Enhancing Privacy-Preserving of Heterogeneous Federated Learning Algorithms Using Data-Free Knowledge Distillation

mohammadreza najafi *

Posted Date: 30 October 2024

doi: 10.20944/preprints202410.2389.v1

Keywords: Federated learning; knowledge distillation; Non-Identical and Independent Distributions; data reconstruction attacks; Dual Decomposition Optimization

 doi:10.20944/preprints202410.2389.v1

Article

Enhancing Privacy-Preserving of Heterogeneous Federated Learning Algorithms Using Data-Free Knowledge Distillation

Mohammadreza Najafi

Chosun University, Republic of Korea; mohamadrezanj76@gmail.com

Abstract: Federated learning (FL) is a decentralized machine learning paradigm that allows multiple local clients to collaboratively train a global model by sharing their model parameters instead of private data, thereby mitigating privacy leakage. However, recent studies have shown that gradient-based data reconstruction attacks (DRA) can still expose private information by exploiting model parameters from local clients. Existing privacy-preserving FL strategies provide some defense against these attacks but at the cost of significantly reduced model accuracy. Moreover, the issue of client heterogeneity further exacerbates these FL methods, resulting in drifted global models, slower convergence, and decreased performance. This study aims to address the two main challenges of FL: data heterogeneity, particularly in Non-Identical and Independent Distributions (Non-IID) clients, and client privacy through DRA. By leveraging the Lagrange duality approach and employing a generator model to facilitate knowledge distillation (KD) between clients, thereby enhancing local model performance, this method aims to concurrently address the primary challenges encountered by FL in real-world applications.

Keywords: federated learning; knowledge distillation; non-Identical and independent distributions; data reconstruction attacks; dual decomposition optimization

1. Introduction

With the advancement of wireless communication networks enabling the Internet of Things (IoT), edge clients are now equipped with increasingly sophisticated sensing, communication, computing, and analysis capabilities [1]. To address data-driven learning tasks arising from various real-world problems, artificial intelligence (AI), particularly deep learning and deep neural networks [2], has been widely adopted for feature extraction and model building to enhance decision-making. This approach is expected to benefit numerous application fields, including smart healthcare [3], smart manufacturing [4], smart cities [5], and smart agriculture [6]. In the traditional centralized deep learning framework, large volumes of raw data collected from edge clients are directly transmitted to a central server for the development of comprehensive deep learning models. However, this data often contains sensitive information (e.g., healthcare records, private photos), posing significant privacy risks. Consequently, ensuring data security and privacy is one of the most critical concerns in this context.

Federated Learning (FL) [7] is an approach to distributed machine learning that enhances privacy by transmitting gradients or model parameters rather than raw data. This framework exchanges parameters between the server and each client, enabling the server to learn from the clients' private data without directly accessing it. According to the Data Processing Inequality [8], communicating via parameters— which are deterministic mappings of local data—reduces the risk of compromising data privacy compared to sharing raw data. However, recent research has demonstrated that even parameter-based communication in FL can still lead to privacy breaches. For instance, multiple studies have shown that adversaries can perform membership inference attacks (MIA) using the transmitted model parameters [9–11], thereby compromising data privacy and anonymity. Additionally, attackers can fully reconstruct training data through data reconstruction attacks (DRA) [12–16], posing significant privacy threats to FL systems. Several innovative privacy-preserving techniques have been proposed to mitigate these risks. For example, [17] introduces a decentralized FL scheme that combines quality-based aggregation with an extended dynamic contribution broadcast encryption (DConBE) and local

 doi:10.20944/preprints202410.2389.v1

differential privacy. [18] proposes an adaptive FL framework that integrates adaptive gradient descent with differential privacy mechanisms. To address the dual challenges of ensuring provable privacy guarantees while minimizing communication and computational overheads in FL, especially for linear regression models, [19] suggests a differential private FL scheme. Lastly, [20] presents the Noising before Model Aggregation FL framework, which enhances differential privacy by adding artificial noise to parameters at the client's side before aggregation.

In addition to privacy concerns, a significant challenge in FL is data heterogeneity, where clients' data exhibit Non-Identical and Independent Distributions (Non-IID). Non-IID data refers to substantial variations across clients in terms of statistical properties, such as data distributions, feature representations, and class imbalances. This heterogeneity complicates model training, as models trained on Non-IID data may struggle to generalize effectively across different clients or accurately represent the overall patterns within the entire dataset. Conventional FL algorithms, such as FedAvg [21], have been observed to result in divergent local models and a considerable loss of global knowledge under Non-IID conditions, leading to decreased performance and slower convergence [22,23]. To address this issue, the concept of Knowledge Distillation (KD), as introduced by [24], offers a potential solution by transferring knowledge from teacher (client) models to a student (global) model. For instance, FedDF [25] employs ensemble distillation for model fusion, where the global model is trained using the averaged logits from local models. FedAUX [26] enhances this approach by determining model initialization for local models and assigning weights to local model logits using (ϵ, δ)-differential private certainty scoring. FedBE [27] takes a Bayesian approach, generating multiple global models from local models and then merging them into a single global model through ensemble KD. FedFTG, as introduced in [28], addresses the Non-IID challenge by training a GAN generator as a data generation model and fine-tuning the global model with the data produced by the GAN. Similarly, FedGen [29] trains a lightweight generator to mitigate the impact of Non-IID data on the global model's performance.

Deploying differential privacy mechanisms for preserving privacy while simultaneously utilizing KD methods to address heterogeneity is a challenging task. This difficulty arises because distilling knowledge from noisy clients is not possible, and basically, privacy and model performance are inversely proportional, meaning that improving privacy typically comes at the cost of model performance, such as accuracy. Consequently, centralized learning often outperforms FL applications in terms of overall performance due to the added privacy constraints in FL. In this work, we propose a novel method, FedSGAN, to tackle the Non-IID challenge in heterogeneous systems while maintaining the privacy of clients against DRA concurrently. FedSGAN employs a dual decomposition optimization approach to tackle the privacy challenge and leverages a Generative Adversarial Network (GAN) model to distill knowledge between clients by generating pseudo data. Consequently, FedSGAN effectively addresses the Non-IID challenge while concurrently preserving client privacy. The primary contributions of this work are outlined as follows:

- We present an innovative method for enhancing client model performance through data-free knowledge distillation.
- We modified GAN training procedure to generate pseudo data, effectively facilitating knowledge transfer between clients to tackle Non-IID challenges.
- We utilize dual decomposition optimization technique to protect clients' private data against DRA.
- FedSGAN simultaneously enhances client model performance and preserves privacy.

