GFL: Federated Learning on Non-IID data via Privacy-preserving Synthetic data

Yihang Cheng*, Lan Zhang*†, Anran Li‡,
*School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
†Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, China
‡Nanyang Technological University, Singapore
whcyh@mail.ustc.edu.cn,zhanglan@ustc.edu.cn,anranLi@mail.ustc.edu.cn

2023 IEEE International Conference on Pervasive Computing and Communications (PerCom) | 978-1-6654-5378-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/PERCOM56429.2023.10099110

***Abstract*—Federated learning (FL) enables large amounts of participants to construct a global learning model, while storing training data privately at local client devices. A fundamental issue in FL systems is the susceptibility to the highly skewed distributed data. A series of methods have been proposed to mitigate the Non-IID problem by limiting the distances between local models and the global model, but they cannot address the root cause of skewed data distribution eventually. Some methods share extra samples from the server to clients, which requires comprehensive data collection by the server and may raise potential privacy risks. In this work, we propose an efficient and adaptive framework, named Generative Federated Learning (GFL), to solve the skewed data problem in FL systems in a privacy-friendly way. We introduce Generative Adversarial Networks (GAN) into FL to generate synthetic data, which can be used by the server to balance data distributions. To keep the distribution and membership of clients' data private, the synthetic samples are generated with random distributions and protected by a differential privacy mechanism. The results show that GFL significantly outperforms existing approaches in terms of achieving more accurate global models (e.g., 17%-50% higher accuracy) as well as building global models with faster convergence speed without increasing much computation or communication costs.**

***Index Terms*—Federated Learning, Non-IID, Membership Inference Attack**

I. Introduction

Federated learning (FL) [1], [2] is one emerging technology, which enables multiple local clients to collaboratively train a machine learning model by iteratively exchanging model parameters between participants and a centralized server, meanwhile keeping their data locally. In reality, the clients can be any endpoints, such as laptops, phones or smart home devices. Thus, FL is able to be applied in the context of pervasive computing and has attract many researchers [3]–[6]. Besides, a series of previous efforts have been devoted to designing FL algorithms for diverse scenarios [7]–[9]. However, FL still faces the statistical challenge. FL leverages stochastic gradient descent (SGD) optimization method, which is widely used in training deep networks with good empirical performance. The independent sampling of the training data is important to ensure that the stochastic gradient is an unbiased estimate of the full gradient. In practice, however, it is unrealistic to assume that the local data on each participant's device is always identically independently distributed (IID). It has been shown that the accuracy of convolutional neural networks trained with FedAVG algorithm suffer a significant reduction in accuracy (e.g., up to 51% reduction for CIFAR-10 and 55% reduction for keyword spotting (KWS) datasets) when faced with the highly skewed non-IID data [10].

Towards improving the performance of FL systems, we focus on addressing the challenge of data heterogeneity. Previous work addressing the data heterogeneity problem in FL can be divided into two branches. One branch focuses on improving the model updating mechanism. FedProx [11] limits the distances between local models and the global model by adding a weighted L_2 regularization term to the local loss function. This method makes few modifications to the original FedAVG method. However, to achieve a more accurate global model, clients are required to spend much more time to tune the weight parameters, which makes it impractical in real FL scenarios. SCAFFOLD [12] proposes to use control variates to correct for the "client-drift" in local updates. The control variates can be calculated in two ways, one is to calculate the gradient of the loss w.r.t. global models, and the other is to re-use the previously computed gradients to update the control variates. However, the first way involves making an additional pass over the local data to compute the gradient, which requires large extra computation cost; while the second way is unstable in producing a high performance global model. The other branch proposes to share a small number of extra samples among the server and local clients to mitigate the effects of non-iid. [10] gives us an intuitive solution, which is to directly share some data with all clients. It shows that 5% of local data contributes to the global accuracy for about 30%. But this method is against the fundamental concept of FL. Sharing raw data should be forbidden. So, instead of sharing the original data, [13] shares synthetic samples generated with methods like Generative Adversarial Network (GAN) to protect the privacy of raw local data. However, some issues caused by GAN still need to be analyzed. For example, a potential privacy problem is illustrated in [14] that membership inference attacks can be conducted on the synthetic data even in black-box settings. Besides, mode collapse as a serious side-effect of GAN can greatly effect the quality of synthetic data.

In this work, we aim to solve the highly skewed data distribution problem by proposing a GAN-based method while preserving clients' privacy, so as to construct high-performance global models. The main challenges are as follows. *(1) How*

to share the synthetic data while protecting local clients' privacy? Existing GAN-based solution towards data heterogeneity [13] does not analyze the leakage of privacy information such as membership and data distribution. Sharing synthetic data with other clients besides the server also introduces another uncontrollable leakage. Therefore, it is necessary to control the privacy leakage with reasonable metrics. *(2) How to use the synthetic data collected from clients properly on the server?* The straight combination of the synthetic data would probably still be non-iid, which cannot assist the server well. We need to train the global model in an iid way. *(3) How to deal with the potential mode collapse and low-quality data introduced by GAN?* Mode collapse and low-quality data are common issues in GAN and hard to avoid. A proper trick is needed to deal with them.

To address above challenges, we propose an efficient and adaptive framework, named Generative Federated Learning (GFL), to solve the non-iid problem in FL systems in a privacy-friendly way. We introduce GAN [15] into FL to help local clients to generate synthetic data, which can be used by the server as extra information about labels apart from the model parameters. To achieve the privacy-preserving goal, we choose to use DPGAN [16], which ensures differential privacy for original data so that the original data which is used to train the local model and GAN model is not leaked. We generate more samples than required in order to ensure that we can downsample a random distribution different from the training dataset. After receiving the synthetic datasets from the clients, the server calculates the number of samples on each label and chooses an iid subset which has the same amount of samples on all labels. Thus, the global model can be trained in the iid way. While training, the mode collapse issue is solved by gradually decreasing the training epochs of the global model with the epoch decay parameter.

The main contributions are summarized as follows:

•We propose GFL to privately solve the data heterogeneity problem by using a GAN-based method, and construct high-performance global models for FL systems. GFL boosts the FL training process while keeping the privacy of the membership and distribution of the original data in clients.

