FEDERATED LEARNING IN NON-IID SETTINGS AIDED BY DIFFERENTIALLY PRIVATE SYNTHETIC DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning (FL) is a privacy-promoting framework that enables potentially large number of clients to collaboratively train machine learning models. In an FL system, a server coordinates the collaboration by collecting and aggregating clients' model updates while the clients' data remains local and private. A major challenge in federated learning arises when the local data is non-iid – the setting in which performance of the learned global model may deteriorate significantly compared to the scenario where the data is identically distributed across the clients. In this paper we propose FedDPMS (Federated Differentially Private Means Sharing), an FL algorithm in which clients augment local datasets with data synthesized using differentially private information collected and communicated by a trusted server. In particular, the server matches the pairs of clients having complementary local datasets and facilitates differentially-private sharing of the means of latent data representations; the clients then deploy variational auto-encoders to enrich their datasets and thus ameliorate the effects of non-iid data distribution. Our experiments on deep image classification tasks demonstrate that FedDPMS outperforms competing state-of-the-art FL methods specifically developed to address the challenge of federated learning on non-iid data.

1 INTRODUCTION

The need for massive amounts of high-quality training data in deep learning (e.g., ImageNet (Deng et al., 2009), COCO (Lin et al., 2014)) creates a major challenge in settings where data is distributed across a potentially large number of users' devices; in particular, constraints on communication resources and users' privacy concerns often prohibit gathering local data and training models at a central location. In response, *federated learning* (FL) where users collaboratively train a global model without revealing personal data has emerged as a privacy-promoting and communication-efficient distributed alternative to centralized learning (Kairouz et al., 2019; Yang et al., 2019; Li et al., 2020b; Augenstein et al., 2019b).

In FL systems, a subset of users' devices updates a global model by training on local data; a server coordinates the training process by selecting the users, collecting the updates, and aggregating them to form a new global model. The original FL algorithm, *Federated Averaging* (FedAvg) (McMahan et al., 2017), chooses users at random and updates the global model by averaging the users' updates; the convergence analysis provided in (McMahan et al., 2017) assumes that local datasets are independent and identically distributed (i.i.d.). Recently, improving performance of FL systems that deploy sophisticated ML models in a variety of practical scenarios has received considerable amount of attention (Li et al., 2018; Karimireddy et al., 2019; Zhang et al., 2020; Reddi et al., 2020; Li et al., 2020a; Hamer et al., 2020; Rasouli et al., 2020; He et al., 2021).

A major challenge in FL presents when the decentralized data is heterogeneous. Indeed, it is unrealistic to expect that the users participating in a FL system train on data generated from identical distributions – instead, distributions of labels will likely differ between the participating devices. This is particularly pronounced in setting where the training data is limited, possibly to the extent that only a subset of classes is present in the users' datasets. Since in such scenarios performance of FedAvg may significantly deteriorate, a number of approaches for learning from heterogeneous distributed data has recently been proposed (Li et al., 2018; Karimireddy et al., 2019; Li et al., 2021). Data heterogeneity in FL systems deploying deep learning networks and training on complex datasets was studied in (Li et al., 2021) where a regularization term is introduced in order to impose contrastive learning on local updates, effectively aligning those updates with the global objective. However, as

we illustrate in our experiments (see Section 4), this approach fails to perform well on imbalanced data partitions. Difficulties in learning a global model under data heterogeneity have also motivated various clustering and personalization approaches to FL including Model-Agnostic Meta-Learning (MAML) and its variants that rely on client clustering (Fraboni et al., 2021; Standley et al., 2020; Fifty et al., 2021; Ghosh et al., 2020; Sattler et al., 2020; Xie et al., 2021).

In this work we address the challenge of data heterogeneity and scarcity in learning a global model by enabling clients to locally synthesize data using parameters acquired and shared in a privacy-preserving manner. In particular, the proposed *federated differentially-private means sharing* (FedDPMS) framework allows each client to locally generate synthetic data by relying on means of latent data representations exchanged in differentially-private (DP) manner between clients matched by a trusted server. The server coordinating the training process is assumed to be provided partial information about the clients' data distributions (in particular, indices of the most abundant and the least abundant classes), and is therefore capable of matching the pairs of clients whose local datasets are complementary. Unlike the existing data augmentation approaches to FL which utilize Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Triastcyn & Faltings, 2020a; Augenstein et al., 2019a), our proposed framework relies on simple Variational Auto-Encoders (VAEs) well-suited to the classification tasks of interest. Privacy of sharing the (noisy) means of latent representations is quantified by utilizing the differential privacy mechanism. To test the proposed FedDPMS framework, we conduct extensive numerical studies on image classification tasks using Fashion-MNIST (Xiao et al., 2017), CIFAR-10 and CIFAR-100 datasets. In those experiments FedDPMS outperforms state-of-the-art approaches, particularly in settings where the local data comes from diverse distributions.

2 Related Work

2.1 Federated Learning on Heterogeneous Data

A training round of the FedAvg algorithm (McMahan et al., 2017) consists of four steps. First, the server broadcasts a global model and selects a subset of clients that will perform model update. Second, the selected clients train on local data, in parallel, starting from the current global model. Third, the clients send the model updates or the computed gradients to the server. Finally, the server aggregates the received updates by averaging them to form a new global model, and starts the next round of training. As discussed in the previous section, performance of FedAvg deteriorates when the data at different clients is non-iid.

Approaches to FL that aim to overcome challenges of training a global model on heterogeneous data can be broadly organized in two categories. The first category includes techniques that attempt to improve the model aggregation step performed by the server. Examples include PFNM (Yurochkin et al., 2019), a Bayesian non-parametric approach for extracting layers from local models and using them to update the corresponding layers in the global model. While PFNM targets relatively simple architectures, FedMA (Wang et al., 2020a) takes a step further and extends the same ideas to CNNs and LSTMs. In (Wang et al., 2020b), the authors present a framework for the analysis of convergence of FL on heterogeneous data, along with a normalized averaging method, FedNova (Wang et al., 2020b), that aims to eliminate objective inconsistencies (i.e., prevent convergence of the global model to a stationary point of the mismatched objectives) caused by naive aggregation of local models. The second category of methods for FL on heterogeneous distributed data is focused on reducing the drift in local training. To this end, FedProx (Li et al., 2018) introduces a proximal term to the learning objective of each client with the goal of making local training aligned with the global objective. The proximal term is defined as the l_2-norm of the difference between the weights of the global model and the local model. SCAFFOLD (Karimireddy et al., 2019) utilizes predictive variance reduction (Johnson & Zhang, 2013) to introduce control variates and correct local updates. The above two studies were verified in experiments on MNIST and EMINIST with multinomial logistic regression and fully connected two-layer networks. Recently, (Li et al., 2021) tested FedProx and SCAFFOLD on more challenging tasks that involve deep learning models, showing that those two methods unfortunately offer little to no advantage over FedAvg in the considered scenarios. As an alternative, (Li et al., 2021) propose the Moon algorithm motivated by the local/global model proximity idea of FedProx but instead of the l_2-norm term, the proximity is imposed via a contrastive term in the objective of local training. The contrastive term is defined using a similarity function expressing preference that the output categorical vectors of the current global and local models are close, while the previous and current local models are distant. Another study, FedDyn (Acar et al., 2020), proposes a dynamic regularizer to promote convergence of the local loss to a stationary

point of the global loss. Building on top of SCAFFOLD, FedDC (Gao et al., 2022) introduces an auxiliary local drift variable which serves as a dynamic regularizer helping narrow the gap between local models and the global model. While the above approaches utilize different loss functions, ultimately they all deploy the same strategy of introducing a regularization term to prevent from overfitting in local training. In this paper, we focus on the problem of learning a (single) global model in a federated system that consists of users with heterogeneous data; study of personalized federated learning (pFL) (Tan et al., 2022) is thus beyond the scope of this work.