2. Preliminaries

In this paper, our methodology tackles Non-IID challenges by distilling knowledge between clients and employing dual decomposition to preserve client parameter privacy against DRA. This section elucidates the characteristics of the methods utilized in the development of the FedSGAN approach.

 doi:10.20944/preprints202410.2389.v1

2.1. Dual Decomposition Optimization

Dual decomposition optimization, as proposed in [30], is a privacy-preserving optimization technique designed to address multi-constraint problems, such as those encountered in FL. In FL, the set of active clients is denoted as A where $|A| = N$, and each client,$I \in A$, has access to its local private dataset, D_i. The goal for each client is to determine the optimal weight vector $\theta_i \in \mathcal{R}^m$, solving the following optimization problem:

$$\min_{\theta} \sum_{i=1}^{N} \mathcal{L}_{X \sim D_i}(\sigma(C(\theta_i, X))). \tag{1}$$

In this context, C denotes the classifier, σ represents the softmax function, and $\mathcal{L}$ refers to the loss function employed for computing the error of the model. Here, θ_i represents the prediction score for (X, Y), where X is the input data and Y is its ground truth label, with $(X, Y) \in D_i$. Additionally, N signifies the total number of clients. For utilizing the Dual Decomposition, the Equation (1) is reformulated into an equivalent constrained optimization problem that mathematically shares a common optimal value which each client aims to solve the following problem:

$$\begin{aligned} &\min_{\theta} \sum_{i=1}^{N} \mathcal{L}_{X \sim D_i}(\sigma(C(\theta_i, X))).,\\ &\text{subject to } \theta_i = \theta_j, \forall i, j \in A \end{aligned} \tag{2}$$

In Equation (2), each client i holds a copy of the weight vector θ_i, and all these copies are required to agree with the mentioned constraint. However, it can be noticed this constraint in 2 can be replaced with pairs of double-sided inequality constraints. Therefore, the Equation (2) can be rewritten as a standard constrained distributed optimization problem within equality constraints as follows:

$$\begin{aligned} &\min_{\theta} \sum_{i=1}^{N} \mathcal{L}_{X \sim D_i}(\sigma(C(\theta_i, X))).,\\ &\text{subject to } \sum_{i=1}^{N} e_i(\theta_i) \leq 0, \end{aligned} \tag{3}$$

where $e_i(\cdot) \in \mathcal{R}^m \rightarrow \mathcal{R}^{2m(N-1)}$ can be designed in infinitely possible ways to represent the equality constraint in Equation (2). For instance, Algorithm 3 provides a detailed explanation for calculating the output vector of $e_i(\theta_i)$.

The $e_i(\cdot)$ function represents one of the infinitely many possible realizations that can satisfy the constraint in Equation (3). In Algorithm 1 , $[\mathbf{0}]$ denotes the zero vector of size m. Considering Equation (3) and the Lagrange multipliers vector with zero initial value, denoted as λ_i for client i, along with the v_i vector calculated as $v_i = m \cdot e_i(\theta_i)$, the training in client i process can be performed as follows:

1. Clients randomly initialize $\theta_i \in \mathcal{R}^m$.
2. Clients initialise $\lambda_i \in \mathcal{R}^{2m(N-1)}$ with zero value and $v_i \in \mathcal{R}^{2m(N-1)}$ with $m \cdot e_i(theta_i)$ value and send them to the server.
3. The $\overline{\lambda}$ and $\overline{v}$ are calculated as Equations (4)-(4), transferred to the clients from server in each communication round.

$$\overline{\lambda} \leftarrow \frac{1}{N} \cdot \left(\sum_{i=1}^{N} \lambda_i \right) \tag{4}$$

$$\overline{v} \leftarrow \frac{1}{N} \cdot \left(\sum_{i=1}^{N} v_i \right). \tag{5}$$

 doi:10.20944/preprints202410.2389.v1

4. Client Calculates the loss value :

$$f_{c_i} \leftarrow \mathcal{L}_{X \sim D_i}(\sigma(C(\theta_i, X))) \tag{6}$$

5. Clients fine-tune its gradient of loss value as below:

$$u_i \leftarrow \nabla f_{c_i} + \nabla e_i(\theta_i)^T \overline{\lambda}. \tag{7}$$

6. Clients update its weight with learning rate α_c as below:

$$\theta_{update} \leftarrow \theta_i - \alpha_c \cdot u_i. \tag{8}$$

7. The new λ_i and ν_i calculate with hyperparameter, σ, form Equations (9)–(11) , and transfer to the server.

$$q_i \leftarrow \overline{\nu} - \sigma.\alpha_c.\overline{\lambda}. \tag{9}$$

$$\lambda_i \leftarrow [\overline{\lambda} + \alpha_c.q_i]. \tag{10}$$

$$\nu_i \leftarrow \overline{\nu} + m(e_i(\theta_{update}) - e_i(\theta_i). \tag{11}$$

8. Repeat steps 3 to 7 till the end of the training phase.

Algorithm 1 Calculating the $e_i(\cdot)$.

Inputs: The client local model parameters θ_i, m is the size of θ_i, N is the total number of clients.
Output: Vector with size of $2m(N-1)$
Case 1: $i = 1$

$$\begin{cases} e_i[1] = \theta_i \\ e_i[2] = -\theta_i \\ e_i[j] = [\mathbf{0}], \quad \text{for } j = 3 \text{ to } 2(N-1) \end{cases}$$

Case 2: $i = N$

$$\begin{cases} e_i[j] = [\mathbf{0}], & \text{for } j = 0 \text{ to } 2(N-2) \\ e_i[2N-3] = -\theta_i \\ e_i[2N-2] = \theta_i \end{cases}$$

Case 3: $1 < i < N-1$

$$\begin{cases} e_i[j] = -\theta_i, & \text{if } j = 2i-3 \\ e_i[j] = \theta_i, & \text{if } j = 2i-2 \\ e_i[j] = \theta_i, & \text{if } j = 2i-1 \\ e_i[j] = -\theta_i, & \text{if } j = 2i \\ e_i[j] = [\mathbf{0}], & \text{otherwise} \end{cases} \begin{cases} \text{Let index} = 2(i-1) \\ e_i[j] = \begin{cases} -\theta_i, & \text{if } j = \text{index} \\ \theta_i, & \text{if } j = \text{index}+1 \text{ or } j = \text{index}+2 \\ -\theta_i, & \text{if } j = \text{index}+3 \\ [\mathbf{0}], & \text{otherwise} \end{cases} \end{cases}$$

2.1.1. *Assumption.1*

For all $i \in A$, we assume that the $e_i(\cdot)$ are not disclosed to the server. In other words, the server is unaware of the exact realization of $e_i(\cdot)$ for all $i \in A$.

