Appendices

List of Appendices

A Related Work

Distributional heterogeneity in FL. Distributional heterogeneity is one of the most important patterns in federated learning [Kairouz et al., 2019, Wang et al., 2021]. Existing literature on FL heterogeneity is mostly focused on the impact of heterogeneity on the training efficiency (convergence and communication) of federated optimizers [Karimireddy et al., 2020, Li et al., 2020b, Reddi et al., 2021]. In this work, we identify that the participation gap is another major outcome of the heterogeneity in FL, and recommend using the participation gap as a natural measurement for dataset heterogeneity.

Personalized FL. In this work, we propose to evaluate and distinguish the generalization performance of clients participating and non-participating in training. Throughout this work, we focus on the classic FL setting [Kairouz et al., 2019, Wang et al., 2021] in which a single global model is learned

from and served to all clients. In the *personalized* FL setting [Hanzely and Richtárik, 2020, Fallah et al., 2020, Singhal et al., 2021], the goal is to learn and serve different models for different clients. While related, our focus and contribution is orthogonal to personalization. In fact, our three-way split framework can be readily applied in various personalized FL settings. For example, for personalization via fine-tuning [Wang et al., 2019, Yu et al., 2020], the participating clients can be defined as the clients that contribute to the training of the base model. The participation gap can then be defined as the difference in post-fine-tuned performance between participating clients and unparticipating clients.

Out-of-distribution generalization. In this work, we propose to train models using a set of participating clients and examine their performance on heldout data from these clients as well as an additional set of non-participating clients. Because each client has a different data distribution, unparticipating clients' data exhibits distributional shift compared to the participating clients' validation data. Therefore, our work is related to the field of domain adaptation [Daumé III, 2009, Ben-David et al., 2007, Shimodaira, 2000, Patel et al., 2015], where a model is explicitly adapted to make predictions on a test set that is not identically distributed to the training set. The participation gap that we observe is consistent with findings from the out-of-distribution research community [Ovadia et al., 2019, Amodei et al., 2016, Lakshminarayanan et al., 2016], which shows on centrally trained (non-federated) models that even small deviations in the distribution of deployment examples can lead to systematic degradations in performance. Our setting differs from these other settings in that our problem framing assumes data is drawn from a distribution of client distributions, making the problem of generalizing to unseen distributions potentially more tractable.

Recent years have observed a booming interest in various aspects of Federated Learning, including communication-efficient learning [McMahan et al., 2017, Konečný et al., 2016, Zhou and Cong, 2018, Haddadpour et al., 2019a, Wang and Joshi, 2018, Yu and Jin, 2019, Yu et al., 2019, Basu et al., 2019, Stich, 2019, Khaled et al., 2020, Yuan and Ma, 2020, Woodworth et al., 2020, Yuan et al., 2021], model ensemble [Bistritz et al., 2020, He et al., 2020, Lin et al., 2020, Chen and Chao, 2021], integration with compression [Faghri et al., 2020, Gorbunov et al., 2020, Sohn et al., 2020, Beznosikov et al., 2020, Horváth and Richtárik, 2020, Albasyoni et al., 2020, Jiang et al., 2020, Islamov et al., 2021], systems heterogeneity [Smith et al., 2017, Diao et al., 2020], data (distributional) heterogeneity [Haddadpour et al., 2019b, Khaled et al., 2020, Li et al., 2020d, Koloskova et al., 2020, Woodworth et al., 2020, Mohri et al., 2019, Zhang et al., 2020, Li et al., 2020b, Wang et al., 2020a, Karimireddy et al., 2020, Pathak and Wainwright, 2020, Al-Shedivat et al., 2021], fairness [Wang et al., 2020b, Li et al., 2020c, Mohri et al., 2019], personalization [Smith et al., 2017, Nichol et al., 2018, Khodak et al., 2019, Balcan et al., 2019, Jiang et al., 2019, Wang et al., 2019, Chen et al., 2019, Fallah et al., 2020, Hanzely et al., 2020, London, 2020, T. Dinh et al., 2020, Yu et al., 2020, Hanzely and Richtárik, 2020, Agarwal et al., 2020, Deng et al., 2020], and privacy [Balle et al., 2020, Chen et al., 2020, Geiping et al., 2020, London, 2020, So et al., 2020, Zhu et al., 2020, Brown et al., 2020]. These works often study generalization and convergence for newly proposed algorithms. But, to our knowledge, there is no existing work that disentangles out-of-sample and participation gaps in federated training. We refer readers to [Kairouz et al., 2019, Wang et al., 2021] for a more comprehensive survey on the recent progress in Federated Learning.