•We propose a privacy-preserving data generation workflow to generate synthetic samples which meet our privacy setting. DPGAN is used to generate samples which meet differential privacy to protect clients from membership inference attacks. Apart from membership, data distribution is another private information. We generate a large set of synthetic data and choose a subset with a random distribution to hide the real data distribution of the clients.

•We "iidify" the synthetic data to train the global model in an iid way, and use the epoch decay parameter to tackle mode collapse and to avoid the use of low-quality synthetic data.

•We evaluate our design via extensive experiments using three FL optimization algorithms, e.g., FedAVG, FedProx and SCAFFOLD, on two public and classic datasets, e.g., EMNIST and CIFAR-10. Results demonstrate that GFL achieves much better results than existing methods, with 47.94%, 14.06%, 27.43%, and 42.79%, 17.67%, 17.32% higher test accuracy than the original FedAvg, FedProx and SCAFFOLD for the two datasets, respectively. Compared with the most related work [13], which does not reduce the training epochs on synthetic data, GFL achieves 23.45% higher test accuracy. Compared to the original FL training, GFL only causes a small amount of extra overhead. Moreover, the privacy leakage of synthetic data is evaluated by the membership inference attack [14] and the results show that the accuracy rate of the attack drops given a small privacy level ϵ.

Algorithm 1: FederatedAveraging (FedAVG)

Input : The number of communication rounds T, the number of total clients K, and the proportion of selected clients ϕ; the local batch size B, the number of local epochs E, and the learning rate η.

Output: The final global model w_T.

Data : The local dataset $\mathcal{P}_k$ with size n_k on each client $k = 1, 2, \cdots, K$.

Server executes:
- initialize w_0
- **for** *each round* $t = 1, 2, \cdots, T$ **do**
  - $m \leftarrow \max(\phi \cdot K, 1)$
  - $C_t \leftarrow$ (random subset of m clients)
  - **for** *each client* $k \in C_t$ **do**
    - $w_t^k \leftarrow \text{ClientUpdate}(k, w_{t-1})$
  - $w_t \leftarrow \sum_{k=1}^{m} \frac{n_k}{n} \cdot w_t^k$
- **return** w_T

ClientUpdate (k, w)**:**
- $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
- **for** *each local epoch* $i = 1, 2, \cdots, E$ **do**
  - **for** *batch* $b \in \mathcal{B}$ **do**
    - $w \leftarrow w - \eta \cdot \nabla l_k(w; b)$
- **return** w

II. Related Works and Preliminaries

A. Data Heterogeneity in FL

Federated learning allows multiple clients to train a global model while keeping local data private [1]. Federated Averaging (FedAVG) [1], a heuristic method based on averaging local SGD updates, has been shown to work well empirically. In the t-th iteration, given a selection ratio ϕ, the server S randomly selects a subset S_t of $m = \max\{\phi \cdot K, 1\}$ clients and distributes the parameters of the current model w_{t-1} to them. Each selected client $k \in S_t$ computes a local model $w_t^k = w_{t-1} - \eta \nabla L(w_{t-1})$ with the learning rate η, and sends w_t^k back to the server. The server aggregates updates from the selected clients and updates $w_t = \sum_{k=1}^{m} \frac{n_k}{n} \cdot w_t^k$. The algorithm is summarized in Algorithm 1.

However, FedAVG suffers from the issue that data is usually distributed with a heterogeneous nature in local clients, where existing works [12], [17], [18] have already illustrated that FedAVG converges slowly when the data distributions on clients are highly skewed.

There are many solutions proposed to improve the model updating mechanism to solve the non-iid problem. FedProx [11] adds a weighted L_2 regularization term to the local loss function to constrain the distance between the global model and the updated local model, so that the updated aggregated model would not deviate from the optimal global model. This method makes few modifications to original FedAVG, and is easy to implement. However, to achieve a highly accurate global model, clients are required to spend much more time to tune the weights, which makes it impractical in real FL scenarios. SCAFFOLD [12] proposes to use control variates or variance reduction to correct for the "client-drift" in local updates. First, SCAFFOLD estimates the update direction for the server model and the update direction for each client. The difference is then an estimate of the "client-drift" which is used to correct the local update. There are two ways to compute the control variates. One is to compute the gradients of the loss w.r.t. global models, and the other instead re-uses the previously computed gradients to update the control variate. However, in practice, the first method involves making an additional pass over the local data to compute the gradient, which requires extra computation cost, while the second method is unstable in producing high performance.

Personalized FL is another solution for solving the data heterogeneity problem. [19] connects the trained global model to a unique personalized layer on each client, so that the new model is able to deal with the local situations in particular.

Beyond improving updating mechanism, some works propose to provide the server more data to mitigate the non-iid effects. [10] propose to share some data among the server and the clients. The authors assume the server to have a small collection of public data related to the task (or to collect some data from clients) and can share with all clients. Thus, the data can be fed into clients' dataset to mitigate the non-iid effects. However, in reality, it is difficult to find high quality public data for a specific FL task and it is also not possible to receive raw data from clients due to privacy. To share data with low privacy risk, [13] designs a framework called SDA-FL, which generates synthetic data with GAN on each client, updates the global model and feeds local dataset with synthetic data. But the method lacks analysis about mode collapse introduced by GAN. Sharing the data with other clients is also not realistic since they may drop the FL training and train on their own. And how well the privacy is preserved is not studied. [14] has conducted successful membership inference attacks on synthetic data generated with GAN even in black-box settings.

B. Generative Adversarial Networks and Membership Attacks

The GAN structure contains a generator and a discriminator. The generator learns how to generate synthetic samples similar to the original ones, and the discriminator tries to distinguish them. Both of them are trained simultaneously and in competition with each other. There have been a plenty of well-designed GAN architectures [20]–[24].

