Federated Learning in Non-IID Settings Aided by Differentially Private Synthetic Data

Huancheng Chen, Haris Vikalo
Universitiy of Texas at Austin, Austin, Texas
huanchengch@utexas.edu,hvikalo@ece.utexas.edu

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 [19], COCO [24]) 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 [2, 15, 22, 42].

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) [26], chooses users at random and updates the global model by averaging the users' updates; the convergence analysis provided in [26] 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 [10, 12, 16, 21, 23, 28, 29, 45].

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 [16, 20, 23]. Data heterogeneity in FL systems deploying deep learning networks and training on complex datasets was studied in [20] 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 [4, 5, 8, 30, 32, 41].

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 gener-

ate 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) [3, 9, 34], 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 [40], 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

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 [44], 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 [37] takes a step further and extends the same ideas to CNNs and LSTMs. In [38], the authors present a framework for the analysis of convergence of FL on heterogeneous data, along with a normalized averaging method, FedNova [38], 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 [23] introduces a proximal term to the learning objective of each client with the goal of making local training aligned with the global objective. SCAFFOLD [16] utilizes predictive variance reduction [14] 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, [20] 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, [20]

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. Another study, FedDyn [1], 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 [7] 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.

2.2. Domain Generalization in Federated Learning

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 [25], 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. However, FedDG is unable to enrich class diversity since the phase spectrum is not shared due to privacy concerns. Another study, [11], employs zero-shot data augmentation and relies on the statistics of the batch normalization (BN) layers to reduce the variance of test accuracy. FedMix [43] presents a framework for sharing clients' averaged local data via Mixup [46]; privacy of clients in FedMix may still be compromised since the only effort to protect it is based on averaging raw data. FedDA [39] utilizes the attention mechanism [36] to enable the server to aggregate local per-label knowledge and create global per-label knowledge, ultimately enabling data augmentation via conditional variational autoencoder [31]. 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. 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 generaliza-

tion 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 up-sampling 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 supplementay material section 1 and 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,

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

$$\ell_3 = \text{MSEloss}(\hat{x}, x)$$

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 3.2, 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.

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 ith 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.

Algorithm 1 (FedDPMS)

Require:
    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 epochs T.

Ensure:
    The final global model w^T

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

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.

Client matching. Let $\mathcal{S}_t$ denote the set of clients selected

---

Algorithm 2 SecTrain

Require:
    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.

Ensure:
    The global model w_e^{t+1}, w_c^{t+1}

1: $\mathcal{R} \leftarrow$ **Matching**$(\mathcal{S}_t, \mathcal{A}, \mathcal{C})$
2: **for** $i \in \mathcal{S}_t$ **in parallel do**
3:     **if** $\mathcal{R} \neq \emptyset$ **AND** $i \notin \mathcal{B}$ **then**
4:         **if** client i did not download $\mathbf{w_d}$ **then**
5:             download the global decoder $\mathbf{w_d}$
6:         **end if**
7:         download $\mathcal{Z}_{\mathbf{R}_i}$ from the server
8:         $\tilde{\mathcal{D}}_i \leftarrow$ **Synthesis**$(\mathbf{w_d}, \mathcal{Z}_{\mathbf{R}_i})$
9:         $\bar{\mathcal{D}}_i \leftarrow \tilde{\mathcal{D}}_i \cup \mathcal{D}_i$
10:         the server executes: $\mathcal{B} \leftarrow \mathcal{B} \cup i$
11:     **end if**
12:     download w_e^t, w_c^t from the server
13:     upload: $(w_{e,i}^t, w_{c,i}^t) \leftarrow$ **Optim**$(w_e^t, w_c^t, \bar{\mathcal{D}}_i)$
14:     **if** $i \notin \mathcal{A}$ **then**
15:         **if** client i did not download $\mathbf{w_d}$ **then**
16:             download the global decoder $\mathbf{w_d}$
17:         **end if**
18:         $(\mathcal{C}_i, \mathcal{Z}_i) \leftarrow$ **DPMS**$(w_{e,i}^t, w_{c,i}^t, \mathbf{w_d}, \mathcal{D}_i, \alpha, \sigma)$
19:     **end if**
20: **end for**
21: **Server executes:**
22: $(\mathcal{A}, \mathcal{C}, \mathcal{Z}) \leftarrow (\mathcal{A}, \mathcal{C}, \mathcal{Z}) \cup_{i \in \mathcal{S}_t} (i, \mathcal{C}_i, \mathcal{Z}_i)$
23: $|\bar{\mathcal{D}}^t| \leftarrow \sum_{i \in \mathcal{S}_t} |\bar{\mathcal{D}}_i|$
24: $w^{t+1} \leftarrow \sum_{i \in \mathcal{S}_t} \frac{|\bar{\mathcal{D}}_i|}{|\bar{\mathcal{D}}^t|}(w_{e,i}^t, w_{c,i}^t)$
25: **return** w^{t+1}

---

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 ith client's "data scarce" classes. Having received $\mathcal{H}_i$, the server identifies client j having the set of data

---

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

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. The proposed method for client matching is formalized as Matching Algorithm in supplementary material section 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 supplementary material section 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. 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 ith 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, and then applies the local encoder/classifier to find its prediction $\tilde{y}$,

$$\tilde{x}^{\mathcal{C}_i^j} = \mathbf{w_d}(\tilde{z}_i^{\mathcal{C}_i^j}), \quad \tilde{y} = w_{c,i}^t(w_{e,i}^t(\tilde{x}^{\mathcal{C}_i^j})). \tag{2}$$

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 supplementary material section 8.