2.2 Domain Generalization in Federated Learning

Recall that the drift of local model updates is caused by differences between local class distributions; in the most severe case, clients' local data may be missing (different) classes. A set of methods complementary to the techniques in Section 2.1 relies on domain generalization, a method which aims to improve generalization ability of models by training them on data from multiple source domains. Such techniques include FedDG (Liu et al., 2021), a method that allows each client to transfer its amplitude spectrum, decomposed from raw data, to a bank at the server. The server shares the collected amplitude spectrum with all clients, enabling them to synthesize new distribution via interpolation and helping improve local training. Another study, (Hao et al., 2021), employs zero-shot data augmentation and relies on the statistics of the batch normalization (BN) layers to reduce the variance of test accuracy. FedMix (Yoon et al., 2021) presents a framework for sharing clients' averaged local data via Mixup (Zhang et al., 2017); privacy of clients in FedMix may still be compromised since the only effort to protect it is based on averaging raw data. FedDA (Wen et al., 2022) utilizes the attention mechanism (Vaswani et al., 2017) to enable the server to aggregate local per-label knowledge and create global per-label knowledge, ultimately enabling data augmentation via conditional variational autoencoder (CVAE) (Sohn et al., 2015). However, aggregation of per-label knowledge in highly non-iid setting (e.g., only a few classes are present in the local dataset) presents the same challenge: local per-label knowledge is drifting from the expectation of the global per-label knowledge. FedDPMS aims to address this problem by matching pairs of client and generating complementary data using shared mean of latent representations.

3 Federated Differentially Private Mean Sharing

3.1 Overview of the Proposed Scheme

Both existing approaches to dealing with data heterogeneity – improving model aggregation at the server and reducing model drift in local training – struggle when the differences between local data distributions are significant (e.g., the clients completely missing some class labels). To this end, in this paper we propose an alternative framework that aims to provide privacy-preserving domain generalization by enhancing and balancing local datasets. In particular, FedDPMS relies on sharing differentially-private information needed to generate representative synthetic data – specifically, the clients share noisy versions of averaged latent representations that their encoders extract from local data. Since the proposed framework facilitates learning in non-iid settings by enriching the diversity of clients' data, it does not compete with the prior work in Section 2 – instead, FedDPMS is complementary to and may potentially be combined with the existing methods for FL on non-iid distributed data. In the upcoming discussions, for illustration purposes we will repeatedly invoke the image classification task as a use case.

3.2 Network Architecture and Training Objective

The global model and local models in an FL system typically share the same network structure. In our proposed framework, the model consists of three components: an encoder, a decoder and a classifier. The encoder learns data representation utilizing convolutional layers; in particular, data samples are encoded into a mean μ and a variance σ_z^2, and represented by a latent variable $z \sim \mathcal{N}(\mu, \sigma_z^2)$. Given the encoder's architecture, a symmetric decoder is designed by relying on upsampling and transposed convolutions. The final component of the model is a classifier that consists of multiple linear layers and maps the latent variable z into a categorical vector p which quantifies the likelihood of each class. The described architecture and the principle of Variational Auto-Encoder (VAE) are illustrated in Appendix A.1 and A.6.

The training starts with a preliminary step which is focused on minimizing a loss that consists of three components: ℓ_1, the cross-entropy between a prediction and the ground truth; ℓ_2, the Kullback-Leibler divergence between the prior distribution $p(z)$ and the approximated distribution $q(z)$ (parametrized by mean and variance in the Gaussian case); and ℓ_3, the reconstruction loss

defined as the mean-square error between the original and reconstructed data points. Formally,

$$\ell_1 = \text{CEloss}\left(F_{w_i^t}(x), y\right),\ \ell_2 = \text{KLDloss}\left(\mu, \sigma_z\right), \ell_3 = \text{MSEloss}\left(\hat{x}, x\right),$$

where $F_{w_i^t}(\cdot)$ denotes the entire network with parameters w_i^t, x and $\hat{x}$ are the input and its reconstruction, respectively, and y denotes the ground-truth. The local training loss is formed as

$$\ell = \ell_1 + \lambda(\ell_2 + \ell_3),$$

where λ is a hyper-parameter. In our experiments, we fine-tune the hyper-parameter and report the performance with the best λ. The training then enters the secondary phase, where we train only the encoder and classifier (i.e., the optimization focuses on ℓ_1). This is elaborated next.

3.3 The Proposed Learning Scheme – FedDPMS

As stated in Section 2, the training of FedDPMS is organized in two stages. I. *Preliminary training.* Each client locally trains a VAE and sends the parameters of the resulting encoder and classifier to the server; the server aggregates the received information and shares the resulting global model (encoder and classifier) with the clients (except in the last round of training, please see Section 3.3.1). II. *Secondary training.* There are four steps in each round t of the secondary training. (1) **Client matching**: The server is given partial information about the selected clients' data distributions (i.e., indices of the most and the least abundant classes) but has no access to raw data which remains private; based on the received information, for each client the server identifies the most informative latent representation statistics (specifically, the most informative means of latent representations); (2) **Data synthesis**: The server communicates (noisy) means of latent representations and the global decoder to the clients in need of certain data classes; using the global decoder and the information received from the server, these clients synthesize samples of locally missing or underrepresented classes and incorporate the synthesized samples into their local datasets. (3) **Model training**: the selected clients locally train encoders and classifiers using augmented datasets; the updated models are collected and aggregated by the server. (4) **DPMS**: The selected clients who prior to the current training round have not shared latent representation information upload the indices of their top-n most abundant classes and differentially-private means of latent data representation to the server. The training procedure outlined in this section is formalized as Algorithm 1. In the next two subsections, we provide further informative details of the preliminary and secondary training.

Algorithm 1 (FedDPMS)

Input:
Local datasets from K clients, $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots \mathcal{D}_K\}$; the number of participating clients k each round; the number of preliminary training rounds T_p; the number of global epoches T.

Output:
The final global model w^T

Server executes:
randomly initialize (w_e^0, w_c^0), $\mathcal{A} = \emptyset, \mathcal{B} = \emptyset, \mathcal{C} = \emptyset, \mathcal{Z} = \emptyset$
for $t = 0, \ldots, T_p - 1$ **do**
  $\mathcal{S}_t \leftarrow k$ clients selected at random
  $(w_e^{t+1}, w_c^{t+1}) \leftarrow$ **PreTrain**$(t, w_e^t, w_c^t, \mathcal{S}_t)$
end for
for $t = T_p, \ldots, T - 1$ **do**
  $\mathcal{S}_t \leftarrow k$ clients selected at random
  $(w_e^{t+1}, w_c^{t+1}) \leftarrow$
  SecTrain$(w_e^t, w_c^t, \mathcal{S}_t, \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{Z})$
end for
return $w^T = (w_e^T, w_c^T)$

3.3.1 Preliminary training

During preliminary training, each client trains a VAE capable of compressing/reconstructing data and recognizing data labels. To curtail communication costs, only the encoder and classifier's weights are uploaded to the server (except in the last round of the preliminary training). Formally, at the beginning of global round t, the server sends the global encoder and classifier models $\{w_e^t, w_c^t\}$ to the clients. The clients use $\{w_e^t, w_c^t\}$ to initialize training VAEs on local data (decoder is not initialized), and run several epochs of the stochastic gradient descent; by the end of those epochs, the i^{th} client has obtained updates $\{w_{i,e}^t, w_{i,c}^t, w_{i,d}^t\}$. Finally, the server aggregates the updates $\{w_{i,e}^t, w_{i,c}^t\}$ into the latest global model $\{w_e^{t+1}, w_c^{t+1}\}$. In the last round of preliminary training, each selected client is requested to upload the full update $\{w_{i,e}^t, w_{i,c}^t, w_{i,d}^t\}$ so that the server can aggregate global decoder $\mathbf{w_d}$; at that point, local decoders may be deleted to free up memory.

