A GAN-based Data Augmentation Framework for Federated Learning on Non-IID Data

YU-MIN CHOU, National Tsing Hua University, Taiwan
FU-CHIANG CHANG, National Tsing Hua University, Taiwan
JERRY CHOU, National Tsing Hua University, Taiwan

Federated Learning (FL) is a distributed learning technique that allows devices to collaboratively learn a shared prediction model while keeping all the training data locally and thus enable more secured and accurate model training. However, the convergence and accuracy of federated learning can be degraded by the non-IID (non independent and identically distributed) data across all edge devices. Hence, we proposed a **GAN**-based **D**ata **A**ugmentation **L**earning **F**ramework named ***GANDALF***, which aims to solve the non-IID problem by performing data augmentation with GAN models based on a mediator-based system architecture. Fault tolerance and data privacy mechanism are also provided to ensure system reliability and security. We conducted experiments to compare GANDALF with the traditional FedAvg and several other recently proposed data augmentation methods, and their top-1 test accuracy can be improved by 2.54% ~ 15.66% on CINIC-10 dataset and 2.85% ~14.8% on EMNIST dataset.

Additional Key Words and Phrases: Federated Learning, Generative Adversarial Network, Non-IID data, Differentail Privacy

ACM Reference Format:
Yu-Min Chou, Fu-Chiang Chang, and Jerry Chou. 2018. A GAN-based Data Augmentation Framework for Federated Learning on Non-IID Data. 1, 1 (December 2018), 24 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With the vigorous development of deep learning (DL), DL has been used in many applications such as sign-to-speech translation [57], medical imaging [18], and self-driving cars [38]. Deep learning is a data-based technology; its current success is based on the centralized training method, which performs model training on the high-quality data collected in a single node or a data center. In recent years, the advancement of technology has caused a large amount of data to be generated in edge side. These distributed data are of great help to the development and quality of the deep learning model. However, as people gradually attach importance to the right to privacy, laws and policies are enforced to protect privacy, making it more difficult to collect available data. The dilemma of not obtaining a large amount of data leads to the failure of deep learning to achieve better performance. To address this challenge, a new learning technique called Federated Learning [39] is proposed, and it has drawn great attention to both academic and industry communities.

Federated Learning (FL) [39] is a distributed framework for deep learning without centralizing data and with privacy by default. FL allows clients to train a global model collaboratively while keeping private data local. During the training process, clients do not need to share their private

Authors' addresses: Yu-Min Chou, National Tsing Hua University, Hsinchu, Taiwan, ymchou@lsalab.cs.nthu.edu.tw; Fu-Chiang Chang, National Tsing Hua University, Hsinchu, Taiwan, @lsalab.cs.nthu.edu.tw; Jerry Chou, National Tsing Hua University, Hsinchu, Taiwan, jchou@lsalab.cs.nthu.edu.tw.

XXXX-XXXX/2018/12-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

data and only exchange the updated model with the FL server. FL can be divided into two settings according to the types of clients. The first one is cross-device, where clients are a large number of mobile or IoT devices, and the second one is cross-silo, where clients are different organizations or distributed datacenters. In reality, FL has been used in cross-device and cross-silo applications, such as improving word prediction [22], query suggestions of Google keyboard [51], and medical data segmentation [8]. In sum, this decentralized framework solves the problem of private data and provides economic benefits.

However, FL encounters several challenges with different deployment environments. The challenges of cross-device FL are mainly from the scalability [4], reliability [29], and heterogeneity of hardware [11] due to a large number of clients and limited computing resources of edge devices. By contrast, non-IID (non Independent and Identically Distributed) is a more significant challenge for cross-silo FL [26, 50] because the data each organization collects may differ a lot due to differences in geography, customer groups, and services of organizations. The non-IID problem means that the data in each local dataset fail to represent the global data, namely, the heterogeneity of local class distributions. Recent work [56] shows that highly skewed non-IID data can make the updated local model weights different from each other and reduce the accuracy of FL significantly. Another work [35] analyzed the convergence of FedAvg [39] on non-IID data and indicates that heterogeneity of data slows down the convergence of deep learning models.

Recently, several works focused on solving the non-IID problem. For example, [56] improves non-IID by establishing a subset of data that is shared between clients, but how to create a shared dataset is still an open question. Another way to address non-IID is performing active client selection in each round of FL to counterbalance the bias introduced by non-IID data, and [45] proposed a reinforcement-learning-based framework to achieve this goal. More intuitively, [12], [47], and [27] combine data augmentation, which is used to solve insufficient data and overfitting in centralized training, with their framework to handle the non-IID problem and further improve the accuracy. However, static data augmentation introduced by [12] only produces limited plausible alternative data, so it performs poorly in severe class distribution skews. Another work [27] requires clients to share data to obtain a well-trained GAN [19], which violates privacy. Thus, although many methods have been proposed, how to address the non-IID problem effectively is still an open question.

In this paper, we consider the non-IID problem and the additional requirements of preserving privacy, reducing the cost, being robust and reliable under the the cross-silo setting where clients can be local computers or powerful IoT devices installed in organizations for training. To solve the problem, we proposed ***GANDALF***, a **GAN**-based **D**ata **A**ugmentation **L**earning **F**ramework, which effectively solves the non-IID problem in FL and meets the requirement mentioned above.

In GANDALF, we solve the non-IID problem by performing data augmentation with multiple GAN models. Since insufficient data diversity of a single client does not limit GAN's data generation, GAN is more robust than other data augmentation methods. Also, we use mediator-based architecture combined with sequential training to speed up the convergence of the model and ensure the diversity of generated data, which helps GANDALF outperform other state-of-the-art GAN data augmentation methods. Furthermore, we prevent the model weights from leaking privacy and satisfy the strict differential privacy guarantee. Finally, We implement fault tolerance mechanisms in our framework to ensure training reliability over networks with unstable network connectivity.

Our approach is extensively evaluated using the CINIC-10 and EMNIST datasets, and the results are compared to the traditional FedAvg, three other recently proposed data augmentation methods [12], [56], and three GAN data augmentation methods [36, 48, 55]. The main results are summarized as follows.

- Comparing to the FedAvg and the two data augmentation methods without sharing local private data, GANDALF improved their top-1 test accuracy by 2.54% ~ 15.66% on CINIC-10 and 2.85% ~8.41% on EMNIST.
- Comparing to data augmentation method sharing partial local private data, GANDALF ensures strict data privacy while achieving comparable results for the top-1 test accuracy of +0.11% and -1.7% on CINIC-10 and EMNIST, respectively.
- Comparing to data augmentation methods using GAN models, GANDALF is able to train high quality GAN models in more flexible ways. GANDALF improved their top-1 test accuracy by 1.84%~2.41% on CINIC-10 and 2.89%~14.8% on EMNIST.
- A significant improvement on accuracy over 30% can be observed on n-class non-IID problems, which contain much higher data skewness.
- In a system environment with unpredictable node or network failures, our proposed fault tolerance mechanism can accelerate the training convergence speed up to 6× and 11× while improving the top-1 test accuracy results by 1.34% ~ 4.92%.

The rest of this paper is organized as follows. Section 2 defines the problem of FL with non-IID data and summarizes the objectives of the system design. Section 3 and 4 detail the design and implementation of *GANDALF*. Section 5 presents the evaluation results, along with the analysis. Related work is discussed in Section 6, and conclusions are given in Section 7.

2 PROBLEM DEFINITION

In this section, we first formally define the problem of federated learning and show the effect of non-IID data on federated learning. Then, in Section 2.2, we summarize the objectives of the system design and briefly explain how we achieve it.

2.1 Mathematical Preliminaries

In this subsection, we formally define the problem of federated learning with non-IID data. We consider a c class classification problem defined over a compact space $\mathcal{X}$ and a label space $\mathcal{Y} = [C]$, where $[C] = \{1...., C\}$. Let $\{x, y\}$ denote a labeled sample, which distributes over $\mathcal{X} \times \mathcal{Y}$ following the class distribution p. Let $f : \mathcal{X} \rightarrow \mathcal{S}$ denote the prediction function, where $\mathcal{S} = \{z \mid \sum_{i=1}^{C} z_i = 1, z_i \geq 0, \forall i \in [C]\}$. Let f_i denote the predicted probability that the sample belongs to the i-th class. Let ω denote the model weight. For training, we define the training loss with the commonly used cross-entropy loss as

$$L(\omega) = \mathbb{E}_{x,y\sim p}[\sum_{i=1}^{C} \mathbb{1}_{y=i} \log f_i(x, \omega)]$$
$$= \sum_{i=1}^{C} p(y = i)\mathbb{E}_{x|y=i}[\log f_i(x, \omega)]$$

We simplify the analysis by ignoring the generalization error. Therefore, the learning problem becomes the following optimization problem.

$$\min_{\omega} \sum_{i=1}^{C} p(y = i)\mathbb{E}_{x|y=i}[\log f_i(x, \omega)]$$

To solve the optimization problem, SGD updates the model weight iteratively to minimize the loss function. Let $\omega_t^{(c)}$ denote the weight after t-th update in the centralized learning and η denote

the learning rate. Next, centralized SGD performs the following update:

$$\begin{aligned}\omega_t^{(c)} &= \omega_{t-1}^{(c)} - \eta \nabla L(\omega_{t-1}^{(c)}) \\ &= \omega_{t-1}^{(c)} - \eta \sum_{i=1}^{C} p(y=i) \nabla \mathbb{E}_{x|y=i}[\log f_i(x, \omega_{t-1}^{(c)})]\end{aligned}$$

We regard $\omega^{(c)}$ as the optimal model weight because the performance of centralized learning is greater than or equal to that of federated learning [41].

