FedDM: Iterative Distribution Matching for Communication-Efficient Federated Learning

Yuanhao Xiong1* Ruochen Wang1*
Minhao Cheng2 Felix Yu3 Cho-Jui Hsieh1
1UCLA 2HKUST 3Google Research

Abstract

Federated learning (FL) has recently attracted increasing attention from academia and industry, with the ultimate goal of achieving collaborative training under privacy and communication constraints. Existing iterative model averaging based FL algorithms require a large number of communication rounds to obtain a well-performed model due to extremely unbalanced and non-i.i.d. data partitioning among different clients. Thus, we propose FedDM to build the global training objective from multiple local surrogate functions, which enables the server to gain a more global view of the loss landscape. In detail, we construct synthetic sets of data on each client to locally match the loss landscape from original data through distribution matching. FedDM reduces communication rounds and improves model quality by transmitting more informative and smaller synthesized data compared with unwieldy model weights. We conduct extensive experiments on three image classification datasets, and show that our method outperforms other FL counterparts in terms of efficiency and model performance given a limited number of communication rounds. Moreover, we demonstrate that FedDM can be adapted to preserve differential privacy with Gaussian mechanism and train a better model under the same privacy budget.

1. Introduction

Traditional machine learning methods are designed with the assumption that all training data can be accessed from a central location. However, due to the growing data size together with the model complexity [10, 23, 26], distributed optimization [7, 8, 43] is necessary over different machines. This leads to the problem of Federated Learning [32] (FL) – multiple clients (e.g. mobile devices or local organizations) collaboratively train a global model under the orchestration of a central server (e.g. service provider) while the training data are kept decentralized and private. Such a practical setting poses two primary challenges [20, 24, 29, 31, 32]: training data of the FL system are highly unbalanced and non-i.i.d. across downstream clients and more efficient communication with fewer costs is expected because of unreliable devices with limited transmission bandwidth.

Most of the existing FL methods [22, 28, 31, 32, 48] adopt an iterative training procedure from FedAvg [32], in which each round takes the following steps: 1) The global model is synchronized with a selected subset of clients; 2) Each client trains the model locally and sends its weight or gradient back to the server; 3) The server updates the global model by aggregating messages from selected clients. This framework works effectively for generic distributed optimization while the difficult and challenging setting of FL, unbalanced data partition in particular, would result in statistical heterogeneity in the whole system [30] and make the gradient from each client inconsistent. It poses a great challenge to the training of the shared model, which requires a substantial number of communication rounds to converge [28]. Although some improvements have been made over FedAvg [32] including modifying loss functions [31], correcting client-shift with control variates [22] and the like, the reduced number of communication round is still considerable and even the amount of information required by the server rises [56].

In our paper, we propose a different iterative surrogate minimization based method, FedDM, referred to Federated Learning with iterative Distribution Matching. Instead of the commonly-used scheme where each client maintains a locally trained model respectively and sends its gradient/weight to the server for aggregation, we take a distinct perspective at the client’s side and attempt to build a local surrogate function to approximate the local training objective. By sending those local surrogate functions to the server, the server can then build a global surrogate function around the current solution and conduct the update by minimizing this surrogate. The question is then how to build local surrogate functions that are informative and with a rel-
ative succinct representation. Inspired by recent progresses in data condensation [54, 55] we build local surrogate functions by learning a synthetic dataset to replace the original one to approximate the objective. It can be achieved by matching the original data distribution in the embedding space [16]. After the optimization of synthesized data, the client can transmit them to the server, which can then leverage the synthetic dataset to recover the global objective function for training. Our method enables the server to have implicit access to the global objective defined by the whole balanced dataset from all clients, and thus outperforms previous algorithms involved in training a local model with unbalanced data in terms of communication efficiency and effectiveness. We also demonstrate that our method can be adapted to preserve differential privacy under, an important factor to the deployment of FL systems.

Our contributions are primarily summarized as follows:

- We propose FedDM, which is based on iterative distribution matching to learn a surrogate function. It sends synthesized data to the server rather than commonly-used local model updates and improves communication efficiency and effectiveness significantly.
- We analyze how to protect privacy of client’s data for our method and show that it is able to guarantee $(\epsilon, \delta)$-differential privacy with the Gaussian mechanism and train a better model under the same privacy budget.
- We conduct comprehensive experiments on three tasks and demonstrate that FedDM is better than its FL counterparts in communication efficiency and the final model performance.

