Unlocking the Potential of Federated Learning: The Symphony of Dataset Distillation via Deep Generative Latents

Yuqi Jia*[1], Saeed Vahidian*[1], Jingwei Sun[1], Jianyi Zhang[1], Vyacheslav Kungurtsev[2], Neil Zhenqiang Gong[1], and Yiran Chen[1]

[1] Duke University
{yuqi.jia,saeed.vahidian,neil.gong,yiran.chen}@duke.edu
[2] Czech Technical University
vyacheslav.kungurtsev@fel.cvut.cz

Abstract. Data heterogeneity presents significant challenges for federated learning (FL). Recently, dataset distillation techniques have been introduced, and performed at the client level, to attempt to mitigate some of these challenges. In this paper, we propose a highly efficient FL dataset distillation framework on the *server* side, significantly reducing both the computational and communication demands on local devices while enhancing the clients' privacy. Unlike previous strategies that perform dataset distillation on local devices and upload synthetic data to the server, our technique enables the server to leverage prior knowledge from pre-trained deep generative models to synthesize essential data representations from a heterogeneous model architecture. This process allows local devices to train smaller surrogate models while enabling the training of a larger global model on the server, effectively minimizing resource utilization. We substantiate our claim with a theoretical analysis, demonstrating the asymptotic resemblance of the process to the hypothetical ideal of completely centralized training on a heterogeneous dataset. Empirical evidence from our comprehensive experiments indicates our method's superiority, delivering an accuracy enhancement of up to 40% over non-dataset-distillation techniques in highly heterogeneous FL contexts, and surpassing existing dataset-distillation methods by 18%. In addition to the high accuracy, our framework converges faster than the baselines because rather than the server trains on several sets of heterogeneous data distributions, it trains on a multi-modal distribution. Our code is available at `https://github.com/jyqhahah/FedDGM.git`

Keywords: Federated Learning · Data Distillation · Generative Model

1 Introduction

Federated Learning (FL), a recently very popular approach in the realm of IoT and AI, enables a multitude of devices to collaboratively learn a shared model,

* These authors contributed equally to this work

while keeping the training data localized, thereby addressing privacy concerns inherent in traditional centralized training methods [19]. In practice, however, FL faces a significant challenge in the form of data heterogeneity due to the diverse and non-i.i.d. nature of client data, shaped by varying user preferences and usage patterns [5,9,20]. To address this complex challenge, which impacts the effectiveness and efficiency of the learning process, several previous methods have been proposed, such as [10, 18]. Recent research, such as FedDM [23], integrates dataset distillation (DD) [7, 11, 21] techniques into FL, performing DD on local devices and uploading synthetic data to the server, which has shown exciting progress in addressing the challenge of data heterogeneity.

However, existing methods incorporating DD in FL are not without their limitations. One significant issue is their inability to enhance knowledge generalization across different model architectures, a challenge especially relevant in FL due to the variability of device resources and capabilities. Furthermore, these methods often compromise data privacy principles, as they require clients to upload synthetic data directly to the server. Addressing these concerns, we propose an advanced, server-centric FL DD framework, which significantly bolsters data privacy and substantially enhances knowledge generalization across different model architectures in various settings to reduce the computational load and communication overhead on client devices. Our methodology is structured into three pivotal components in each communication round. Firstly, it facilitates the training of compact surrogate models on local devices, which are then updated to the server, accommodating diverse resource constraints. Secondly, leveraging pre-trained deep generative models, the server synthesizes distilled data representations via matching training trajectories of the local surrogate models. Subsequently, the server employs this distilled synthetic data to refine a more comprehensive global model. Our extensive experimental analysis underscores the effectiveness of our approach. We demonstrate a notable 40% increase in accuracy compared to traditional non-dataset-distillation techniques within varied FL environments, as exemplified by our results on the CIFAR-10 benchmark. Furthermore, our method outperforms existing DD techniques by 18% and shows a remarkable performance improvement of around 10% on high-resolution image datasets like ImageNet.

To motivate the principle, consider Figure 1. While standard aggregation-based FL frameworks such as FedAvg average the gradients/parameters from each local training rounds together, this effectively can be thought of as training on the datasets individually and computing the arithmetic mean of the result. Recall fundamentally that the ideal problem of centrally learning the entire dataset, that in the FL context is impossible due to privacy or technical constraints. By contrast, in our schema, we attempt to solve the ideal problem by attempting to recreate the distribution by combining the datasets from those distilled on training each client's data. Thus, if the distribution, as far as SGD iterates are concerned, is entirely identical, we would recreate the ideal training scenario. Of course, this is not possible in practice, however, fundamentally the quality of the training is only limited to the quality of the dataset distillation.

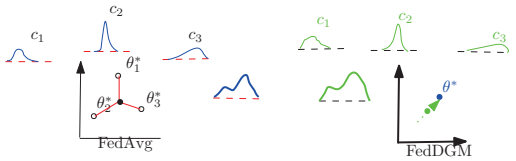


Fig. 1: The ideal problem would be to train on the central multimodal dataset, including all of the data. FedAvg can be considered as solving each problem individually and averaging the final solution, which is not guaranteed to be closer than a certain distance to the true minimum regardless of the quality of the local training. FedDGM attempts to recreate the dataset in the most relevant way for training, by combining distilled data from each client.

The contributions of our work are as follows:

1. We present a novel FL dataset distillation framework, allowing clients to train smaller models to mitigate computational costs, while the server aggregates this information to train a larger model.
2. We conduct a theoretical analysis demonstrating that, from an asymptotic perspective, our method is equivalent to the theoretical ideal of centralized training on a multi-modal distribution, representing the sum of heterogeneous data.
3. We conduct extensive experiments to underscore the effectiveness of our approach. Our empirical evidence demonstrates a significant improvement in model performance in highly heterogeneous FL contexts, setting a new benchmark for dataset distillation methods in FL.