In federated learning, we assume there are K clients in total. Let $n^{(k)}$ denote the amount of samples and $p^{(k)}$ denote the class distribution of local dataset of client $k \in [K]$. For local training, each client performs SGD to update the local model weight separately. At iteration t on client $k \in [K]$, local SGD performs the following update:

$$\begin{aligned}\omega_t^{(k)} &= \omega_{t-1}^{(k)} - \eta \nabla L(\omega_{t-1}^{(k)}) \\ &= \omega_{t-1}^{(k)} - \eta \sum_{i=1}^{C} p^{(k)}(y=i) \nabla \mathbb{E}_{x|y=i}[\log f_i(x, \omega_{t-1}^{(k)})]\end{aligned}$$

We assume the synchronization, that is, the server aggregates model weights from all K clients, is conducted every E local epochs, and let $\omega_{mE}^{(f)}$ denote the global model weight updated after the m-th synchronization. Finally, we have:

$$\omega_{mE}^{(f)} = \sum_{k=1}^{K} \frac{n^{(k)}}{\sum_{k=1}^{K} n^{(k)}} \omega_{mE}^{(k)}$$

Based on the analysis of non-IID derived by [56], weight divergence between $\omega_{mE}^{(c)}$ and $\omega_{mE}^{(f)}$ is affected by two factors. First, if the clients in FL starts from different initialization, regardless of whether the data is non-IID or not, a considerable weight divergence is still expected. Second, the class distribution difference between the local datasets and the total data $\sum_{i=1}^{C} ||p^{(k)}(y=i) - p(y=i)||$ is another reason for the weight divergence. It implies that the more non-IID the local datasets are, $\omega_{mE}^{(f)}$ is farther away from the optimal model weight $\omega_{mE}^{(c)}$, and it leads to degraded accuracy of federated learning. We assume all the clients start from the same initialization as the centralized learning, so the non-IID data is the root cause of degraded accuracy and is the problem we try to solve.

2.2 Objectives of system design

In this subsection, we summarize the objectives of the system design and describe how we achieve those objectives.

Accuracy. To solve the non-IID problem and further increase the top-1 test accuracy of the model that solves the main task, we replenish the lack of data in local datasets and deal with the bias of the model weights introduced by non-IID data. The above two steps make $\omega_{mE}^{(f)}$ be more close to the optimal model weight $\omega_{mE}^{(c)}$ through changing the class distribution of each client and model weights respectively.

Cost. Cost and time complexity are significant considerations for whether an algorithm can be applied in practice. To reduce the cost of solving the non-IID problem, we introduced the concept of parallelism. In addition, we design an algorithm that allows us to avoid generating unnecessary data to reduce computation and storage costs.

Robustness. Making our framework robustness is a critical objective. We hope that GANDALF can handle non-IID in different situations, even in highly skewed non-IID data. To achieve it, we

Table 1. Comparison of recently proposed data augmentation methods, where ✓ means the objective is well achieved. △ means the work considered the objective but can be improved. Finally, × means the objective was not considered or achieved by the work.

Objective	Privacy	Robustness	Reliability
GANDALF	✓	✓	✓
RFA-RFD [47]	△	✓	×
FAug [27]	△	✓	×
Astraea [12]	△	×	×
Fedmix [52]	×	×	×

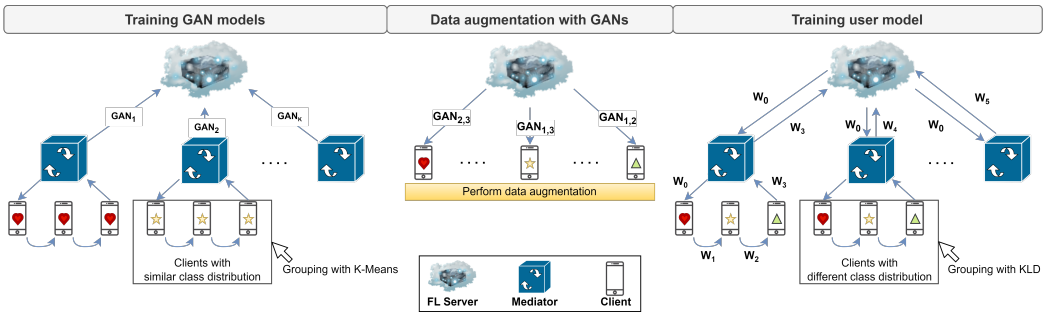


Fig. 1. The design of our proposed framework, *GANDALF*. The training process of our framework consists of three phases: training GAN models, data augmentation with GANs, and training user model.

introduce GAN, which offers valuable data for data augmentation and is more flexible than other data augmentation methods.

Reliability. Due to the unstable network connectivity of distributed systems, the training of FL is often interrupted, and a lot of overheads are caused. To handle it, we design several fault tolerance mechanisms which can avoid unavailable clients and stabilize the training process.

Privacy. Last but not least, privacy is essential for FL and also GANDALF. To provide a strict privacy guarantee in the whole process, we ensure that all private data are kept in the edge nodes while solving the non-IID problem. Furthermore, we introduce differential privacy with a Gaussian mechanism to make the model weights passed to each other leak less privacy.

3 FRAMEWORK DESIGN

In this section, we first discuss the design principles of our approach. Then the framework design and architecture of *GANDALF* are presented. Finally, we explain how reliability and security are ensured in *GANDALF*. The detail of our implementation is given in the next section.

3.1 Design Principals

Deep learning is a data-based technology. The gradient contributed by the clients with insufficient data or severe class distribution skews may cause the aggregated network to generalize poorly. One common method is performing data augmentation to replenish the target classes. However, existing data augmentation methods for solving the non-IID problem have some areas for improvement. As summarized in Table 1, our proposed framework has the advantage in privacy, robustness, and reliability, as discussed below.

Privacy. Astraea [12] performs static (Z-score-based) data augmentation with only local data and does not expose any sensitive information about private data. However, the server needs to know the probability distributions of clients, which could cause privacy leakage. In comparison, GANDALF does not expose clients' data and data distribution to other clients or the server. Fedmix [52] has the highest privacy vulnerability because it requires mashed (averaged) private data shared among clients, and the data leakage problem is only highly dependent on the number of data instances used in computing the average. FAug [27] uses the seed data samples uploaded and labeled by clients to train a global GAN model at the server for data augmentation. Also, FAug [27] attempts to reduce privacy leakage by adding noises to the uploaded data samples. In comparison, similar to RFA-RFD [47], GANDALF trains the augmentation generator locally at clients and only exchanges the generator model, not the data among clients. To provide a more strict privacy guarantee, GANDALF introduces differential privacy to ensure that the generated data and model weights passed between clients and the server do not leak privacy while FAug and RFA-RFD do not. The detail of the privacy guarantee is discussed in Section 3.5.

Robustness. To achieve robustness, we must ensure the quality and diversity of the augmented data. The static data augmentation (rotation, shift, flip, etc.) used by Astraea [12] and the mashed (averaged) data augmentation used by Fedmix [52] only produce limited plausible alternative data. Thus, they perform poorly in severe class distribution skews. In comparison, both the variational autoencoder (VAE) method used by RFA-RFD [47] and the GAN method used by FAug [27] and GANDALF can greatly improve the data quality with proven results [3, 16, 44, 58]. While the data quality of GAN is known to be better than VAE, GAN more easily suffers from the mode collapse problem, which limits the diversity of generated data. Hence, RFA-RFD [47] and FAug [27] address the problem by using conditional GAN/VAE, which can generate data for specific labels. On the other hand, we address this issue by training multiple generators for different classes of labels as described in Section 3.3. Therefore, RFA-RFD [47], FAug [27], and GANDALF can achieve robustness, while Astraea [12] and Fedmix [52] cannot.

Reliability. Even in the cross-silo environment, federated learning will still suffer from unstable network and node failure. Its instability often causes additional overhead and reduces the performance of DL models. If a framework does not consider reliability, it will not be applied in practice. To make GANDALF not only an algorithm but also a framework that can be applied in the real environment, we expect that GANDALF is reliable enough to exclude unavailable clients from participating in training and keep the system usually operating when clients disconnect arbitrarily. Although the previous works solved the non-IID problem with a well-designed algorithm, they all neglect the reliability discussion and analysis. In comparison, we implement some fault tolerance mechanisms to provide reliability as detailed in Section 3.4.

In sum, GANDALF solves the non-IID problem by performing data augmentation with multiple GAN models, ensuring the diversity of generated data. In addition, in the process, GANDALF keeps private data local and introduces differential privacy to ensure that the model weight and generated data do not leak privacy. Finally, We have implemented GANDALF that can work in a real environment and designed several fault tolerance mechanisms to ensure its reliability while other works do not. Therefore, we believe that GANDALF is a better solution for performing FL on non-IID data with enhanced security and reliability for practical uses.

3.2 Framework Overview

The overview of the proposed framework is shown in Fig. 1. This framework consists of three phases: Training GAN models, Data augmentation with GANs, and Training user model. In addition, our framework also introduces the mediator-based architecture, which brings optimizations and

reduces overhead. The detail of the framework and mediator-based architecture are discussed below.