B Experimental Evaluation

We conduct experiments on four image classification tasks: EMNIST-10 (digits only), EMNIST-62 (digits and characters) [Cohen et al., 2017, Caldas et al., 2019], CIFAR-10 and CIFAR-100 [Krizhevsky et al., 2009]; and two next character/word prediction tasks: Shakespeare [Caldas et al., 2019] and StackOverflow [Reddi et al., 2021]. The detailed setups (including model, dataset preprocessing, hyperparameter tuning) are relegated to Appendix D. We summarize our main results in Figure 1 and Table 1. In the following subsections, we provide more detailed ablation studies regarding the effect of various aspects of the training process on generalization performance.

B.1 Effect of the number of participating clients

In this subsection we study the effect of the number of participating clients on the generalization performance on various tasks. To this end, we randomly sample subsets of clients of different scales as partcipating clients, and perform federated training with the same settings otherwise. The results

Table 1: **Summary of experimental results.** We perform federated and centralized training across six tasks. EMNIST, Shakespeare, and StackOverflow are naturally-partitioned, and CIFAR is semantically partitioned. Observe that the participation gaps (gap between `part_val` and `unpart`) are consistent across tasks. We provide other metric statistics across clients (such as percentiles of metrics) in Table 2 of Appendix C.2.

	Federated Training			Centralized Training		
	`part_train`	`part_val`	`unpart`	`part_train`	`part_val`	`unpart`
EMNIST-10	100.0	99.6	98.9	100.0	99.5	98.9
EMNIST-62	91.8	86.3	85.4	93.8	87.1	86.1
CIFAR-10	97.5	83.3	81.6	99.7	86.3	84.9
CIFAR-100	99.8	57.2	54.2	99.9	59.7	55.4
Shakespeare	58.8	56.5	56.2	60.7	57.2	56.8
StackOverflow	26.4	25.5	25.2	27.9	26.2	25.6

are shown in Figure 7. As the number of participating clients increases, the unparticipating accuracy monotonically improves, and the participation gap tends to decrease.[7] This is consistent with our theoretical understanding, as the participating clients can be interpreted as a discretization of the overall clients population distribution.

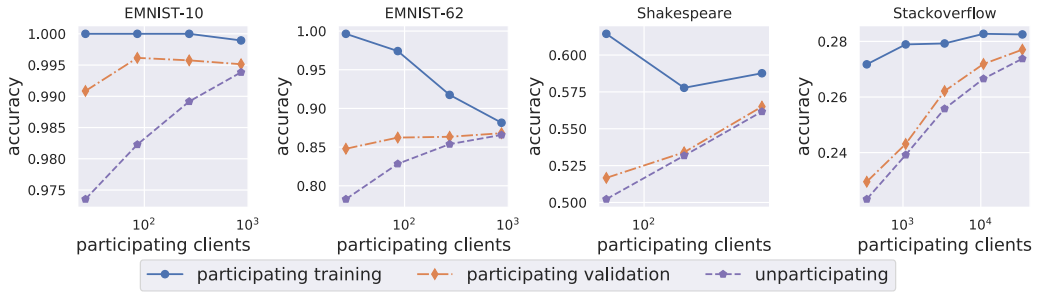

Figure 7: **Effect of the number of participating clients.** See Appendix B.1 for discussion.

B.2 Effect of client diversity

In this subsection, we study the effect of client diversity on generalization performance. Recall that in the previous subsection, we vary the number of participating clients while keeping the amount of training data per client unchanged. As a result, the total amount of training data will grow proportionally with the number of participating clients.

To disentangle the effect of diversity and the growth of training data size, in the following experiment, we instead fix the total amount of the training data, while varying the concentration across clients. The experiment is conducted on the EMNIST digits dataset. As shown in Figure 8, the training data from a new participating client can be more valuable than those contributed by the existing participating clients. The intuition can also be justified by our model in that the data from a new participating client is drawn from the overall population distribution $\int_{c\in\mathfrak{C}} p_{\mathcal{P}}(c)p_{\mathcal{D}_c}(\xi)\mathrm{d}c$, whereas the data from existing clients are drawn from the distribution aggregated by existing clients $\frac{1}{|\hat{\mathfrak{C}}|}\sum_{c\in\hat{\mathfrak{C}}} p_{\mathcal{D}_c}(\xi)$. This reveals the importance of client diversity in federated training.