DCGAN [20] is the first de-convolutional neural network (de-CNN) structural design which significantly stabilizes the training of GAN. It consists of two networks: one CNN network works as the generator, and the other de-CNN network works as the discriminator. It produces high visual quality images efficiently. WGAN [21] proposes a substitute loss function derived through Wasserstein distance. It uses this alternate cost function to avoid the vanishing gradient problem and partially remove the mode collapse obstacle to stabilize the GAN training. InfoGAN [22] attempts to learn representations with the idea of maximizing the mutual information between labels and the generated samples by using an additional classifier to predict the labels of the samples. We use DCGAN in the implementation of GFL, due to its simplicity and efficiency.

Along with the ability to generate data, GAN also brings the privacy issue such as membership inference attacks [25]–[28] in which an adversary aims to predict whether or not a particular sample was contained in the target models training dataset. Membership inference attacks can be conducted in many fields and overfitting is a sufficient condition for membership vulnerability [29]. LOGAN [14] follows the original idea of membership inference attacks in [25] by training shadow GAN models and using the discriminator to distinguish whether data is used in the training dataset. To deal with possible attacks, several works [16], [30], [31] have been proposed. DPGAN [16] adds carefully designed noise to gradients during the learning procedure to achieve differential privacy.

C. Differential Privacy

Differential privacy (DP) [32] is a system for publicly sharing information about a dataset by describing the patterns of groups within the dataset while withholding information about individuals in the dataset. Denote a randomized algorithm with the DP property as $\mathcal{A}_p(\cdot)$. Then DP is defined as following:

Definition 1 (Differential Privacy)**.** A randomized algorithm $\mathcal{A}_p$ is (ϵ, δ)-differentially private if for any two databases $\mathcal{D}$ and $\mathcal{D}'$ differing in a single point and for any subset of outputs $\mathcal{S}$: $\mathbb{P}(\mathcal{A}_p(\mathcal{D}) \in \mathcal{S}) \leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{A}_p(\mathcal{D}') \in \mathcal{S}) + \delta$, where $\mathcal{A}_p(\mathcal{D})$ and $\mathcal{A}_p(\mathcal{D}')$ are the outputs of the algorithm for input databases $\mathcal{D}$ and $\mathcal{D}'$, respectively, and $\mathbb{P}$ is the randomness of the noise in the algorithm.

It can be shown that the definition is equivalent to

$$\left| \log \left(\frac{P(\mathcal{A}_p(\mathcal{D})) = s}{P(\mathcal{A}_p(\mathcal{D}')) = s} \right) \right| \leq \epsilon,$$

with probability $1 - \delta$ for every point s in the output range, where ϵ reflects the privacy level.

DP has been widely used in GAN and FL recently. For example, [33]–[35] make use of DP for privacy protection to collaboratively train a GAN model. However, our proposed method relies on a local GAN model but shared synthetic samples, which is different from the settings above. Thus, we

TABLE I
THE EXPERIMENT RESULTS ON CIFAR-10 WITH FEDAVG ACROSS DIFFERENT DATA DISTRIBUTIONS.

	$\alpha \to \infty$	$\alpha \to 10$	$\alpha \to 1$	$\alpha \to 0$
Accuracy	89.84%	78.15%	46.67%	13.43%
Drop Rate	—	13.01%	48.05%	85.05%

seek for a GAN network which is able to produce synthetic samples with DP protection so that we can safely share them. DPGAN [16] applies DP to GAN and proofs its DP guarantee. The parameters of discriminator guarantee (ϵ, δ)-DP if the noise standard deviation σ_n satisfies

$$\sigma_n = 2q \sqrt{n_d \log\left(\frac{1}{\delta}\right)} \Big/ \epsilon.$$

And it also shows that the outputs of the generator, i.e. the synthetic samples, also guarantee (ϵ, δ)-DP.

III. PROBLEM FORMULATION AND SYSTEM OVERVIEW

We aim to design an efficient framework for FL that enables the server and clients to collaboratively solve the non-iid problem, and construct a high-performance global model with higher accuracy and faster convergence rate.

A. Motivation Examples

Here, we perform data driven analysis to demonstrate the negative impacts of data heterogeneity on the global, thereby highlighting the necessity to address this issue. We use the well-known image dataset CIFAR-10 [36] as training data, which contains 50,000 images from 10 categories. We employ 10 clients to train a ResNet-18 [37] model using the typical federated optimization algorithm FedAvg. We divided the dataset into 100 shards of size 500, and assigned them to 10 clients. Further, we leverage the Dirichlet distribution to quantify the degree of the data heterogeneity, which is controlled by the parameter α. The smaller the α is, the greater the degree of the uneven distribution is.

The results are shown in Table I. The drop rate P_{drop} is defined as

$$P_{drop} = 1 - \frac{P_{target}}{P_{base}},$$

where P_{base} is the accuracy when the data distribution is IID ($\alpha \to \infty$), and P_{target} is the accuracy of another target distribution. It shows that as α decreases and the non-iid situation gets severe, the test accuracy of the global model decreases dramatically. For example, when $\alpha \to \infty$ (iid), FedAVG achieves high test accuracy, while when $\alpha \to 0$ (non-iid), where each client holds samples from only one random category out of all 10 categories, the accuracy is only 13.43%, which decreases 85.05% compared to the iid setting.

These results show that the statistically heterogeneous training data with severe imbalance distributions among categories would result in low accuracy and unstable convergence. An efficient and privacy-preserving method to solve the non-iid problem is urgently needed for FL.

B. Problem Definition

Here, we take a classification task as an example, which is defined over a compact space $\mathcal{X}$ and a label space $\mathcal{Y}$. There are K clients, each client k has a local dataset $\mathcal{P}_k = \{z_{k,1}, z_{k,2}, \cdots, z_{k,n_k}\}$, where $z_{k,i} = (x_{k,i}, y_{k,i}) \in \mathcal{X} \times \mathcal{Y}$. $n = \sum_{k=1}^{K} n_k$ is the total number of all clients' samples. Function $l_k(\cdot)$ represents loss functions of on client k. The goal of a standard federated optimization problem is to find the optimal global model,

$$w^* = \arg\min_w \left\{ l(w) \coloneqq \sum_{k=1}^{K} p_k \cdot l_k(w) \right\},$$

where $p_k > 0$ is the weight of the loss of client k, and $\sum_{k=1}^{K} p_k = 1$. Suppose there are n_k samples on client k, we can set $p_k = \frac{n_k}{n}$. In general, the local objectives measure the local empirical risk over different data distribution $\mathcal{P}_k$, i.e. $l_k(w) \coloneqq \mathbb{E}_{z_{k,i} \in \mathcal{P}_k}[l_i(w; z_{k,i})]$, where $k = \{1, 2, \cdots, K\}$.