Training GAN models. In this phase, we train multiple GAN models. At first, the clients with similar class distribution are grouped into the same group based on a clustering algorithm, and each group will be assigned a mediator. Next, each group trains a GAN model independently in parallel. In each group, the GAN model weight is passed between the clients and updated sequentially until it converges. Mediators in this phase are responsible for scheduling the training process of each group and returning the converged GAN model to the server.

Data augmentation with GANs. In this phase, each client performs data augmentation with GAN models. After the first phase, the server already has multiple trained GAN models. Next, the server sends the appropriate GAN models to each client according to its class distribution. Then, each client performs data augmentation with received GAN models to replenish the minority classes. Considering the computation cost of data generation and randomness of GAN data augmentation, we design an algorithm used to calculate the amount of data that each received GAN model should generate to minimize the computation cost. Furthermore, we also combine GAN data augmentation with static data augmentation to handle the randomness of GAN data augmentation. The detail of this algorithm is shown in section. 4.2.

Training user model. We train the model which solves the main task here. At first, we perform grouping, which makes the class distribution of each group as close to uniform as possible. Then each group will be assigned a mediator. Next, the server initializes the global model and broadcasts it to all mediators. After that, each group updates the received model weight independently in parallel. In each group, the model weight is passed between the clients and updated sequentially. Finally, the server aggregates the updated model weight of all mediators, performs weighted FedAvg, and updates the global model. Then the server broadcast the global model to all mediators again to start the next round of training. After several training rounds, we get a well-trained model at the end.

3.3 Mediator-based architecture

Mediator-based architecture brings two significant optimizations. One is handling the problem of mode collapse where the generator only concentrates on generating data lying on a few modes instead of the entire data space; namely, the diversity of generated data is insufficient. GANDALF trains a set of GAN models in parallel through the mediator-based architecture. It uses different collections of local datasets for training to encourage the generators to specialize in different data modes. That is why we perform grouping, which groups clients with similar class distribution into the same group in the first phase. A similar approach is also used by [24], [17]. With this optimization, we ensure the diversity of generated data. Compared with training a single GAN model for data augmentation [27], GANDALF can better balance local datasets. We will show this in Section 5.2

The second optimization is sequential training, where each client performs local training and updates the model sequentially. In addition to accelerating model convergence [28], sequential training can also counterbalance the bias of model weight introduced by non-IID data [12]. However, the communication overhead and time overhead of sequential training are K times that of FedAvg, where K is the number of clients. With mediator-based architecture, we significantly reduce the overhead by performing grouping and making each group perform sequential training parallelly under the coordination of the mediators. Hence, the overhead reduces to $\frac{K}{M}$ times, where M is the number of mediators. Thus, with this optimization, we accelerate the convergence of the models and reduce the time overhead of training. Furthermore, in the third phase, to better counterbalance the bias introduced by non-IID data instead of accumulating it, we make each group's class distribution

as close to uniform as possible. We will show the advantage brought by sequential training in Section 5.2.

3.4 Fault Tolerance

In reality, the clients may not complete the local training task within the time limit due to network issues, hardware limitations, and node failures. The common solution for reliability-agnostic clients is to define a maximum waiting time threshold T_R, discard clients that cannot complete the task in time and aggregate only for clients who successfully return the result. The difficulty of this solution lies in the choice of T_R. If T_R is too large, it will lead to spending too much time waiting for a few clients, and the training process will become inefficient. On the other hand, if T_R is too small, the server may miss too many clients that contribute generously to the global model, resulting in adverse effects on accuracy and model convergence. The mediator-based architecture also faces the same problem in such an unstable environment. To solve it, we design three fault tolerance mechanisms to ensure reliability and further reduce the time and communication overhead.

Training Order. For mediator-based architecture, the fail of local training tasks aggravates the weight divergence and cancels out the improvement brought by sequential training mentioned in Section. 3.3. In order to make each client's contribution to the global model more evenly in each round, we let each mediator determine the training order of the corresponding group based on a maintained table, which records the number of times each client completes a local training task. Whenever we need to decide which the next client is, the mediator prioritizes the clients based on the maintained table, and those less involved in training are given higher priority. On the other hand, to avoid client selection bias, we set an upper limit for the number of times each client is selected in a round and a minimum interval between two tasks for each client. Moreover, smaller T_R is allowed in our approach because we reduce the cost of missing a client by iteratively confirming the level of each client's contribution and re-balancing it. In short, we use shorter waiting time intervals to minimize the time overhead from the possible failed clients while avoiding the bias of model weight caused by the frequently selected clients.

Heartbeat. After a client is selected, its local training process still may not be complete due to node failure or degraded performance. To overcome this issue, our framework introduces the heartbeat message, a periodic signal generated by the client holding the model weight. If the mediator does not receive a heartbeat for a certain period, we assume the client is unavailable and perform a rollback to resume training as detailed next.

Rollback. As described in Section 3.2, the model weight is passed between clients without going through the mediator to avoid performance bottleneck. Hence, when the client holding the latest model weight becomes unavailable, we let the mediator perform the rollback mechanism to resume the training processing from the available client having the latest model weight. The rollback mechanism consists of the following steps. First, the mediator decides which is the next client to perform training. Then, the mediator requests the previous client that completed training to send the model weight to the next one. If the previous client is unavailable, the mediator asks the one before the previous instead until an available client is found.

3.5 Data Privacy

To provide a strict privacy guarantee on the model weight and generated data, we combine differential privacy(DP) with our framework. We formally define DP as follows,

Definition 1. *(ϵ, δ)-DP [14]. A randomized algorithm $\mathcal{M}$ with domain $\mathcal{X}$ is (ϵ, δ)-differential private if for all $\mathcal{S} \subseteq Range(\mathcal{M})$ and for all adjacent databases $x, y \in \mathcal{X}$,*

$$Pr[\mathcal{M}(x) \in \mathcal{S}] \leq e^{\epsilon} Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta \tag{1}$$

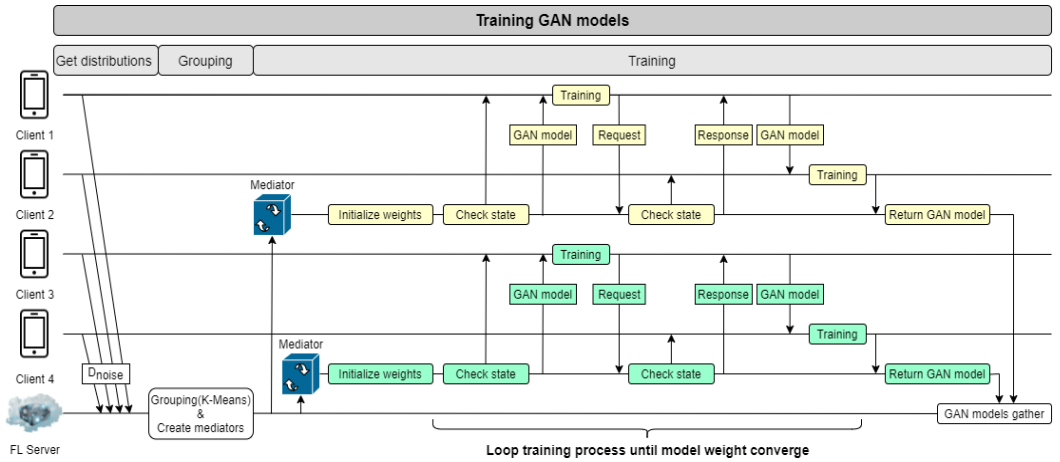


Fig. 2. Workflow of training GAN models for data augmentation.

According to [14], a Gaussian mechanism can be used to guarantee (ϵ, δ)-DP. We present this as follows,

Theorem 1. *The given noise* $n \sim \mathcal{N}(0, \sigma^2)$ *satisfies* (ϵ, δ)*-DP if the parameter* $\sigma \geq c\Delta s/\epsilon$*, and the constant* $c \geq \sqrt{2ln(1.25/\epsilon)}$ *for* $\epsilon \in (0, 1)$*, where* $\Delta s = max_{x,y}||\mathcal{M}(x) - \mathcal{M}(y)||$*, which is the sensitivity of the algorithm* $\mathcal{M}$

We achieve differential privacy by adding Gaussian noise satisfying Theorem. 1 to the model weights and other private information such as class distribution. In the first and third phases, clients always add noise to the model weight before passing it to another.

Finally, according to [14], DP not only protects the model weight from violating privacy but also the generated data. We provide proof in Theorem 2 below.

Definition 2. *Post-processing property of DP [14]. Let a randomized algorithm* $\mathcal{M}$ *be* (ϵ, δ)*-differential private, and let* F *be any deterministic or randomized function. Then* $F(\mathcal{M})$ *also satisfies* (ϵ, δ)*-differential privacy.*

Theorem 2. *The output of a generator guarantees* (ϵ, δ)*-differential privacy.*

Proof. According to Theorem. 1, the discriminator and the generator trained in each client has satisfied DP, and with the Definition. 2, the data generated by the generator also satisfies DP.

Although differential privacy provides adequate protection for privacy without expensive computational cost, it affects the model's performance. We will show the trade-off between privacy and accuracy in section 5.2.

4 IMPLEMENTATION DETAILS

In this section, we explain the detailed workflow implementation of the three phases mentioned in Section. 3.