C Additional Experimental Results

In this section, we present several experimental results omitted from the main body due to space constraints. Additional task-specific ablation experiments can be found in Appendix D.

[7] One exception is the Stack Overflow next word prediction task, in which the participating validation accuracy also improves significantly with increasing number of participating clients. This is likely due to the fact that the model is under-fitted when the number of participating clients is too small.

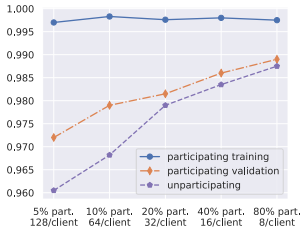

Figure 8: **Effect of diversity on generalization.** We fix the total amount of training data while varying the concentration across clients. The concentration varies from taking only 5% clients as participating clients where each client contributes 128 training data, to the most diverse distribution with 80% clients as participating clients but each client only contributes 8 training data. Observe that while the total amount of training data is identical, the more diverse settings exhibit better performance in terms of both participating validation and unparticipating accuracy.

C.1 Training progress of centralized optimizers

In this subsection, we repeat the experiment in Figure 1 with centralized optimizers. The results are shown in Figure 9. Observe that participation gap still exists with centralized optimizers. This is because participation gap is an intrinsic outcome of the heterogeneity of federated dataset.

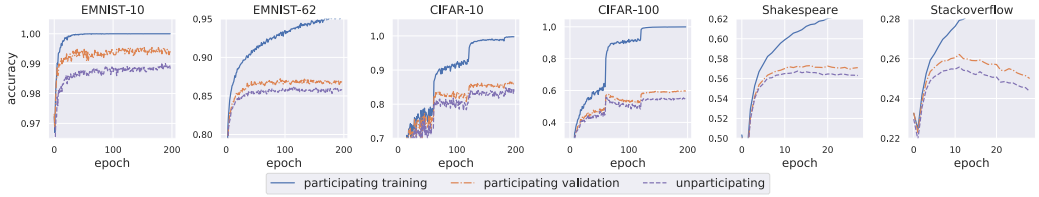

Figure 9: **Centralized training progress on six different federated tasks.** Observe that the participation gap still exists even with centralized optimizers. We refer readers to Table 1 for a quantitative comparison between federated optimizers and centralized optimizers.

C.2 Percentiles of metrics across clients

In this subsection, we report the detailed statistics of metrics across clients. Recall that in Table 1, we aggregated the metrics across clients by weighted averaging, where the weights are determined by the number of elements contributed by each client. In the following Table 2, we report five percentiles of metrics across clients: 95^{th}, 75^{th}, 50^{th} (a.k.a. median), 25^{th}, and 5^{th}. These statistics provide a detailed characterization on the metrics distribution across clients.[8]

C.3 Federated training progress at the 25^{th} percentile acorss clients

To further inspect the distribution of metrics across clients, we plot the 25^{th} percentile of accuracy across clients versus the communication rounds (training progress). The results are shown in Figure 10.

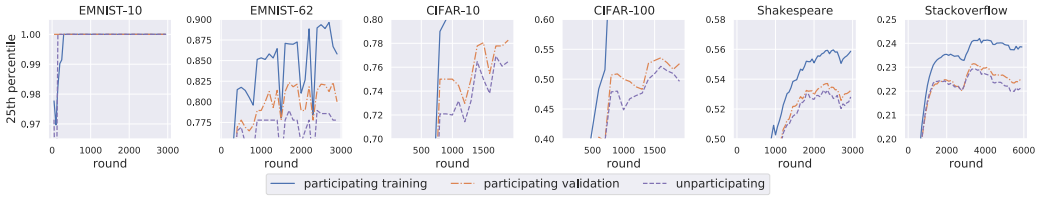

Figure 10: **Accuracies of the client at the 25^{th} percentile versus the communication rounds.**

[8] To make the percentiles comparable, we ensure the un-participating clients and participating validation clients to have the same scale of elements per client.