3.3.2 SECONDARY TRAINING

Recall that in this stage clients update the encoder and classifier by optimizing the cross-entropy between a prediction and the ground truth while disregarding the divergence and reconstruction loss; these updates are formed by training on augmented datasets. For the sake of computation and communication efficiency, each selected client shares its means of latent data representations and augments the local dataset only once, regardless of how many times the client is selected by the server; as our results demonstrate, this is sufficient to improve the performance of the learned model. To avoid repeating information sharing / dataset augmentation, the server maintains four sets: (1) the set of clients who shared information (i.e. who encoded original data into latent representations and shared noisy means of latent representations to the server), $\mathcal{A}$; (2) the set of clients who benefited from the shared information (i.e., who augmented their local datasets with synthesized artificial data), $\mathcal{B}$; (3) the set $\mathcal{C} = \{\mathcal{C}_i | i \in \mathcal{A}\}$, where $\mathcal{C}_i = (\mathcal{C}_i^1, \ldots, \mathcal{C}_i^n)$ indicates the n most abundant classes in the local dataset of client $i \in \mathcal{A}\}$; and (4) the set of shared latent representation means, $\mathcal{Z} = \{\mathcal{Z}_i | i \in \mathcal{A}\}$, where $\mathcal{Z}_i = \{(\tilde{z}_{i,1}^1, \ldots, \tilde{z}_{i,\alpha}^1), \ldots, (\tilde{z}_{i,1}^n, \ldots, \tilde{z}_{i,\alpha}^n)\}$, α is the number of repeatedly generated noisy means $\tilde{z}_i^c$ of the latent representation of the (abundant) class c in the local dataset available to client i. These four sets are initialized as empty at the beginning of the secondary training. The described procedure for secondary training is formalized as Algorithm 2.

Algorithm 2 SecTrain

Input:
global model w_e^t, w_c^t; selected k clients $\mathcal{S}_t$; assisting clients $\mathcal{A}$; benefited clients $\mathcal{B}$; abundant class $\mathcal{C}$; shared means $\mathcal{Z}$; local datasets at k clients, $\mathcal{D} = \{\mathcal{D}_i | i \in \mathcal{S}_t\}$; generation quota α; standard deviation of additive noise σ.
Output:
The global model w_e^{t+1}, w_c^{t+1}

$\mathcal{R} \leftarrow$ **Matching**$(\mathcal{S}_t, \mathcal{A}, \mathcal{C})$
for $i \in \mathcal{S}_t$ **in parallel do**
  if $\mathcal{R} \neq \emptyset$ **AND** $i \notin \mathcal{B}$ **then**
    if client i did not download $\mathbf{w_d}$ **then**
      download the global decoder $\mathbf{w_d}$
    end if
    download $\mathcal{Z}_{\mathbf{R}_i}$ from the server
    $\tilde{\mathcal{D}}_i \leftarrow$ **Synthesis**$(\mathbf{w_d}, \mathcal{Z}_{\mathbf{R}_i})$
    $\bar{\mathcal{D}}_i \leftarrow \tilde{\mathcal{D}}_i \cup \mathcal{D}_i$
    the server executes: $\mathcal{B} \leftarrow \mathcal{B} \cup i$
  end if
  download w_e^t, w_c^t from the server
  upload: $(w_{e,i}^t, w_{c,i}^t) \leftarrow$ **Optim**$(w_e^t, w_c^t, \bar{\mathcal{D}}_i)$
  if $i \notin \mathcal{A}$ **then**
    if client i did not download $\mathbf{w_d}$ **then**
      download the global decoder $\mathbf{w_d}$
    end if
    $(\mathcal{C}_i, \mathcal{Z}_i) \leftarrow$ **DPMS**$(w_{e,i}^t, w_{c,i}^t, \mathbf{w_d}, \mathcal{D}_i, \alpha, \sigma)$
  end if
end for
Server executes:
$(\mathcal{A}, \mathcal{C}, \mathcal{Z}) \leftarrow (\mathcal{A}, \mathcal{C}, \mathcal{Z}) \cup_{i \in \mathcal{S}_t} (i, \mathcal{C}_i, \mathcal{Z}_i)$
$|\bar{\mathcal{D}}^t| \leftarrow \sum_{i \in \mathcal{S}_t} |\bar{\mathcal{D}}_i|$
$w^{t+1} \leftarrow \sum_{i \in \mathcal{S}_t} \frac{|\bar{\mathcal{D}}_i|}{|\bar{\mathcal{D}}^t|} (w_{e,i}^t, w_{c,i}^t)$
return w^{t+1}

Client matching. Let $\mathcal{S}_t$ denote the set of clients selected in training round t. Based on the information about the most and least abundant classes in local datasets, the server decides for each client who should they receive assistance from (i.e., which information in $\mathcal{Z}$ should they be given). The matching is based on the distance between the clients' data distributions; in particular, client i is scheduled to be the recipients of client j's latent space information if the server identifies that client j's data distribution is such that the samples drawn from it would significantly diversify client i's data.[1] Specifically, each client $i \in \mathcal{S}_t$ still seeking to diversify local data sends indices of n classes in its dataset with the fewest samples to the server; let $\mathcal{H}_i$ denote the set of the i^{th} client's "data scarce" classes. Having received $\mathcal{H}_i$, the server identifies client j having the set of data abundant classes $\mathcal{C}_j$ that intersects with $\mathcal{H}_i$ more than any other set of data abundant classes. After matching, $\mathcal{Z}_j$ and the global decoder are sent to client i. Note that client j's means of latent data representations $\mathcal{Z}_j$ are sent to client i by the server – there is no direct connection between clients i and j. In the first round of secondary training, the server does not pursue matching since $\mathcal{A}, \mathcal{B}, \mathcal{C}$ and $\mathcal{Z}$ are empty sets; the sets are augmented with new elements in the DPMS step. While our experiments demonstrate remarkable performance improvements of FedDPMS over competing methods despite providing a client in need of synthetic data with the latent representation of only one of its peers, the matching algorithm can readily be extended to identifying several "assisting" clients for a "benefitting" client.

[1] For convenience, we refer to client i as the "benefiting client" and to client j as the "assisting client".

The proposed method for client matching is formalized as Algorithm 3 in Appendix A.3.

Data synthesis. When client i receives a set of noisy means of latent data representations and the global decoder, it utilizes the global decoder $\mathbf{w_d}$ to generate synthetic data $\tilde{\mathcal{D}}_i$; the synthetic data is merged with the local dataset $\mathcal{D}_i$ to form $\bar{\mathcal{D}}_i$. The client then proceed to perform local model update by training on $\bar{\mathcal{D}}_i$, and uploads the result to the server. Upon receiving the update, the server adds index i to the set of indices of clients who completed data diversification, $\mathcal{B}$. Samples of synthetic images are provided in A.4.

Model training After generating synthetic data, each client i has access to an augmented local dataset $\bar{\mathcal{D}}_i$. When updating the model, the client samples $|\mathcal{D}_i|$ points uniformly at random from $\bar{\mathcal{D}}_i$ and utilizes only the sampled points for gradient computation ($|\cdot|$ denotes the set cardinality). This is to maintain the same complexity of the update step as FedAvg.