4.1 Training GAN Models

As shown in Fig. 2 and Algorithm. 1, we train multiple GAN models for data augmentation in this phase. First, to determine the clients participating in the training and obtain the class distributions used to perform grouping, each client needs to join the training task by uploading its

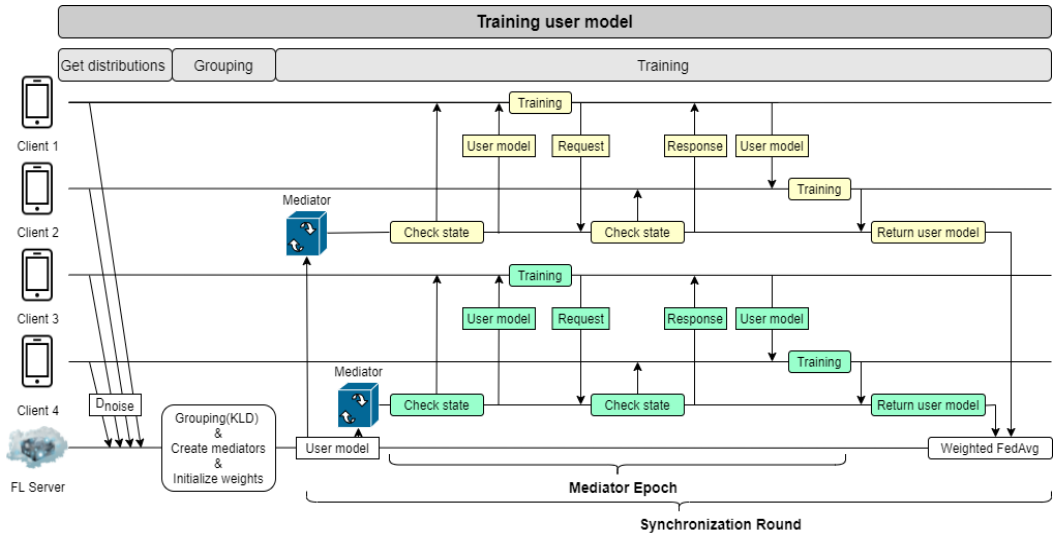


Fig. 3. Workflow of training the user model based on FedAvg algorithm.

Algorithm 1 Training WGAN with gradient penalty based on FL

Input: s_g, sensitivity of generator model weight. s_d, sensitivity of discriminator model weight. T_d, number of discriminator iterations per generator iteration. T_g, generator iteration. σ_n, noise scale. λ, penalty coefficient. M_g, number of mediators. b, batch size.

Output: M_g GAN models

for each mediator m in $m = 1, ..., M_g$ parallelly **do**
  Initialize : weights of discriminator ω, weights of generator θ
  while θ has not converged **do**
    Take out a client from mediator m
    for t_g = 1,...,T_g **do**
      for t_d = 1,...,T_d **do**
        for i = 1,...,b **do**
          Sample real data $x \sim P_r$, latent variable $z \sim p(z)$, a random number $\gamma \sim U[0, 1]$
          $\tilde{x} \leftarrow G_\theta(z)$
          $\hat{x} \leftarrow \gamma x + (1 - \gamma)\tilde{x}$
          $g \leftarrow \lambda(||\nabla_{\hat{x}} D_\omega(\hat{x})||_2 - 1)^2$
          $L^{(i)} \leftarrow D_\omega(\tilde{x}) - D_\omega(x) + g$
        $\omega \leftarrow Adam(\nabla_\omega \frac{1}{b} \sum_{i=1}^{b} L^i, \omega, \alpha_d)$
      Sample a batch of variables$\{z_i\}_{i=1}^{b} \sim p(z)$
      $\theta \leftarrow Adam(\nabla_\theta \frac{1}{b} \sum_{i=1}^{b} -D_\omega(G_\theta(z)), \theta, \alpha_g)$
    $\theta \leftarrow \theta + N(0, (\sigma_n s_g)^2 I)$
    $\omega \leftarrow \omega + N(0, (\sigma_n s_d)^2 I)$
  Send θ to server

Algorithm 2 Grouping with K-Means

Input: M_g, number of mediators. $iter$, termination condition of $K\text{-}Means$.
Output: $S_{mediator}$, result of grouping.
1: $Initialize : S_{mediator} \leftarrow \emptyset, S_{km} \leftarrow \emptyset, P \leftarrow P_1, ..., P_K$.
2: **repeat**
3: $S_{km} \leftarrow K\text{-}Means(P, M_g, iter)$.
4: **until** sensitivity of $S_{km} < M_g$
5: **for** each set s in S_{km} **do**
6: Create mediator m
7: **for** each client c in s **do**
8: Mediator m add client c
9: $S_{mediator} \cup m$

class distribution with Gaussian noise to the FL server. After all clients join, the server performs grouping, which groups clients with similar class distribution into the same group to solve the mode collapse problem. The policy of grouping is shown in Algorithm. 2. This algorithm is based on K-Means, a clustering algorithm, and P is the input of K-Means, where P means the class distributions of all clients. Because K-Means can not guarantee that the number of clients in each group is close, leading to an increase in time overhead, we execute K-Means a few times until the difference in the number of clients each mediator has is less than M_g (Algorithm. 2, line 2). Once grouping finish, we create M_g mediators, which are responsible for scheduling the training process of the groups (Algorithm. 2, line 5).

As mentioned in Section. 3.3, to solve the mode collapse problem, we use different collections of local datasets to train a set of GAN models. In the implementation, we achieve the goal by making each mediator train a GAN model with the data in its group independently. The process is as follows. First, each mediator initializes the model weight respectively and decides on a subordinate client performing local training (Algorithm. 1, line 4). Then the mediator sends it to the chosen client. Next, the client holding model weight performs training with local data. In local training (Algorithm. 1, line 5), the generator is updated after the discriminator is trained T_d times, and the generator is updated T_g times. Finally, at the end of local training, the client adds Gaussian noise with noise scale σ_n to the model weight to satisfy the differential privacy (Algorithm. 1, line 16, 17).

After local training, the model weight will be passed to the next client. As mentioned in Section. 3.4, the training order is determined by the mediator to avoid additional time overhead caused by unavailable clients. First, the client requests the mediator to obtain information about the next client (Fig. 2, Request). After that, the mediator decides which next client is and sends the information about the next one to the client, which sends a request. Finally, the model weight is passed to the next client. Then the model weight is continuously updated and given between clients until the model converges. We judge whether the model converges mainly based on whether the loss change of the generator and discriminator tends to be flat. Once the model converges, the mediator lets the client holding the model weight return the model to the mediator. Finally, the mediator returns it to the server.

4.2 Data Augmentation with GANs

After the first phase, the server has M_g GAN models. Given communication overhead, it is unreasonable to send all the GAN models to each client. Instead, the server determines which GAN models a client needs based on its class distribution. We assume that the training of each GAN model goes well, the class distribution of data generated by the GAN model is the same as that

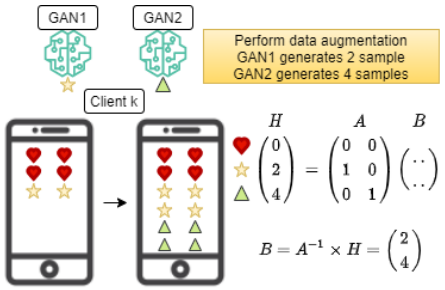


Fig. 4. Example of GAN data augmentation. A client k with 4 hearts and 2 stars receives two GAN models that focus on generating stars and triangles, respectively. H means how many samples of each class this client lacks to reach uniform distribution. A means the class distributions of received GAN models, namely the first and second columns of A represent the class distribution of GAN1 and GAN2, respectively. Finally, B, the target matrix, means how much data each GAN should generate. After calculation, we obtain $B = [2, 4]$, which implies that GAN1 and GAN2 should generate 2 and 4 samples, respectively.

of the training dataset. Then we calculate the Kullback–Leibler divergence (KLD) between GAN models and the client. The larger the value, the more likely the GAN model generates the data that the client lacks. After calculation, the server sends J GAN models to each client respectively, where $J \leq M_g$.

Given that using GAN models to generate data has randomness and brings computation costs, we design an algorithm used to calculate how many samples each GAN model should generate theoretically for a client to achieve the best balance and the least computation costs. As shown by the example in Fig. 4 and the algorithm is described as follows. First, we simplify the problem to a matrix multiplication problem. To do so, we express H as the product of two matrices: $H = AB$, where $A \in \mathbb{R}^{C \times J}$, $B \in \mathbb{R}^{J \times 1}$, C is the number of classes, and J is the number of GAN models the client receives. The matrix A means the class distribution of J GAN models, the matrix B means how many samples each GAN model should generate, and matrix H means how many samples of each class this client lacks to reach uniform distribution. A and H can be obtained from the server and its dataset, and B is the target we need to calculate. Finally, We obtain B by calculating the pseudo inverse matrix of A and basic matrix multiplication operations.

Zhao *et al.*[56] shows that when the data become less non-IID to a certain degree, its decrease does not bring a significant improvement to the accuracy. In other words, we do not need to generate many samples to reach the uniform distribution. Based on the observation, We finally multiply matrix B by τ, where $\tau \in (0, 1]$. τ indicates the degree of balancing. Taking Fig. 4 as an example, if $\tau = 0.5$, the matrix B becomes [1,2], which implies that the client only needs to generate half of the number of data. As a result, the computation and storage cost can be reduced, but the degree of data balancing also decreases. Thus, τ is a parameter for adjusting the trade-off between accuracy and cost, and its impact is evaluated by our experiments in Section. 5.2.