Table 2: **Percentiles of metrics across clients on six federated tasks.** We observe that the unparticipating clients tend to exhibit longer tails on the lower side of accuracy. For example, the participating clients of EMNIST-10 have perfect (100%) accuracy even for clients at the 5th percentile, whereas the unparticipating clients only achieve 91.7%.

	percentile	Federated Training		Centralized Training	
		`part_val`	`unpart`	`part_val`	`unpart`
EMNIST-10	95th	100.0	100.0	100.0	100.0
	75th	100.0	100.0	100.0	100.0
	50th	100.0	100.0	100.0	100.0
	25th	100.0	100.0	100.0	100.0
	5th	100.0	91.7	100.0	91.7
EMNIST-62	95th	100.0	100.0	100.0	100.0
	75th	93.3	92.3	93.5	93.5
	50th	88.2	87.1	87.5	87.5
	25th	82.1	78.6	81.8	79.2
	5th	66.7	64.7	71.4	70.6
CIFAR-10	95th	93.2	90.0	95.5	92.9
	75th	88.2	86.0	90.9	89.1
	50th	83.0	81.3	87.0	85.7
	25th	79.2	76.7	82.1	81.5
	5th	71.1	69.6	77.1	73.7
CIFAR-100	95th	65.4	62.4	66.7	62.6
	75th	61.1	56.7	64.2	56.6
	50th	57.3	53.8	61.3	55.7
	25th	53.9	51.9	55.5	52.8
	5th	46.9	46.4	50.4	50.4
Shakespeare	95th	68.4	71.4	70.1	71.4
	75th	60.8	60.0	61.7	61.1
	50th	57.3	56.8	58.2	57.5
	25th	54.0	53.1	54.5	53.7
	5th	38.4	38.2	42.6	40.9
StackOverflow	95th	31.0	31.3	31.8	31.4
	75th	27.7	27.7	28.8	27.9
	50th	25.6	25.4	26.3	25.9
	25th	23.5	23.2	24.2	23.7
	5th	20.1	20.0	20.8	20.7

D Additional Details on Experimental Setup and Task-Specific Experiments

In this section we provide the details of the experimental setup, including dataset preparation/preprocessing, model choice and hyperparameter tuning. We also include task-specific experiments with ablation study.

For every setting, unless otherwise stated, we tune the learning rate(s) to achieve the best sum of participating validation accuracy and unparticipating accuracy (so that the result will not be biased towards one of the accuracy).

D.1 EMNIST Hand-written Character Recognition Task

Federated Dataset Description and Proprocessing. The EMNIST dataset [Cohen et al., 2017] is a hand-written character recognition dataset derived from the NIST Special Database 19 [Grother and Flanagan, 1995]. We used the Federated version of EMNIST [Caldas et al., 2019] dataset, which is partitioned based on the writer identification. We consider both the full version (62 classes) as well as the numbers-only version (10 classes). We adopt the federated EMNIST hosted by Tensorflow Federated (TFF). In TFF, the federated EMNIST was by default split intra-cliently, namely all the clients appeared in both the "training" and "validation" dataset. To construct a three-way split, we hold out 20% of the total clients as unparticipating clients. Within each participating client, we keep the original training/validation split, e.g., the original training data that are assigned to participating clients will become participating training data. We tested the performance under various number of participating clients, as shown in Figure 7. The results reported in Table 1 are for the case with 272 participating clients.

Model, Optimizer, and Hyperparameters. We train a shallow convolutional neural network with approximately one million trainable parameters as in [Reddi et al., 2021]. For centralized training, we run 200 epochs of SGD with momentum = 0.9 with constant learning rate with batch size 50. The (centralized) learning rate is tuned from $\{10^{-2.5}, 10^{-2}, \ldots, 10^{-0.5}\}$. For federated training, we run 3000 rounds of FEDAVGM [Reddi et al., 2021] with server momentum = 0.9 and constant server and client learning rates. For each communication round, we uniformly sample 20 clients to train for 1 epoch with client batch size 20. The client and server learning rates are both tuned from $\{10^{-2}, 10^{-1.5}, \ldots, 1\}$.

D.1.1 Consistency across various hyperparameters choices

In Table 1, we only presented the best hyperparameter choice (learning rate combinations). In this subsubsection, we show that the pattern of generalization gap is consistent across various hyperparameter choices. The result is shown in Figure 11.