DPMS. Following model training at round t, selected client i forms updated encoder $w_{e,i}^t$ and classifier $w_{c,i}^t$; recall that the client received the global decoder $\mathbf{w_d}$ which is no longer being updated. If the client did not share latent representation information in the previous rounds, it utilizes $w_{e,i}^t$ to compute the means of latent representations for its n most abundant classes; note that in this step we only utilize the original (real) data to form latent representations. Let us denote the computed means by $(\bar{z}_i^{\mathcal{C}_i^1}, \ldots, \bar{z}_i^{\mathcal{C}_i^n})$, where $\mathcal{C}_i^1, \ldots, \mathcal{C}_i^n$ denote the indices of the most abundant classes in the i^{th} client's dataset. The means are then perturbed by an additive zero-mean Gaussian noise $\mathcal{N}(0, \sigma^2)$ (see Figure 1). There are two benefits of adding the noise: first, it introduces diversity in the synthetically generated data; and, second, it endues the shared information with differential privacy (more on differential privacy in the next section). The noisy corruption of latent representation means, however, may be so severe to result in unusable synthesized data; we would like to identify if this is the case before the client communicates such means to the server. To this end, the client applies the global decoder $\mathbf{w_d}$ and utilizes the (highly accurate) local classifier $w_{c,i}^t$ to attempt recognizing the reconstructed data point. Specifically, the noisy latent representation means are formed as

$$\tilde{z}_i^{\mathcal{C}_i^j} = \bar{z}_i^{\mathcal{C}_i^j} + \delta, \tag{1}$$

where $\delta \sim \mathcal{N}(0, \sigma^2)$ and $j \in \{1, \ldots, n\}$. Using the global decoder, the client reconstructs $\tilde{x}^{\mathcal{C}_i^j}$ in the original space,

$$\tilde{x}^{\mathcal{C}_i^j} = \mathbf{w_d}(\tilde{z}_i^{\mathcal{C}_i^j}), \tag{2}$$

and then applies the local encoder/classifier to find its prediction $\tilde{y}$,

$$\tilde{y} = w_{c,i}^t(w_{e,i}^t(\tilde{x}^{\mathcal{C}_i^j})). \tag{3}$$

If the prediction of the classifier is correct, i.e., $\tilde{y} = \mathcal{C}_i^j$, we retain the noisy latent representation mean $\tilde{z}_i^{\mathcal{C}_i^j}$ used to synthesize the classified point; otherwise, we declare that $\tilde{z}_i^{\mathcal{C}_i^j}$ is unusable and discard it. We continue this procedure until the number of usable noisy latent representation means in each abundant class $\mathcal{C}_i^j$ meets a predetermined quota α (a tunable hyperparameter). Finally, each client sends its set of noisy encoded means $\mathcal{Z}_i = \{(\tilde{z}_{i,1}^{\mathcal{C}_i^1}, \ldots, \tilde{z}_{i,\alpha}^{\mathcal{C}_i^1}), \ldots, (\tilde{z}_{i,1}^{\mathcal{C}_i^n}, \ldots, \tilde{z}_{i,\alpha}^{\mathcal{C}_i^n})\}$ and their corresponding labels to the server.[2] Following the DPMS step, the server appends i, $\mathcal{C}_i$ and $\mathcal{Z}_i$ to $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{Z}$, respectively. In future training rounds, even if sampled again (which in large-scale systems is highly unlikely), client i will not be asked to share latent representation information. Moreover, the client can now delete the global decoder $\mathbf{w_d}$ to free up memory; in fact, a selected client maintains the global decoder model for at most one round and thus its impact on average memory consumption is only minor. Further discussion of communication, computation and memory consumption can be found in Section A.9.

It is worth pointing out that the assisting and benefiting clients each utilize their respective *local* encoder and *global* decoder: the former (in conjunction with its local classifier and global decoder) to learn latent data representations, the latter to synthesize artificial data. Such a strategy is desirable for two reasons. First, the local classifier achieves performance superior to the global classifier since

[2]To ensure that discarding noisy latent representation means which fail the test does not affect privacy, we empirically evaluate the standard deviation of the noise present in the perturbed means that have passed the test and are consequently shared; in experiments, this standard deviation was verified to be virtually identical to σ in (1), implying that sub-selecting the means has no impact on the level of privacy introduced by the noise corruption of latent representation means.

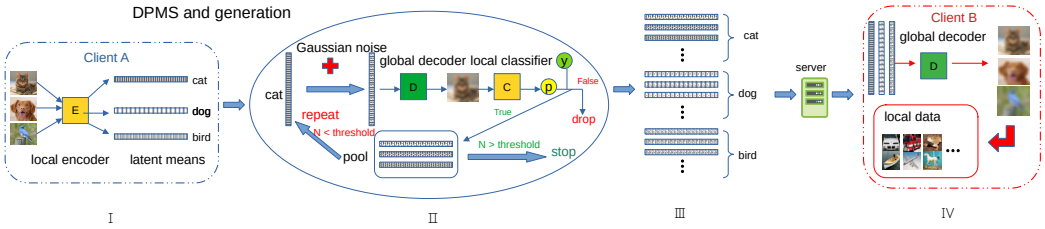


Figure 1: FedDPMS and synthetic data generation. The four parts of the figure depict: (1) finding latent representation of raw data via a local encoder; (2) creating noisy latent means (by adding Gaussian noise to the means of latent data representations) and filtering out unusable ones with the help of a local classifier; (3) uploading usable noisy latent means to the server; (4) a benefiting client utilizing the global decoder to generate synthetic data from the received noisy latent means, expanding its local dataset.

the latter is agglomerated from local models that may have drifted apart; therefore, local classifiers are more trustworthy decision-makers for selecting usable noisy means. Second, we encode raw data using the local encoder of the assisting client but decode the latent representation means with the global decoder at the benefiting client; this helps separate the benefiting client's synthetic data from the assisting client's raw data in a way that is complementary to the separation induced by averaging latent information or adding noise. While its impact appears challenging to formalize analytically, such a strategy intuitively helps protect the assisting client's privacy.

3.4 Privacy concerns

While prior work on data augmentation in federated learning relied on data averaging for privacy protection (Liu et al., 2021; Yoon et al., 2021), differential attacks may be used to extract individual information from the averages. This motivates using concepts from differential privacy to quantify the privacy provided by noisy perturbations that FedDPMS injects into latent means. In related prior work, a number of methods that attempt to prevent privacy leaks from uploaded local gradients or models have been proposed in literature (Fredrikson et al., 2015; Hitaj et al., 2017; Triastcyn & Faltings, 2020b; Torkzadehmahani et al., 2019). For brevity, the definitions of differential privacy are provided in Appendix A.7. These methods can readily be adapted to model training and client matching in FedDPMS. For simplicity, in this paper we limit our attention to characterizing differential privacy of sharing noisy latent means. To this end, consider m latent representations $\mathbf{z_i} \in \mathcal{R}^l$, where l is the dimension of latent representation, $1 \leq i \leq m$; each client computes the j^{th} element of the latent mean $\bar{\mathbf{z}}$ by averaging the corresponding elements of the aforementioned m vectors, $1 \leq j \leq l$. Since clients reveal means of latent local data representations to the server, they may become exposed to the risk of leaking the latent representation of individual data points; this, in turn, could potentially be exploited to attempt reconstruction of raw data. To protect individual latent representations from a differential attack, we construct the following differential privacy mechanism. Let us interpret the averaging operation as a deterministic function $f(\mathbf{e})$, where vector $\mathbf{e}$ collects the j^{th} elements of m latent representations. Recall the definition of S_f,

$$S_f^2 = \max \left(f(\mathbf{e}) - f(\mathbf{e}') \right)^2,$$

where $\mathbf{e}'$ is an adjacent input (similar to Def. 2) which matches $\mathbf{e}$ in $m-1$ elements. Since we use sigmoids as the activation functions of the layer generating latent representations (applied to all models in this paper), the latent means are between $[0, 1]$; therefore,