2. Related Work

Federated Learning. Federated learning [20, 32] has aroused heated discussion nowadays from both research and applied areas. With the goal to train the model collaboratively, it incorporates the principles of focused data collection and minimization [20]. FedAvg [32] was proposed along with the concept of FL as the first effective method to train the global model under the coordination of multiple devices. Since it is based on iterative model averaging, FedAvg suffers from heterogeneity in the FL system, especially the non-i.i.d. data partitioning, which degrades the performance of the global model and adds to the burden of communication [30]. To mitigate the issue, some variants have been developed upon FedAvg including [22, 31, 48]. For instance, FedProx [31] modifies the loss function while FedNova [48] and SCAFFOLD [22] leverage auxiliary information to balance the distribution shift. Apart from better learning algorithms with faster convergence rate, another perspective at improving efficiency is to reduce communication costs explicitly [5, 6, 40, 42, 51]. An intuitive approach is to quantize and sparsify the uploaded weights directly [40]. Efforts have also been made towards one-shot federated learning [17, 41, 44, 56], expecting to obtain a satisfactory model through only one communication round.

Differential Privacy. To measure and quantify information disclosure about individuals, researchers usually adopt the state-of-the-art model, differential privacy (DP) [11, 14, 15]. DP describes the patterns of groups while withholding information about individuals in the dataset. There are many scenarios in which DP guarantee is necessary [1, 2, 12, 13, 21, 34, 38]. For example, Abadi et.al [1] developed differentially private SGD (DP-SGD) which enabled training deep neural networks with non-convex objectives under a certain privacy budget. It was further extended to settings of federated learning, where various techniques have been designed to guarantee client-level or instance-level differential privacy [33, 37, 52]. Recently, DP has been taken into account for hyperparameter tuning [38].

Dataset Distillation. With the explosive growing of the size of training data, it becomes much more challenging and costly to acquire large datasets and train a neural network within moderate time [35, 36]. Thus, constructing smaller but still informative datasets is of vital importance. The traditional way to reduce the size is through coreset selection [3], which select samples based on particular heuristic criteria. However, this kind of method has to deal with a trade-off between performance and data size [35, 55]. To improve the expressiveness of the smaller dataset, recent approaches consider learning a synthetic set of data from the original set, or data distillation for simplicity. Along this line, different methods are proposed using meta-learning [46, 50], gradient matching [53, 55], distribution matching [49, 54], neural kernels [35, 36] or generative models [45]. These methods demonstrate great potentials in datasets such as CIFAR10 but face challenges in scaling up to larger ones like ImageNet. Besides, a recent work [9] has analyzed the privacy property of dataset distillation methods, focusing on membership inference attacks. It provides a complementary prospective to $(\epsilon, \delta)$-differential privacy discussed in our paper.

3. Methodology

In this part, we first present the iterative surrogate minimization framework in Section 3.1, and then expand on the details of our implementation of FedDM in Section 3.2. In addition, we discuss preserving differential privacy of our method through Gaussian mechanism in Section 3.3.

3.1. Iterative surrogate minimization framework for federated learning

Neural network training can be formulated as solving the finite sum minimization problem:

$$\min_w f(D; w) \quad \text{where} \quad f(D; w) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i; w),$$

(1)
where $w \in \mathbb{R}^d$ is the parameter to be optimized, $\mathcal{D}$ is the dataset and $\ell(x_i, y_i; w)$ is the loss of the prediction on sample $(x_i, y_i) \in \mathcal{D}$ w.r.t. $w$ such as cross entropy. We will abbreviate these terms as $f(w)$ and $\ell_i(w)$ for simplicity. Equation 1 is typically solved by stochastic optimizers when training data are gathered in a single machine. However, the scenario is different under the setting of federated learning with $K$ clients. In detail, each client $k$ has access to its local dataset of the size $n_k$ with the set of indices $\mathcal{I}_k$ ($n_k = |\mathcal{I}_k|$), and we can rewrite the objective as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} f_k(w) \quad \text{where} \quad f_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} \ell_i(w).$$

(2)

Figure 1. A 1-D example showing advantages of minimizing the surrogate function.