Because the data augmentation with GANs has randomness, the actual class distribution of generated data is usually not as expected. So, finally, we make some adjustments to the local dataset and the generated data. If the amount of data actually generated is more than theoretically generated, we delete the redundant generated data for each class. Otherwise, we use static data augmentation to generate at most twice the amount of source data to make up for the gap.

Algorithm 3 Hierarchical distributed neural network training

Input: s_c, sensitivity of model weight. M_c, number of mediators. E_m, number of mediator epoch. E, number of local epoch. η, learning rate. σ_n, noise scale.
Output: A trained user model.

1: $Initialize : \omega_1$
2: **for** each synchronization round $r = 1, ..., R$ **do**
3: **for** each mediator m in $1, ..., M_c$ parallelly **do**
4: $\omega^* \leftarrow \omega_r$
5: **for** each mediator epoch $e_m = 1, ..., E_m$ **do**
6: **for** each available client i in mediator m **do**
7: **for** each local epoch $e = 1, ...E$ **do**
8: $\omega_r \leftarrow \omega_r - \eta \nabla L(\omega; \mathbb{X}^{(i)}, \mathbb{Y}^{(i)})$
9: $\omega_r \leftarrow \omega_r + N(0, (\sigma_n s_c)^2 I)$
10: $\Delta\omega^m_{r+1} \leftarrow \omega_r - \omega^*$
11: $\omega_{r+1} \leftarrow \omega_r - \sum_{m=1}^{M_c} \frac{n_m}{n} \Delta\omega^m_{r+1}$

Algorithm 4 Grouping with KLD,D_{KL} is Kullback-Leibler divergence

Input: M_c, number of mediators.
Output: $S_{mediator}$, result of grouping

1: $Initialize : S_{mediator} \leftarrow \emptyset, S_{client} \leftarrow 1, ...K$
2: **while** S_{client} is not $\emptyset$ **do**
3: Create mediator m
4: **while** S_{client} is not $\emptyset$ and $|m| < \lceil \frac{K}{M_c} \rceil$ **do**
5: $c \leftarrow argmin_i D_{KL}(P_m + P_i || P_u), i \in S_{client}$
6: Mediator m add client c
7: $S_{mediator} \leftarrow S_{mediator} \cup m$

4.3 Training user Model

As shown in Fig. 3 and Algorithm. 3, we train a model which solves the main task here. Same as the first phase, each client needs to join the training task by sending the class distribution with Gaussian noise to the FL server. After determining the clients participating in the training and obtaining the class distributions, the server performs grouping, making each group's class distribution close to the uniform distribution to allow sequential training to improve non-IID better. The policy of grouping of this phase is shown in Algorithm. 4, which is a greedy algorithm. The server traverses the class distribution of all clients and selects the one that minimizes the KLD between the class distribution $P_m + P_i$ and uniform distribution P_u (Algorithm. 4, line 5). When the number of clients in the group reaches the max limitation, the server will create a new mediator and repeat the above process until all clients have been grouped (Algorithm. 4, line 4).

After grouping, the server initializes the global model and broadcasts it to all mediators to start a new synchronization round. Next, each mediator schedules the training of its group in parallel (Algorithm. 3, line 3) to reduce the overhead brought by sequential training. The process is as follows. First, each mediator decides on a client ready to perform local training. After that, the mediator sends the model weight to the chosen client. Next, the client trains the model for E local epochs and then sends the model weight with Gaussian noise to the next client through the information provided by the mediator. Under the scheduling of the mediator, the unavailable

clients will be skipped to avoid additional overhead. We call it a mediator epoch that all available clients in the group have been trained once (Algorithm. 3, line 5). After looping mediator epochs E_m times, the server performs aggregation and makes all clients holding the model weights send it to the server through mediators (Algorithm. 3, line 10). Next, because the amount of data in each mediator is various, we adopt weighted FedAvg, which is more reasonable (Algorithm. 3, line 11). The server aggregates the model weights with the weight of n_m/n, where n_m is the size of data in the group m, and n is the size of data of all groups. Then, the server updates the global model to ω_{r+1}. We call the process from broadcasting to aggregation a synchronization round. Finally, the server broadcast updated model weight to mediators to start the next synchronization round.

5 EVALUATION

In this section, we introduce our experimental setup and then present the results of our evaluation in four parts. First is to analyze the training accuracy of our framework under various system settings, including noise scale σ_n, degree of balancing τ, and the number of mediators M_g, M_c. The second is to show that our framework can outperform other existing solutions for better training accuracy. The third is to show that our solution is capable of solving the more extreme n-class non-IID problem, which the traditional static data augmentation approaches cannot solve. Finally, we evaluate the impact of the fault tolerance mechanisms on test accuracy and model convergence.

5.1 Experimental Setup

Implementation: We implement the proposed solution based on Tensorflow [1]. The communication between clients, mediators, and the server is implemented using the NATS messaging library. The IBM differential privacy library [25] is used to add noise to the model weights for privacy. Our experiments were conducted on an in-house cluster with two physical nodes. Each node has 2 Intel Xeon E5-2620 v4 2.1GHz CPUs, 4 V100 GPUs, and 128GB memory.

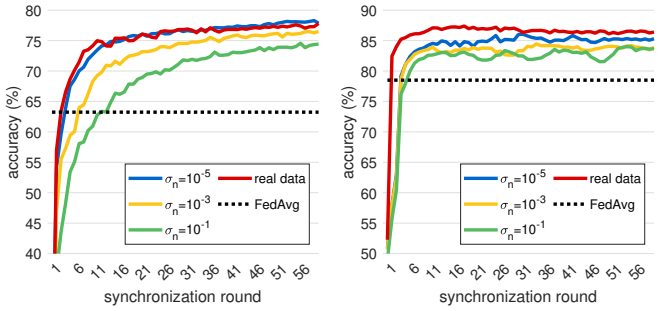
Dataset: We evaluate our solution on CINIC-10 [9] and EMNIST [7] that have 10 and 47 classes, respectively. CINIC-10 has a total of 270k images, 4.5 times that of CIFAR-10 [31](and more challenging). EMNIST(balanced), an extension of MNIST [10], has 131k images and constitutes a more challenging classification task involving letters and digits. For the non-IID setting, we divide the training dataset into 20 partitions of the same size. Then each partition has four major classes, whose size is about ten times that of the other minority classes. Finally, a total of 20 clients will be randomly assigned one partition.

Models: The implemented GAN model is as same as WGAN with gradient penalty [21], and we set the batch size to 256, and the learning rate of discriminator and generator to 1.0×10^{-4} and 5.0×10^{-5} respectively. We use a widely used model, ResNet-18 [23], as the user model. For training, we set the batch size to 128, and the optimizer of ResNet-18 is SGD with learning rate 1.0×10^{-1} and momentum 0.9. Finally, we evaluate the user model with top-1 accuracy on the testing set and consider the maximum accuracy while training as the result of the model.

FL setting: For FL setting, we set iteration of generator T_g to 3, iteration of discriminator T_d to 2, the local epoch of the user model E to 3, mediator epoch E_m to 1, and the number of clients to 20, as same as the number of partitions mentioned in *Dataset*.

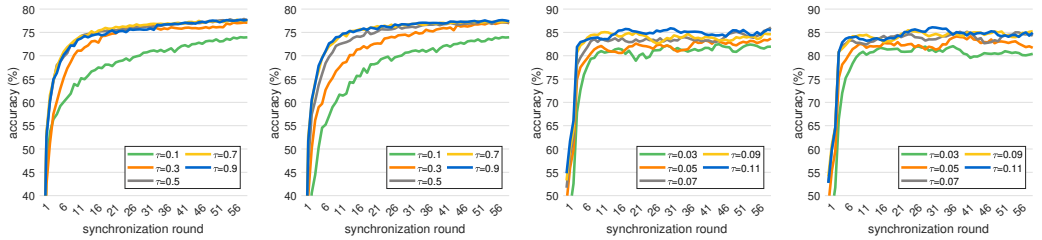
5.2 Effect of Framework Parameters

Our framework has three important parameter settings, naming the noise scale σ_n, degree of balancing τ, and number of mediators M_g, M_c. We evaluate the impact of these settings to the model accuracy result as follows. The default settings of these parameters for our approach in all the experiments are $M_g = 2$, $M_c = 2$, $\tau = 0.5$ for CINIC-10 [9], $\tau = 0.07$ for EMNIST [7], and $\sigma_n = 1 \times 10^{-5}$. The result of FedAvg [39] is also shown in the plots as a comparison baseline.



(a) On CINIC-10 (b) On EMNIST

Fig. 5. Test accuracy of GANDALF under varied noise scale training settings of σ_n. The results are compared with the idea result from using real data for data augmentation and the result from FedAvg without data augmentation.