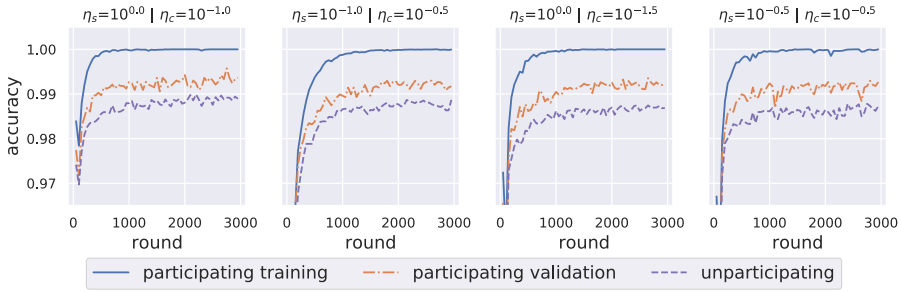

Figure 11: **Consistency of participation gaps across hyperparameter choice (learning rates configuration).** We present the best four (4) combination of learning rates for federated training of EMNIST-10. Here η_c stands for client learning rate, and η_s stands for server learning rate. We observe that the participation gap is consistent across various configurations of learning rates.

D.1.2 Effect of multiple local epochs per communication round

In the main experiments we by default let each sampled client run one (1) local epoch every communication round. In this subsubsection, we evaluate the effect of multiple local epochs on the generalization performance. The result is shown in Figure 12.

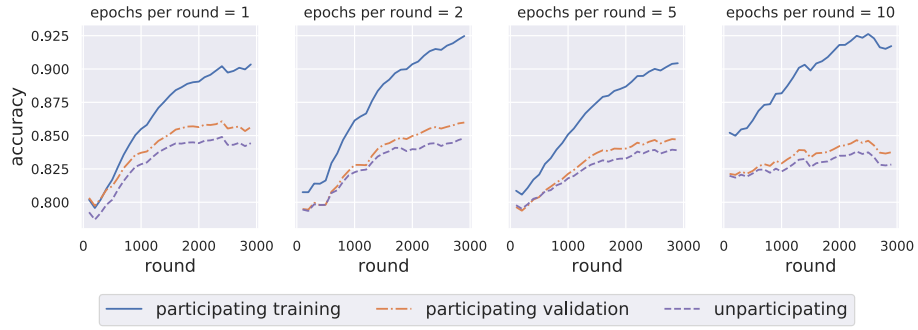

Figure 12: **Effect of multiple client epochs per round on EMNIST-62.** We repeat the experiment on EMNIST-62 but instead let each sampled client run multiple local epochs per communication round. The other settings (including the total communication rounds) remain the same. We observe that the participation gap is consistent across various settings of local epochs.

D.2 CIFAR 10/100 Image Classification Task

Federated Dataset Preprocessing. The CIFAR-10 and CIFAR-100 datasets [Krizhevsky et al., 2009] are datasets of natural images distributed into 10 and 100 classes respectively. Since the dataset does not come with user assignment, we first shuffle the original dataset and assign to clients by applying our proposed semantic synthesized partitioning. The CIFAR-10 and CIFAR-100 dataset are partitioned into 300 and 100 clients, respectively. For three-way split, we hold out 20% (60 for CIFAR-10, 20 for CIFAR-100) clients as unparticipating clients, and leave the remaining client as participating clients. Within each participating client, we hold out 20% of data as (participating) validation data.

Model, Optimizer, and Hyperparameters We train a ResNet-18 [He et al., 2016] in which the batch normalization is replaced by group normalization [Wu and He, 2018] for improved stability in federated setting, as recommended by Hsieh et al. [2019]. For centralized training, we run 200 epochs of SGD with momentum = 0.9 with batch size 50, and decay the learning rate by 5x every 60 epochs. The initial learning rate is tuned from $\{10^{-2.5}, 10^{-2}, \ldots, 10^{-0.5}\}$. For federated training, we run 2,000 rounds of FedAvgM [Reddi et al., 2021] with server momentum = 0.9, and decay the server learning rate by 5x every 600 communication rounds. For each communication round, we uniformly sample 10 clients (for CIFAR-100) or 30 clients (for CIFAR-10), and let each client train for 1 local epoch with batch size 20. The client learning rate is tuned from $\{10^{-2}, 10^{-1.5}, \ldots, 1\}$; the server learning rate is tuned from $\{10^{-1.5}, 10^{-1}, \ldots, 10^{0.5}\}$.