Since information can only be communicated between the server and clients, previous methods [22, 31, 32, 48] train the global model by aggregation of local model updates, as introduced in Section 1. However, as each client only sees local data which could be biased and skewed, the local updates is often insufficient to capture the global information. Further, since local weight update consists limited information, it is hard for the server to obtain better joint update direction by considering higher order interactions between different clients. We are motivated to leverage the surrogate function by the example in Figure 1. Specifically, we synthesize a 1-D binary classification problem and learn a surrogate for the objective function. We learn the surrogate function via distribution matching introduced in Section 3.2 around the weight of 0. Compared with the tangent line computed by the gradient, the surrogate function in orange matches the original one accurately and minimizing it leads to a satisfactory solution. More details can be checked in Appendix B. Thus, we hope to develop a novel scheme such that each client can send a local surrogate function instead of a single gradient or weight update to the server, so the server has a more global view to loss landscape to obtain a better update instead of pure averaging.

To achieve this goal, we propose to conduct federated learning with an iterative surrogate minimization frame-

work. At each round, let $w_r$ be the current solution, we build a surrogate training objective $\hat{f}_r(\cdot)$ to approximate the original training objective in the local area around $w_r$, and then update the model by minimizing the local surrogate function. The update rule can be written as

$$w_{r+1} = \min_{w \in B_{\rho}(w_r)} \hat{f}_r(w),$$

(3)

where $\hat{f}_r(w) \approx f(w)$, $\forall w \in B_{\rho}(w_r)$ and $B_{\rho}(w_r)$ is a $\rho$-radius ball around $w_r$. We do not expect to build a good surrogate function in the entire parameter space; instead, we only construct it near $w_r$ and obtain the update by minimizing the surrogate function within this space. Many optimization algorithms can be described under this framework. For instance, if $f_r(w) = \nabla f(w_r)^T (w - w_r)$ (based on the first-order Taylor expansion), then Equation 3 leads to the gradient descent update where $\rho$ controls the step size.

To apply this framework in the federated learning setting, we consider the decomposition of Equation 2 and try to build surrogate functions to approximate each $f_k(w)$ on each client. More specifically, each client aims to find

$$\hat{f}_{r,k}(w) \approx f_k(w), \quad \forall w \in B_{\rho}(w_t)$$

(4)

and send the local surrogate function $\hat{f}_{r,k}(\cdot)$ instead of gradient or weights to the server. The server then form the aggregated surrogate function

$$\hat{f}_r(w) = \hat{f}_{r,1}(w) + \cdots + \hat{f}_{r,K}(w)$$

(5)

and then use Equation 3 to obtain the update. Again, if each $\hat{f}_{r,k}$ is the Taylor expansion based on local data, it is sufficient for the client to send local gradient to the server, and the update will be equivalent to (large batch) gradient descent. However, we will show that there exists other ways to build local approximations to make federated learning more communication efficient.

### 3.2. Local distribution matching

Inspired by recent progresses in data distillation [35, 36, 53–55], it is possible to learn a set of synthesized data for each client to represent original data in terms of the objective function. Therefore, we propose to build local surrogate models based on the following approximation for the $r$-th round:

$$f_k(w) \approx \frac{1}{n_k} \sum_{j \in \mathcal{I}_k} \ell_j(\hat{x}_j, \hat{y}_j; w) = \hat{f}_{r,k}(\mathcal{S}; w), \quad \forall w \in B_{\rho}(w_r),$$

(6)

where $\mathcal{S}$ denotes the set of synthesized data and $\mathcal{I}^S_k$ is the corresponding set of indices. Note that we aim to approximate $f_k$ only in a local region around $w_r$ instead of finding the approximation globally, which is hard as demonstrated.
in [54, 55]. To form the approximation function in Equation 6, we solve the following minimization problem:

$$\min_{\mathcal{S}} \mathbb{E}_{w \sim \mathcal{P}_w} \| f_k(w) - \hat{f}_{r,k}(\mathcal{S}; w) \|^2$$  

(7)

where $w$ is sampled from distribution $\mathcal{P}_w$, which is a Gaussian distribution truncated at radius $\rho$. A different perspective at Equation 7 is that we can just match the distribution between the real data and synthesized ones given $f_k(w)$ and $\hat{f}_{r,k}(\mathcal{S}; w)$ are just empirical risks. A common way to achieve this is to estimate the real data distribution in the latent space with a lower dimension by maximum mean discrepancy (MMD) [49, 54]:

$$\sup_{\|h\|_{\mathcal{H}} \leq 1} \left| \mathbb{E}[h(w(D))] - \mathbb{E}[h(w(S))] \right|.$$  