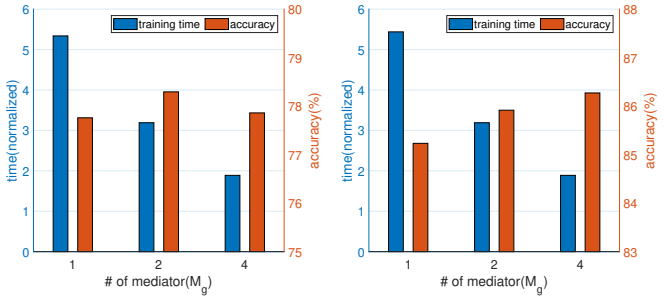
(a) With static data augmentation, on CINIC-10 (b) Without static data augmentation, on CINIC-10 (c) With static data augmentation, on EMNIST (d) Without static data augmentation, on EMNIST

Fig. 6. Test accuracy of GANDALF under varied degree of balancing settings of τ. The results of both with and without using static data augmentation to make up the lack of generated data are shown.

5.2.1 Quality of generated data. Noise scale σ_n, the standard deviation of the Gaussian noise used to satisfy differential privacy mentioned in Section. 3.5, determines how large the noise is added while training GAN models to achieve differential privacy. Fig. 5 shows that the accuracy is reduced, and the model converges slower when the noise scale increase from 10^{-5} to 10^{-1}. The reason for the decline is that the considerable noise reduces the quality of generated data, and the class distribution of generated data has a more significant gap from the theoretical value. This results in the inability to balance the local datasets and adversely affects training.

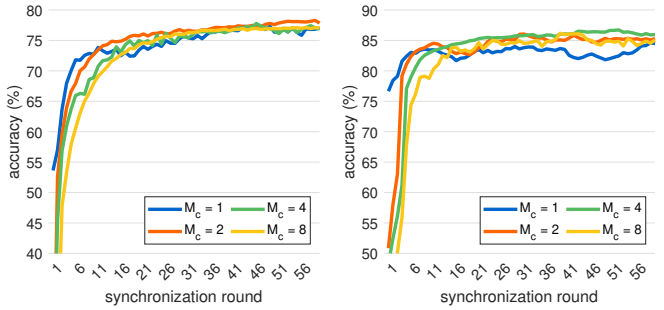
In order to show that the data generated by GAN has the potential to be more beneficial to training than the well-processed data in the training dataset, we also show the result of using real data for data augmentation in Fig. 5 labeled as "real data". Surprisingly, we found that if σ_n is small enough, the generated data could obtain a similar or even better accuracy than the real data. We believe the reason is that a small amount of data generated by GAN not only can solve the non-IID problem without exposing private data but also makes the model more robust with more diverse features and information such as image size, shape, and location of key components. Therefore, our approach achieves both the goals of accuracy and privacy.

5.2.2 Degree of balancing and static data augmentation. Degree of balancing τ is a parameter that determines the number of data GAN generated. As shown in Fig. 6, The more data GAN generates,



(a) On CINIC-10 (b) On EMNIST

Fig. 7. Test accuracy and training time of *G*ANDALF under varied number of mediators while training GAN models.



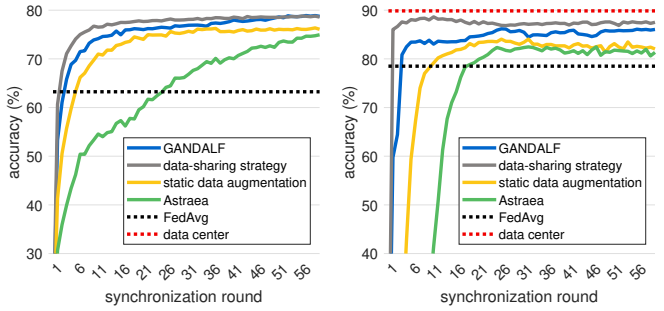
(a) On CINIC-10 (b) On EMNIST

Fig. 8. Test accuracy of *G*ANDALF under varied number of mediators while training the user model.

the higher the accuracy. However, after τ reaches 0.5, its increase will not bring significant changes to accuracy. This observation shows that generating too much generated data may be inefficient to balance local datasets because computation cost is proportional to the amount of generated data, but the accuracy is not. Therefore, our approach provides a tuning knob to address the performance and accuracy trade-off.

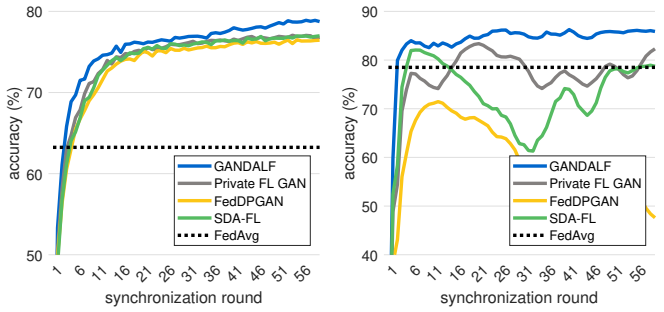
To observe the benefit of using static data augmentation to make up for the lack of generated data, we plot the result without static data augmentation in Fig. 6. We found that static data augmentation only improves the accuracy when τ is small enough. The results of with and without static augmentation are almost the same when τ is larger than 0.5 for CINIC10 and 0.07 for EMNIST. Therefore, our GAN augmentation is robust enough by itself. However, static augmentation does have the advantage of consuming less computing power for data generation than GAN augmentation. Thus, static augmentation can still be helpful to combine with our method to reach similar accuracy results with less computing power requirement.

The above results show that our proposed framework provides the mechanisms to address the trade-off between model accuracy, cost, and performance. However, the discussion of finding the best setting for optimizing performance and cost is considered to be our future work and out of the scope of this paper.



(a) On CINIC-10 (b) On EMNIST

Fig. 9. Test accuracy comparison of different data augmentation algorithms.

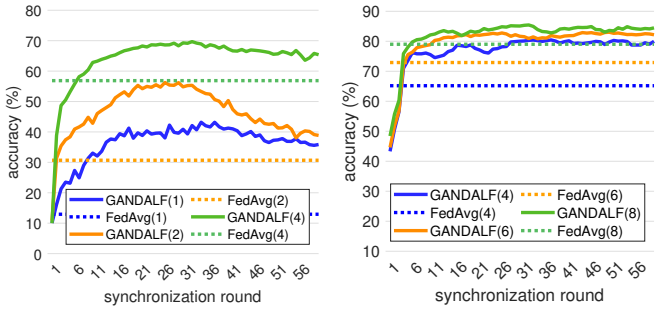


(a) On CINIC-10 (b) On EMNIST

Fig. 10. Test accuracy comparison of different GAN data augmentation algorithms.

5.2.3 Number of mediators. Fig. 7 shows the accuracy of the user model and training time of GAN models under the different number of mediators M_g while training GAN models. To evaluate the relationship of the time overhead between training GAN models and the user model, we normalize the training time of GAN models by dividing it by the training time of the user model. In Fig. 7, as the number of mediators increases, the time overhead of training GAN models also decreases. Because each mediator performs training in parallel, increasing the number of mediators is equivalent to increasing parallelism and reducing time overhead. For accuracy, we observe that training with only one mediator (one GAN model) got the lowest accuracy in both datasets. This result shows that training multiple GAN models in parallel instead of one can mitigate the mode collapse problem and slightly improve accuracy.

Next, We evaluate the different number of mediators M_c while training a CNN model. Fig. 8 shows that with the advantage of sequential training, we could accelerate the convergence of the model without affecting accuracy by decreasing the number of mediators. However, reducing the number of mediators will also reduce the parallelism of training. Thus, it is a trade-off between convergence and parallelism, and how to set this system parameter to achieve the minimum time overhead is another interesting and challenging problem for our future works.



(a) On CINIC-10 (b) On EMNIST

Fig. 11. Test accuracy of GANDALF for the n-class non-IID problem, where each client only has data from n classes (n = {1,2,4}).

5.3 Solution Comparison

Next, we conducted experiments to compare the accuracy result of our framework with other types of data augmentation and state-of-the-art GAN data augmentation methods.

5.3.1 Data augmentation algorithms. In this subsection, we first compare GANDALF with the following two baselines and three data augmentation methods which do not use GAN models.

- FedAvg [39]: a widely used FL algorithm, which has been deployed in a large-scale system to test its effectiveness [4]. Also, [39] shows that FedAvg is robust to non-IID data.
- Datacenter: a standard ML approach that trains with a centralized training dataset on a machine or in a data center.
- Data-sharing strategy [56]: we establish a globally shared dataset G that each client contributes and consists of a uniform distribution over classes. The size of G is $0.2 \times ||D||$, where $||D||$ represents the number of data in training dataset. And then, a random 20% of G is distributed to each client.
- Static data augmentation: each client doubles the data size of the six minority classes of its local dataset using the static data augmentation operations, including rotation, shift, and flip.
- Astraea [12]: a self-balancing FL framework that alleviates the non-IID by i) Z-score-based data augmentation and ii) multi-client rescheduling.

As shown in Fig. 9, although our solution converges slower than the data-sharing strategy, it achieves a similar or even better accuracy. This similar observation can also be found in Fig. 5: Generated data can bring performance close to that of real data. The generated data not only balances the local datasets but also enhances the robustness of the model. Among them, static data augmentation has a slower convergence speed. The reason for the slower convergence of static data augmentation is that the amount of supplemented data is not as enough as GANDALF. On the other hand, it is not appropriate to use static data augmentation to generate a large amount of data because it may cause a model to overfit the minority class, which is being oversampled. Similarly, the Z-scored-based data augmentation proposed by Astraea [12] can not effectively supplement the data for the differences of each local dataset. Therefore, the two works have a slower convergence rate and less accuracy. In short, static data augmentation is not appropriate for severe class distribution skew.