D.2.1 Consistency across various hyperparameters choice

In the main result Table 1 we only present the best hyperparameter choice (learning rate combinations). In this subsubsection, we show that the pattern of generalization gap is consistent across hyperparameter choice. The result is shown in Figures 13 and 14.

D.2.2 Effect of Weight Decay Strength

In the main experiments we by default set the weight decay of ResNet-18 to be 10^{-4}. In this subsubsection, we experiment various other options of weight decay from 10^{-5} to 10^{-2}

The result is shown in Figure 15.

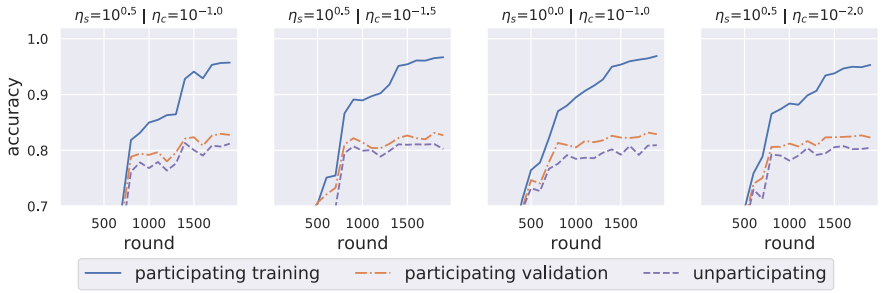

Figure 13: **Consistency of participation gaps across hyperparameter choice (learning rates configuration).** We present the best four (4) combination of learning rates for federated training of CIFAR-10. Here η_c stands for client learning rate, and η_s stands for server learning rate. We observe that the participation gap is largely consistent across various configurations of learning rates.

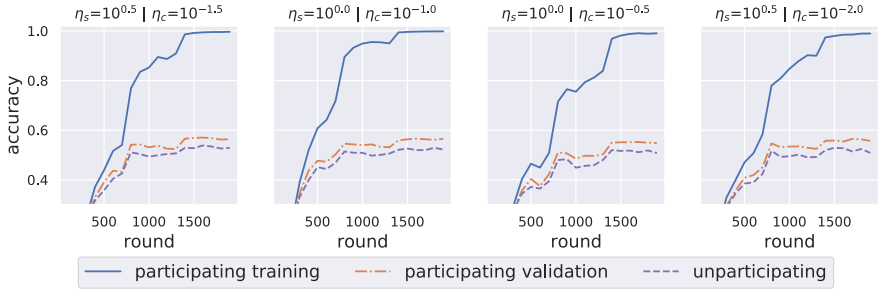

Figure 14: **Consistency of participation gaps across hyperparameter choice (learning rates configuration).** We present the best four (4) combination of learning rates for federated training of CIFAR-100. Here η_c stands for client learning rate, and η_s stands for server learning rate. We observe that the participation gap is consistent across various configurations of learning rates.

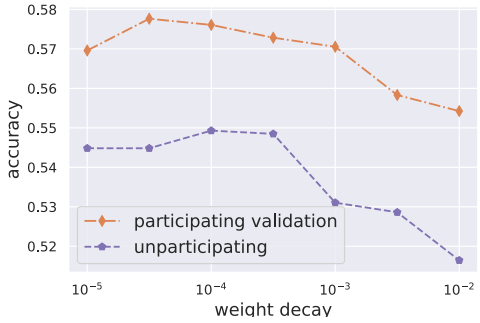

Figure 15: **Effect of ℓ_2 weight decay on CIFAR-100 training.** We federated train the ResNet-18 networks for CIFAR-100 with various levels of weight decay ranging from 10^{-5} to 10^{-2}. We observe that a moderate scale of weight decay might improve the unparticipating accuracy and therefore decrease the participation gap. However, an overlarge weight decay might hurt both participating validation and unparticipating performance.

D.2.3 Effect of Model Depth

In the main experiments we by default train a ResNet-18 for the CIFAR task. In this subsubsection, we experiment a deeper model (ResNet-50) for the CIFAR-100. The result is shown in Figure 16.