2.2. GAN

Generative Adversarial Networks (GANs), introduced by [31], are a class of deep learning models designed to generate new data samples that resemble a given training dataset. GANs have gained

 doi:10.20944/preprints202410.2389.v1

significant attention in privacy-preserving data generation due to their ability to generate pseudo data that preserves the statistical properties of the original data without revealing sensitive information. This pseudo data can then be used for the training phase without compromising privacy. A GAN consists of two neural networks, namely a Generator $\mathcal{G}$ with a set of model parameters denoted as ω and a Discriminator $\mathcal{D}$, which are trained simultaneously. Through an adversarial process, the generator takes as input a random noise vector z sampled from a prior distribution, which typically is a uniform or Gaussian distribution. The generator maps this noise vector to the data space, aiming to produce pseudo data indistinguishable from real data. As illustrated in Figure 1, the discriminator receives either a real data sample (X, Y) or a generated sample X' as input and outputs a probability representing the likelihood that the input is real rather than generated.

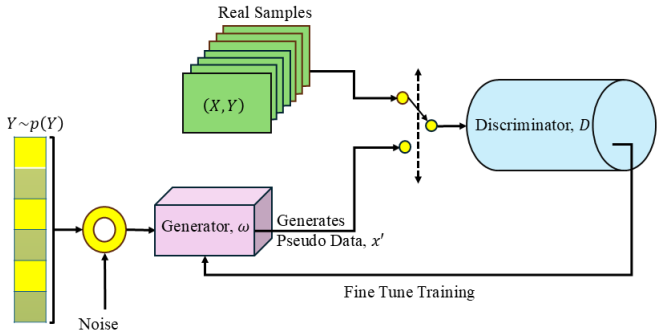


Figure 1. The GAN model training procedure architecture.

The GAN framework is mathematically formulated as a minimax optimization problem, where the objective is to find an equilibrium between the generator and the discriminator. The objective function for GANs can be expressed as Equation (12) where $\sim \mathcal{N}(0,1)$ and $Y \sim p(Y)$:

$$\min_{\omega} \max_{\mathcal{D}} \mathbb{E}_x[log\ \mathcal{D}(X)] + \mathbb{E}_{z,Y}[1 - log\ \mathcal{D}(\mathcal{G}(\omega, z, Y))] \tag{12}$$

The generator aims to minimize the loss function by generating samples that maximize the discriminator's error, while the discriminator tries to maximize the loss function by correctly classifying real and generated samples.

3. Methodology

In this section, we delineate the FedSGAN training phase with Non-IID clients. Figure 2 represents that each client initializes two models: a local model for the FL objective task, denoted as $\theta \in \mathcal{R}^m$, and a generator model, denoted as $\mathcal{G}$ with model parameters set ω, aimed at enhancing local model performance under Non-IID conditions.

During each communication round, clients employ a local optimizer loss function, $\mathcal{L}$, to train their local model on their private dataset, D_i. Furthermore, the local model at each client is refined using pseudo data generated by $\mathcal{G}$. Unlike conventional FL methods, where clients transfer their local model gradients or parameters, θ_i, for aggregation, FedSGAN clients employ the modified dual decomposition optimization method, as explained in the Section 2.1. They transfer only λ_i and v_i vectors, and as a result, FedSGAN protects clients from sharing their local model parameters, thereby defending them against DRA. The FedSGAN client-side and server-side processes are further explained in the subsequent subsections.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 October 2024 doi:10.20944/preprints202410.2389.v1

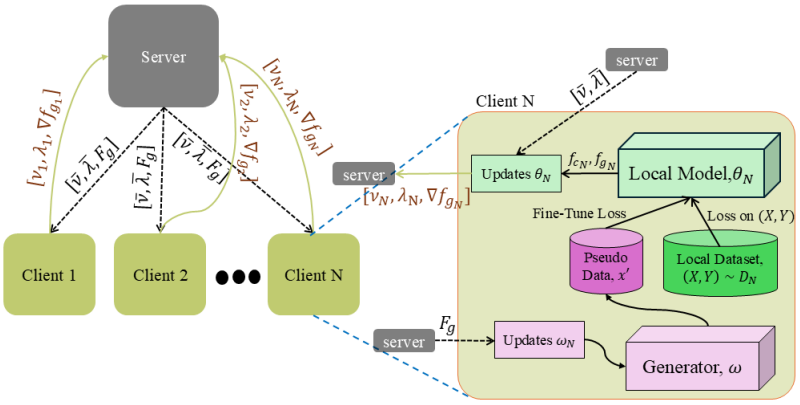


Figure 2. The proposed FedSGAN model architecture.

3.1. Client Side

Clients in FedSGAN utilize the decomposition optimization method described in Section 2.1 for aggregation and to keep their data privacy against DRA. But decomposition optimization causes the performance loss as it shown in Table **??** which compare the decomposition FL mehtod with FedAVG and FedProx, the FedAVG and FedProx have better accuracy on MNIST classification and also decomposition FL method could not address the Non-IID challenges in Heterogeneous systems. Therefore, The proposed FedSGAN utilizes the $\mathcal{G}$ model with GAN architecture to distillate knowledge from other clients and improve the performance. During the training phase, $\mathcal{G}$ gets noise z, and y grand true label for generating pseudo data, x', as described in Equation (13).

$$x' = \mathcal{G}(z, Y; \omega_i). \tag{13}$$

The ω_i is the $\mathcal{G}$ model parameter for client i, z is Gaussian noise $z \sim \mathcal{N}(0,1)$ and $Y \sim p(Y)$. The number of grand true data, $(X,Y) \in D_i$, with label Y label is denoted as $n^i{}_Y$ and $p(Y)$ for generating pseud data is calculated from Equation (14) while it is objective to each client and is not disclosed to the server. In Equation (14), for each class, Rn_Y represents a random number for each label Y and is significantly greater than n_Y.

$$p(Y) \propto \sum_{j=1}^{n^i{}_Y} \mathbb{E}_{(X_j,Y_j)\sim\mathcal{D}_i}\left[1_{Y_j=Y}\right] = Rn_Y. \tag{14}$$

The pseudo data x' generated by $\mathcal{G}$ is employed to fine-tune the client loss function, f_{c_i}. By evaluating the loss of the local model, θ_i, on this pseudo data, F_{g_i}, as detailed in Equation (15), the performance loss induced by decomposition optimization is mitigated. Moreover, the $p(Y)$ distribution compels ω_i to generate pseudo data that guide the client towards a more uniform data distribution for training. This approach mitigates the drift during aggregation caused by the Non-IID challenge, as demonstrated by the experimental results in the subsequent section.

$$f_{c_i} \leftarrow \mathcal{L}_{X\sim D_i}(\sigma(C(\theta_i, X)))) + \underbrace{\mathcal{L}(\sigma(C(\theta_i, x')}_{f_{g_i}}). \tag{15}$$

After calculating the f_{c_i}, the u_i is computed from Equation (7) and then clients updates their model parameters, θ_i, with α_c learning rate from Equation (8). In the end, refer to the Equations (9)–(11) clients update the λ_i and ν_i for transferring them to the server for aggregation.