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

---

[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.

achieves performance superior to the global classifier since 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

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*

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} \Pr\left[\mathcal{M}\left(d'\right) \in S\right] + \delta. \qquad (3)$$

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.

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

$$\mathcal{M}(d) = f(d) + \mathcal{N}\left(0, S_f^2 \cdot \sigma^2\right), \qquad (4)$$

where S_f denotes the maximum of the absolute distance $|f(d) - f(d')|$, 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 $\delta \geq \frac{4}{5} \exp\left(-(\sigma\epsilon)^2/2\right)$.

While prior work on data augmentation in federated learning relied on data averaging for privacy protection [25, 43], 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 [6, 13, 33, 35]. 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 jth 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 jth 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. 1) 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 [27], using Adam [17] 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
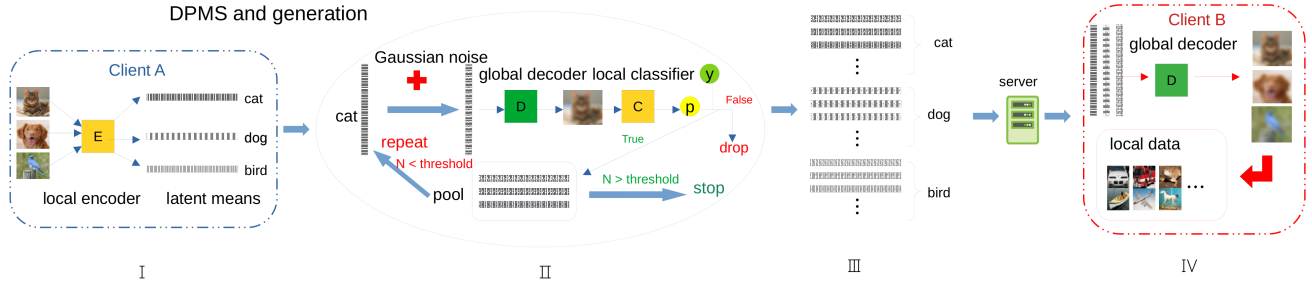
Figure 1. FedDPMS and synthetic data generation. The four parts of the figure depict: (1) finding latent representation of raw data via a local encoder; (2) creating noisy latent means (by adding Gaussian noise to the means of latent data representations) and filtering out unusable ones with the help 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.

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 [40], CIFAR10, and CIFAR100 [18]. To control the degree of imbalance in the partitioned data, we utilize Dirichlet distribution [20, 43, 44] 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 supplementary material section 5. We compare the test accuracy of FedDPMS with four state-of-the-art federated learning methods including FedAvg [26], FedProx [23], FedMix [43] and Moon [20]. The benchmarking experiments utilize VAEs and a convolutional neural network trained to perform image classification tasks. For detailed specification of the network architecture and hyperparameters, please see supplementary material section 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.

Effect of data heterogeneity. As previously stated, the concentration parameter β controls heterogeneity of the data partitions – smaller β leads to more severe class im-

Table 1. Test accuracy of FedDPMS and the competing methods on FMNIST, CIFAR10 and CIFAR100 datasets. We run five trials with different random seeds and report the mean accuracy.

Scheme	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
FedDPMS	**0.8759**	**0.6797**	**0.3861**

balance. 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 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 ex-
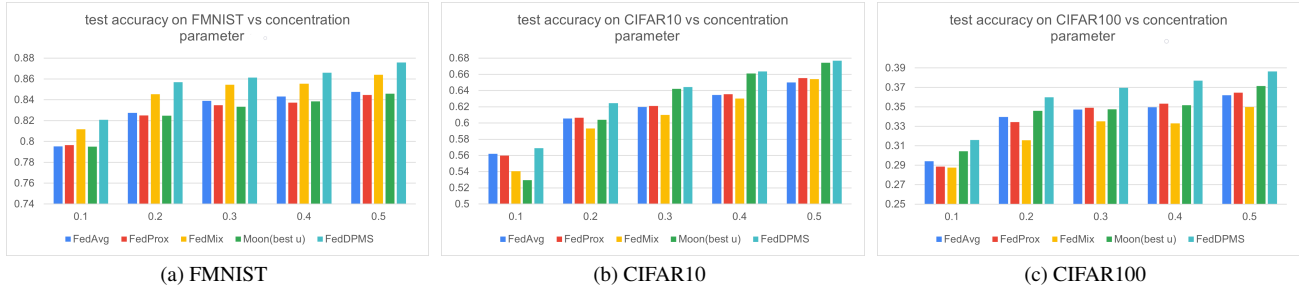
(a) FMNIST   (b) CIFAR10   (c) CIFAR100