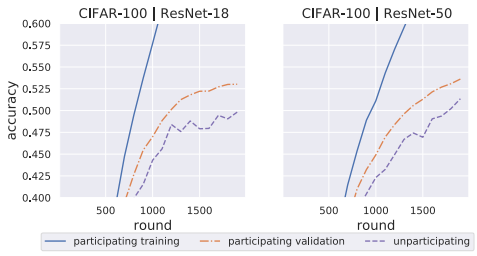

Figure 16: **Effect of a deeper ResNet on CIFAR-100 training.** We federatedly train a ResNet-50 for CIFAR-100 to compare with our default choice (ResNet-18). We apply a constant learning rate (instead of step decay learning rate) for easy comparison. We observe that while using a deeper model improves the overall accuracy, the participation gap is still reasonably large for ResNet-50.

D.3 Shakespeare Next Character Prediction Task

Federated Dataset Description and Preprocessing. The Shakespeare dataset [Caldas et al., 2019] is a next character prediction dataset containing lines from *the Complete Works of William Shakespeare* where each client is a different character from one of the plays. We adopt the federated shakespeare dataset hosted by Tensorflow Federated (TFF). In TFF, the federated shakespeare dataset was by default split intra-cliently, namely all the clients appeared in both the "training" and "validation" dataset. To construct a three-way split, we hold out 20% of the total clients as unparticipating clients, and leave the remaining (80%) clients as participating clients (which gives the result reported in Table 1). Within each participating client, we keep the original training/validation split, e.g., the original training data that are assigned to these participating clients will become participating training data. We also tested the performance under other numbers of participating clients, as shown in Figure 7.

Model, Optimizer, and Hyperparameters We train the same recurrent neural network as in [Reddi et al., 2021]. For centralized training, we run 30 epochs of Adam (with $\epsilon = 10^{-4}$) with batch size 20. We tune the centralized learning rate from $\{10^{-3}, 10^{-2.5}, \ldots, 10^{-1}\}$. For federated training, we run 3,000 rounds of FEDADAM [Reddi et al., 2021] with server $\epsilon = 10^{-4}$. For each communictaion round, we uniformly sample 10 clients, and let each client train for 1 local epoch with batch size 10. Both client and server learning rates are tuned from $\{10^{-2}, 10^{-1.5}, \ldots, 1\}$.

D.4 Stackoverflow Next Word Prediction Task

Federated Dataset Description and Preprocessing. The Stack Overflow dataset consists of questions and answers taken from the website Stack Overflow. Each client is a different user of the website. We adopt the stackoverflow dataset hosted by Tensorflow Federated (TFF). In TFF, the federated stackoverflow dataset is splitted **inter-cliently**, namely the training data and validation data belong to two disjoint subsets of clients. To construct a three-way split, we will treat the original "validation" clients as unparticipating clients. Within each participating client, we randomly hold out the max of 20% or 1000 elements as (participating) validation data, and the max of 80% or 1000 elements as (participating) training data. Due to the abundance of stackoverflow data, we randomly sample a subset of clients from the original "training" clients as participating clients. The result shown in Table 1 is for the case with 3425 participating clients. We also tested other various levels of participating clients, shown in Figure 7.

Model, Optimizer, and Hyperparameters We train the same recurent neural network as in [Reddi et al., 2021]. For centralized training, we run 30 epochs of Adam (with $\epsilon = 10^{-4}$) with batch size 200. We tune the centralized learning rate from $\{10^{-3}, 10^{-2.5}, \ldots, 10^{-1.5}\}$. For federated training, we run 6,000 rounds of FEDADAM [Reddi et al., 2021] with server $\epsilon = 10^{-4}$. For each communictaion round, we randomly sample 100 clients, and let each client train for 1 local epoch with batch size 50. Both client and server learning rates are tuned from $\{10^{-2}, 10^{-1.5}, \ldots, 1\}$. The client learning rate is tuned from $\{10^{-3}, 10^{-1.5}, \ldots, 10^{-1}\}$; the server learning rate is tuned from $\{10^{-2}, 10^{-1.5}, \ldots, 1\}$.

E Details of Semanticically Partitioned Federated Dataset

E.1 Details of the Semantic Partitioning Scheme

In this section we provide the details of the proposed algorithm to semantically partition a federated dataset for CIFAR-10 and CIFAR-100. For clarify, we use K to denote the number of classes, and C to denote the number of clients partitioned into.

The first stage aims to cluster each label into C clusters.