FedSGAN addresses the privacy performance loss and Non-IID challenge by using ω_i during the training phase, but training the ω_i still remains a critical task. The $\mathcal{G}$ can not use client's local private dataset, D_i, for training ω_i because in the Non-IID situation training ω_i on the clients private dataset caused to generated pseudo data in a similar distribution as client has and it can not help us to

 doi:10.20944/preprints202410.2389.v1

address the Non-IID. Therefore, for training the ω_i, FedSGAN modified the formal GAN architecture as illustrated in Figure 3. In taring phase of ω_i, FedSGAN remove Discriminator $\mathcal{D}$, Figure 3, and utilize f_{g_i} loss value as it describe in Equation (15). F_{g_i} calculates the loss of the local model on the pseudo data generated by ω_i. Clients transfer the gradient of f_{g_i} to the server for aggregation and receive the F_g from the server for updating the ω_i with α_g learning rates as described in Equation (16). The method for calculating F_g from ∇f_{g_i} is detailed in Equation (17) in the server side subsection.

$$\omega_i \leftarrow \omega_i - \alpha_g.F_g. \tag{16}$$

As illustrated in Figure 2, clients in FedSGAN, in addition to transferring the λ_i and v_i to a server in the training phase, also transfer the ∇f_{g_i} for training their ω_i. The step-by-step FedSGAN training phase on the client side has been shown in Algorithm 2.

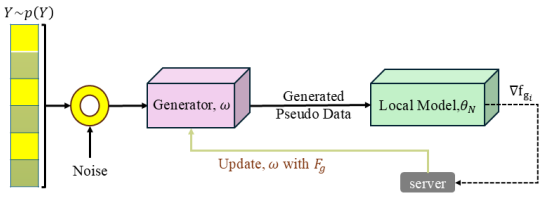


Figure 3. The modified GAN training procedure.

Algorithm 2 FedSGAN, Client side.

Inputs: Average Lagrange multiplier vector: $\overline{v}$, Average Lagrange dual vector: $\overline{\lambda}$,Average gradient loss function of generator model: F_g, Local model learning rate: α_c, Generator model learning rate: α_g, σ.

repeat
  for all client $i \in N$ in parallel **do**
    Client i^{th} receive $\overline{v}$, $\overline{\lambda}$ and F_g from server.
    $\omega_i \leftarrow$ from Equation (16).
    $(z, y) \leftarrow (z \sim \mathcal{N}(0,1), Y \sim p(Y))$.
    $x' \leftarrow$ from Equation (13).
    $f_{g_i} \leftarrow$ from Equation (15).
    $f_{c_i} \leftarrow$ from Equation (15).
    $u_i \leftarrow \nabla f_{c_i} + \nabla e_i(\theta_i)^T \overline{\lambda}$
    $q_i \leftarrow$ from Equation (9)
    $\theta_{update} \leftarrow \theta_i - \alpha_c.u_i$
    $\lambda_i \leftarrow$ from Equation (10).
    $v_i \leftarrow$ from Equation (11).
    $\theta_i \leftarrow \theta_{update}$.
    client sends v_i, λ_i and ∇f_{g_i} to the server.
  end for
until training stop

3.1.1. *Remark.1*

The *Assumption. 1* Section 2.1.1 ensures that when executing Algorithm 2, the server is unable to reconstruct θ_i, thereby preventing access to information about the clients' private local dataset D_i. It is important to note that the ∇f_{g_i} transferred between clients and the server represents the gradient of the loss function of the local model θ_i on the pseudo data x' generated from Equation (13). Consequently, state-of-the-art privacy attacks can only reconstruct information about x' (the pseudo data) and not the clients' private data. Additionally, due to $R_{nY} >> n^i{}_Y$ in Equation (14), the distribution of the private data D_i's ground truth labels cannot be reconstructed from the pseudo data's ground truth label distribution. Therefore, Algorithm 2 strengthens the privacy of FedSGAN by safeguarding the clients' private data.

 doi:10.20944/preprints202410.2389.v1

3.2. Server Side

As previously mentioned, unlike conventional FL methods, FedSGAN protects client DRA by avoiding the transfer of the client's local model parameters, θ_i, to the server. Consequently, there is no global model at the server side for training or aggregation. Additionally, the server lacks any information about the clients' data distributions and the $e_i(\cdot)$ function used for decomposition optimization, as explained in *Remark.1* 3.1.1, thereby ensuring the privacy of FedSGAN's clients. Server received the λ_i, ν_i and ∇f_{g_i} from clients and process to calculate the average of these parameters as formulated in Equations (4), (5), (17) and send back the $\overline{\lambda}$,$\overline{\nu}$ and F_g to the clients for updates their models. The server-side FedSGAN algorithm has been represented in Algorithm 3.

$$F_g \leftarrow \frac{1}{N}.\left(\sum_{i=1}^{N} \nabla f_{g_i}\right). \tag{17}$$

Algorithm 3 FedSGAN, Server side.

Inputs: clients average Lagrange multiplier vector: $(\nu_i)_{i=1}^N$, clients average Lagrange dual vector: $(\lambda_i)_{i=1}^N$ and clients average gradient loss function of generator model:$\nabla(f_{g_i})_{i=1}^N$, Number of communication round: T.
for t=1,...,T **do**
Collect ν_i, λ_i and f_{g_i} from clients.
$\overline{\lambda} \leftarrow$ from Equation (4)
$\overline{\nu} \leftarrow$ from Equation (5)
$F_g \leftarrow$ from Equation (17)
sends $\overline{\lambda}$, $\overline{\nu}$ and F_g to the clients.
end for

4. Experiments

This section will elucidate the implementation aspects of FedSGAN and provide a comparative analysis of its performance against state-of-the-art methods for addressing Non-IID challenges. Additionally, a comparison of its performance in situations involving DRA with state-of-the-art methods will be conducted in Section 4.2.

4.1. Implementation Details

In this section, we conduct a comparative analysis to evaluate the performance of FedSGAN against other notable related works. Detailed implementation specifics and comprehensive experimental results are discussed in the subsequent subsections. In addition, the Python source code of FedSGAN is accessible via the following link: https://anonymous.4open.science/r/FedSGAN-2816/README.md.