We assume that all participants including the server and the clients are semi-honest, i.e., they follow the exact protocol of FL and debugging but may be curious about others' local data. We also assume that each client has enough samples for at least one category (see Assumption 1). This assumption is practical for the reason that if no client has enough samples for one category, it is almost impossible to achieve a high performance global model that performs well on that category.

Assumption 1. A client k has enough samples for at least one category if $\exists$ a category i satisfies,

$$l_k(w; \mathcal{P}_k^i) < \tau,$$

where w is a well-trained local model, $\mathcal{P}_k^i$ is the data on category i, and τ is a threshold to control the convergence.

C. Overview of GFL

In FL, the server, e.g., usually a remote cloud server, plays the role that coordinates the joint training of locally distributed clients, usually e.g., edge devices. So the server often has larger computation and communication resources and power than the resource-constrained clients. However, in tradition FL settings, the server only conducts the procedure that aggregates the model parameters and communicates with the clients. This mode not only wastes the power of the server, but also consumes more power of the clients, since more communication rounds are needed for the global model to converge in the non-iid setting.

The main idea of GFL is to use synthetic samples on the server to improve the converge speed even when the data distributions of clients are non-iid. All of the synthetic samples come from the clients. According to Lemma 1, high quality synthetic samples from all labels can be produced. In order to protect the privacy of original data on the clients, we add DP noises to the synthetic samples and the distribution is different from the original data. The server chooses an iid subset dynamically from all the synthetic data collected from the clients. Thus, the server can train on the iid synthetic dataset to encourage the global model to converge.

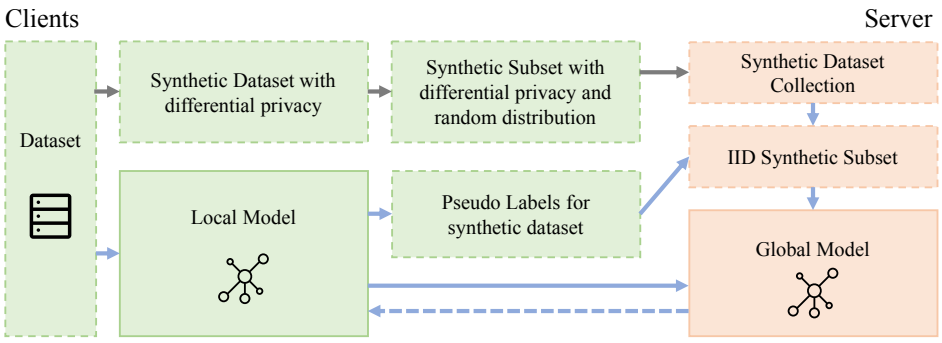


Fig. 1. An overview of GFL framework. The green and orange parts are the client and server side, respectively. The grey lines denote the synthetic data generation process, in which each client trains its DPGAN model, generates synthetic data with differential privacy and a random distribution, and uploads them to the server for further training. The blue arrows indicate the federated learning stage. The server receives local models as well as the updated pseudo labels for the synthetic samples. The server then extracts an iid subset from the synthetic data collection to further train the global model.

Lemma 1. When all clients satisfy Assumption 1, the server can have high quality synthetic samples from all labels by collecting synthetic samples from all clients.

To be specific, GFL can be divided into two stages: the offline synthetic data generation stage and the online federated learning stage with synthetic data. It is summarized in Fig. 1. During the synthetic data generation stage (the grey arrows), each client k trains DPGAN, generates synthetic data S_k from $\mathcal{P}_k$, and then sends S_k to the server. At the federated learning stage (the blue arrows), all clients follow the normal federated learning process, while the server aggregates the local models. However, unlike the normal process, the server trains the aggregated model additionally on the iid synthetic dataset $S = \{S_1, \cdots, S_K\}$ so that the global model converges faster based on the balanced extra information provided by S.

IV. Generative Federated Learning

Here, we present the design of GFL method. We first give the solution to the synthetic data generation against the threat model, e.g., membership attacks. Then, we illustrate how to conduct federated learning with the generated synthetic data to achieve high-performance global model.

A. Synthetic Data Generation

In this part, we give our privacy-preserving data generation algorithm, which adds DP noises to the synthetic samples and generates a different distribution from the original dataset.

1) Threat Model: We introduce GAN to federated learning in GFL. However, there may be a chance that the synthetic data provided by GAN leaks private information of clients. One of the most severe leakages in GAN is membership [16]. For example, nobody wants his/her face appearing in the generated images even he/she takes part in the GAN training. Thus, it is necessary to analyze the privacy of the synthetic data in case of any possible issue.

We consider an adversary (the server in GFL) who aims to infer whether a single known record was included in the training set of the victim (the target GAN model), which is called membership inference attacks. The adversary is semi-honest, who follows the normal training process but is curious about membership. We focus only the black-box setting, in which the adversary has no access to either the discriminator or the generator. It is reasonable since the server can only get the synthetic samples from the clients but have no information about how they are generated. The accuracy of the attack is measured as the fraction of the records correctly inferred as members of the training set.

Besides, the adversary is also curious about the data distribution of the victim, which may contain important privacy information about the victim.

a) Assumptions: The semi-honest server as an adversary is in a black-box setting as described below. It has no prior or side information about training records or the GAN model. To make it clear, the adversary tries to conduct a membership inference attack on an victim client to predict whether a sample belongs to the training set with only its synthetic samples but no knowledge of the following:

•The architecture of the victim. When the adversary reconstructs the GAN model, it does not know if its architecture is the same as the victim. In other words, the adversary may choose a different GAN model from the victim.

•The parameters and training hyper-parameters of the victim. The adversary can neither get access to the parameters to construct the same model as the victim, nor know the loss function or the total epochs during training.