We test our ideas on two widely adopted network architectures on popular datasets in the FL community (CIFAR10, ImageNet) under a wide range of choices of architecture backbones and compare them with several global state-of-the-art baselines. We have the following findings:

– **F1:** The advantages of our dataset distillation-driven FL approach are accentuated in comparison to aggregation-based FL training methods, particularly when distinct groups of clients manifest substantial differences in the distributions of their local data (Section 4.2 of the main paper and Section 5 of the supplementary material).
– **F2:** Theoretically, we find interesting implications of the distributional limit from the bilevel structure of the dataset distillation problem. Considering marginals at this limit motivates the approximation to the ideal problem with an approximate dataset (Section 3).
– **F3:** Our method is architecture-agnostic and significantly enhances state-of-the-art performance on high-resolution images by leveraging prior information from a pre-trained generative model (Section 4.2).
– **F4:** Our framework converges significantly faster than the state-of-the-arts. The reason is that our framework aggregates the knowledge from the clients

by gathering synthetic data rather than averaging the local model parameters, which preserves the local data information much better (Section 4.2).

2 Method

2.1 Overview

In this work, we introduce a novel FL training approach grounded in Dataset distillation with deep Generative Models (FedDGM). Diverging from conventional aggregation based FL training techniques like FedAvg [12], FedDGM distinguishes itself by enabling collaborative training of a large global model with enhanced performance while maintaining a lower computational burden. In contrast to data distillation based FL methods, exemplified by FedDM [23], FedDGM sets itself apart by addressing privacy concerns more effectively than FedDM. This is achieved by exclusively transferring model parameters rather than synthetic data, ensuring a heightened level of privacy preservation.

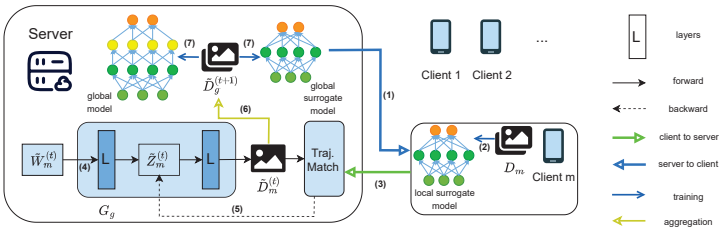


Fig. 2: An overview of FedDGM. At each global communication round t, (1) the server sends global surrogate model to clients, (2) clients train local surrogate models on local data, (3) clients send local surrogate models to the server, (4) the server initializes latent vectors and generates synthetic data for each client m, (5) the server uses MTT to update latent vectors, (6) after distilling data, the server aggregates all synthetic data and gets the synthetic dataset $\tilde{D}_g^{(t+1)}$, and (7) the server uses the synthetic dataset to train the global model and global surrogate model.

In particular, to reduce the computational costs for clients, we enable clients to utilize a small *surrogate model*, which typically has fewer model parameters than the global model. Clients use this surrogate model for local training and send their trained models to the server. To avoid additional communication costs and privacy concerns, when the server receives the surrogate models from clients, it employs the Matching Training Trajectories (MTT) method [1] to distill synthetic data for each client. To enhance the training effectiveness of the synthetic data on the global model and the performance of FedDGM on high-resolution datasets, we leverage prior knowledge from pre-trained deep generative models to distill data. Subsequently, the server trains the global model on the

distilled data. Moreover, the server uses the distilled data to train an extra global surrogate model, and then sends it back to the clients, enabling them to perform their local training in the next communication round. Fig. 2 is an overview of FedDGM, and Algorithm 1 summarizes our FedDGM.

2.2 Problem Formulation

We begin by introducing some notations in FedDGM. Assuming that each client m has its own local training dataset D_m and a surrogate model parameterized by $\theta_m^{(0)}$. In each global communication round t, client m addresses the following optimization problem:

$$\theta_m^{(t+1)} = \min_{\theta \in B_r(\theta_g^{(t)})} \mathcal{L}(D_m; \theta), \tag{1}$$

Algorithm 1 The FedDGM Framework

Input: M clients indexed by m, participating-client number K, communication rounds T_g, server global model f with $w_g^{(0)}$, server surrogate model f_s with $\theta_g^{(0)}$, pre-trained deep generative model G_g.

Server executes:
initialize global model f with $w_g^{(0)}$ and global surrogate model f_s with $\theta_g^{(0)}$
for each round $t = 0, 1, 2, ...$ **do**
  $\mathbb{S}_t \leftarrow$ (random set of K clients)
  for each client $m \in \mathbb{S}_t$ **in parallel do**
    broadcast $\theta_g^{(t)}$ to clients
    $\theta_m^{(t+1)} \leftarrow \text{ClientUpdate}(\theta_g^{(t)}, D_m)$
    transmit $\theta_m^{(t+1)}$ to the server
  for each client $m \in \mathbb{S}_t$ **do**
    randomly initialize vectors $\tilde{W}_m^{(t)}$
    calculate latent vectors $\tilde{Z}_m^{(t)}$ using $\tilde{W}_m^{(t)}$
    for t_d = 1,...,T_d **do**
      generate distilled data $\tilde{D}_m^{(t)} = G_g(\tilde{Z}_m^{(t)})$
      obtain $\hat{\theta}_g^{(t+T_s)}$ from $\hat{\theta}_g^{(t)} = \theta_g^{(t)}$ by SGD on $\tilde{D}_m^{(t)}$ for T_s iterations
      compute the Trajectory Matching loss $\mathcal{L}_{MTT}$
      update $\tilde{Z}_m^{(t)}$ with respect to $\mathcal{L}_{MTT}$
  $\tilde{D}_g^{(t+1)} = \{\tilde{D}_m^{(t)} | \tilde{D}_m^{(t)} = G_g(\tilde{Z}_m^{(t)}), m \in \mathbb{S}_t\}$
  update parameters to $\theta_g^{(t+1)}$ and $w_g^{(t+1)}$ on $\tilde{D}_g^{(t+1)}$ by SGD, respectively

ClientUpdate $(\theta_g^{(t)}, D_m)$**:**
initialize local surrogate model parameters: $\theta_m^{(t+1)} = \theta_g^{(t)}$
update $\theta_m^{(t+1)}$ on D_m by SGD for T_l iterations
return $\theta_m^{(t+1)}$

where $\theta_g^{(t)}$ is the global surrogate model parameters sent from the server, and $B_r(\theta_g^{(t)})$ is a r-radius ball around $\theta_g^{(t)}$. After client m sends its local surrogate

model parameters $\theta_m^{(t+1)}$ to the server, the server distills synthetic data $\tilde{D}_m^{(t)}$ based on $\theta_m^{(t+1)}$. The server gets the synthetic dataset $\tilde{D}_g^{(t+1)}$ by aggregating all $\tilde{D}_m^{(t+1)}$, and trains the global model parameterized by $w_g^{(t)}$, together with the global surrogate model parameterized by $\theta_g^{(t)}$, on $\tilde{D}_g^{(t+1)}$. Subsequently, the server sends the global surrogate model parameters back to clients.