5.3.2 GAN model training method. This subsection aims to prove our proposed GAN model training method described in Section 4.1 can help the GANDALF framework to achieve better user model accuracy on non-IID data by comparing to the following three state-of-the-art GAN data augmentation methods. Each of them proposes its own method to train a GAN model in the FL environment as described below.

- Private FL GAN [48]: a method combining serialized model-training and differential privacy into GAN model training. Specifically, all clients sequentially train the GAN model and add random noises to the GAN model while communicating.
- FedDPGAN [55]: a method which trains GAN models with FedAvg algorithm.
- SDA-FL [36]: a method making each client trains a GAN model based on its private data independently. After training, we will get K GAN models, where K is the number of clients.

To focus our comparison on the GAN model training method, we only replace how the GAN model is trained in our GANDALF framework by the three compared methods above, while the rest of steps, including data augmentation and user model training, remain unchanged.

As observed from the final user model accuracy comparison shown in Fig. 10, GANDALF with our proposed GAN model training method successfully achieves the best model performance and convergence comparing to all other state-of-the-art methods. Among the three methods, private FL GAN is the special case of GANDALF, where the number of mediators M_g equals 1. The benefits and impact of the number of mediators on training GAN have been discussed in Section. 5.2.3. Compared with private FL GAN, we introduce mediators to improve the parallelism of training and better solve mode collapse. In addition, SDA-FL is also a special case of GANDALF, where the number of mediators M_g equals the number of clients. Although SDA-FL lets all clients perform training to achieve better parallelism, the trained GAN models may be overfitting due to the limited size of a single local dataset. Overfitting GAN models may not be suitable for data augmentation since they simply memorize a few data points and cause the user model to overfit the generated data. Finally, FedDPGAN is the most naive method. Compared with FedDPGAN, private FL GAN and SDA-FL show the benefit of sequential training and multiple GAN models, respectively. Furthermore, GANDALF cleverly combined the two mechanisms, thus achieving the best results.

5.4 n-class non-IID Problem

In order to further demonstrate that our solution can be applied to more extreme cases than static data augmentation, we evaluate our solution in three extreme cases where static data augmentation cannot work at all. The extreme cases are named n-class non-IID, where each client only has data from n classes, and n = {1,2,4} for CINIC-10 [9], and n = {4,6,8} for EMNIST [7]. In these extreme cases, each client does not have any data of minority class, so it is impossible to balance the local datasets through static data augmentation. As shown in Fig. 11, our solution effectively improves the accuracy in all cases.

In summary, the above experiments show that our solution not only can obtain higher accuracy than other solutions with almost no invasion of privacy but also can be applied to more extreme non-IID cases, which the traditional static data augmentation approaches cannot solve.

5.5 Fault Tolerance

Finally, we conducted experiments to show the fault tolerance (FT) mechanisms we designed for improving test accuracy and accelerating model convergence. To simulate the unpredictable network and node failures or performance degradation in the real-world environment, we introduce a randomly generated delay time to the local training process of each client in the experiment. The baseline comparison method without our fault tolerance mechanism will wait for the response

Table 2. Test accuracy and Model convergence with/without Fault Tolerance

		Stop @R_{max} = 40						Stop @Acc = 0.7/0.8					
		Best accuracy			Round length(min)			Round needed			Total time(min)		
Dropout rate		0.1	0.4	0.7	0.1	0.4	0.7	0.1	0.4	0.7	0.1	0.4	0.7
CINIC-10	With FT	78.17	77.85	76.89	3.92	3.81	3.69	7	7	12	27.44	16.75	44.27
	Without FT	76.80	74.58	71.97	5.66	10.96	16.20	9	11	18	50.91	120.53	291.57
EMNIST	With FT	86.82	86.20	85.58	5.07	4.81	4.17	4	6	7	20.28	28.89	29.17
	Without FT	85.48	83.93	81.74	6.98	12.08	16.46	6	10	21	41.90	120.83	345.70

of clients for a maximum waiting time threshold T_R. Once the waiting time threshold is reached, mediators will skip the client from the training round and move on to the next client in the fixed training sequence. In contrast, our fault tolerance mechanism dynamically adjusts each client's training priority and sets smaller T_R with less cost. Therefore, with FT, we can use shorter waiting time intervals to minimize the wait time overhead from the possible failed clients while avoiding the bias of model weight caused by the frequently selected clients.

The improvement of our approach is evaluated in two use case scenarios: i) stop the training at a maximum preset synchronization round R_{max}. ii) stop the training when a preset test accuracy is achieved for the global model. In Table. 2, we present the result in terms of maximum test accuracy in R_{max} rounds, the average length of a round, the number of needed rounds, and the total time duration for achieving sure accuracy. The results in the table were collected from the experiments when we gradually increased the degree of delay time from failures. The dropped rate in the table indicates the percentage of client requests that exceeds the maximum waiting time threshold under the training method without FT. As shown in Table. 2, training with FT gets better accuracy and reduces the number of needed rounds and time to achieve the target accuracy in all cases, especially in the cases where the dropout rate is greater than 0.4. Since the FT tolerance mechanisms mentioned in Section. 3.4 adjust the training sequence of clients to avoid the bias of model weight introduced by clients frequently selected. Furthermore, it also introduces heartbeat mechanism to avoid additional time overhead caused by node failure.

6 RELATED WORK

Federated Learning enables edge nodes to train a model collaboratively while keeping all the data on local clients and ensuring privacy. Existing FL research mainly focuses on addressing key challenges such as heterogeneity of local datasets, communication overhead, privacy issues, etc.

Non-IID: Recently, Zhao *et al.* [56] showed that non-IID data reduce the accuracy of federated learning through experiment and mathematical demonstration. To solve the non-IID issue, Li *et al.* [33] proposed FedProx, which introduces a proximal term to the loss function to alleviate weight divergence. Similarly, Li *et al.* [32] proposed MOON, which utilizes the similarity between model representations to correct the local loss function. In addition, several works introduce existing DL techniques to solve non-IID. Kavya Kopparapu *et al.* [30] combine federated learning and lifelong learning to tackle catastrophic forgetting brought by non-IID data. Park *et al.* [42] proposed GPAL, which utilizes meta-learning to prepare a well-initialized model that can adapt various local datasets within a few rounds. Another intuitive way is to use data augmentation. Yoon *et al.* [52] proposed FedMix, which appropriately modifies Mixup [54] to generate synthetic data by linear interpolation between samples. Yoshida *et al.* [53] proposed HybridFL, which gathers private data from volunteered clients to build a shared dataset and use it to update the global model. Compared with the above two data augmentation works [52, 53], our work combines static and GAN data augmentation to make our method more robust. Furthermore, we keep the data on the

local side instead of uploading it to the server [53] throughout the process of training GAN models to minimize the privacy cost.

GAN data augmentation: The existing literature has proved that GAN can indeed be applied to data augmentation. Bowles *et al.* [5] and Frid-Adar *et al.* [15] show that GAN can be used for data augmentation and improve the performance of the model for medical imaging. Tanaka *et al.* [44] show that the fake data generated by GAN has the potential to help a decision tree and different networks achieve better accuracy. Some works noticed that mode collapse hinders the performance of GAN data augmentation. Li *et al.* [34] add an independent classifier and introduce Wasserstein distance to the loss functions to overcome model collapse and gradient vanishing. Guo *et al.* [20] introduce multiple generators to ensure the diversity of generated data. Recently, how to train a GAN model based on distributed learning or FL effectively has been a focus of active work. Rasouli *et al.* [43] proposed a communication-efficient distributed GAN which is robust to reduced communications. Xin *et al.* [49] proposed a differential privacy GAN based on FL, which provides a strict privacy guarantee. In addition, some works try to apply GAN data augmentation into FL to solve the non-IID issue. Zhang *et al.* [55] proposed FedDPGAN, which used GAN to generate patient data of COVID-19 privately. Nguyen *et al.* [40] proposed a scheme for COVID-19 detection by the joint design of FL, blockchain, and GAN. Li *et al.* [36] proposed SDA-FL, which makes each client train a GAN model independently and generate synthetic data. The applications [40, 55] widely used in COVID-19 verified the feasibility of GAN in a cross-silo setting. Compared with them, GANDALF uses multiple GAN models to ensure the diversity of generated data and the effectiveness of data augmentation. On the other hand, SDA-FL [36] train multiple GAN models as we do, but it does not allow clients to perform GAN training collaboratively and has a higher risk of overfitting. Compared with SDA-FL, we train GAN models in more flexible ways and bring more improvements in the final accuracy.

Privacy issues: Regarding privacy, Liu *et al.* [37] proposed a framework, Federated Transfer Learning (FTL), that is compatible with two secure approaches, namely, homomorphic encryption and secret sharing. Differential privacy [13] is widely used to protect the recovery of specific data information during training. The impact of differential privacy on the quality of ML and FL training is explored by Abadi *et al.* [2] and Wei *et al.* [46]. Our work introduced differential privacy to reduce privacy leakage, but it will affect performance. Research related to privacy will help us achieve better performance and reduce costs.