•The dataset used in the victim. The adversary has zero-knowledge of the original dataset, since the target of the membership inference attack is to infer which sample is used during training.

However, in order to give a clear view of the performance of the attack, we give the adversary a dataset which contains the data from both the training set and other sources. Its only usage is to give a clear metric about the performance of the attack and it cannot be used to interfere the training of the attack model. Thus, this dataset does not affect the whole attack process and can be safely ignored in the real scenario.

2) Privacy-preserving Data Generation: The reason that leads to the non-iid problem is that the clients hold datasets of different distributions. Clients know nothing about the distributions of each other. However, since the server plays the

Algorithm 2: Privacy-preserving Data Generation

Input : The synthetic data amount n_{S_k}.

Output: The synthetic dataset S_k.

Data : The local dataset $\mathcal{P}_k$ with size n_k.

Client k **executes:**
- $P_{S_k} \leftarrow$ (a random distribution)
- $d_k, g_k \leftarrow$ DPGANTraining($\mathcal{P}_k$) 根据本地数据集训练带DP的数据生成模型
- $S'_k \leftarrow \emptyset$
- **while** *not* EnoughData(S'_k, n_{S_k}, P_{S_k}) **do**
  - $noises \leftarrow$ (random noises)
  - $S'_k \leftarrow S'_k \cup g_k(noises)$
- $S_k \leftarrow$ CutData(S'_k, n_{S_k}, P_{S_k})

EnoughData(S'_k, n_{S_k}, P_{S_k}**):**
- **for** *each label* i **do**
  - **if** $S'^{i}_{k} < n_{S_k} \cdot P^i_{S_k}$ **then**
    - **return** *false*
- **return** *true*

生成数据的标签由其local model预测，在每个通讯轮可能变化

CutData(S'_k, n_{S_k}, P_{S_k}**):**
- $S_k \leftarrow \emptyset$
- **for** *each label* i **do**
  - $temp \leftarrow$ (a random subset of S'^{i}_{k} with the size of $n_{S_k} \cdot P^i_{S_k}$)
  - $S_k \leftarrow S_k \cup temp$
- **return** S_k

role that coordinates the whole FL training, we wish it to take part in the training actively instead of passively aggregating parameters. Due to the lack of feature information, we want to create a small iid dataset for the server to use to help find the global optima in a short time. The data in this iid dataset should completely come from the clients since there is no data on the server. But we still face the challenge that privacy information can leak when sharing. We focus on membership inference attacks and data distribution leakage. In the synthetic data generation stage, we provide a solution to achieve privacy-preserving generation of synthetic data.

Suppose we need to generate n_{S_k} synthetic samples on client k. We first train a DPGAN on the client with its local dataset $\mathcal{P}_k$ and generate a random distribution P_{S_k} which will be the distribution of the synthetic dataset S_k. Then the generator g_k of DPGAN starts to generate synthetic samples until each label i contains at least $n_{S_k} \cdot P^i_{S_k}$ samples. Finally, the client extracts a random subset which meets the distribution P_{S_k} as S_k. Detailed steps are described in Algorithm 2.

It is worth noticing that the synthetic data generation stage is done offline before the federated learning stage, and DPGAN is an unsupervised method which does not need label information. So the generator g_k on client k can be reused in different tasks without retraining every time.

B. Federated Learning with Synthetic Data

After the synthetic data generation, it is time to step into the federated learning stage. In this stage, we "iidify" the synthetic samples from the clients to train the global model in an iid way. And the potential mode collapse issue is solved by adding an epoch decay parameter to gradually decrease the training epochs on the global model. In order to achieve the purpose above, there are some modifications to be made from the traditional federated learning. The detailed steps are summarized in Algorithm 3.

First, since DPGAN is an unsupervised learning method, all the synthetic samples are unlabeled. Although it is possible for a client to use the local model for labeling at the beginning, considering that it takes part in the federated learning task, it may have difficulty in training a good enough local model to label the synthetic samples confidently alone. So we decide to dynamically update the labels during the training process. At each communication round t, after the training on each client k is finished, the updated local model w^k_t is used to update the labels of the synthetic data S_k. The updated labels y_{S_k} are then packed and sent along with the new local model parameters to the server for further training.

Second, after aggregation, instead of sending the aggregated model w'_t immediately back to the clients, the server continues to train w'_t on the synthetic dataset S, where S is a random iid subset of all the data collected from all the clients. This strategy can extremely speed up the training process, especially at the first few epochs, because it encourages the model to directly converge to the optimum with the help of S. On the contrast, traditional federated learning methods may take more communication rounds for the clients to negotiate and find out a consistent convergence direction.

Third, in order to deal with potential problems introduced by GAN such as mode collapse and low-quality synthetic data, we gradually reduce the training epochs on the server as the communication rounds go up. We define an epoch decay parameter τ. For the communication round $t = 1, 2, \cdots, T$ and a given default epoch startpoint E_s, the actual epochs of this round is

$$E_s \cdot e^{-\tau \cdot (t-1)}.$$

We see that the total epochs start from E_s, and gradually reduce below 1 when $t > \lceil \frac{\ln E_s}{\tau} \rceil$, which means that there is no need to continue using synthetic dataset, because the clients start to work together and mode collapse and low-quality synthetic data begin to harm rather than help.

By updating labels of synthetic data and training the aggregated model with gradually reduced epochs, we achieve a better convergence speed and performance than other methods.

V. Experiments

We run experiments on two different datasets to confirm the superiority of our GFL framework. Our main discoveries are that i) GFL outperforms other methods such as FedAVG, FedProx and SCAFFOLD in various types of non-iid settings, ii) the benefit of speed up continues to show up even in the iid

Algorithm 3: Online Federated Learning Stage of GFL

Input : The number of communication rounds T, the number of total clients K, the proportion of selected clients ϕ, the number of server epochs E_s, and the epoch decay parameter τ.

Output: The final global model w_T.

Data : The local dataset $\mathcal{P}_k$ with size n_k and the synthetic dataset S_k with size n_{S_k} on each client $k = 1, 2, \cdots, K$.