In the next section, we will delve into the core component of FedDGM, which is the data distillation on the server.

2.3 Global Data Distillation

Our data distillation incorporates two key techniques to address different issues: distillation via Matching Training Trajectories and optimization via deep generative latents.

Distillation via Matching Training Trajectories Previous FL training methods that incorporate data distillation directly transmit synthetic data between the server and clients [23], leading to privacy concerns. In contrast, FedDGM adopts a different approach where the server only receives surrogate models from clients. This prevents the server from using dataset distillation methods that require access to the actual local data of clients, such as data condensation [25] and distribution matching [24]. Hence, FedDGM employs an adaptive version of Matching Training Trajectories, a technique that solely utilizes checkpoints of model parameters saved throughout the training of clients' local surrogate models on their real datasets to distill data.

Specifically, after client m uploads its local surrogate model parameters $\theta_m^{(t+1)}$, the server trains a student network initialized with $\theta_g^{(t)}$ on a randomly initialized synthetic dataset $\tilde{D}_m^{(t)}$ for T_s epochs, where $\theta_g^{(t)}$ is the global surrogate model parameters received by client m in the last communication round. We denote the post-training student model parameters as $\hat{\theta}_g^{(t+T_s)}$, and define the training trajectory matching loss as the normalized L_2 distances between $\hat{\theta}_g^{(t+T_s)}$ and client m's local surrogate model parameters $\theta_m^{(t+1)}$, i.e.,

$$\mathcal{L}_{MTT} = \frac{||\hat{\theta}_g^{(t+T_s)} - \theta_m^{(t+1)}||^2}{||\theta_g^{(t)} - \theta_m^{(t+1)}||^2}. \tag{2}$$

Then the server updates the synthetic dataset $\tilde{D}_m^{(t)}$ according to the training trajectory matching loss $\mathcal{L}_{MTT}$. The server repeats the above steps T_d times to obtain the updated synthetic dataset $\tilde{D}_m^{(t)}$.

In the above process, we directly use $\theta_g^{(t)}$ to initialize the student network instead of randomly selecting a starting epoch from client m's training trajectories. This is done to reduce communication costs, allowing the server to distill data even if it only receives updated surrogate model parameters $\theta_m^{(t+1)}$ from client m. Due to the server's direct use of synthetic data for global model training, as opposed to aggregating model parameters or gradients sent by clients, data

distillation significantly enhances the convergence speed of training the global model, noticeably reducing the global communication rounds. More importantly, it allows the global model on the server to have a different architecture from the local surrogate models on clients' devices, thereby reducing the computational costs for clients. To further improve the generalization of synthetic data on the global model, which often has a distinct architecture from clients' local surrogate models, we employ another key technique, optimization via deep generative latents, as introduced in the next section.

Optimization via Deep Generative Latents In the previous section, unlike the methods for distilling data under centralized learning and other FL training methods that distill data on the clients' side using real data for initializing synthetic data, we randomly initialize synthetic data for each global communication round. This is due to the server's inability to access the local training data of clients. However, distilling data directly using randomly initialized synthetic data tends to result in low-quality synthetic data, subsequently leading to poor performance of the global model. Furthermore, the global model and surrogate model often have different architectures, and data distillation employs the surrogate model, resulting in synthetic data that does not generalize well on the global model. Therefore, during data distillation on the server, we employ a novel technique called optimization via deep generative latents, where the server leverages prior knowledge obtained from a pre-trained generative model to optimize the distillation process.

In our FedDGM, the server possesses a pre-trained generative model denoted as G_g. Optimizing distilled data in the latent space of a generative model can result in better generalization on models with different architectures compared to directly optimizing in pixel space [2]. Hence, in each global communication round t, instead of using randomly initialized synthetic data, the server begins by generating latent vectors $\tilde{Z}_m^{(t)}$ for each client m through some initialization methods. These latent vectors are then employed with the help of G_g to generate the distilled data $\tilde{D}_m^{(t)}$. Because G_g can propagate gradients, it doesn't interfere with the use of $\mathcal{L}_{MTT}$ for optimizing the latent vectors $\tilde{Z}_m^{(t)}$.

We utilized the pre-trained StyleGAN-XL as our generative model. Since StyleGAN-XL has multiple latent spaces in its synthesis network, the choice of which latent space to use for distilling synthetic data is an intriguing question. We denote the distillation space corresponding to the n-th intermediate layer of G_g as Fn space. A smaller n implies that distillation relies more on the prior knowledge of the generative model, resulting in more realistic synthetic data. Conversely, a larger n makes the synthetic data more expressive. In our experiments, we investigate the impact of different latent spaces on both the quality of the distilled data and the training performance.

We can also leverage prior knowledge from StyleGAN to initialize latent vectors, thus improving the quality of the synthetic data. StyleGAN incorporates a mapping network, which is typically a multi-layer perceptron, responsible for mapping input vector $\tilde{W}$ to an intermediate latent space. The mapping network

transforms the latent space into a more expressive style space, offering fine-grained control over styles. Therefore, we can randomly generate vectors $\tilde{W}_m^{(t)}$ for client m and use G_g's mapping network, along with the earlier layers of the synthesis network, to initialize latent vectors $\tilde{Z}_m^{(t)}$ that possess prior knowledge.