(8)

where $h$ is reproducing kernel Hilbert space and $h$ is the embedding function that maps the input into the hidden representation. We use the empirical estimate of MMD in [54] since the underlying distribution is inaccessible. To make our approximation more accurate and effective, we match the outputs of the logit layer which corresponds with the Equation 7, together with the preceding embedding layer:

$$\mathcal{L}_{DM} = \mathbb{E}_{w \sim \mathcal{P}_w} \left[ \sum_{(x,y) \in D} h_{w}(x) - \sum_{(x,y) \in S} h_{w}(\tilde{x}) \right]^2$$

$$+ \mathbb{E}_{w \sim \mathcal{P}_w} \left[ \sum_{(x,y) \in D} z_{w}(x) - \sum_{(x,y) \in S} z_{w}(\tilde{x}) \right]^2,$$

(9)

where $h_{w}(x)$ again denotes intermediate features of the input while $z_{w}(x) \in \mathbb{R}^C$ represents the output of the final logit layer. It should be emphasized that we learn synthesized data for each class respectively, which means samples in $\mathcal{D}$ and $\mathcal{S}$ belong to the same class. For training, we adopt mini-batch based optimizers to make it more efficiently. Specifically, a batch of real data and a batch of synthetic data are sampled randomly for each class independently by $B_{\mathcal{D}} \sim \mathcal{D}_k$ and $B_{\mathcal{S}} \sim \mathcal{S}_k$. We plug these two batches into Equation 8 to compute $\mathcal{L}$, and $\mathcal{L} = \sum_{c=0}^{C-1} \mathcal{L}_c$, $\mathcal{S}_k$ can be updated with SGD by minimizing $\mathcal{L}$ for each client.

Then we aggregate all synthesized data from $K$ clients at the server’s side to approximate the global objective function, which is computed as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} f_k(w) \approx \sum_{k=1}^{K} \frac{n_k}{n} \hat{f}_{r,k}(\mathcal{S}_k; w), \ \forall w \in B_p(w_t).$$

(10)

Moreover, since synthesized data are trained based on a specific distribution around the current value of $w$, we need to iteratively synchronize the global weights with all the clients and obtain proper $\mathcal{S}$ according to the latest $w$ for the next communication round.

Therefore, instead of transmitting information such as parameters or gradients in previous FL algorithms, we propose federated learning with iterative distribution matching (FedDM) in Algorithm 1, following the steps below to train the global model:

(a) At each communication round, for each client, we adopt Equation 8 as the objective function to train synthesized data for each class.

(b) The server receives synthesized data and leverages them to update the global model.

(c) The current weight is then synchronized with all the clients and a new communication round starts by repeating step (a) and (b).

It should be noticed that through estimating the local objective, FedDM extracts richer information than existing model averaging based methods, and enables the server to explore the loss landscape from a more global view. It reduces communication rounds significantly. On the other hand, the explicit message uploaded to the server, or the number of float parameters, is relatively smaller. This is especially true when training large neural network models, where the size of neural network parameters (and therefore gradient update) is much larger than the size of the input. Take CIFAR10 as an example, when training data are distributed obeying Dir($0.5$), the average number of classes per client (cpc) is $9$. When we adopt the number of images per class (ipc) of $10$ for the synthetic set, the total number of float parameters uploaded to the server is: the number of clients $\times$ cpc $\times$ ipc $\times$ image size $= 10 \times 9 \times 10 \times 3 \times 32 \times 32 \approx 2.8 \times 10^6$. For those iterative model averaging model methods, the number of float parameters is equal to the product of weight size and the number of clients, which is $320010 \times 10 \approx 3.2 \times 10^6$ for ConvNet [55] and comparatively larger than FedDM. An extensive comparison is presented in Appendix C.

### 3.3. Differential privacy of FedDM

An important factor to evaluate a federated learning algorithm is whether it can preserve differential privacy. Before analyzing our method, we first review some fundamentals of differential privacy.

