

where x_r and x_r' are real samples drawn from real data distribution P_r, x_g and x_g' are synthesized samples drawn from fake data distribution P_g. $g(\cdot)$ represents a kernel function that maps the sample space to the Hilbert space. It can be observed that the communication overhead of FGAN-I is proportional to the batch size b, while the communication overhead of FGAN-II is mainly determined by the size of the model parameters. Compared to FGAN-II, FGAN-I consumes less running time per training round. With the number of training rounds increases, both FGAN-I and FGAN-II can obtain lower MMD scores which means that the quality of synthesized sample is improved.

To evaluate the performance of classification , we consider three evaluation metrics: Recall, model precision and F1 score, we consider two scenaries for each solution: (1) original: only 8 known services have been evaluated and classified, and (2) updated: total 10 services including 8 known and 2 unknown services have been evaluated. We

TABLE II
EVALUATION OF FGAN-I AND FGAN-II

I	b	N	MMD		Com.overhead (MB)		Time per round (s)	
			FGAN-I	FGAN-II	FGAN-I	FGAN-II	FGAN-I	FGAN-II
5	64	30	2.25	1.94	6.3	4.08	651.32	1571.77
10	64	30	0.67	1.45	6.3	4.08	651.32	1571.77
20	64	30	0.26	0.56	6.3	4.08	651.32	1571.77
5	128	30	0.96	0.96	12.5	4.08	347.41	1051.89
10	128	30	0.83	0.83	12.5	4.08	347.41	1051.89
20	128	30	0.35	0.38	12.5	4.08	347.41	1051.89
5	128	50	1.07	1.33	12.5	4.08	405.74	1288.84
10	128	50	0.79	0.62	12.5	4.08	405.74	1288.84
20	128	50	0.34	0.43	12.5	4.08	405.74	1288.84

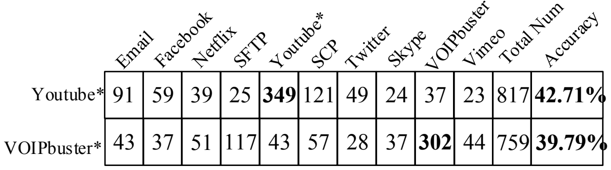
TABLE III
PERFORMANCE EVALUATION RESULTS OF THE CLASSIFIERS

		Recall	**Precision**	**F1 score**
MLP	original	0.3746	0.3787	0.3725
	updated	0.3219	0.3273	0.3298
1D-CNN	original	0.4128	0.4181	0.4135
	updated	0.3767	0.3748	0.3654
2D-CNN	original	0.4584	0.4564	0.4627
	updated	0.4289	0.4263	0.4257

use TP to denote the true positive decision that assigns two similar samples to the same cluster, and TN to denote the true negative decision that assigns two different samples to different clusters. We name the possible erroneous outputs of a classifier as false positive (FP) decision that assigns two different samples to the same cluster and false negative (FN) decision that assigns two similar packets to different clusters. We can then calculate those evaluation metrics as follows:

$$R = \frac{TP}{TP+FN}, \quad P = \frac{TP}{TP+FP}, \quad F1 = \frac{2P \times R}{P+R} \qquad (12)$$

We considere three network architectures for classifier: Multilayer Perceptron (MLP), 1D-CNN and 2D-CNN. Note that we resize the input data to 1 × 1600 for both MLP and 1D-CNN, and 40 × 40 for 2D-CNN. It can be observed that the 2D-CNN-based classifier offers the best performance. The MLP-based classifier has the lowest classification accuracy among three solutions which is around 37%. We can also observe that compared to the original classifier, the performance of the updated classifier also degrades, which may be attributed to two aspects: some unknown service samples are not detected, and some errors occured in the clustering process for the unknown service types, to slightly worsen the subsequent classification.