3 Theoretical Analysis

In this section, we articulate the argument that, from an asymptotic standpoint, the described procedure becomes equivalent to the theoretical ideal of centralized training on the cumulative heterogeneous dataset. The central server is performing GD with gradually changing data, with iteration,

$$\theta_g^{t+1} = \theta_g^t - s_g \sum_{m \in S_t} \nabla f(\theta_g^t; \tilde{D}_m^t)$$

where s_g is the step-size and f is the loss function. Recall that the dataset distillation optimization problem for each agent is defined by,

$$\begin{aligned} &\min_{\tilde{D}_m^t} f(\theta^*(\tilde{D}_m^t), D_m^t) \\ &\text{s.t.} \quad \theta^*(\tilde{D}_m^t) \in \arg\min f(\theta, \tilde{D}_m^t) \end{aligned}$$

One can consider the federated DD component as an iterative optimization process for this particular problem, for simplicity writing each major iteration as a full gradient update,

$$\tilde{D}_m^{t+1} = \tilde{D}_m^t - s_D \nabla_\theta f(\theta^*(\tilde{D}_m^t), D_m^t) \frac{d\theta^*(\tilde{D}_m^t)}{dD_m^t}$$

where the term $\frac{d\theta^*(\tilde{D}_m^t)}{dD_m^t}$ represents the sensitivity of the training solution with respect to the distilled dataset. Consider a Stochastic Differential Equation (SDE) model of the training:

$$d\theta_g^t = -\frac{1}{|S_t|} \sum_{m \in S_t} \nabla f(\theta_g^t; \tilde{D}_m^t) dt + dW_t,$$

$$d\tilde{D}_t = -\nabla_\theta f(\theta^*(\tilde{D}_m^t), D_m^t) \frac{d\theta^*(\tilde{D}_m^t)}{d\tilde{D}_m^t} + d\tilde{W}_t$$

Note that the second does not depend on θ_g, it thus drifts independently. This is of course a simplification as Local SGD iterations for each client are constrained to be near θ_g. As we do not know how far local iterations are from optimality, and the ultimate intention is to perform dataset distillation as in the solution of the bilevel problem, we study the results with idealistic formalism.

We would need to prove some conditions for this to be the case, but the stationary distribution of this second diffusion can be written by taking the antiderivative of the drift term with respect to $\tilde{D}^t$ and taking the negative exponential

$$\pi(\tilde{D}_m^t | \theta^*) \propto \exp\left\{-\beta_D f(\theta^*(\tilde{D}_m^t), D_m^t)\right\}$$

where $\beta = 1/s$, the step-size of the SGD, which also corresponds to the entropy. As $\beta \to \infty$, the distribution converges to a delta distribution on interpolating the sample, and as $\beta \to 0$, far from mean samples are more likely to be taken. Now, $\theta^*(\tilde{D}_m^t)$ also exhibits a distribution, considering the stationary Gibbs for the neural network,

$$\pi(\theta^*|\tilde{D}_m^t) \propto \exp\left\{-\beta_* f(\theta, \tilde{D}_m^t)\right\}.$$

As this network is overparametrized relative to the data, this distribution has multiple modes and a connected zero loss region in θ space. If we consider iteratively sampling in an alternating fashion from $\pi(\tilde{D}_m^t|\theta^*)$ to $\pi(\theta^*|\tilde{D}_m^t)$, then we know from [15, Theorem 1] that the stationary distributions of $\tilde{D}_m^t$ and θ^* converge to their marginals. Let us now study the properties of the limiting marginal distributions. Detailed balance (as required for ergodicity, or convergence in distribution, see e.g. [17]) requires that,

$$\pi(\theta^*) \exp\left\{-\beta_D f(\theta^*, D_m^t)\right\} = \pi(\tilde{D}_m^t) \exp\left\{-\beta_* f(\theta^*, \tilde{D}_m^t)\right\}$$

Taking $\beta_* \to \infty$ we see that we have convergence in distribution of,

$$\pi(\tilde{D}_m^t) = \exp\left\{-\beta_D f(\theta^*, D_m^t)\right\} \pi(\theta^*)$$

with,

$$\operatorname{supp}\left(\pi(\theta^*)\right) \subseteq \arg\min f(\theta, \tilde{D}_m^t)$$

Consider taking $\beta_D \to \infty$ now, and thus,

$$\operatorname{supp}\left(\pi(\tilde{D}_m^t)\right) \subseteq \left\{\tilde{D}_m^t : \arg\min f(\theta, \tilde{D}_m^t) \cap \arg\min f(\theta, D_m^t) \neq \emptyset\right\}.$$

Finally, the solution of θ_g is based on a deterministic gradient descent, i.e., there is no stochasticity, because the entire dataset $\tilde{D}_m^t$ is small and hence can be loaded in memory. However, $\tilde{D}_m^t$ itself is a stochastic process. Furthermore, as the network is overparametrized relative to the size of $\tilde{D}_m^t$, it has a submanifold of zero loss solutions.

$$\theta_g^* \in \arg\min_\theta \sum_m f(\theta; \tilde{D}_m^t) \tag{3}$$

Now, if there exists θ^* such that $\theta^* = \arg\min f(\theta, D_m^t)$ for all m, then it also solves (3). Otherwise, SGD iterations for (3) involve descent by $\sum_m \nabla_\theta f(\theta; \tilde{D}_m^t)$, which is the same as a linear combination of the m computations to find $\arg\min f(\theta, D_m^t)$ for all m. Thus we have established the equivalency.

4 Experiment

4.1 Experimental Setup

Datasets. We evaluate our method on the CIFAR-10 dataset and five 10-class subsets of ImageNet. Specifically, we use CIFAR-10 (32×32) to evaluate the

performance of FedDGM on low-resolution data and employ subsets of ImageNet (128×128) to evaluate performance on high-resolution data. Previous works introduced some subsets of ImageNet, such as ImageWoof (dogs) [4], ImageMeow (cats) and ImageFruit (fruits) [1], as well as ImageFood (food) and ImageMisc (miscellaneous items) [2]. We provide a detailed list of the categories contained within each subset of ImageNet in the supplementary material.