**Definition 3.1 (Differential Privacy [12]).** A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies $(\epsilon, \delta)$-differential privacy if for any two adjacent datasets $D_1, D_2$ and any measurable subset $S \subseteq \mathcal{R}$,

$$\Pr(\mathcal{M}(D_1) \in S) \leq e^\epsilon \Pr(\mathcal{M}(D_2) \in S) + \delta.$$  

(11)

In this paper, we focus on instance-level differential privacy, which indicates that $D_1$ and $D_2$ differ on a single element. Typically, the randomized mechanism is applied to a query function of the dataset, $f : \mathcal{D} \to \mathcal{X}$. Without loss of generality, we assume that the output spaces $\mathcal{R}, \mathcal{X} \subseteq \mathbb{R}^m$. A key quantity in characterizing differential privacy for various mechanisms is the sensitivity of a query [15] $f : \mathcal{D} \to \mathbb{R}^m$ in a given norm $\ell_p$. Formally this is defined as

$$\Delta_p \triangleq \max_{D_1, D_2} \| f(D_1) - f(D_2) \|_p.$$  

(12)
Algorithm 1 FedDM: Federated Learning with Iterative Distribution Matching

1: Input: Training set \( D \), set of synthetic samples \( S \), deep neural network parameterized with \( w \), probability distribution over parameters \( P_w \), training iterations of distribution matching \( T \), learning rate \( \eta_c \) and \( \eta_s \).

2: Server executes:
   3: for each round \( r = 1, \ldots, R \) do
   4:     for client \( k = 1, \ldots, K \) do
   5:         \( S_k \leftarrow \text{ClientUpdate}(k, w_r) \)
   6:         Transmit \( S_k \) to the server
   7:     end for
   8:     Aggregate synthesized data from each client and build the surrogate function by Equation 9
   9:     Update weights to \( w_{r+1} \) on \( S \) by SGD with the learning rate \( \eta_s \)
   10: end for

11: ClientUpdate\((k, w_r)\):
12: Initialize \( S_k \) from random noise or real examples.
13: for \( t = 0, \ldots, T - 1 \) do
14:     Sample \( w \sim P_w(w_r) \)
15:     Sample mini-batch pairs \( B_c^D \sim D_k \) and \( B_c^S \sim S_k \) for each class \( c \)
16:     Compute \( L_c \) based on Equation 8, \( L_c \leftarrow \sum_{c=0}^{C-1} L_c \)
17:     Update \( S_k \leftarrow S_k - \eta_c \nabla S_k L \)
18: end for

---

Gaussian mechanism [15] is one simple and effective method to achieve \((\epsilon, \delta)\)-differential privacy:

\[
\mathcal{M}(D) \triangleq f(D) + Z, \quad \text{where} \quad Z \sim \mathcal{N}(0, \sigma^2 \Delta^2_p I). \quad (12)
\]

It has been proved that under Gaussian mechanism, \((\epsilon, \delta)\)-differential privacy is satisfied for the function \( f \) of sensitivity \( \Delta_p \), if we choose \( \sigma \geq \sqrt{2 \log \frac{1.25}{\delta}} / \epsilon \) [15]. Differentially private SGD (DP-SGD) [1] then applies Gaussian mechanism to deep learning optimization with hundreds of steps and demonstrates the following theorem:

**Theorem 3.1 (Differential Privacy of DP-SGD).** There exist constants \( c_1 \) and \( c_2 \) so that given the sampling probability \( q \) and the number of steps \( T \), for any \( \epsilon < c_1 q^2 T \), DP-SGD is \((\epsilon, \delta)\)-differentially private for any \( \delta > 0 \) if

\[
\sigma \geq c_2 q \sqrt{T \log(1/\delta)} / \epsilon. \quad (13)
\]

We then prove that by leveraging DP-SGD to update the synthetic dataset which is initialized from random Gaussian noise, FedDM can preserve differential privacy of the original dataset. We present this DP guarantee of FedDM in the theorem below:

**Theorem 3.2 (Differential privacy of FedDM).** Given the synthetic dataset \( S \) is initialized from random noise, FedDM trained with DP-SGD can guarantee \((\epsilon, \delta)\)-differential privacy in a K-client federated learning system, with \( \sigma \geq \sqrt{\frac{\log(\delta)}{T q^2}} \) or \( \sigma \geq \sqrt{\frac{2 \log(1/\delta)}{\epsilon}} \) if \( T q^2 \leq \epsilon / 2 \) in each communication round.

A complete proof and an initial analysis of differential privacy for \( R \) communication rounds are included in Appendix D. We also present the whole procedure of FedDM integrated with DP-SGD in Appendix E.