4.1.1. Reference Methods

In this study, we compare FedSGAN with several significant related works, including FedAvg [7], FedProx [32], SCAFFOLD [33], FedFTG [28], FedGen [34], FedRand [35], FedKD [36], and f-differential [37]. Table 1 outlines the features of these state-of-the-art methods and demonstrates that FedGen and FedFTG utilize KD to address the Non-IID challenge. FedFTG employs a classification model for the client's local model $\sigma(C(\theta_i))$, utilizing ResNet18 [38], and uses a GAN architecture [31] for its $\mathcal{G}$ model. Similarly, FedGen also employs ResNet18 as the local model and utilizes a simple Autoencoder model for its $\mathcal{G}$ architecture. Notably, FedAvg, FedProx, SCAFFOLD, FedGen, and FedFTG do not mention any techniques in their methodologies to ensure client privacy against DRA. Meanwhile, the FedKD local model employs a Multi-Layer Perceptron (MLP) architecture and utilizes the KD for performance improvement but does not employ any generator model.

 doi:10.20944/preprints202410.2389.v1

Table 1. The state-of-the-art methods main features.

Model Name	Local Model	KD	Generator	Privacy against DRA
FedAvg	CNN	×	-	×
FedProx	CNN	×	-	×
SCAFFOLD	CNN	×	-	×
FedGen	ResNet18	✓	Autoencoder	×
FedFTG	ResNet18	✓	GAN	×
FedRand	CNN	×	-	✓
FedKD	MLP	✓	No generator	✓
f-differential	CNN	×	-	✓
FedSGAN	ResNet18	✓	GAN	✓

4.1.2. FedSGAN Networks Architecture

FedSGAN, similar to FedFTG and FedGen, has ResNet18 architecture for clients local model $\sigma(C(\theta_i))$. Additionally, The GAN proposed in [39] is employed as the $\mathcal{G}$ model.

4.1.3. Datasets

The effectiveness of FedSGAN is evaluated using the CIFAR-10 [40] and MNIST [41] datasets, which feature heterogeneous dataset partitions. To emulate Non-IID data distribution among clients, a practice adopted by previous studies [28,42–44], we utilize the Dirichlet distribution Dir(ψ) on label ratios. Here, a smaller value of ψ signifies increased data heterogeneity. For FedSGAN implementation, and $\psi = 0.3, 0.6$.

4.1.4. Differential Privacy

We also compare FedSGAN with reference methods under differential privacy conditions as proposed by [45]. This comparison involves deploying Gaussian noise with varying privacy budgets (ϵ) to the clients' weight models to protect their privacy against DRA.

4.1.5. Hyperparameters

The number of clients $N = 100$, number of local training epochs 50, communication rounds $T = 1000$, for local training, the batch size is 50. Moreover the α_c and α_g are both initialized to 0.1 and σ is 0.3 . The dimension of X and x' are $32 \times 32 \times 3$ for CIFAR-10 and $28 \times 28 \times 1$ for MNIST.

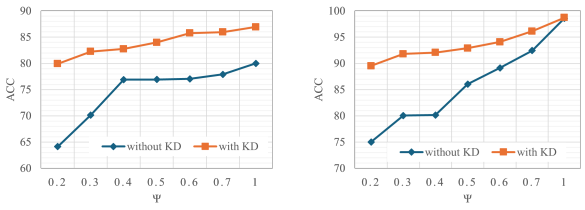
4.2. Performance Comparison

In this section, we first compare the proposed FedSGAN without the $\mathcal{G}$ model, utilizing only decomposition optimization, to the full implementation containing $\mathcal{G}$. This analysis highlights the significance of the $\mathcal{G}$ model and utilizing KD method, particularly in the context of Non-IID clients. Subsequently, we conduct a comparative analysis of FedSGAN's accuracy against existing state-of-the-art methods in three scenarios: *I*. with Non-IID clients and without privacy consideration, *II*. with implemented privacy measures and all clients are IID, and *III*. with Non-IID clients while considering private training by utilizing the DP method.

4.2.1. Comparison of FedSGAN Model Without Utilizing KD Method

We compare the accuracy of FedSGAN with and without using the KD method. The experimental results of this comparison across different values of ψ are illustrated in Figure 4 for the MNIST and CIFAR-10 datasets. As shown in Figure 4, the accuracy of FedSGAN with the KD method significantly surpasses without KD, particularly in the presence of Non-IID clients. This highlights the importance of the KD method in enhancing FedSGAN's performance.

 doi:10.20944/preprints202410.2389.v1



(a) CIFAR-10. **(b)** MNIST.

Figure 4. Comparison of FedSGAN With and Without KD on MNIST and CIFAR-10.

4.2.2. *I.Performance Without Privacy Consideration*

The overall accuracy comparison between FedSGAN and other state-of-the-art methods is presented in Table 2 for the CIFAR-10 and MNIST datasets across two various ψ values. All experiments were conducted using three random seeds. As shown in Table 2, FedSGAN surpasses most of the methods for $\psi = 0.3$ and $\psi = 0.6$, except for FedFTG. FedFTG demonstrates strong performance in Non-IID client FL applications. However, as illustrated in Figure 5 which presents a comparison of test accuracy across various ψ, the accuracy of FedSGAN is comparable to FedFTG when the ψ value is close to 1, indicating clients are IID. For ψ values less than 0.3, SCAFFOLD's accuracy exceeds that of FedSGAN, likely due to FedSGAN's approach of not transferring clients' local model parameters θ_i Unlike other methods, and utilizing decomposition optimization for privacy considerations. Nonetheless, FedSGAN's accuracy remains competitive with state-of-the-art methods, particularly for non-IID clients in CIFAR-10 classification applications, and it shows minimal differences from FedFTG.

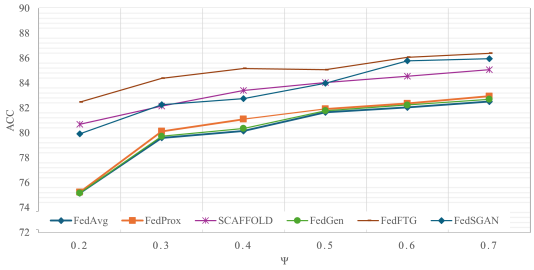


Figure 5. Compare the accuracy of state-of-the-art Methods on different ψ (data heterogeneity) on CIFAR-10.

Table 2. The Average Test Accuracy (%) of different FL methods on CIFAR-10 and MNIST.