A distinguishing feature of FL is the non-identical distribution (non-i.i.d.) nature of local training data across clients. To simulate non-i.i.d. settings for M clients, we partition the training data by Dirichlet distribution $\text{Dir}_M(\alpha)$ [13,14], where $\alpha > 0$ controls the degree of non-i.i.d. A smaller α, implies a higher degree of non-i.i.d. By default, we have a total of $M = 10$ clients, with α set to 0.5. Furthermore, in Section 4.2, we will investigate the performance of FedDGM under various non-i.i.d. scenarios.

Compared Methods. To provide a more comprehensive understanding of the efficiency of FedDGM, we select three FL training methods based on aggregation: FedAvg [12], FedProx [10], and FedNova [22], as well as a method based on data distillation, FedDM [23].

FL Training Settings. We have a total of $M = 10$ clients. For each global training round, the server selects all clients for aggregation. For the CIFAR-10 dataset, the training batch size is 256, and for subsets of ImageNet, it is 32. By default, each client conducts local training for $T_l = 20$ epochs on its own local training data, and we tune T_l within the range $[5, 10, 20, 30]$. On the server side, for each global training round, the server updates the latent vector for $T_d = 100$ iterations, and for each distillation step, the server updates the student network for $T_s = 20$ epochs. The number of images per class (IPC) is 10. The images are distilled into F5 space (the 5th layer of StyleGAN-XL) for CIFAR-10 and F12 for subsets of ImageNet. After distilling the data, the server trains the global model for $T_g = 1000$ epochs with a training batch size of 256 for CIFAR-10 and 32 for subsets of ImageNet. In particular, we tune the layer of StyleGAN-XL in $[0, 3, 4, 5, 6, 9]$ and IPC in $[1, 5, 10, 20]$ for CIFAR-10.

To ensure a fair comparison, we maintain the same local training batch size as FedDGM for baseline methods. For FedAvg, FedProx, and FedNova, we also use the same local training epochs of $T_l = 20$ as in FedDGM. For FedDM, we adopt the same IPC=10 and train the global model for $T_g = 1000$ epochs with the same training batch size, using the synthetic dataset. All the experiments are run for three times, and we report the mean validation accuracy $\pm$ 1 standard deviation for each evaluation case.

Network Architectures. To demonstrate the cross-architecture generalization of FedDGM, for data distillation-based FL methods (i.e., FedDM and FedDGM), we set the local models of clients as a 5-layer ConvNet [6] and set the global model to be some larger models. Specifically, for CIFAR-10, we evaluate global models such as ConvNet, ResNet18 [8], VGG11 [16], and ViT [3], while for subsets of ImageNet, we consider global models like ConvNet, ResNet18, and ResNet34 [8]. For the remaining baseline methods, the structure of clients' models remains consistent with the server's global model.

Table 1: Using data distillation for FL training significantly enhances the performance of global models in extremely non-i.i.d. scenarios. As data heterogeneity increases, FedDGM exhibits a significant advantage over all baseline methods. Impact of different values of α on the CIFAR-10 dataset. Each column represents a specific architecture of the global model.

α	Algorithm	ConvNet	ResNet18	VGG11	ViT	Average
0.9	FedAvg	$70.1_{\pm 0.5}$	$62.1_{\pm 0.7}$	$66.2_{\pm 0.6}$	$54.8_{\pm 0.2}$	$63.3_{\pm 0.5}$
	FedProx	$70.7_{\pm 0.3}$	$63.0_{\pm 0.6}$	$66.2_{\pm 0.5}$	$55.0_{\pm 0.8}$	$63.7_{\pm 0.5}$
	FedNova	$70.6_{\pm 0.7}$	$62.1_{\pm 1.0}$	$66.6_{\pm 0.6}$	$54.5_{\pm 0.4}$	$63.5_{\pm 0.7}$
	FedDM	$70.2_{\pm 0.2}$	$73.4_{\pm 0.2}$	$68.7_{\pm 0.4}$	$51.6_{\pm 0.1}$	$66.0_{\pm 0.3}$
	FedDGM	$70.8_{\pm 0.5}$	$73.8_{\pm 0.2}$	$70.1_{\pm 0.4}$	$55.3_{\pm 0.3}$	$67.5_{\pm 0.4}$
0.5	FedAvg	$69.4_{\pm 0.1}$	$61.1_{\pm 0.8}$	$64.4_{\pm 0.7}$	$53.5_{\pm 0.2}$	$62.1_{\pm 0.4}$
	FedProx	$69.5_{\pm 0.5}$	$62.3_{\pm 0.5}$	$65.3_{\pm 0.4}$	$53.4_{\pm 0.3}$	$62.6_{\pm 0.4}$
	FedNova	$69.5_{\pm 0.4}$	$60.9_{\pm 0.7}$	$64.8_{\pm 0.6}$	$54.3_{\pm 1.2}$	$62.4_{\pm 0.7}$
	FedDM	$68.9_{\pm 0.3}$	$72.9_{\pm 0.1}$	$68.0_{\pm 0.3}$	$51.0_{\pm 0.6}$	$65.2_{\pm 0.3}$
	FedDGM	$70.8_{\pm 0.9}$	$73.7_{\pm 0.5}$	$69.8_{\pm 0.3}$	$55.9_{\pm 0.7}$	$67.5_{\pm 0.6}$
0.1	FedAvg	$45.3_{\pm 3.2}$	$38.0_{\pm 0.9}$	$41.1_{\pm 2.2}$	$43.6_{\pm 0.3}$	$42.0_{\pm 1.7}$
	FedProx	$45.2_{\pm 2.2}$	$40.2_{\pm 0.4}$	$40.3_{\pm 1.9}$	$45.6_{\pm 0.6}$	$42.8_{\pm 1.3}$
	FedNova	$42.9_{\pm 1.7}$	$35.8_{\pm 0.6}$	$35.8_{\pm 1.7}$	$42.1_{\pm 0.7}$	$39.2_{\pm 1.2}$
	FedDM	$62.0_{\pm 0.3}$	$63.6_{\pm 0.3}$	$61.4_{\pm 0.2}$	$45.0_{\pm 0.2}$	$58.0_{\pm 0.3}$
	FedDGM	$68.9_{\pm 0.6}$	$71.6_{\pm 0.2}$	$68.1_{\pm 0.2}$	$54.2_{\pm 0.7}$	$65.7_{\pm 0.4}$
0.01	FedAvg	$14.3_{\pm 2.7}$	$16.9_{\pm 2.6}$	$19.3_{\pm 2.9}$	$31.1_{\pm 1.7}$	$20.4_{\pm 2.5}$
	FedProx	$18.3_{\pm 2.5}$	$16.7_{\pm 2.5}$	$14.8_{\pm 3.4}$	$28.8_{\pm 3.3}$	$19.7_{\pm 2.9}$
	FedNova	$10.1_{\pm 0.1}$	$16.4_{\pm 1.8}$	$13.3_{\pm 2.6}$	$13.3_{\pm 2.4}$	$13.3_{\pm 1.7}$
	FedDM	$47.8_{\pm 0.4}$	$48.1_{\pm 0.7}$	$48.9_{\pm 0.5}$	$36.3_{\pm 0.4}$	$45.3_{\pm 0.5}$
	FedDGM	$66.1_{\pm 0.7}$	$69.5_{\pm 0.1}$	$66.4_{\pm 0.3}$	$51.5_{\pm 0.2}$	$63.4_{\pm 0.3}$