$$f(\mathbf{e}) - f(\mathbf{e}') \leq \frac{\mathbf{1}^T\mathbf{e}' + 1}{m} - \frac{\mathbf{1}^T\mathbf{e}'}{m-1} = \frac{m - 1 - \mathbf{1}^T\mathbf{e}'}{m(m-1)} \leq \frac{1}{m},$$

where $\mathbf{1}^T$ denotes the vector of all ones having the same dimension as $\mathbf{e}'$. Thus, $S_f = \frac{1}{m}$. For any $\epsilon > 0$ and $\delta > 0$, we can always identify σ needed to ensure that $f(d)$ achieves (ϵ, δ) differential privacy according to Definition 2. As an illustration, for $m = 100$, $\epsilon = 0.5$ and $\delta = 0.01$, selecting $\sigma > 3.9$ (i.e., the standard deviation of noise $S_f\sigma > 0.039$), we are 99% confident that the privacy is not broken. In fact, the standard deviation of noise added to latent means in our experiments is > 1 while $m > 100$; thus there is abundant room for lower privacy budget ϵ or higher confidence $1 - \delta$. In these settings, FedDPMS is very unlikely to break the privacy of the latent data representations.

4 EXPERIMENTS

4.1 DATASETS AND BASELINES

We implemented all models and ran the experiments in Pytorch (Paszke et al., 2019), using Adam (Kingma & Ba, 2014) optimizer with a learning rate 0.001 for all methods. The period of learning rate decay was set to 10, while the hyper-parameter γ in Adam was set to 0.5. We used the data batch size of 64. Unless stated otherwise, the number of local epochs was set to 5, while the number of global communication rounds was 50; none of the methods improved performance by further increasing the number of global communication rounds. The default number of clients participating in federated learning is 10. For simplicity and due to the relatively small number of clients, all clients participate in all rounds of the federated learning process.

To test the performance of our proposed framework, we conduct experiments on three datasets – Fashion-MNIST (Xiao et al., 2017), CIFAR10, and CIFAR100 (Krizhevsky et al.). To control the degree of imbalance in the partitioned data, we utilize Dirichlet distribution (Yoon et al., 2021; Yurochkin et al., 2019; Li et al., 2021) and generate non-iid partitions with varied concentration parameter β. Note that when the concentration parameter is very small, e.g. $\beta = 0.5$, a client may have very few samples (possible none) in some classes, potentially rendering the partition highly imbalanced. Examples of clients' local dataset class distributions are shown in Appendix A.5. We compare the test accuracy of FedDPMS with four state-of-the-art federated learning methods including FedAvg (McMahan et al., 2017), FedProx (Li et al., 2018), FedMix (Yoon et al., 2021) and Moon (Li et al., 2021). The benchmarking experiments utilize VAEs and a convolutional neural network trained to perform image classification tasks. For detailed specification of the network architecture and hyper-parameters, please see Appendix A.1 and A.2.

4.2 RESULTS ANALYSIS

Accuracy comparison. Table 1 compares the test accuracy of FedDPMS with the concentration parameter $\beta = 0.5$ and default parameter settings against the competing methods. In such a severely heterogeneous setting, relative improvement of FedDPMS over FedAvg is 3.3%, 4.5% and 6.6% on FMNIST, CIFAR10 and CIFAR100, respectively; overall, FedDPMS achieves the best performance among all approaches. As for the performance of other methods relative to each other: FedMix has a significant advantage over FedAvg on FMNIST while closely tracking, along with FedProx, performance of FedAvg on the other two datasets. Moon exhibits a slight improvement over FedAvg on CIFAR100 and achieves close 2nd (behind FedDPMS) performance on CIFAR10.

Table 1: Test accuracy of DPMS and other methods on FMNIST, CIFAR10 and CIFAR100 datasets with concentration parameter $\beta = 0.5$ among 10 clients.

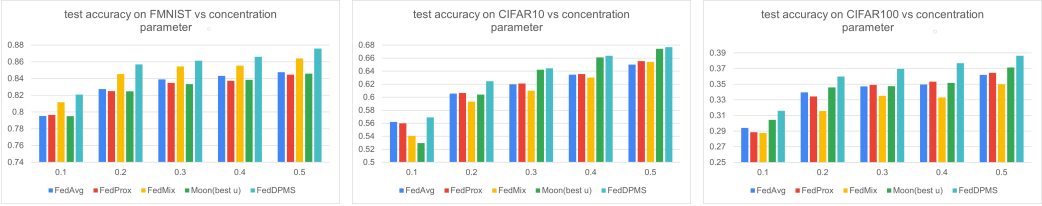
	Fmnist	Cifar10	Cifar100
FedAvg	0.8476	0.6501	0.3621
FedProx	0.8446	0.6553	0.3646
FedMix	**0.8640**	0.6542	0.3498
Moon	0.8458	0.6742	0.3715
FedVAE	0.8452	0.6631	0.3712
FedDPMS	0.8604	**0.6797**	**0.3861**

Table 2: Test accuracy among different numbers of clients from $\{10, 20, 30, 40, 50\}$ with 10%-50% samples of CIFAR100 with concentration parameter $\beta = 0.5$.

	10	20	30	40	50
FedAvg	0.1881	0.2401	0.2738	0.2842	0.3066
FedProx	0.1844	0.2408	0.2780	0.2914	0.3081
FedMix	0.1820	0.2365	0.2676	0.2877	0.3045
Moon	0.1871	0.2375	0.2707	0.2912	0.3095
FedVAE	0.1905	0.2428	0.2746	0.2858	0.3073
FedDPMS	**0.2177**	**0.2677**	**0.2940**	**0.3061**	**0.3208**

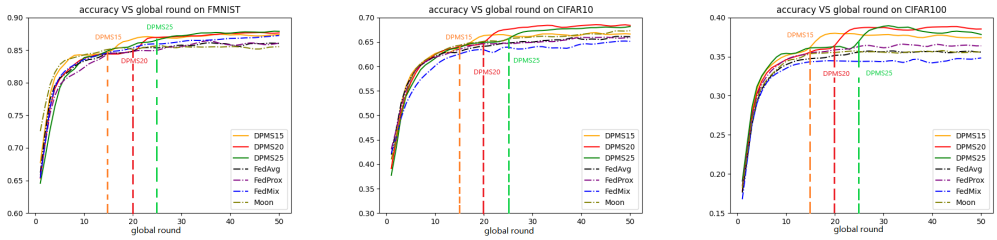
Effect of data heterogeneity. As previously stated, the concentration parameter β controls heterogeneity of the data partitions – smaller β leads to more severe class imbalance. We use the Dirichlet distribution with $\beta = \{0.1, 0.2, 0.3, 0.4, 0.5\}$ to generate varying data partitions and study the effect of heterogeneity on different methods. As the results in Fig. 2 demonstrate, FedDPMS consistently achieves the best performance on FMNIST and CIFAR10/100. Among other methods, FedProx closely tracks FedAvg in all experiments, while Moon is competitive on CIFAR10 yet performs badly for $\beta = 0.1$. The takeaway from competitive performance of FedMix on FMNIST, but poor on both CIFAR10 and CIFAR100, is that utilizing interpolation to synthesize samples from raw data succeeds on simple datasets but does not on more complex ones; moreover, the approach deployed by FedMix is risky in terms of potential privacy leaks. In all, the experiments demonstrate consistently best performance of FedDPMS across different data distributions and levels of heterogeneity.

Scalability. So far, our experiments involved simulating federated learning systems with 10 clients. To investigate the scalability of FedDPMS, we vary the number of clients in experiments on CIFAR100. In particular, the number of clients is varied across $\{10, 20, 30, 40, 50\}$; to maintain the



(a) FMNIST (b) CIFAR10 (c) CIFAR100

Figure 2: Test accuracy of different approaches as the concentration parameter β takes values from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. We run five trials with different random seeds and report the mean accuracy.