	Email	Facebook	Netflix	SFTP	Youtube*	SCP	Twitter	Skype	VOIPbuster*	Vimeo	Total Num	Accuracy
Youtube*	91	59	39	25	**349**	121	49	24	37	23	817	**42.71%**
VOIPbuster*	43	37	51	117	43	57	28	37	**302**	44	759	**39.79%**

Fig. 3. Classification results for unknown classes(2D-CNN)

We present the classification result of the unknown services in Figure 3 which we present a confusion matrix to show the detailed classification results of two unknown service traffics. The confusion matrix calculates the classification accuracy by comparing the actual class of each data sample with the estimated one. We can observe that the accuracy for Youtube has reached roughly 42.71%, while the accuracy for VOIPbuster is around 39.79%.

VI. CONCLUSION

In this paper, we have proposed a novel framework that not only synthesizes the mixture of distribution of decentralized dataset but also classifies multiple unknown classes of data traffic. Furthermore, we have compared two types of FGAN approaches and propose an automatic traffic classification framework to autonomous classify unknown services and assign new pseudo-labels. Extensive simulations have been conducted and the numerical results demonstrate that our proposed framework is able to synthesize high-quality mixed service data traffic as well as significantly improve the service classification performance.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grants 62071193 and 61632019, the Key R & D Program of Hubei Province of China under Grant 2020BAA002, and China Postdoctoral Science Foundation under Grant 2020M672357.

REFERENCES

[1] P. Wang *et al.*, “Datanet: Deep learning based encrypted network traffic classification in sdn home gateway,” *IEEE Access*, vol. 6, pp. 55 380–55 391, Sep. 2018.

[2] M. Lotfollahi *et al.*, “Deep packet: A novel approach for encrypted traffic classification using deep learning,” May. 2019.

[3] I. Masi *et al.*, “Learning pose-aware models for pose-invariant face recognition in the wild,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 379–393, Feb. 2019.

[4] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, “Toward self-learning edge intelligence in 6g,” *IEEE Communications Magazine*, vol. 58, no. 12, pp. 34–40, Dec. 2020.

[5] Y. Xiao and M. Krunz, “Distributed optimization for energy-efficient fog computing in the tactile Internet,” *IEEE J. Sel. Area Commun.*, vol. 36, no. 11, pp. 2390–2400, Nov. 2018.

[6] Augenstein *et al.*, “Generative models for effective ml on private, decentralized datasets,” *arXiv preprint arXiv:1911.06679*, 2019.

[7] B. McMahan *et al.*, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in *The Proceedings of AISTATS*, vol. 54, Fort Lauderdale, FL, Apr. 2017, pp. 1273–1282.

[8] Goodfellow *et al.*, “Generative adversarial networks,” *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, Mar. 2014.

[9] J. Konecný *et al.*, “Federated learning: Strategies for improving communication efficiency,” *NIPS*, vol. abs/1610.05492, Mon. 2016.

[10] Bonawitz *et al.*, “Practical secure aggregation for privacy-preserving machine learning,” Oct. 2017, pp. 1175–1191.

[11] W. Y. B. Lim *et al.*, “Federated learning in mobile edge networks: A comprehensive survey,” *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, Sep. 2020.

[12] Wen *et al.*, “A unified federated learning framework for wireless communications: towards privacy, efficiency, and security,” Toronto, ON, Canada, Jul. 2020, pp. 653–658.

[13] C. Hardy *et al.*, “Md-gan: Multi-discriminator generative adversarial networks for distributed datasets,” in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, United States,Epub, Nov. 2019, pp. 866–877.

[14] Rasouli *et al.*, “Fedgan: Federated generative adversarial networks for distributed data,” *arXiv preprint arXiv:2006.07228*, Jun. 2020.

[15] D. J. Im *et al.*, “Generative adversarial parallelization,” *arXiv e-prints*, p. arXiv:1612.04021, Dec. 2016.

[16] S. Deena *et al.*, “Recurrent neural network language model adaptation for multi-genre broadcast speech recognition and alignment,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 3, pp. 572–582, Sep. 2019.

[17] Li *et al.*, “Byte segment neural network for network traffic classification,” in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–10.

[18] J. Zhang *et al.*, “Autonomous unknown-application filtering and labeling for dl-based traffic classifier update,” in *IEEE INFOCOM 2020*, Toronto, Jul. 2020, pp. 397–405.

[19] J. Xie *et al.*, “Unsupervised deep embedding for clustering analysis,” in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, June*, vol. 48, pp. 478–487.