Table 2: The benefit of FedDGM is particularly pronounced when the data distributions are extremely non-i.i.d. Impact of different α on the average accuracy across global models with different architectures on ImageNet.

Dataset	Algorithm	ConvNet	ResNet18	ResNet34	Average
ImMeow	FedAvg	$48.3_{\pm 1.5}$	$34.8_{\pm 2.1}$	$40.7_{\pm 1.3}$	$41.3_{\pm 1.6}$
	FedProx	$49.7_{\pm 0.4}$	$38.5_{\pm 2.0}$	$44.3_{\pm 1.3}$	$44.2_{\pm 1.2}$
	FedNova	$50.6_{\pm 0.3}$	$34.0_{\pm 0.6}$	$38.3_{\pm 1.5}$	$41.0_{\pm 0.8}$
	FedDM	$51.7_{\pm 1.3}$	$53.3_{\pm 0.7}$	$61.2_{\pm 1.4}$	$55.4_{\pm 1.1}$
	FedDGM	$57.8_{\pm 1.2}$	$56.2_{\pm 1.9}$	$65.1_{\pm 0.6}$	$59.7_{\pm 1.2}$
ImWoof	FedAvg	$38.5_{\pm 1.7}$	$25.7_{\pm 1.5}$	$29.5_{\pm 2.1}$	$31.2_{\pm 1.8}$
	FedProx	$40.5_{\pm 1.2}$	$25.3_{\pm 0.4}$	$29.7_{\pm 2.0}$	$31.8_{\pm 1.2}$
	FedNova	$40.7_{\pm 1.2}$	$26.5_{\pm 1.6}$	$27.0_{\pm 1.6}$	$31.4_{\pm 1.4}$
	FedDM	$47.0_{\pm 1.8}$	$52.7_{\pm 0.9}$	$60.3_{\pm 0.2}$	$53.3_{\pm 1.0}$
	FedDGM	$57.4_{\pm 2.6}$	$62.3_{\pm 1.2}$	$72.7_{\pm 0.6}$	$64.1_{\pm 1.5}$
ImFruit	FedAvg	$42.7_{\pm 1.9}$	$36.2_{\pm 1.4}$	$39.5_{\pm 2.6}$	$39.5_{\pm 2.0}$
	FedProx	$43.9_{\pm 0.2}$	$34.7_{\pm 1.8}$	$42.4_{\pm 3.3}$	$40.4_{\pm 1.8}$
	FedNova	$45.7_{\pm 1.2}$	$35.0_{\pm 0.6}$	$39.1_{\pm 1.0}$	$40.0_{\pm 1.0}$
	FedDM	$45.7_{\pm 1.7}$	$49.7_{\pm 0.2}$	$53.7_{\pm 1.5}$	$49.7_{\pm 1.1}$
	FedDGM	$54.5_{\pm 1.5}$	$55.1_{\pm 0.7}$	$58.9_{\pm 0.4}$	$56.2_{\pm 0.8}$
ImFood	FedAvg	$45.1_{\pm 0.7}$	$32.8_{\pm 1.3}$	$39.3_{\pm 1.2}$	$39.1_{\pm 1.1}$
	FedProx	$46.3_{\pm 2.8}$	$33.4_{\pm 1.9}$	$36.7_{\pm 0.7}$	$38.8_{\pm 1.8}$
	FedNova	$46.7_{\pm 2.3}$	$32.7_{\pm 1.6}$	$38.4_{\pm 1.1}$	$39.3_{\pm 1.7}$
	FedDM	$50.6_{\pm 1.4}$	$50.7_{\pm 0.9}$	$59.8_{\pm 0.4}$	$53.7_{\pm 0.9}$
	FedDGM	$54.5_{\pm 2.2}$	$52.9_{\pm 1.3}$	$60.0_{\pm 0.8}$	$55.8_{\pm 1.5}$
ImMisc	FedAvg	$56.8_{\pm 0.9}$	$42.4_{\pm 0.7}$	$45.5_{\pm 3.2}$	$48.2_{\pm 1.6}$
	FedProx	$59.1_{\pm 1.9}$	$44.7_{\pm 1.1}$	$46.1_{\pm 1.8}$	$50.0_{\pm 1.6}$
	FedNova	$58.8_{\pm 1.0}$	$42.7_{\pm 0.4}$	$48.3_{\pm 0.2}$	$50.0_{\pm 0.5}$
	FedDM	$64.3_{\pm 1.1}$	$63.9_{\pm 0.6}$	$72.2_{\pm 0.3}$	$66.8_{\pm 0.7}$
	FedDGM	$65.6_{\pm 0.1}$	$66.6_{\pm 2.1}$	$72.5_{\pm 0.4}$	$68.2_{\pm 1.1}$