(a) FMNIST (b) CIFAR10 (c) CIFAR100

Figure 3: Test accuracy of different approaches on FMNIST, CIFAR10 and CIFAR100. All experiments are conducted with the concentration parameter $\beta = 0.5$ and 10 clients. We run three trials with different random seeds and report the mean accuracy.

same amount of data per client as the number of clients grow, an increasing fraction of CIFAR100 data $(10\%, 20\%, 30\%, 40\%, 50\%)$ is partitioned and allocated to the clients in these experiments. Meanwhile, we keep the default heterogeneity setting $\beta = 0.5$. The results of the experiments are reported in Table 2, showing that FedDPMS outperforms all other approaches. For example, relative improvement of FedDPMS over FedAvg is more than $22.3\%, 15.6\%$, 7.3%, 7.7% and 4.6% in the experiments involving 10, 20, 30, 40, and 50 clients, respectively.

Convergence rate. Fig. 3 shows the test accuracy of all approaches across the federated training rounds. For FedDPMS, we further experimented by varying duration of the preliminary training phase (15, 20 and 25). Among all methods, FedDPMS exhibits the fastest convergence rate and best accuracy on CIFAR10 and CIFAR100. FedMix has the slowest convergence rate on CIFAR10/100 but a slightly higher accuracy than FedDPMS on FMNIST. After sharing the latent representation means and augmenting local datasets with synthetic data, the test accuracy of each FedDPMS model suddenly increases. The results imply that the quality of augmented data improves with the number of preliminary training rounds; however, the longer we wait, the smaller the number of rounds to train the model on the augmented dataset. The experiments suggest that setting the number of preliminary training rounds to 40% of the total number of allotted rounds is a suitable choice.

5 Conclusion

Data heterogeneity is a critical challenge hindering practical federated learning systems, potentially causing major performance deterioration. We propose a novel framework, FedDPMS, aiming to enable accurate and robust performance of federated deep learning models trained on heterogeneous distributed datasets. This is accomplished by sharing differentially-private information which the clients use to enrich local datasets and thus combat the local model drift. As our experimental results demonstrate, FedDPMS outperforms state-of-the-art federated learning methods on image classification tasks with varied levels of heterogeneity across clients while requiring only a minor increase in communication cost. FedDPMS does requires additional computation and memory resources, the amount of which depends on the specifics of the training process. In applications where the high accuracy is imperative, FedDPMS provides an attractive and effective framework for overcoming challenges presented by data heterogeneity.

REFERENCES

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2020.

Sean Augenstein, H Brendan McMahan, Daniel Ramage, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019a.

Sean Augenstein, H Brendan McMahan, Daniel Ramage, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019b.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *arXiv preprint arXiv:2109.04617*, 2021.

Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *International Conference on Machine Learning*, pp. 3407–3416. PMLR, 2021.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333, 2015.

Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10112–10121, 2022.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *International Conference on Machine Learning*, pp. 3973–3983. PMLR, 2020.

Weituo Hao, Mostafa El-Khamy, Jungwon Lee, Jianyi Zhang, Kevin J Liang, Changyou Chen, and Lawrence Carin Duke. Towards fair federated learning with zero-shot data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3310–3319, 2021.

Chaoyang He, Shen Li, Mahdi Soltanolkotabi, and Salman Avestimehr. Pipetransformer: Automated elastic pipelining for distributed training of transformers. *arXiv preprint arXiv:2102.03161*, 2021.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 603–618, 2017.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

Peter Kairouz, H Brendan McMahan, Brendan Avent, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Qinbin Li, Zeyi Wen, and Bingsheng He. Practical federated gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4642–4649, 2020a.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1013–1023, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Adam Paszke, Sam Gross, Francisco Massa, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Mohammad Rasouli, Tao Sun, et al. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228*, 2020.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 2020.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pp. 9120–9132. PMLR, 2020.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

Aleksei Triastcyn and Boi Faltings. Federated generative privacy. *IEEE Intelligent Systems*, 35(4): 50–57, 2020a.

Aleksei Triastcyn and Boi Faltings. Federated generative privacy. *IEEE Intelligent Systems*, 35(4): 50–57, 2020b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020a.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020b.

Hui Wen, Yue Wu, Jingjing Li, and Hancong Duan. Communication-efficient federated data augmentation on non-iid data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3377–3386, 2022.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning. *arXiv preprint arXiv:2108.08647*, 2021.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. *arXiv preprint arXiv:2107.00233*, 2021.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pp. 7252–7261. PMLR, 2019.

Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *arXiv preprint arXiv:2010.08982*, 2020.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

A APPENDIX

A.1 NETWORK ARCHITECTURE

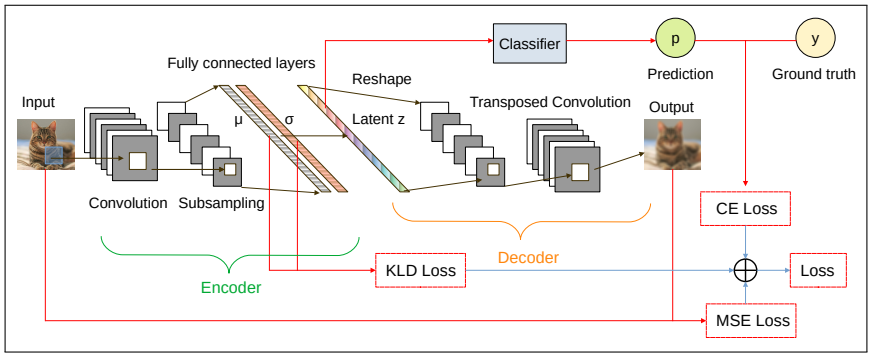


Figure 4: The architecture of the VAE. The KLD, MSE and CE stand for Kullback-Leibler divergence, mean-square error, and cross-entropy, respectively. The sum of the MSE loss and the KLD loss is weighted by the hyper-parameter λ and added to the CE loss.

For FMNIST, we use a CNN network with three 3×3 convolutional layers connected by two 2×2 max pooling layers as the base encoder. For CIFAR10 and CIFAR100, we use a CNN network with four 3×3 convolutional layers connected by two 2×2 max pooling layers as the base encoder. Both architectures utilize two fully connected layers. Moreover, we construct a decoder of a symmetric structure, composing a variational auto-encoder when implementing FedVAE and FedDPMS. We set the latent dimension l to 32 and 128 for FMNIST and CIFAR10/100, respectively. The details are illustrated in Tables 3 and 4:

Table 3: The layers of the base networks for all approaches other than FedVAE and FedDPMS.

Layers	FMNIST	Layers	CIFAR10/100
Input	$28 \times 28 \times 1$	Input	$32 \times 32 \times 3$
Convolution	$3 \times 3(16)$	Convolution	$3 \times 3(32)$
Maxpool	2×2	Convolution	$3 \times 3(32)$
Convolution	$3 \times 3(32)$	Maxpool	2×2
Maxpool	2×2	Convolution	$3 \times 3(64)$
FC	$7 \times 7 \times 32$	Convolution	$3 \times 3(64)$
FC	32	Maxpool	2×2
Ouput	10	FC	$8 \times 8 \times 64$
		FC	128
		Output	10/100

A.2 SELECTION OF HYPER-PARAMETERS

The hyper-parameter controlling the proximal term in FedProx is set to $\mu_{prox} = 0.001$. For Moon, we set the temperature parameter to 0.5, tune μ_{moon} across $\{1, 5, 10\}$, and report the best result. The mixup ratio parameter in FedMix was set to $\lambda_{mix} = 0.05$. In FedDPMS, we set the hyper-parameter $\lambda = 0.05$. Moreover, FedDPMS's predetermined quota for the synthetic data of each abundant class and the standard deviation of the additive Gaussian noise in experiments on FMNIST, CIFAR10 and CIFAR100 were set to $\alpha = \{50, 200, 50\}$, $\sigma = \{3, 4, 4\}$ respectively. To choose the values of n, we recall that the data is generated according to the Dirichlet distribution with a small concentration parameter β; this implies presence of relatively few classes in each local dataset. Consequently, for FMNIST/CIFAR10 we set $n = 3$, while for CIFAR100 we set $n = 10$. In practice, the value of n can be chosen with the clients' help; e.g., client i reports to the server n_i, the number of the client's data classes decidedly less abundant than the rest of the classes. The server then sets $n = \min\{n_1, n_2, \ldots, n_K\}$. The communication cost and delay of establishing n in this way is

Table 4: The layers for base networks of FedVAE and FedDPMS.