Model Name	CIFAR-10		MNIST	
	$\psi = 0.6$	$\psi = 0.3$	$\psi = 0.6$	$\psi = 0.3$
FedAvg	82.04	79.59	93.84	90.16
FedProx	82.36	80.12	93.83	90.10
SCAFFOLD	84.55	82.14	97.14	95.94
FedGen	82.23	79.72	95.52	93.03
FedSGAN	85.78	82.254	94.07	91.81
FedFTG	**86.06**	**84.38**	**98.91**	**97.01**

4.2.3. *II.Performance with Privacy Consideration*

The strength of FedSGAN lies in its ability to ensure client privacy against DRA while maintaining performance in heterogeneous systems, as it mentioned in *Remark.1*3.1.1. Differential privacy is a well-known method utilized to address the privacy challenges of DRA in FL applications. Table 3 compares various state-of-the-art FL methods that employ differential privacy techniques with FedSGAN, using different ϵ values on the MNIST dataset. As shown, for lower ϵ values, the accuracy of f-differential

 doi:10.20944/preprints202410.2389.v1

slightly surpasses FedSGAN by less than 1%. However, for higher ϵ values, FedSGAN outperforms all methods with significant difference and remains independent of ϵ values because it maintains privacy without relying on differential privacy techniques. Like FedSGAN, FedAKD employs the KD method to preserve accuracy, but its performance does not compare favorably to FedSGAN.

Table 3. Comparison of various FL methods on MNIST with IID clients ($\psi = 0$) considering privacy with different ϵ).

ϵ	FedRand	f-differential	FedAvg	FedAKD	FedSGAN
0	97.1	**98.74**	78.12	60.15	98.71
5	96.2	**98.72**	78.16	60.14	98.71
10	94.3	98.55	70.15	60.16	**98.71**
20	34.5	90.11	46.15	20.48	**98.71**

Table 4 compares state-of-the-art methods with accuracy levels similar to FedSGAN in MNIST classification, now applied to CIFAR-10 classification across varying ϵ values. This comparison underscores how FedSGAN performs relative to these leading methods in maintaining accuracy while addressing privacy concerns. Unlike in MNIST classification, where FedSGAN's accuracy is comparable, in CIFAR-10 classification, FedSGAN's accuracy is notably higher than that of other methods, demonstrating significant performance improvements at larger ϵ.

Table 4. Comparison of various FL methods on CIFAR-10 with IID clients ($\psi = 0$) considering privacy for different ϵ).

ϵ	FedRand	f-differential	FedSGAN
0	64.20	64.72	**86.06**
5	60.00	62.55	**86.06**
10	42.20	59.61	**86.06**
20	22.40	34.12	**86.06**

4.2.4. *III.Performance with Privacy Consideration and Non-IID Clients*

FedSGAN effectively preserves client privacy against DRA while simultaneously addressing the challenges posed by Non-IID clients. As discussed in previous subsections, its performance in independent scenarios, Non-IID environments, and privacy protection against DRA is comparable with state-of-the-art methods. We now compare FedSGAN's performance with other leading methods that either outperform or match FedSGAN in the previously discussed scenarios. Figure 6 illustrates its accuracy in scenarios where both privacy must be maintained and clients are Non-IID, with a ψ value of 0.5 for CIFAR-10 classification. Based on *Remark* 13.1.1, FedSGAN ensures privacy against DRA, whereas other methods utilize differential privacy with varying ϵ values for comparison. As shown in Figure 6, FedSGAN demonstrates a significant advantage over other methods at ϵ values greater than zero, specially in compare to FedFTG that has better performance than FedSGAN when $\psi = 0.5$. However, its performance under differential privacy is significantly reduced and less effective than FedSGAN, underscoring FedSGAN's strength for real-world federated learning applications.

 doi:10.20944/preprints202410.2389.v1

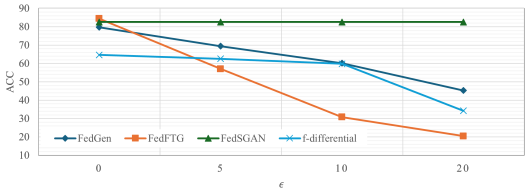


Figure 6. Compare the accuracy of state-of-the-art Methods on different ϵ while $\psi = 0.5$(Non-IID clients) on CIFAR-10.

5. Discussion

The primary limitation of FedSGAN lies in the parameters transferred between clients and servers. As detailed in Methodology section, FedSGAN clients transmit λ_i, ν_i, and ∇f_{g_i} instead of their model parameters. While the size of ∇f_{g_i} depends on the client's model architecture, the parameters λ_i and ν_i have a size of $\mathcal{R}^{2m(N-1)}$, which depends on the number of clients, N. In scenarios with a large number of clients and bandwidth constraints, FedSGAN becomes unsuitable due to the large data size transferred between clients and the server. To address this challenge, one potential solution involves optimizing the parameter communication process with techniques such as parameter compression, or quantization, which can significantly reduce the data volume without compromising model performance.

6. Conclusion

We have introduced a novel data-free knowledge distillation approach in FL that effectively balances accuracy and privacy preservation. By transmitting alternative parameters (λ_i, ν_i, and ∇f_{g_i}) instead of direct model parameters, FedSGAN maintains privacy against DRA while addressing the challenges posed by Non-IID client distributions. Leveraging a GAN model, FedSGAN efficiently transfers knowledge between clients, overcoming the discrepancies of Non-IID data. FedSGAN achieves over 50% better accuracy compared to the FedFTG method, which uses the same data-free knowledge distillation approach, when $\psi = 0.5$ and $\epsilon = 10$ in CIFAR-10 classification, demonstrating significant improvements in overall accuracy. Comprehensive evaluations across two benchmark datasets confirm the effectiveness of FedSGAN, positioning it as a promising solution for privacy preservation with Non-IID clients in FL applications.

Appendix A. Appendix

In this appendix, we provide detailed proofs for the optimization Equations (3) to (11) proposed in Section 2.1, which describe the Dual Decomposition Optimization method used in FedSGAN.

Appendix A.1. Optimization Problem Formulation

The optimization problem for FedSGAN is formulated as follows:

$$\begin{aligned} \min_{\theta} \sum_{i=1}^{N} \mathcal{L}_{X\sim D_i}(\sigma(C(\theta_i, X))), \\ \text{subject to } \theta_i = \theta_j, \forall i, j \in A \end{aligned} \tag{A1}$$

This can be reformulated into a set of inequalities:

$$\theta_i = \theta_j \rightarrow \begin{cases} \theta_i - \theta_j \leq 0 & \forall i, j \in A \\ -\theta_i + \theta_j \leq 0 & \forall i, j \in A \end{cases} \tag{A2}$$

By using Equation (A2) we could defined the $e_i(\cdot)$ function as explained in Equation (A3) which had describe in Algorithm1. Additionally, from Equation (A3) we impose the following constraint:

 doi:10.20944/preprints202410.2389.v1

$$\sum_{i=1}^{N} e_i(\theta_i) \leq 0,$$

Appendix A.2. Lagrangian Formulation

With relies on [30], the Lagrangian for the optimization problem 3 could formulate given by:

$$\mathbf{L}(\theta, \lambda) = \sum_{i=1}^{N} (\mathcal{L}_{X \sim D_i}(\sigma(C(\theta_i, X)))) + \frac{1}{N} \sum_{i=1}^{N} (e_i(\theta_i)\lambda_i)$$

where λ_i are the dual variables associated with the constraint $\sum_{i=1}^{N}(e_i(\theta_i)) \leq 0$.