4.2 Performance and Convergence Rate

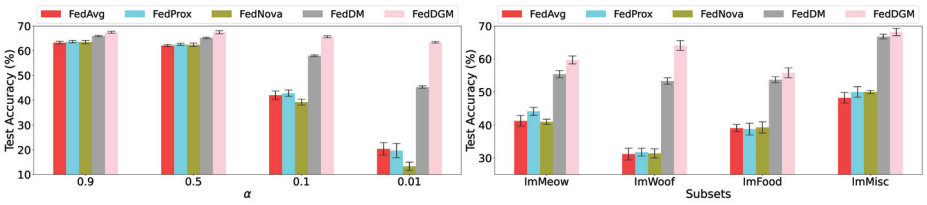
Performance Across Different Datasets. We investigate the performance of FedDGM on both low-resolution data using CIFAR-10 and high-resolution data using ImageNet subsets across various non-i.i.d. degrees (different α values) and architectures. As shown in Table 1, using data distillation for FL training significantly enhances the performance of global models in extremely non-i.i.d. scenarios (e.g., $\alpha = 0.01$). More importantly, our FedDGM consistently outperforms the baseline methods on CIFAR-10 across different global model architectures. Particularly, as data heterogeneity increases (i.e., $\alpha = 0.9 \rightarrow 0.01$), FedDGM exhibits a significant advantage over the baseline methods. This could be attributed to the way we distill data, which compensates for the challenges in training posed by high data heterogeneity. When client i lacks data in class j, the global model parameters received by client i already contain information about data in class j from other clients, which allows the server to use client i's local model parameters to distill data in class j with relatively good quality, as depicted in Fig. 3. To facilitate a quick understanding of the key findings in these extensive tables, we have presented the average accuracy, represented by the last column in Table 1, in Fig. 4a. It is evident that, with increased heterogeneity (i.e., smaller α values), the superiority of FedDGM becomes more

pronounced. Unlike other baseline methods, which experience a substantial decline in performance as α decreases, FedDGM maintains a relatively stable level of performance. We also conducted experiments on a high resolution dataset i.e., one of the subsets of ImageNet, ImageFruit under different data heterogeneity, and the results are shown in the supplementary material.

In the case of high-resolution data, according to results in Table 2, utilizing data distillation for FL training significantly enhances the performance of models on high-resolution data, and FedDGM always outperforms baseline methods on all five ImageNet subsets when the server possesses global models with different architectures, such as ConvNet, ResNet18, and ResNet34. Specifically, our algorithm significantly outperforms FedDM and other aggregation-based methods on ImageMeow, ImageWoof, and ImageFruit. For example, FedDGM achieves 57.4±0.6% on ImageWoof when the global model is ConvNet, while the best-performing baseline method, FedDM, only achieves 47.0±1.8%. This demonstrates a substantial improvement in our algorithm for high-resolution data. Also on ImageWoof, our algorithm achieves 72.7±0.6% on ResNet34, exceeding the next best method by over 10% improvement. *This significant improvement highlights its strong generalization capacity across diverse model architectures.* To facilitate ease of comparison, Fig. 4b depicts the average results across architectures, as presented in the last column of Table 2.

Fig. 3: Illustrations of distilled data examples labeled as "truck". These examples are distilled using a client's local surrogate model whose local data does not contain "truck".

Convergence Rate. Fig. 5 illustrates the relationship between test accuracy and communication rounds for FedDGM and baseline methods on CIFAR-10 across different model architectures. The data partitioning distribution is $\text{Dir}_{10}(0.5)$. First, we observe that FedDGM converges significantly faster than baseline methods on ConvNet, ResNet, and VGG11. It also converges faster than FedDM on ViT and has a comparable convergence speed with other baseline methods. In addition, we observe that data distillation-based methods consistently exhibit a better convergence rate compared to aggregation-based methods, except when the global model is ViT. *This is because the global model is trained directly on synthetic data, rather than being obtained through the aggregation of local model parameters or model updates.* The prior knowledge from deep generative models contributes to FedDGM having a better convergence rate compared to FedDM.



(a) Different levels of heterogeneity **(b)** Different ImageNet subsets

Fig. 4: (a): The benefit of FedDGM is particularly pronounced when the data distributions are extremely non-i.i.d. Impact of different α on the average accuracy across global models with different architectures on CIFAR10. **(b): FedDGM is architecture-agnostic.** Average accuracy across global models with different architectures on various ImageNet subsets. FedDGM outperforms all baseline methods across diverse architectures.

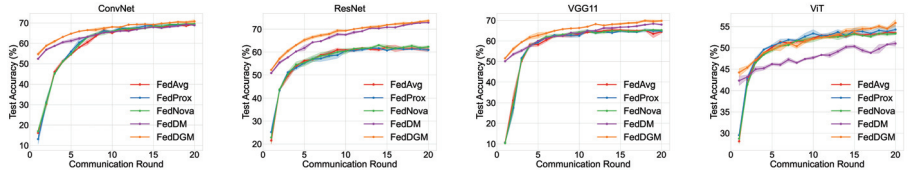


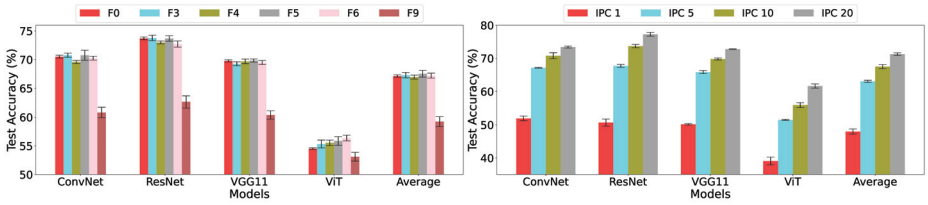
Fig. 5: FedDGM Demonstrates better performance and faster convergence rates compared to other baselines. The relationship between test accuracy and communication rounds on CIFAR-10.

4.3 Impact of Hyperparameters

Fig. 6: Illustrations of distilled data generated from different latent spaces. Each row corresponds to a distinct latent space, arranged from top to bottom: F0, F3, F4, F5, F6, and F9. Employing latent spaces associated with earlier layers generally results in more realistic distilled data (i.e., those images at the top rows).

Impact of Different Latent Spaces. As mentioned earlier in Section 2.3, data distilled from different latent spaces may exhibit noticeable differences. Keeping the data partitioning fixed, we randomly select a client and show the synthetic data belonging to the "dog" class generated through data distillation using six different latent spaces, as in Fig. 6. We can observe that using the latent space corresponding to an earlier layer tends to make the distilled data more realistic.