### 4. Experiments

#### 4.1. Experimental setup

**Datasets.** In this paper, we focus on image classification tasks, and select three commonly-used datasets: MNIST [27], CIFAR10 [25], and CIFAR100 [25]. We adopt the standard training and testing split. Following commonly-used scheme [47], we simulate non-i.i.d. data partitioning with Dirichlet distribution \( \text{Dir}_K(\alpha) \), where \( K \) is the number of clients and \( \alpha \) determines the non-i.i.d. level, and allocate divided subsets to clients respectively. A smaller value of \( \alpha \) leads to more unbalanced data distribution. The default data partitioning is based on Dirichlet(0.5) with 10 clients. Furthermore, we also take into account different scenarios of data distribution, including Dirichlet(0.1), Dirichlet(0.01). Results of Dirichlet(0.5) and Dirichlet(50) (i.i.d. scenario) can be found in Appendix F.1. We evaluate a realistic dataset CelebA as well, following splits in the LEAF benchmark [4] in Appendix F.1.

**Baseline methods.** We compare FedDM with four representative iterative model averaging based methods: FedAvgM [32], FedProx [31], FedNova [48], and SCAFFOLD [22]. We summarize the action of the client and the server, and the transmitted message for all methods in Table 1. Two stronger methods, FedAvgM [19] and FedAdam [39], are compared in Appendix F.1.

**Hyperparameters.** For FedDM, following [54], we select the batch size as 256 for real images, and update the synthetic set \( S \) for \( T = 1,000 \) iterations with \( \eta_c = 1 \) for each client in each communication round, and tune the number of images per class (ipc) within [3, 5, 10]. Synthetic images are initialized as randomly sampled real images with corresponding labels suggested by [54, 55], and random noise initialization is leveraged when differential privacy is required. Considering the trade-off between communication efficiency and model performance, we choose ipc to be 10 for MNIST and CIFAR10, 5 for CIFAR100 when there are 10 clients. The choice of radius \( \rho = 5 \) is discussed in Section 4.5. On the server’s side, the global model is trained with the batch size 256 for 500 epochs by...
In particular, we tune \( \mu \in [0, 0.01, 0.1, 1] \) and \( \eta \in [0.01, 0.1, 1] \), and local epoch from \([1, 2, 5, 10, 15, 20]\). For a fair comparison, all methods share the fixed number of communication rounds as a fair comparison, all methods share the fixed number of communication rounds in Appendix F.1 and a different network ResNet-18 [18] in Section 4.5 as well. All experiments are run for three times with different random seeds with one NVIDIA 2080Ti GPU and the average performance is reported. More implementation details and experimental results can be found in Appendix F.1.

### 4.2. Communication efficiency & convergence rate

We first evaluate our method in terms of communication efficiency and convergence rate on all three datasets on the default data partitioning Dir\(_{10}(0.5)\). As we can see in Figure 2(a)-(c), our method FedDM performs the best among all considered algorithms by a large margin on MNIST, CIFAR10, and CIFAR100. Specifically, for CIFAR10, FedDM achieves \(69.66 \pm 0.13\%\) on test accuracy while the best baseline SCAFFOLD only has \(66.12 \pm 0.17\%\) after 20 communication rounds. FedDM also has the best convergence rate and it significantly outperforms baseline methods within the initial few rounds. Advantages of FedDM are more evident when we evaluate convergence as a function of the message size. As mentioned in 3.2, FedDM re-

---

**Table 1. Summary of different FL methods.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Client</th>
<th>Message</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedAvg [32]</td>
<td>( \min f_k(w) )</td>
<td>( \Delta^*_w )</td>
<td>model averaging</td>
</tr>
<tr>
<td>FedProx [31]</td>
<td>( \min f_k(w) + \mu |w - w_r|/2 )</td>
<td>( \Delta^*_w )</td>
<td>model averaging</td>
</tr>
<tr>
<td>FedNova [48]</td>
<td>( \min f_k(w) )</td>
<td>( d ) and ( \alpha^*_w )</td>
<td>normalized model averaging</td>
</tr>
<tr>
<td>SCAFFOLD [22]</td>
<td>( \min f_k(w, c) )</td>
<td>( \Delta_w ) and ( \Delta_c^* )</td>
<td>model averaging for both ( w ) and ( c )</td>
</tr>
<tr>
<td>FedDM(Ours)</td>
<td>( \min \text{Equation 8} )</td>
<td>( S )</td>
<td>model updating on ( S )</td>
</tr>
</tbody>
</table>

* \( \Delta_w \) denotes the model update, \( d \) is the aggregated gradient and \( \alpha \) is the coefficient vector, \( \Delta_c \) is the change of control variates. Refer to original papers for more details.
(b) Medium noise ($\epsilon = 2.46$).
(c) Large noise ($\epsilon = 1.35$).