Layers	FMNIST	Layers	CIFAR10/100
Input	$28 \times 28 \times 1$	Input	$32 \times 32 \times 3$
Convolution	$3 \times 3(16)$	Convolution	$3 \times 3(32)$
Maxpool	2×2	Convolution	$3 \times 3(32)$
Convolution	$3 \times 3(32)$	Maxpool	2×2
Maxpool	2×2	Convolution	$3 \times 3(64)$
FC	$7 \times 7 \times 32$	Convolution	$3 \times 3(64)$
FC	32	Maxpool	2×2
FC	$7 \times 7 \times 32$	FC	$8 \times 8 \times 64$
Transposed-Con	$3 \times 3(16)$	FC	128
Transposed-Con	$3 \times 3(1)$	FC	$8 \times 8 \times 64$
Output	$28 \times 28 \times 1$	Transposed-Con	$3 \times 3(64)$
		Transposed-Con	$3 \times 3(64)$
		Transposed-Con	$3 \times 3(32)$
		Transposed-Con	$3 \times 3(3)$
		Output	$32 \times 32 \times 3$

negligible. Without a loss of generality, in the experiments we set the participation rate of clients k/K to 1.

A.3 Algorithm 3 Matching

The matching algorithm aims to match pairs of clients having complementary local data; the result of matching depends on class distribution of each client's local dataset. One of the options is to find the size of the intersection between "abundant classes" $\mathcal{C}_i$ and "data scarce" classes $\mathcal{H}_j$, described as Algorithm 3.

It is worth mentioning that the information sharing may be uni-directional: if client i benefits from client j's latent data representation, the reverse need not be true. In fact, it may be the case that one client supplies latent representation information to several clients if the latter lack in data classes that the former has in its local dataset. Note that while the presented framework assumes categorical data labels, the methodology can be extended to regression tasks by relying on data binning.

Algorithm 3 Matching

Input:
the set of clients who shared information, $\mathcal{A}$; the set of abundant classes at the server, $\mathcal{C}$; the selected clients, $\mathcal{S}_t$ client's "data scarce" classes $\mathcal{H} = \{\mathcal{H}_i | i \in \mathcal{S}_t\}$.

Output:
Receiving source $\mathbf{R}$

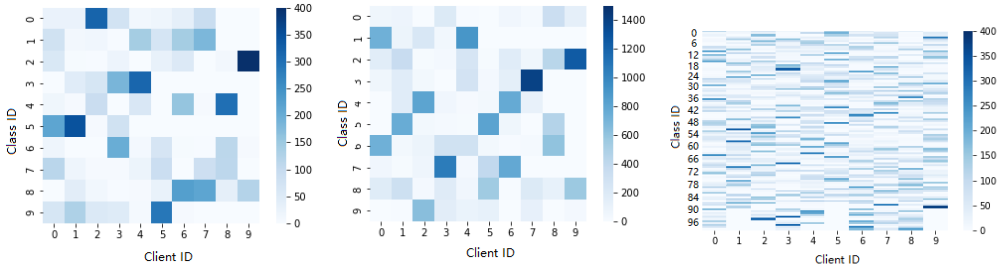
Server executes:
if $\mathcal{C} = \emptyset$ **then**
  $\mathbf{R} \leftarrow \emptyset$
else
  for $i \in \mathcal{S}_t$ **do**
    $R_i \leftarrow \textbf{argmax}_{j \in \mathcal{A}} |\mathcal{H}_i \cap \mathcal{C}_j|$
  end for
  $\mathbf{R} \leftarrow \{R_i | i \in \mathcal{S}_t\}$
end if
return $\mathbf{R}$

A.4 Synthetic Images

Figure.5 presents examples of artificial images generated by the variational auto-encoder on FMNIST, CIFAR10 and CIFAR100; since the FMNIST's representation space is relatively simple, one

(a) FMNIST (b) CIFAR10 (c) CIFAR100

Figure 5: The class distribution of each client in non-iid data partitions generated by Dirichlet distribution with $\beta = 0.5$. Dark color implies more data while light color implies less data.



(a) FMNIST (b) CIFAR10 (c) CIFAR100

Figure 6: The class distribution of each client in non-iid data partitions generated by Dirichlet distribution with $\beta = 0.5$. Dark color implies more data while light color implies less data.

may recognize the type of synthetic images, while the synthetic images generated to augment CIFAR10/100 datasets are incomprehensible. Nevertheless, even though incomprehensible to humans, synthesized images carry information that helps FedDPMS improve local training.

A.5 Visualization of Class Distribution in Data Partitions

Here we visualize class distributions of local clients induced by data partitioning of FMNIST, CIFAR10 and CIFRAR100. For example, the first column in Fig. 6, representing the class distribution of client '0' on FMNIST partition, shows that client '0' has samples in classes '1,2,5,7,9' while lacking classes '0,3,4,6,8'. As we can observe in Fig. 6, data partitions generated by the Dirichlet distribution may be extremely heterogeneous, becoming more so for smaller β.

A.6 Generative Model

In contrast to FedDG (Liu et al., 2021) and FedMix (Yoon et al., 2021) which utilize interpolation to synthesize augmented data, FedDPMS relies on Variational Auto-Encoders (VAEs) (Kingma & Welling, 2013) to learn latent data representation and enable data synthesis from DP-protected transformations of those representations. As its conventional counterparts, a VAE consists of two main components: an encoder and a decoder. The encoder is an approximate inference network $q_\phi(z \mid x)$ that maps input x to the latent vector z which approximates prior probability $p(z)$. The decoder is a generative network $p_\theta(x \mid z)$ which synthesizes x from the latent vector z. The goal of the VAE training is to maximize the likelihood of data points x, i.e., solve $\max_\theta \log p_\theta(x)$. This likelihood is challenging to compute/maximize; observing that

$$\log p_\theta(x) \geq E_{q_\phi(z|x)} \log p_\theta(x \mid z) - D_{KL}\left(q_\phi(z \mid x) \mid p(z)\right) \tag{4}$$

one can instead choose to maximize $E_{q_\phi(z|x)} \log p_\theta(x \mid z)$, the so-called evidence lower bound (ELBO). For convenience, we often assume that z has a Gaussian prior distribution, somewhat limiting the expressiveness of the VAE. In applications involving images, for example, this typically renders VAE-generated samples to be blurry. More broadly, when it comes to generating high-resolution images VAEs may be lagging behind GANs (Goodfellow et al., 2014), but are faster, more reliable and theoretically better understood. Besides, for the problem studied in this paper, the utility of generated images is not measured by their resolution but by the robustness of the classification model trained on synthetically diversified distributed datasets.

A.7 Differential Privacy Mechanism

For completeness, we here define the differential privacy mechanism.