Appendix A.3. Gradient Descent Update

The parameter update rule using gradient descent is:

$$\theta_{update} \leftarrow \theta_i - \alpha_c(\nabla \mathbf{Loss})$$

Expanding this for the specific Primal update, we have:

$$\theta_{update} \leftarrow \theta_i - \alpha_c(\underbrace{\nabla \mathcal{L}_{X \sim D_i}(\sigma(C(\theta_i, X))) + \nabla e_i(\theta_i)^T \underbrace{\frac{1}{N} \sum_{i=1}^{N} \lambda_i}_{\overline{\lambda}}}_{u_i})$$

where α_c is the learning rate, $\mathcal{L}_{X \sim D_i}$ is the loss function for the data distribution D_i.
Moreover Let θ^* denote the optimal solution of 3. We define the regret functions as follows:

$$Reg^{\mathcal{L}}(T) \triangleq \sum_{i=1}^{N} (\sum_{k=1}^{T} (\mathcal{L}(\sigma(C(\theta_i, X)))_k - \mathcal{L}(\sigma(C(\theta^*, X)))))$$

$$Reg^{e}(T) \triangleq |||[\sum_{i=1}^{N} \sum_{k=1}^{T} e_i(\theta_i)_k]|||$$

where T represents the number of communications. A successful FL algorithm must ensure that:

$$\lim_{T \to \infty} \frac{Reg^{\mathcal{L}}(T)}{T} = 0$$

$$\lim_{T \to \infty} \frac{Reg^{e}(T)}{T} = 0$$

The success criteria, as expressed through A.3, articulate the convergence of regrets over time, indicating the effectiveness and convergence properties of the FL algorithm. The following theorem guarantees the desired performance.

Appendix A.3.1. *Theorem.1*

Suppose for all clients $i \in A$, we assume that the respective gradients $\mathcal{L}(\cdot)$ and $e_i(\cdot)$ are convex and uniformly bounded. In other words, there exists a positive constant C such that $||\nabla \mathcal{L}_{X \sim D_i}(\sigma(C(\theta_i, X))) \leq C$ and $||\nabla e_i(\theta_i)|| \leq C$, where $||\nabla e_i(\theta_i)||$ represents a Jacobian matrix at θ_i. Consider that FedSGAN Algorithm 2 is executed by all clients with $\alpha = \sigma^{-1}T^{-\beta}$, where $\sigma = 2N(NC^2 + 1)$ and $\beta \in (0, 1)$.Then the limits introduced in A.3 tend to 0.

 doi:10.20944/preprints202410.2389.v1

Appendix A.3.2. *Proof:*

The aggregation process in the server during Algorithm3 closely resembles the scenario where clients communicate in a complete graph, assigning $\frac{1}{N}$ to their incoming edges. In this context, the graph's adjacency matrix is doubly stochastic and strongly connected. Consequently, noticing the proposed values for α_c and σ, the fulfillment of above assumption satisfies all the essential conditions outlined in *Theorem.1* and as a result, it holds concluding the proof.

$$\lim_{T\to\infty} \frac{Reg^{\mathcal{L}}(T)}{T} = 0$$

$$\lim_{T\to\infty} \frac{Reg^{e}(T)}{T} = 0$$

$$\underbrace{\begin{bmatrix} \theta_1 \\ -\theta_1 \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ \vdots \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \end{bmatrix}_{2m(N-1)}}_{e_1(\theta_1)} + \underbrace{\begin{bmatrix} -\theta_2 \\ \theta_2 \\ \theta_2 \\ -\theta_2 \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \end{bmatrix}_{2m(N-1)}}_{e_2(\theta_2)} + \underbrace{\begin{bmatrix} [\mathbf{0}] \\ [\mathbf{0}] \\ -\theta_3 \\ \theta_3 \\ \theta_3 \\ -\theta_3 \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \end{bmatrix}_{2m(N-1)}}_{e_3(\theta_3)} + \underbrace{\begin{bmatrix} [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ -\theta_4 \\ \theta_4 \\ \theta_4 \\ -\theta_4 \\ \vdots \\ [\mathbf{0}] \end{bmatrix}_{2m(N-1)}}_{e_4(\theta_4)} + \cdots + \underbrace{\begin{bmatrix} [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ \vdots \\ -\theta_{N-1} \\ \theta_{N-1} \\ \theta_{N-1} \\ -\theta_{N-1} \end{bmatrix}_{2m(N-1)}}_{e_{N-1}(\theta_{N-1})} + \underbrace{\begin{bmatrix} [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ [\mathbf{0}] \\ \vdots \\ -\theta_N \\ \theta_N \end{bmatrix}_{2m(N-1)}}_{e_N(\theta_N)} \leq 0 \tag{A3}$$

References

1. Nguyen, D.C.; Ding, M.; Pathirana. 6G Internet of Things: A comprehensive survey. *IEEE Internet of Things Journal* **2021**, *9*, 359–383.
2. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep learning*; MIT press, 2016.
3. Gupta, R.; Shukla, A.; Tanwar, S. BATS: A blockchain and AI-empowered drone-assisted telesurgery system towards 6G. *IEEE Transactions on Network Science and Engineering* **2020**, *8*, 2958–2967.
4. Lin, C.C.; Deng, D.J.; Chih, Y.L.; Chiu, H.T. Smart manufacturing scheduling with edge computing using multiclass deep Q network. *IEEE Transactions on Industrial Informatics* **2019**, *15*, 4276–4284.
5. Singh, S.K.; Jeong, Y.S.; Park, J.H. A deep learning-based IoT-oriented infrastructure for secure smart city. *Sustainable Cities and Society* **2020**, *60*, 102252.
6. Vimalajeewa, D.; Kulatunga. A service-based joint model used for distributed learning: Application for smart agriculture. *IEEE Transactions on Emerging Topics in Computing* **2021**, *10*, 838–854.
7. McMahan, B.; Moore. Communication-efficient learning of deep networks from decentralized data. Artificial intelligence and statistics. PMLR, 2017, pp. 1273–1282.
8. Cover, T.M. *Elements of information theory*; John Wiley & Sons, 1999.
9. Carlini, N.; Chien, S.; Nasr. Membership inference attacks from first principles. 2022 IEEE Symposium on Security and Privacy (SP). IEEE, 2022, pp. 1897–1914.
10. Choquette-Choo, C.A. Label-only membership inference attacks. International conference on machine learning. PMLR, 2021, pp. 1964–1974.
11. Long, Y.; Wang. A pragmatic approach to membership inferences on machine learning models. 2020 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2020, pp. 521–534.
12. Carlini, N.; Tramer. Extracting training data from large language models. 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 2633–2650.