Figure 3. Performance of FL methods with different levels of noise. To preserve differential privacy, FedDM initializes the synthetic data from random Gaussian noise.

Table 2. Test accuracy of FL methods with different level of non-uniform data partitioning.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
<th>MNIST</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedAvg</td>
<td>96.92 ± 0.09</td>
<td>57.32 ± 0.04</td>
<td>32.00 ± 0.50</td>
<td>91.04 ± 0.80</td>
<td>39.28 ± 0.25</td>
<td>27.05 ± 0.45</td>
</tr>
<tr>
<td>FedProx</td>
<td>96.72 ± 0.04</td>
<td>56.92 ± 0.30</td>
<td>30.77 ± 0.52</td>
<td>91.18 ± 0.16</td>
<td>40.30 ± 0.15</td>
<td>25.88 ± 0.39</td>
</tr>
<tr>
<td>FedNova</td>
<td>98.04 ± 0.03</td>
<td>60.76 ± 0.14</td>
<td>31.92 ± 0.42</td>
<td>90.27 ± 0.49</td>
<td>36.46 ± 0.42</td>
<td>27.52 ± 0.43</td>
</tr>
<tr>
<td>SCAFFOLD</td>
<td>98.32 ± 0.06</td>
<td>60.96 ± 1.20</td>
<td>34.39 ± 0.25</td>
<td>88.37 ± 0.25</td>
<td>32.42 ± 1.13</td>
<td>31.14 ± 0.20</td>
</tr>
<tr>
<td>FedDM</td>
<td>98.67 ± 0.01</td>
<td>67.38 ± 0.32</td>
<td>37.58 ± 0.27</td>
<td>98.21 ± 0.23</td>
<td>63.82 ± 0.17</td>
<td>34.98 ± 0.17</td>
</tr>
</tbody>
</table>

As discussed in Section 3.3, if the synthetic dataset is initialized from random noise, using DP-SGD in local training of FedDM can satisfy $(\epsilon, \delta)$-differential privacy, with $\sigma \geq \sqrt{2\log(1/\delta)} \epsilon$ for any $Tq^2 \leq \epsilon/2$, a relatively loose bound independent of training steps $T$. To make a fair comparison, we use tensorflow privacy to compute $\epsilon$ with a tight bound given the number of examples, batch size, training steps, $\delta = 10^{-5}$ under $\sigma \in 1, 3, 5$ for FedDM, and obtain noise levels for baseline methods accordingly which are $[0.44, 0.76, 0.95]$ respectively. We tune clipping norm $C \in [1, 3, 5, 10]$. $S$ is initialized from $\mathcal{N}(0, 1)$ to guarantee differential privacy. We notice in Figure 3 that under the same DP guarantee, FedDM outperforms other FL counterparts in terms of convergence rate and final performance. Moreover, the accuracy of FedDM does not drop significantly compared with all considered methods when the noise level increases, indicating that FedDM is more resistant to the perturbed optimization. Visualization of the synthetic dataset can be found in Appendix F.2.

4.5. Analysis of FedDM

In this section, we analyze FedDM to investigate effects of hyperparameters including the initialization of the synthetic dataset, image-per-class (ipc), network structure and selection of $\rho$-radius ball. Besides, we compare our method with a strong baseline of sending real images with the same size. Extensive results are reported in Appendix F.1.

Initialization of the synthetic dataset. We conduct an ablation study on the initialization of the synthetic dataset $\mathcal{S}$ on CIFAR10 with the default partition $\text{Dir}_{10}(0.5)$. In detail, random initialize $\mathcal{S}$ based on the standard normal distribution $\mathcal{N}(0, 1)$ while real samples instances from the original dataset to be distilled. It can be observed in Table 3 that random performs consistently better than random, which concurs with the conclusion in [54] and justifies the choice of real in our experiments. Note that even random can outperform model averaging methods compared with results in Figure 2 and Table 2. In addition, random with DP-SGD
helps preserve differential privacy of FedDM, and still improves the efficiency and accuracy significantly in Figure 3.