Server executes:
  $S' \leftarrow \{S_1, S_2, \cdots, S_K\}$ 1，收集所有客户端的生成数据
  initialize w_0
  for *each round* $t = 1, 2, \cdots, T$ **do**
    $m \leftarrow \max(\phi \cdot K, 1)$
    $C_t \leftarrow$ (random subset of m clients)
    for *each client* $k \in C_t$ **do** 2，FedAvg
      $y_{S_k}, w_t^k \leftarrow \text{ClientUpdate}(k, w_{t-1})$
    $w_t' \leftarrow \sum_{k=1}^{m} \frac{n_k}{n} \cdot w_t^k$
    update labels of S' with $\{y_{S_k}, k \in C_t\}$
    $S \leftarrow \text{IIDify}(S')$
    $w_t \leftarrow \text{ServerUpdate}(t, w_t')$ 3，Server基于生成数据再训练global model
  return w_T

IIDify(S' **):**
  $mini \leftarrow \min\{|S'^i|, i = 1, 2, \cdots, \mathcal{Y}\}$
  $S \leftarrow \emptyset$
  for *each label* i **do**
    $temp \leftarrow$ (a random subset of S'^i with the size of $mini$)
    $S \leftarrow S \cup temp$
  return S

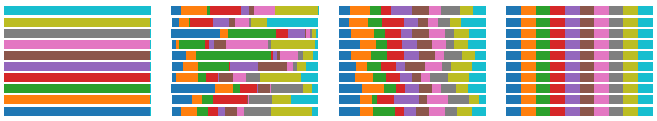
ClientUpdate(k, w **):**
  ... (similar to ClientUpdate in Algorithm 1)
  $y_{S_k} \leftarrow w(S_k)$
  return y_{S_k}, w

ServerUpdate(t, w **):**
  $E \leftarrow E_s \cdot e^{-\tau \cdot (t-1)}$
  ... (similar to ClientUpdate in Algorithm 1)
  return w

setting, and iii) the accuracy of membership inference attacks on synthetic data drops as privacy level ϵ goes down.

A. Setup

We first carefully define the non-iid problem. To simulate a population of non-iid clients, we introduce the Dirichlet distribution. We draw $q \sim Dir(\alpha \cdot P)$, where P characterizes a prior distribution of N labels, and α is a control parameter which measures the difference among clients. By assuming that the distribution of the total dataset is iid, we can simply set p to be the same on all labels and vary α. When $\alpha \to 0$, all clients hold samples only from one single category; when $\alpha \to \infty$, each client has an identical distribution which contains the same amount of samples from all categories, which is the same as iid.



(a) $\alpha \to 0$ (b) $\alpha \to 1$ (c) $\alpha \to 10$ (d) $\alpha \to \infty$

Fig. 2. Data distributions of non-iid clients sampled from different Dirichlet distributions. Different colors represent different labels, and each row is the distribution of a client. (a) $\alpha \to 0$. An extreme situation when each client holds data in only one category. (b) $\alpha \to 1$. A common simulated non-iid situation. (c) $\alpha \to 10$. A slightly non-iid situation. (d) $\alpha \to \infty$. IID distributions among all clients.

Fig. 2 illustrates populations that we use in our experiments from Dirichlet distributions with $\alpha \in \{0, 1, 10, \infty\}$. We have 10 clients in the figure to give a clear view of the distributions, which is also our client setting in the following experiments.

Then, we use DPGAN [16] as the local GAN and ResNet-18 [37] as the training model in federated learning on EMNIST [38] and CIFAR-10 [36] in our experiments.

- EMNIST. We use its "Digits" subset in our experiments. It contains 10 classes with a total $280,000$ samples, which is divided into $240,000$ training samples and $40,000$ testing samples. Compared to MNIST, it has more data and is completely balanced.
- CIFAR-10. It contains $60,000$ 32×32 color images in 10 different classes, which is a more complicated task compared to EMNIST. We use this dataset to measure the stability of GFL in hard tasks.

Finally, we choose to compare our GFL framework with the following three algorithms:

- FedAVG [1], as the most classic federated learning method, should give us the baseline of the training results.
- FedProx [11] uses a regulation parameter μ to make sure the local model not deviate a lot from the global model. We fix the regulation parameter μ to 1.
- SCAFFOLD [12] uses control variates to instruct the clients to update in the similar direction to reduce the effect of non-iid. In our experiments, we follow its idea and choose to re-uses the previously computed gradients to update the control variates, which is cheaper to compute but sufficient.

GFL trains 200 epochs at the synthetic data generation stage. Then, we randomly select $n_{S_k} = 500$ synthetic samples from each client k. During the federated learning stage, the initial global training epoch E_s is set to 10 and the epoch decay parameter τ is set to 0.1 if not specified.

All the algorithms above share the same basic training settings. The training contains 100 communication rounds, and in each round each client trains for 20 epochs, and the batch size on both the server and the clients is set to 128.

TABLE II
THE ACCURACY RESULTS OF DIFFERENT METHODS ON EMNIST AND CIFAR-10. GFL IS THE BEST IN MOST SITUATIONS, ESPECIALLY WHEN THE DATA IS EXTREMELY NON-IID SUCH AS $\alpha \to 0, 1$. AS THE DATA DISTRIBUTION MOVES TOWARDS IID, GFL MAINTAINS THE ACCURACY SIMILAR TO FEDAVG.