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.
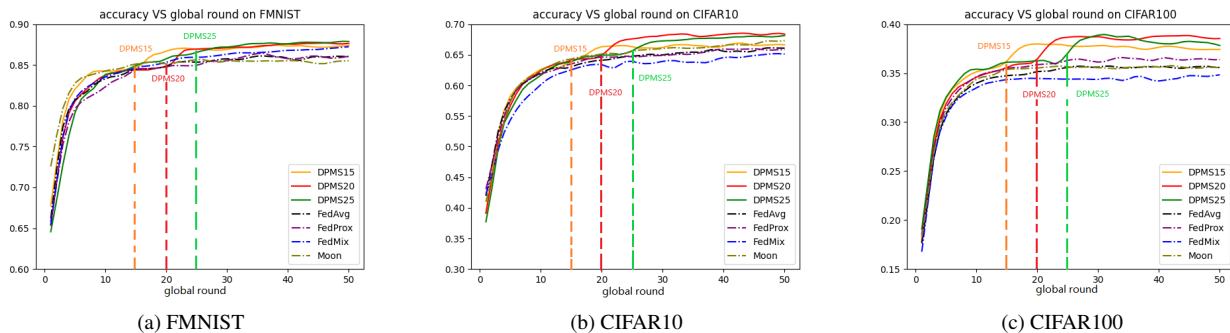

(a) FMNIST   (b) CIFAR10   (c) CIFAR100

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.

Table 2. Test accuracy as the number of clients varies from 10 to 50 in steps of 10; the clients utilize 10%-50% samples of CIFAR100 with class partitions generated via Dirichlet distribution with the concentration parameter $\beta = 0.5$. We run three trials with different random seeds and report the mean accuracy.

SCHEME	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
FEDDPMS	**0.2301**	**0.2695**	**0.2940**	**0.3061**	**0.3208**

ample, relative improvement of FedDPMS over FedAvg is more than 22.3%, 15.6%, 7.3%, 7.7% and 4.6% in the experiments involving 10-50 clients, respectively.

Convergence rate. Fig. 3 shows the test accuracy of all approaches across the federated training rounds. For Fed-DPMS, 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

[1] 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. 2

[2] Sean Augenstein, H Brendan McMahan, Daniel Ramage, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019. 1

[3] Sean Augenstein, H Brendan McMahan, Daniel Ramage, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019. 2

[4] 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. 1

[5] 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*, pages 3407–3416. PMLR, 2021. 1

[6] 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*, pages 1322–1333, 2015. 6

[7] 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*, pages 10112–10121, 2022. 2

[8] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*, 2020. 1

[9] 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. 2

[10] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *International Conference on Machine Learning*, pages 3973–3983. PMLR, 2020. 1

[11] 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*, pages 3310–3319, 2021. 2

[12] Chaoyang He, Shen Li, Mahdi Soltanolkotabi, and Salman Avestimehr. Pipetransformer: Automated elastic pipelining for distributed training of transformers. *arXiv preprint arXiv:2102.03161*, 2021. 1

[13] 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*, pages 603–618, 2017. 6

[14] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013. 2

[15] Peter Kairouz, H Brendan McMahan, Brendan Avent, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019. 1

[16] 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. 1, 2

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[18] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 7

[19] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 1

[20] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1, 2, 7

[21] Qinbin Li, Zeyi Wen, and Bingsheng He. Practical federated gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4642–4649, 2020. 1

[22] 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, 2020. 1

[23] 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. 1, 2, 7

[24] 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*, pages 740–755. Springer, 2014. 1

[25] 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*, pages 1013–1023, 2021. 2, 6

[26] 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*, pages 1273–1282. PMLR, 2017. 1, 7

[27] 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. 6

[28] Mohammad Rasouli, Tao Sun, et al. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228*, 2020. 1

[29] 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. 1

[30] 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. 1

[31] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. 2

[32] 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*, pages 9120–9132. PMLR, 2020. 1

[33] 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*, pages 0–0, 2019. 6

[34] Aleksei Triastcyn and Boi Faltings. Federated generative privacy. *IEEE Intelligent Systems*, 35(4):50–57, 2020. 2

[35] Aleksei Triastcyn and Boi Faltings. Federated generative privacy. *IEEE Intelligent Systems*, 35(4):50–57, 2020. 6

[36] 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. 2

[37] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020. 2

[38] 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, 2020. 2

[39] 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*, pages 3377–3386, 2022. 2

[40] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 2, 7

[41] 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. 1

[42] 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. 1

[43] 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. 2, 6, 7

[44] 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*, pages 7252–7261. PMLR, 2019. 2, 7

[45] 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. 1

[46] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2