<table>
<thead>
<tr>
<th>Partitioning</th>
<th>random</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>α = 0.5</td>
<td>64.12 ± 0.15</td>
<td><strong>69.66 ± 0.13</strong></td>
</tr>
<tr>
<td>α = 0.1</td>
<td>62.75 ± 0.24</td>
<td><strong>67.38 ± 0.32</strong></td>
</tr>
<tr>
<td>α = 0.01</td>
<td>59.00 ± 0.26</td>
<td><strong>63.82 ± 0.17</strong></td>
</tr>
</tbody>
</table>

Table 3. Test accuracy of FedDM with random and real S initialization on CIFAR10. Three data partitionings are evaluated.

**Effects of ipc.** Experiments are conducted on CIFAR10 with the distribution Dir10(0.5) with three different ipc values from [3, 5, 10]. As the ipc increases, the performance gradually get better from 53.64 ± 0.35%, 62.24 ± 0.04% to 69.62 ± 0.14%. On the other hand, more images per class indicates a heavier communication burden in the meanwhile. We need to trade off the model performance against the communication cost, and thus choose an appropriate ipc value based on the task.

![Figure 4. Performance under different ipc values.](image)

**Different network structures.** Besides ConvNet, we evaluate FedDM under the default CIFAR10 setting on ResNet-18. It can be observed that our method works well even for this more complicated and larger model in Figure 5. It should also be emphasized that for FL baseline methods, they have to transmit a larger amount of message while FedDM maintains the original size. This makes FedDM more efficient in larger networks.

![Figure 5. Test accuracy on ResNet-18.](image)

**Selection of ρ-radius ball.** It has been discussed in Section 3.1 that \( B_\rho(w_r) \) is a ρ-radius ball around \( w_r \):

$$B_\rho(w_r) = \{ w | \|w - w_r\|_2 \leq \rho \}. \quad (14)$$

In FedDM, we sample \( w \) based on a truncated Gaussian distribution below:

$$P_w(w_r) = \text{Clip}(\mathcal{N}(w_r, 1), \rho), \quad (15)$$

where we clip the sampled weight to guarantee that \( \|w - w_r\|_2 \leq \rho \). At the server’s side, when training the global model, we also clip the weight to the ρ-radius ball. We conduct experiments to choose \( \rho \) from [3, 5, 10] and present the test accuracy after 20 communication rounds on CIFAR10 under the default Dir10(0.5) setting in Table 4. We find that performance is similar and FedDM is not very sensitive to the choice of \( \rho \). \( \rho = 5 \) performs relatively the best and we hypothesize that a too small weight restricts the optimization to a limited range and a too big one adds to the difficulty of learning a surrogate function. Based on results in Table 4, we select \( \rho = 5 \) for all our experiments.

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>Test accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \rho = 3 )</td>
<td>69.15 ± 0.09</td>
</tr>
<tr>
<td>( \rho = 5 )</td>
<td><strong>69.66 ± 0.13</strong></td>
</tr>
<tr>
<td>( \rho = 10 )</td>
<td>69.32 ± 0.24</td>
</tr>
</tbody>
</table>

Table 4. Test accuracy of FedDM under different \( \rho \).

**Comparison with transmitting real images.** Our method is compared with REAL, which sends real images of the same size as FedDM (ipc = 10). In particular, REAL achieves test accuracy of 68.66 ± 0.08% on CIFAR10 with the default setting, but cannot beat FedDM with 69.62 ± 0.14%. It indicates that our learned synthetic set can capture richer information of the whole dataset rather than just a few images.

5. **Conclusions and Limitations**

In this paper we propose an iterative distribution matching based method, FedDM, to achieve more communication-efficient federated learning. By learning a synthetic dataset for each client to approximate the local objective function, the server can obtain a global view of the loss landscape better than aggregating local model updates. We also show that FedDM can preserve differential privacy with Gaussian mechanism. However, there is still a trade-off between the size of the synthetic set and the final performance, especially for classification tasks with hundreds of clients or classes. How to reduce the synthetic set to save communication costs can be a future direction.

**Acknowledgements.** This work is supported by NSF under IIS-2008173, IIS-2048280, an Okawa research grant, a Cisco research grant and Google.
References

[28] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental
study. In *IEEE International Conference on Data Engineering*, 2022. 1, 6

[29] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 2021. 1


[42] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019. 2