EMNIST	$\alpha \to 0$		$\alpha \to 1$		$\alpha \to 10$		$\alpha \to \infty$	
FedAVG	30.84%	(x1.00)	66.05%	(x1.00)	98.55%	(x1.00)	98.65%	(x1.00)
FedProx	64.72%	(x2.10)	77.66%	(x1.18)	98.94%	(x1.00)	**98.89%**	**(x1.00)**
SCAFFOLD	51.35%	(x1.67)	78.38%	(x1.19)	97.88%	(x0.99)	98.03%	(x0.99)
GFL	**78.78%**	**(x2.55)**	**89.77%**	**(x1.36)**	**99.39%**	**(x1.01)**	98.13%	(x0.99)

CIFAR-10	$\alpha \to 0$		$\alpha \to 1$		$\alpha \to 10$		$\alpha \to \infty$	
FedAVG	13.43%	(x1.00)	46.67%	(x1.00)	78.15%	(x1.00)	89.84%	(x1.00)
FedProx	38.55%	(x2.87)	46.19%	(x0.99)	83.63%	(x1.07)	89.31%	(x0.99)
SCAFFOLD	38.90%	(x2.90)	56.82%	(x1.22)	**88.50%**	**(x1.13)**	89.48%	(x1.00)
GFL	**56.22%**	**(x4.19)**	**67.42%**	**(x1.44)**	84.26%	(x1.08)	**89.93%**	**(x1.00)**

B. Training Results

First, we run federated learning on EMNIST and CIFAR-10 in different distribution settings on GFL and other three methods. Comparing with other algorithms, we have some interesting findings according to the results in Table II:

GFL outperforms other algorithms in all settings. For most settings we choose in the experiments, the accuracy of GFL is higher than other three methods. FedAVG is hard to converge and find the global optima in non-iid settings. FedProx and SCAFFOLD are also getting slower because in order to keep the update direction close to each other, the clients can only update their local models with constrains such as extra terms in the loss function. Compared to them, GFL has a higher accuracy thanks to the synthetic dataset. Training on the synthetic dataset makes the global model converge much faster from the beginning. And the clients benefit from it as well, since the closer the global model is to the global optima, the more likely the local optima would be to the global optima. Thus, the clients naturally have a similar updating direction and encourage convergence, which results in a virtuous circle.

The more severe non-iid is, the more obvious the advantage of GFL becomes. As α gradually decreases from ∞ to 0, we find that the relative accuracy rate of GFL compared to others increases. When α decreases, the weakness of FedProx and SCAFFOLD starts to show up. Although we can see improvement compared to FedAVG, the slow converge speed leads to an increase in the number of communication rounds required to achieve the same accuracy compared to GFL.

The more complicated a dataset is, the more superior GFL is. We have our results on both EMNIST and CIFAR-10 datasets. The pictures in CIFAR-10 are 3 channels compared to only 1 channel in EMNIST. In a harder CIFAR-10 task, the relevant accuracy improvement is even greater. The reason is the same as above.

C. Analysis of Epoch Decay

We then take a closer look at the epoch decay parameter τ in GFL. τ is introduced to prevent mode collapse in GAN and low-quality synthetic samples from harming the global model. Table III gives us the results with different τ.

TABLE III
ACCURACY COMPARISON OF FEDAVG AND GFL WITH DIFFERENT EPOCH DECAY PARAMETER τ.

CIFAR-10	τ	$\alpha \to 0$	$\alpha \to 1$
FedAVG	∞	13.43%	46.67%
GFL	0.1	**56.22%**	**67.42%**
	0	32.77%	38.21%
	$-\infty$	17.26%	24.94%

The epoch decay parameter is essential in GFL. If $\tau \to \infty$, there is no training epoch on the server $E_s \cdot e^{-\tau \cdot (t-1)} \to 0$, which means we fall back to FedAVG. If $\tau \to 0$, the training epochs on the server freeze at $E_s \cdot e^{-\tau \cdot (t-1)} = E_s$, which means we keep training on the synthetic dataset until the end. However, the result is not good due to mode collapse and low-quality synthetic samples. Thus, a proper τ is needed to train a good global model in GFL.

The way to use only synthetic data is useless. We also analyze the situation when $\tau \to -\infty$, i.e., when the server uses only synthetic data without any aggregation. The result is also not ideal as the quality and the number of synthetic samples are not sufficient.

D. Privacy Evaluation

In this part, we focus on potential privacy attacks in GFL, i.e. membership inference attacks. To be specific, we use LOGAN [14] to attack the synthetic CIFAR-10 data generated by DPGAN with different privacy level ϵ. DPGAN is trained for $1,000$ epochs and in each epoch it generates 500 samples for a pre-trained LOGAN to infer membership of samples from a static given dataset which contains 10% samples from the trainset of DPGAN and another 90% unrelated samples. Fig. 3 shows the attack results when DPGAN is trained with $\epsilon \in \{\infty, 1, 0, 0.1, 0.01\}$.

When $\epsilon \to \infty$, which means we do not add any noise at all, the accuracy of attack achieves 31.16%. Then as ϵ decreases, the attack accuracy drops below 20%. This reflects the efficiency of adding noises in DPGAN.

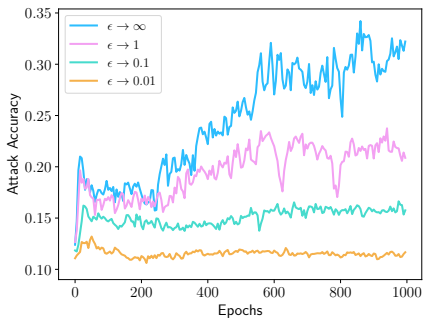


Fig. 3. The line chart of the privacy evaluation results on different privacy level ϵ. The membership inference attack accuracy drops as ϵ decreases.

TABLE IV
THE TABLE OF THE PRIVACY EVALUATION AND FL TRAINING RESULTS ON DIFFERENT PRIVACY LEVEL ϵ.

CIFAR-10	ϵ	Attack Acc	FL Acc
GFL	∞	31.16%	69.31%
	1	21.58%	68.12%
	0.1	14.80%	67.42%
	0.01	11.66%	52.79%
FedAVG	—	—	46.67%

However, too small ϵ in DPGAN would also affect the quality of synthetic data, which would result in a pool global model in GFL. From Table IV, we can see that the performance of the global model drops as well when ϵ decreases. A small $\epsilon \to 0.01$ means more noises in the training of DPGAN, which results in protecting the original dataset (with an attack accuracy at 11.66%) but generating pool-quality synthetic samples. These samples are then sent to the server to help training the global model. However, due to the pool quality, fitting the global model on these samples can hardly improve the performance. The convergence of the global model is slowed down by them instead. Because τ generally reduces the training epochs of global model, the performance still maintains near the baseline (with the FL accuracy at 52.79% which is close to 46.67% by FedAVG).

Overall, we need to find a trade-off between privacy and performance. According to the experiments in Fig. 3 and Table IV, we choose $\epsilon = 0.1$, which prevents membership inference attacks while keeping good performance.