7 CONCLUSION

In this paper, we proposed GANDALF, which solves the non-IID problem under the cross-silo setting effectively and meets the requirements of preserving privacy, reducing the cost, robustness, and reliability. In GANDALF, we introduce GAN to avoid data augmentation relying on existing data in a single local dataset like traditional static data augmentation. Thus it is robust to more extreme non-IID problems. Also, we introduce mediator-based architecture combined with sequential training, which considerably improves the model's convergence speed and further reduces overhead. With this architecture, we also cleverly ensure the diversity of the generated data by training multiple GAN models in parallel. Furthermore, in terms of privacy, we do not share private data and ensure that model weight does not leak privacy by adding artificial noise that satisfies differential privacy, enabling GANDALF to provide strict privacy guarantees. Finally, we design multiple fault tolerance mechanisms to provide reliability and avoid the overhead caused by unavailable clients. Experiment results show that GANDALF improved 15.66 and 14.8 top-1 test accuracy over FedAvg on imbalanced CINIC-10 and imbalanced EMNIST, respectively. Also, we demonstrate the robustness of GANDALF by applying it to n-class non-IID, and we significantly improve the accuracy in all cases. Finally, we

show that the fault tolerance mechanisms for GANDALF bring better accuracy and faster model convergences while training with highly unavailable clients.

Finally, there are some limitations and promising areas of improvement for GANDALF. First, data labeling, an unavoidable step for GAN data augmentation, causes additional overhead and affects test accuracy. Although many studies and automated data labeling services such as AWS sagemaker can assist in labeling, performing labeling effectively in decentralized environment is still an open question. Second, training GAN model is known to be time consuming and power demanding. But as known, the computing power of edge devices has proliferated over the past few years, and the increasing rate is not expected to slow down. Many recent research also aims to reduce the computational cost of model training, such as the model compression [6] technique. Hence, we would also like to investigate and adopt these solutions in our work, so that our approach can be applied in broader computing environments. Finally, our framework offers several system parameters that could affect the trade-off between computation cost, convergence speed, and accuracy. Hence, it will be our future direction to conduct more quantitative and qualitative studies on the impact of these settings for meeting the desired expectation of cost and accuracy under different data characteristics and computing environments.

REFERENCES

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *USENIX symposium on operating systems design and implementation* (2016), pp. 265–283.

[2] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (2016), pp. 308–318.

[3] Antoniou, A., Storkey, A., and Edwards, H. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).

[4] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, H. B., et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* (2019).

[5] Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., Dickie, D. A., Hernández, M. V., Wardlaw, J., and Rueckert, D. GAN augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863* (2018).

[6] Cheng, Y., Wang, D., Zhou, P., and Zhang, T. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* (2017).

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

[8] Courtiol, P., Maussion, C., Moarii, M., Pronier, E., Pilcer, S., Sefta, M., Manceron, P., Toldo, S., Zaslavskiy, M., Le Stang, N., et al. Deep learning-based classification of mesothelioma improves prediction of patient outcome. *Nature medicine 25*, 10 (2019), 1519–1525.

[9] Darlow, L. N., Crowley, E. J., Antoniou, A., and Storkey, A. J. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505* (2018).

[10] Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine 29*, 6 (2012), 141–142.

[11] Diao, E., Ding, J., and Tarokh, V. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264* (2020).

[12] Duan, M., Liu, D., Chen, X., Liu, R., Tan, Y., and Liang, L. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems 32*, 1 (2021), 59–71.

[13] Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference* (2006), Springer, pp. 265–284.

[14] Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci. 9*, 3-4 (2014), 211–407.

[15] Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing 321* (2018), 321–331.

[16] Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. Synthetic data augmentation using gan for improved liver lesion classification. In *IEEE international symposium on biomedical imaging* (2018), IEEE, pp. 289–293.

[17] Ghosh, A., Kulharia, V., Namboodiri, V. P., Torr, P. H., and Dokania, P. K. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 8513–8521.

[18] Giger, M. L. Machine learning in medical imaging. *Journal of the American College of Radiology 15*, 3 (2018), 512–520.

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

[20] Guan, Q., Chen, Y., Wei, Z., Heidari, A. A., Hu, H., Yang, X.-H., Zheng, J., Zhou, Q., Chen, H., and Chen, F. Medical image augmentation for lesion detection using a texture-constrained multichannel progressive gan. *Computers in Biology and Medicine 145* (2022), 105444.

[21] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028* (2017).

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

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

[24] Hoang, Q., Nguyen, T. D., Le, T., and Phung, D. MGAN: Training generative adversarial nets with multiple generators. In *International conference on learning representations* (2018).

[25] Holohan, N., Braghin, S., Mac Aonghusa, P., and Levacher, K. Diffprivlib: the ibm differential privacy library. *arXiv preprint arXiv:1907.02444* (2019).

[26] Huang, Y., Chu, L., Zhou, Z., Wang, L., Liu, J., Pei, J., and Zhang, Y. Personalized cross-silo federated learning on non-iid data. In *AAAI* (2021), pp. 7865–7873.

[27] Jeong, E., Oh, S., Kim, H., Park, J., Bennis, M., and Kim, S.-L. Communication-efficient On-device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data. *arXiv preprint arXiv:1811.11479* (2018).

[28] Jin, P. H., Yuan, Q., Iandola, F., and Keutzer, K. How to scale distributed deep learning? *arXiv preprint arXiv:1611.04581* (2016).

[29] Kang, J., Xiong, Z., Niyato, D., Zou, Y., Zhang, Y., and Guizani, M. Reliable federated learning for mobile networks. *IEEE Wireless Communications 27*, 2 (2020), 72–80.

[30] Kopparapu, K., and Lin, E. Fedfmc: Sequential Efficient Federated Learning on Non-IID Data. *arXiv preprint arXiv:2006.10937* (2020).

[31] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto* (2009).

[32] Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 10713–10722.

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

[34] Li, W., Zhong, X., Shao, H., Cai, B., and Yang, X. Multi-mode data augmentation and fault diagnosis of rotating machinery using modified acgan designed with new framework. *Advanced Engineering Informatics 52* (2022), 101552.

[35] Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the Convergence of FedAvg on Non-IID Data. *arXiv preprint arXiv:1907.02189* (2019).

[36] Li, Z., Shao, J., Mao, Y., Wang, J. H., and Zhang, J. Federated learning with gan-based data synthesis for non-iid clients. *arXiv preprint arXiv:2206.05507* (2022).

[37] Liu, Y., Kang, Y., Xing, C., Chen, T., and Yang, Q. A secure federated transfer learning framework. *IEEE Intelligent Systems 35*, 4 (2020), 70–82.

[38] Maqueda, A. I., Loquercio, A., Gallego, G., García, N., and Scaramuzza, D. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018).

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

[40] Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., and Zomaya, A. Y. Federated learning for covid-19 detection with generative adversarial networks in edge cloud computing. *IEEE Internet of Things Journal* (2021).

[41] Nilsson, A., Smith, S., Ulm, G., Gustavsson, E., and Jirstrand, M. A performance evaluation of federated learning algorithms. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning* (2018), pp. 1–8.

[42] Park, Y., Han, D.-J., Kim, D.-Y., Seo, J., and Moon, J. Few-round learning for federated learning. *Advances in Neural Information Processing Systems 34* (2021), 28612–28622.

[43] Rasouli, M., Sun, T., and Rajagopal, R. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228* (2020).

[44] Tanaka, F. H. K. d. S., and Aranha, C. Data augmentation using GANs. *arXiv preprint arXiv:1904.09135* (2019).

[45] Wang, H., Kaplan, Z., Niu, D., and Li, B. Optimizing Federated Learning on Non-IID Data with Reinforcement

Learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications* (2020), IEEE, pp. 1698–1707.

[46] Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q., and Poor, H. V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security 15* (2020), 3454–3469.

[47] Wen, H., Wu, Y., Yang, C., Duan, H., and Yu, S. A unified federated learning framework for wireless communications: Towards privacy, efficiency, and security. In *IEEE Conference on Computer Communications Workshops* (2020), pp. 653–658.

[48] Xin, B., Geng, Y., Hu, T., Chen, S., Yang, W., Wang, S., and Huang, L. Federated synthetic data generation with differential privacy. *Neurocomputing 468* (2022), 1–10.

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

[50] Xu, A., Li, W., Guo, P., Yang, D., Roth, H. R., Hatamizadeh, A., Zhao, C., Xu, D., Huang, H., and Xu, Z. Closing the generalization gap of cross-silo federated medical image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 20866–20875.

[51] Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., and Beaufays, F. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).

[52] Yoon, T., Shin, S., Hwang, S. J., and Yang, E. Fedmix: Approximation of mixup under mean augmented federated learning. *arXiv preprint arXiv:2107.00233* (2021).

[53] Yoshida, N., Nishio, T., Morikura, M., Yamamoto, K., and Yonetani, R. Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)* (2020), IEEE, pp. 1–7.

[54] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[55] Zhang, L., Shen, B., Barnawi, A., Xi, S., Kumar, N., and Wu, Y. Feddpgan: federated differentially private generative adversarial networks framework for the detection of covid-19 pneumonia. *Information Systems Frontiers 23*, 6 (2021), 1403–1415.

[56] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated Learning with Non-IID Data. *arXiv preprint arXiv:1806.00582* (2018).

[57] Zhou, Z., Chen, K., Li, X., Zhang, S., Wu, Y., Zhou, Y., Meng, K., Sun, C., He, Q., Fan, W., et al. Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. *Nature Electronics 3*, 9 (2020), 571–578.

[58] Zhuang, P., Schwing, A. G., and Koyejo, O. Fmri data augmentation via synthesis. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)* (2019), IEEE, pp. 1783–1787.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009