To investigate which latent space results in better data generalization, we experimented with these six different latent spaces of StyleGAN-XL on CIFAR-10, and the corresponding global model performance is shown in Fig. 7a. We observe that using the F5 space results in the best generalization of distilled synthetic data across different architectures. Additionally, except for the F9 space, the performance difference is relatively small when parameterizing the synthetic data in other latent spaces and consistently outperforms baseline methods.



(a) Different latent spaces **(b)** Impact of Different IPCs

Fig. 7: (a): Impact of distilling data in different latent spaces. Except for the F9 space, the performance difference among the other latent spaces is minimal. By default, we utilize the F5 space. **(b):** As the value of IPC increases, the test accuracy across different global model architectures significantly improves. Impact of different IPCs. By default, we set IPC to 10.

Impact of Different IPCs. To analyze the impact of different IPC values on FedDGM, we conduct experiments to test the performance of different global model architectures with IPC values of 1, 5, 10, and 20 on CIFAR-10. The results are presented in Fig. 7b. As the IPC value increases, the test accuracy of global models with different architectures significantly improves. In particular, when IPC is set to 5, the performance of FedDGM is comparable to aggregation-based baseline methods. When IPC is greater than or equal to 10, FedDGM outperforms all baseline methods. Considering the trade-off between global model performance and computational cost on the server, we set the IPC value to 10.

5 Conclusion

This paper introduces a server-side federated learning (FL) framework that leverages pre-trained deep generative models for efficient and privacy-enhanced training. This approach reduces computational demands on local devices, enabling smaller local models and facilitating the training of a larger global model on the server. Theoretical analysis shows an asymptotic resemblance to centralized training on a heterogeneous dataset. Empirical results demonstrate up to a 40% accuracy improvement over non-dataset-distillation techniques in highly heterogeneous FL contexts, outperforming existing methods by 18%. Notably, our framework achieves around a 10% performance increase on high-resolution image datasets and exhibits faster convergence.

Acknowledgements

This work was supported in part by NSF-2112562 and ARO W911NF-23-2-0224. Also, VK would like to acknowledge support to the Czech National Science Foundation Project 24-11664S.

References

1. Cazenavette, G., Wang, T., Torralba, A., Efros, A.A., Zhu, J.: Dataset distillation by matching training trajectories. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. pp. 10708–10717. IEEE (2022)
2. Cazenavette, G., Wang, T., Torralba, A., Efros, A.A., Zhu, J.Y.: Generalizing dataset distillation via deep generative prior (2023)
3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
4. Fastai: Fastai/imagenette: A smaller subset of 10 easily classified classes from imagenet, and a little more french.
5. Ghosh, A., Chung, J., Yin, D., Ramchandran, K.: An efficient framework for clustered federated learning. arXiv preprint arXiv:2006.04088 (2020)
6. Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4367–4375 (2018)
7. Gu, J., Vahidian, S., Kungurtsev, V., Wang, H., Jiang, W., You, Y., Chen, Y.: Efficient dataset distillation via minimax diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15793–15803 (2024)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
9. Khalid, U., Iqbal, H., Vahidian, S., Hua, J., Chen, C.: CEFHRI: A communication efficient federated learning framework for recognizing industrial human-robot interaction. CoRR **abs/2308.14965** (2023). https://doi.org/10.48550/ARXIV.2308.14965, https://doi.org/10.48550/arXiv.2308.14965
10. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018)
11. Lu, Y., Gu, J., Chen, X., Vahidian, S., Xuan, Q.: Exploring the impact of dataset bias on dataset distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7656–7663 (2024)
12. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
13. Morafah, M., Vahidian, S., Wang, W., Lin, B.: Flis: Clustered federated learning via inference similarity for non-iid data distribution. IEEE Open Journal of the Computer Society **4**, 109–120 (2023). https://doi.org/10.1109/OJCS.2023.3262203
14. Morafah, M., Vahidian, S., Wang, W., Lin, B.: Flis: Clustered federated learning via inference similarity for non-iid data distribution. IEEE Open Journal of the Computer Society **4**, 109–120 (2023). https://doi.org/10.1109/OJCS.2023.3262203

15. Schervish, M.J., Carlin, B.P.: On the convergence of successive substitution sampling. Journal of Computational and Graphical statistics **1**(2), 111–127 (1992)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
17. Stroock, D.W.: An introduction to Markov processes, vol. 230. Springer Science & Business Media (2013)
18. Vahidian, S., Kadaveru, S., Baek, W., Wang, W., Kungurtsev, V., Chen, C., Shah, M., Lin, B.: When do curricula work in federated learning? vol. abs/2212.12712 (2022). https://doi.org/10.48550/arXiv.2212.12712, https://doi.org/10.48550/arXiv.2212.12712
19. Vahidian, S., Morafah, M., Lin, B.: Personalized federated learning by structured and unstructured pruning under data heterogeneity. In: 2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW). pp. 27–34 (2021). https://doi.org/10.1109/ICDCSW53096.2021.00012
20. Vahidian, S., Morafah, M., Wang, W., Kungurtsev, V., Chen, C., Shah, M., Lin, B.: Efficient distribution similarity identification in clustered federated learning via principal angles between client data subspaces. https://arxiv.org/abs/2209.10526 (2022)
21. Vahidian, S., Wang, M., Gu, J., Kungurtsev, V., Jiang, W., Chen, Y.: Group distributionally robust dataset distillation with risk minimization. arXiv preprint arXiv:2402.04676 (2024)
22. Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V.: Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization (Jul 2020). https://doi.org/10.48550/arXiv.2007.07481, arXiv:2007.07481 [cs, stat]
23. Xiong, Y., Wang, R., Cheng, M., Yu, F., Hsieh, C.J.: Feddm: Iterative distribution matching for communication-efficient federated learning (2022)
24. Zhao, B., Bilen, H.: Dataset condensation with distribution matching. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 6514–6523 (2023)
25. Zhao, B., Mopuri, K.R., Bilen, H.: Dataset condensation with gradient matching (2021)