E. Other necessary metrics

By using DPGAN in GFL, the clients spend some extra computation (training DPGAN) and communication (uploading synthetic samples) costs. We analyze these effects in this part in order to show that the extra costs are affordable compared to the traditional federated learning such as FedAVG. The results are shown in Table V.

FLOPS (FLoating-point Operations Per Second) is a common metric to measure the computation cost. We use it to compare the training cost of ResNet-18 and DPGAN on a batch of 128 CIFAR-10 images. As shown in Table Va, the cost of ResNet-18 and DPGAN is similar, which means the total extra cost of DPGAN would not be larger than ResNet-18 if trained the same epochs. Besides, the trained DPGAN can be re-used on the same dataset because it is an unsupervised method. Thus, in a future FL task, the training cost of DPGAN would be zero, i.e., there is no extra computation cost.

TABLE V
COMPARISON OF THE EXTRA COMPUTATION AND COMMUNICATION COST OF FEDAVG AND GFL.

(a) Computation Cost.

Model	FLOPS (G)
ResNet-18	71.25
DPGAN	68.17

(b) Communication Cost.

Data	Size (MB)
Parameters of ResNet-18	42.6
Synthetic samples from DPGAN	23.4

The extra communication cost introduced by GAN is the uploaded synthetic samples. As the settings in the experiments above, we assume that each client uploads $n_{S_k} = 500$ samples to the server. The communication cost of FedAVG is the parameters of the local model. Thus, we compare the size of 500 synthetic CIFAR-10 images with the parameters of ResNet-18 in our settings. As shown in Table Vb, the extra cost is much smaller. Considering that the parameters of the local model need to upload in every communication round while the synthetic samples upload only once at the beginning, the extra cost can be safely ignored.

In summary, DPGAN does not increase too much computation and communication cost. Thus, GFL can be widely adopted in pervasive computing applications as long as the devices are capable of training original FedAVG.

VI. CONCLUSION

In this paper, we propose a new framework called Generative Federated Learning (GFL) to deal with the non-iid problem. We introduce DPGAN to GFL in order to deal with potential privacy attacks, especially membership inference attacks. Extensive experiments demonstrate that GFL can work on non-iid situations as well as iid and outperform other frameworks such as FedProx and SCAFFOLD in most situations. The privacy evaluation shows that by adding proper noises to the training of DPGAN, we can defense membership inference attacks efficiently. We also analyze the necessity of the epoch decay parameter τ introduced by GFL, which is important to prevent mode collapse and low-quality samples from harming the global model. Finally, we evaluate the extra computation and communication costs of GFL and show that compared to FedAVG, the extra costs are affordable for various pervasive computing applications.

ACKNOWLEDGEMENTS

Lan Zhang is the corresponding author. This research was supported by the National Key R&D Program of China 2021YFB2900103, China National Natural Science Foundation with No. 61932016 and "the Fundamental Research Funds for the Central Universities" WK2150110024.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[2] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[3] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *NIPS*, 2017.

[4] K. Sozinov, V. Vlassov, and S. Girdzijauskas, "Human activity recognition using federated learning," *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pp. 1103–1111, 2018.

[5] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent iot applications: A cloud-edge based framework," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 35–44, 2020.

[6] S. Ek, F. Portet, P. Lalanda, and G. Vega, "A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison," *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, 2021.

[7] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," *arXiv preprint arXiv:2007.14513*, 2020.

[8] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *AAAI*, 2020.

[9] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, 2020.

[10] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[11] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[12] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *ICML*, 2020.

[13] Z. Li, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Federated learning with gan-based data synthesis for non-iid clients," *ArXiv*, vol. abs/2206.05507, 2022.

[14] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro, "Logan: Membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, pp. 133 – 152, 2019.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[16] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," *ArXiv*, vol. abs/1802.06739, 2018.

[17] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[18] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[19] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *ArXiv*, vol. abs/2003.13461, 2020.

[20] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[21] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[22] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2180–2188, 2016.

[23] J. Wen, B.-Y. Chen, C.-D. Wang, and Z. Tian, "Prgan: Personalized recommendation with conditional generative adversarial networks," *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 729–738, 2021.

[24] G. Guarino, A. Samet, A. Nafi, and D. Cavallucci, "Pagan: Generative adversarial network for patent understanding," *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 1084–1089, 2021.

[25] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2017.

[26] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *ArXiv*, vol. abs/1806.01246, 2018.

[27] L. Song, R. Shokri, and P. Mittal, "Membership inference attacks against adversarially robust deep learning models," *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56, 2019.

[28] Y. Liu, Z. Zhao, M. Backes, and Y. Zhang, "Membership inference attacks by exploiting loss trajectory," *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[29] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282, 2017.

[30] J. Chen, J. Konrad, and P. Ishwar, "Vgan-based image representation learning for privacy-preserving facial expression recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1570–1579, 2018.

[31] R. Shetty, B. Schiele, and M. Fritz, "A4nt: author attribute anonymity by adversarial training of neural machine translation," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 1633–1650, 2018.

[32] C. Dwork, "Differential privacy," in *Encyclopedia of Cryptography and Security*, 2006.

[33] B. Xin, W. Yang, Y. Geng, S. Chen, S. Wang, and L. Huang, "Private fl-gan: Differential privacy synthetic data generation based on federated learning," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2927–2931, 2020.

[34] L. Zhang, B. Shen, A. Barnawi, S. Xi, N. Kumar, and Y. Wu, "Feddpgan: Federated differentially private generative adversarial networks framework for the detection of covid-19 pneumonia," *Information Systems Frontiers*, vol. 23, pp. 1403 – 1415, 2021.

[35] J.-F. Rajotte, S. Mukherjee, C. Robinson, A. Ortiz, C. West, J. M. L. Ferres, and R. T. Ng, "Reducing bias and increasing utility by federated generative modeling of medical images using a centralized adversary," *Proceedings of the Conference on Information Technology for Social Good*, 2021.

[36] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep. 0, University of Toronto, Toronto, Ontario, 2009.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[38] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, IEEE, 2017.