Definition 1 (Differential Privacy) *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies (ϵ, δ) differential privacy if for any two adjacent databases $d, d' \in \mathcal{D}$ with only one different sample, and for any subset of the output $S \subseteq \mathcal{R}$, it holds that*

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} \Pr\left[\mathcal{M}\left(d'\right) \in S\right] + \delta. \tag{5}$$

The output of the random mechanism $\mathcal{M}$ is a random distribution; ϵ denotes an upper bound on the distance between distributions $\mathcal{M}(d)$ and $\mathcal{M}(d')$ and can be interpreted as the privacy budget, while the relaxing factor δ is the probability that the ε-differential privacy is broken.

A common method for enduing a deterministic function $f : \mathcal{D} \to \mathbf{R}$ with differential privacy is via the Gaussian noise mechanism, which is defined as follow:

Definition 2 (Gaussian noise mechanism) *The Gaussian noise mechanism achieving (ε, δ) differential privacy for function $f : \mathcal{D} \to \mathbf{R}$ is defined as*

$$\mathcal{M}(d) = f(d) + \mathcal{N}\left(0, S_f^2 \cdot \sigma^2\right), \tag{6}$$

where S_f denotes the maximum of the absolute distance $|f(d) - f\left(d'\right)|$, and $\sigma > \sqrt{2 \log \frac{5}{4\delta}}/\varepsilon$. In other word, if we add a zero-mean Gaussian noise with variance $S_f^2\sigma^2$ to the output of f and set the privacy budget ϵ, the confidence of the resulting mechanism $\mathcal{M}$ is $\delta \geq \frac{4}{5} \exp\left(-(\sigma\epsilon)^2/2\right)$.

A.8 Effects of the synthetic quota α and the additive Gaussian noise

We rely on empirical studies to investigate the impact of (predetermined) synthetic data quota α (see Section 3.3.2) and the variance of the additive Gaussian noise σ^2 upon the performance of FedDPMS; the results of the experiments conducted on CIFAR10 are shown in Table 5. As implied by the table, when σ^2 is too large, the data synthesized using noisy means is unrecognizible by the local encoder / classifier. Note that the variance of the added noise is $S_f^2\sigma^2$, where $S_f^2 = \frac{1}{m}$ (see Section 3.4). The synthetic data quota α controls the proportion of the synthesized data in the augmented local datasets. On the one hand, if the quota is too small, augmentation does not help improve diversity of local data and ultimately fails to help local training; on the other hand, large quota limits the impact of real data and in fact adversely affects accuracy of the learned models. We experimentally find that the synthesized data should amount to approximately 10% of a local dataset. The same value of α is used by all clients.

A.9 Communication, Computation and Memory

Here we compare communication, computation and memory consumption of FedDPMS with state-of-the-art competing methods. To facilitate the comparison, we set FedAvg as the baseline and summarize the information in Table. 6. For clarity, we introduce the following notations: Θ is the number of parameters of the final model (encoder and classifier); l denotes the dimension of the latent space in FedDPMS; G is the size of a raw data point (e.g., if data is an image with width a and height b, then $G = ab$); n denotes the number of abundant classes; α is the number of synthetic samples in an abundant class; $t_{avg}, t_{avg} + t_{prox}, t_{avg} + t_{moon}$ denotes the time to update the base model in FedAvg, FedProx and Moon methods in a round, respectively; K denotes the number of

Table 5: Test accuracy for different generation quotas α and standard deviations σ. All results are based on $\beta = 0.5$. We set $\alpha = 200$ when running experiments for varied σ, while $\sigma = 4$ when running experiments for varied α. '-' in the table means VAE is not generating qualified data.

σ	ACCURACY (α = 200)	α	ACCURACY (σ = 4)
3	0.6734	50	0.6618
4	**0.6797**	100	0.6721
5	0.6719	150	0.6795
6	-	200	**0.6824**
		250	0.6712
		300	0.6655

clients in the system; k is the number of clients selected in a round; $\nu = \frac{k}{K}$ denotes the sampling rate; T denotes the overall number of training epochs and T_p denotes the number of preliminary training epochs.

Communication costs. Compared to the baselines (e.g., FedProx (Li et al., 2018) and Moon (Li et al., 2021)), FedDPMS requires additional communication resources for two reasons. First, to enable diversification of local data, the server acquires noisy latent representation means from an assisting client and shares them with a matching benefitting clients. We emphasize that each client can assist or benefit from another at most once during the entire training process. The worst-case scenario from the communications perspective is if every client in the system fulfills both roles. In that case, the relative overhead introduced by transmitting the means of latent representation is $r_1 = \alpha nl/\Theta$. Typically, this overhead is rather small; for instance, in our experiments with CIFAR10, $n = 3$, $\alpha = 200$, $l = 128$, $\Theta > 1000000$, and thus $\alpha nl/\Theta \approx 7.68\%$. The second part of the additional communication cost is due to uploading local decoder model (a single event at the end of preliminary training) and the download of the global decoder by the clients seeking to diversify local dataset during the secondary training phase. To compute E, the expected number of selected clients who download the global decoder during secondary training (lasting $T - T_p$ rounds), we denote by $\mathcal{E}_t$ an event that a client selected at round t has not benefitted from latent representation information in the previous $t - 1$ rounds of secondary training. Clearly,

$$\Pr(\mathcal{E}_t) = (1 - \nu)^{t-1}. \tag{7}$$

Then, the expected number of the selected clients is found as

$$E = \left((1 - \nu)^0 + \cdots + (1 - \nu)^{T-T_p-1}\right) k = \frac{1 - (1 - \nu)^{T-T_p}}{\nu} k. \tag{8}$$

The ratio between the communication overhead and the baseline is

$$r_2 = \frac{E\Theta}{2k\Theta T} = \frac{1 - (1 - \nu)^{T-T_p}}{2\nu T}. \tag{9}$$

If $\nu \ll 1$ then $r_2 = \frac{1}{2} - \frac{T_p}{2T}$, implying that all of the clients selected in a secondary training round actually download the global decoder. For large T, $r_2 = \mathcal{O}(\frac{1}{2\nu T})$. For convenience, we specify the communication cost of FedDPMS in Table 6 as $UB = r_1 + r_2$.

Computational cost. While FedAvg schemes deploy and train only a classifier (i.e., a single neural network), FedDPMS relies on VAEs and in the preliminary phase requires training of encoder/classifier/decoder. Specifically, the computational overhead of FedDPMS relative to FedAvg is $\frac{T_p}{T}$. In our experiments, FedDPMS achieves the best performance when $T_p/T = 0.4$, implying 40% more computation; clearly, varying T_p/T explores the trade-off between performance and computational cost. Note that when training large models, FedDPMS requires more secondary training rounds and thus ratio $\frac{T_p}{T}$ becomes small, implying less computational overhead.

Memory requirements. Since FedDPMS relies on VAEs, in the preliminary training it requires approximately twice as much memory as FedAvg schemes that deploy the same neural network architectures. In the secondary training, each client stores the global decoder's parameters for at most one round, and hence the average additional memory is negligible. FedDPMS further requires additional memory to store synthetic data; that overhead is typically negligible compared to the total memory use. Note that among the competing methods, FedProx and Moon each introduce a regularization term and thus they too need additional memory – the former requires twice as much memory as FedAvg while the latter needs three times as much.

Table 6: A comparison of the communication, computation and memory requirement for FedDPMS and the competing methods.

SCHEME	COMMUNICATION	COMPUTATION	MEMORY
FEDAVG	1	1	1
FEDPROX	1	1 + $\frac{t_{prox}}{t_{avg}}$	2
FEDMIX	1 + G/Θ	1	1
MOON	1	1 + $\frac{t_{moon}}{t_{avg}}$	3
FEDDPMS	1 + UB	1+T_p/T	1+T_p/T