 doi:10.20944/preprints202410.2389.v1

13. Chen, J.; Zhao, Y.; Li, Q.; Feng, X.; Xu, K. FedDef: defense against gradient leakage in federated learning-based network intrusion detection systems. *IEEE Transactions on Information Forensics and Security* **2023**.
14. Haim, N.; Vardi, G.; Yehudai, G.; Shamir, O.; Irani, M. Reconstructing training data from trained neural networks. *Advances in Neural Information Processing Systems* **2022**, *35*, 22911–22924.
15. Hitaj, B.; Ateniese, G.; Perez-Cruz, F. Deep models under the GAN: information leakage from collaborative deep learning. Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, 2017, pp. 603–618.
16. Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. *Advances in neural information processing systems* **2019**, *32*.
17. Gao, Y.; Zhang, L.; Wang, L.; Choo, K.K.R.; Zhang, R. Privacy-preserving and reliable decentralized federated learning. *IEEE Transactions on Services Computing* **2023**, *16*, 2879–2891.
18. Wu, X.; Zhang, Y.; Shi, M.; Li, P.; Li, R.; Xiong, N.N. An adaptive federated learning scheme with differential privacy preserving. *Future Generation Computer Systems* **2022**, *127*, 362–372.
19. Anand, A.; Dhakal, S.; Akdeniz. Differentially private coded federated linear regression. 2021 IEEE Data Science and Learning Workshop (DSLW). IEEE, 2021, pp. 1–6.
20. Wei, K.; Li, J.; Ding. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security* **2020**, *15*, 3454–3469.
21. McMahan, B.; Moore. Communication-Efficient Learning of Deep Networks from Decentralized Data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics; Singh, A.; Zhu, J., Eds. PMLR, 2017, Vol. 54, *Proceedings of Machine Learning Research*, pp. 1273–1282.
22. Hsu, T.M.H.; Qi, H.; Brown, M. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification, 2019, [arXiv:cs.LG/1909.06335].
23. Khaled, A.; Mishchenko, K.; Richtarik, P. Tighter Theory for Local SGD on Identical and Heterogeneous Data. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics; Chiappa, S.; Calandra, R., Eds. PMLR, 2020, Vol. 108, *Proceedings of Machine Learning Research*, pp. 4519–4529.
24. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network, 2015, [arXiv:stat.ML/1503.02531].
25. Lin, T.; Kong, L.; Stich, S.U.; Jaggi, M. Ensemble Distillation for Robust Model Fusion in Federated Learning. Advances in Neural Information Processing Systems; Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; Lin, H., Eds. Curran Associates, Inc., 2020, Vol. 33, pp. 2351–2363.
26. Sattler, F.; Korjakow, T.; Rischke, R.; Samek, W. FedAUX: Leveraging Unlabeled Auxiliary Data in Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems* **2023**, *34*, 5531–5543. doi:10.1109/TNNLS.2021.3129371.
27. Chen, H.Y.; Chao, W.L. FedBE: Making Bayesian Model Ensemble Applicable to Federated Learning, 2021, [arXiv:cs.LG/2009.01974].
28. Zhang, L.; Shen, L.; Ding, L.; Tao, D.; Duan, L.Y. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10174–10183.
29. Zhu, Z.; Hong, J.; Zhou, J. Data-Free Knowledge Distillation for Heterogeneous Federated Learning. Proceedings of the 38th International Conference on Machine Learning; Meila, M.; Zhang, T., Eds. PMLR, 2021, Vol. 139, *Proceedings of Machine Learning Research*, pp. 12878–12889.
30. Tjell, K.; Wisniewski, R. Privacy preservation in distributed optimization via dual decomposition and ADMM. 2019 IEEE 58th Conference on Decision and Control (CDC). IEEE, 2019, pp. 7203–7208.
31. Goodfellow, I.; Pouget-Abadie, J. Generative adversarial nets. *Advances in neural information processing systems* **2014**, *27*.
32. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabia. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* **2020**, *2*, 429–450.
33. Karimireddy, S.P.; Kale, S.; Mohri. Scaffold: Stochastic controlled averaging for federated learning. International conference on machine learning. PMLR, 2020, pp. 5132–5143.
34. Zhu, Z.; Hong, J.; Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. International conference on machine learning. PMLR, 2021, pp. 12878–12889.
35. Varun, M.; Feng, S.; Wang, H.; Sural, S.; Hong, Y. Towards Accurate and Stronger Local Differential Privacy for Federated Learning with Staircase Randomized Response. Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy, 2024, pp. 307–318.

 doi:10.20944/preprints202410.2389.v1

36. Gad, G.; Gad, E.; Fadlullah, Z.M.; Fouda, M.M.; Kato, N. Communication-Efficient and Privacy-Preserving Federated Learning Via Joint Knowledge Distillation and Differential Privacy in Bandwidth-Constrained Networks. *IEEE Transactions on Vehicular Technology* **2024**.
37. Zheng, Q.; Chen, S.; Long, Q.; Su, W. Federated f-differential privacy. International conference on artificial intelligence and statistics. PMLR, 2021, pp. 2251–2259.
38. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
39. Fang, G.; Song, J.; Shen, C.; Wang, X.; Chen, D.; Song, M. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006* **2019**.
40. Krizhevsky, A.; Hinton, G.; others. Learning multiple layers of features from tiny images **2009**.
41. MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges — yann.lecun.com. https://yann.lecun.com/exdb/mnist/, 2010. [Accessed 14-08-2024].
42. Acar, D.A.E.; Zhao. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263* **2021**.
43. He, C.; Li, S.; So. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518* **2020**.
44. Yurochkin, M.; Agarwal. Bayesian nonparametric federated learning of neural networks. International conference on machine learning. PMLR, 2019, pp. 7252–7261.
45. Talaei, M.; Izadi, I. Adaptive Differential Privacy in Federated Learning: A Priority-Based Approach. *arXiv preprint arXiv:2401.02453* **2024**.