1. Embed the original inputs of dataset using a pretrained EfficientNetB3. This gives a embedding of dimension 1280 for each input.
2. Reduce the dimension of the above embeddings to 256 dimensions via PCA.
3. For each label, fit the corresponding input with a Gaussian mixture model with C clusters. This step yields C gaussian distribution for each of the K labels. Formally, we let $\mathcal{D}_c^k$ denote the (Gaussian) distribution of the cluster c of label k.

The second stage will package the clusters from different labels across clients. We aim to compute an optimal multi-partite matching with cost-matrix defined by KL-divergence between the Gaussian clusters. To reduce complexity, we heuristically solve the optimal multi-partite matching by progressively solving the optimal bipartite matching at each time for some randomly-chosen label pairs. Formally, we run the following procedure

---

Initialize $S_{\text{unmatched}} \leftarrow \{1, \dots, K\}$
Randomly sample a label k from $S_{\text{unmatched}}$, and remove k from $S_{\text{unmatched}}$.
while $S_{\text{unmatched}} \neq \emptyset$ **do**
    Randomly sample a label k' from $S_{\text{unmatched}}$, and remove k' from $S_{\text{unmatched}}$.
    Compute a cost matrix $\boldsymbol{A}$ of dimension $C \times C$, where $\boldsymbol{A}_{ij} \leftarrow D_{\text{KL}}(\mathcal{D}_i^k || \mathcal{D}_j^{k'})$.
    Solve and record the optimal bipartite matching with cost matrix $\boldsymbol{A}$.
    Set $k \leftarrow k'$
return the aggregation of all the bipartite matchings computed.

E.2 Visualization of Semanticically Partitioned CIFAR-100 Dataset

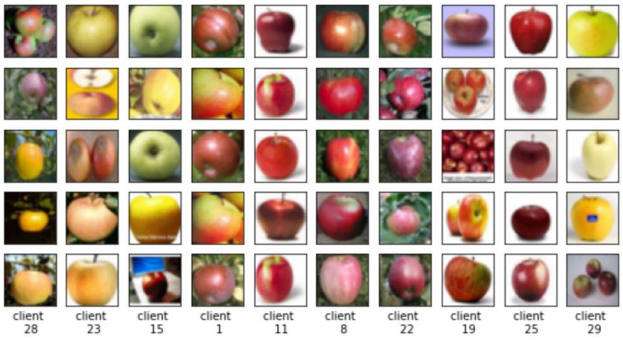

Figure 17: **Visualization of semantic partitioning of CIFAR-100.** We partition the CIFAR-100 dataset into 100 clients without resorting to external user information (such as writer identification). Here we show 10 out of 100 clients featuring the label "apple".

E.3 Visualization of Semantically Partitioned MNIST Dataset

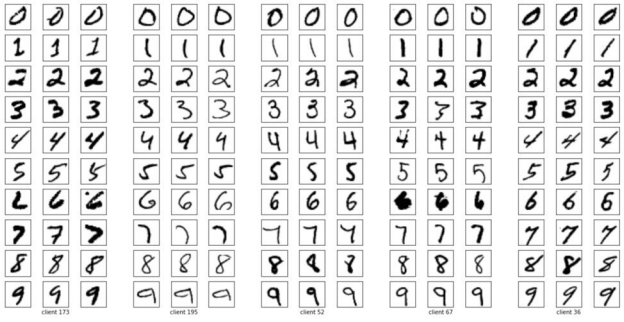

Figure 18: **Visualization of semantic partitioning of MNIST.** We partition the (classic) MNIST dataset into 300 clients without resorting to external user information (such as writer identification). Here we show 5 out of 300 clients. Observe that the images within each client demonstrates consistent writing styles both within label and across labels.

F Methodology for Computing Entropy

We hypothesize that a participation gap exists for naturally partitioned datasets and not for synthetically partitioned datasets because the naturally partitioned datasets inherently contain correlated inputs not drawn IID from the full data generating distribution. Put another way, the entropy of the input data for a given label from a naturally partitioned client is lower than the entropy for that same label from a synthetically partitioned client. To evaluate this claim, we need to (approximately) infer the data generating distribution for each client, and then measure the entropy of this distribution, defined as:

$$H(q) = -\mathbb{E}_{x \sim q(x)} \log q(x) \tag{7}$$

To infer the client data generating distribution, we used deep generative models. Because our clients possess relatively few training examples ($\mathcal{O}(10)$ for a particular class), many deep generative models such as Glow [Kingma and Dhariwal, 2018] or PixelCNN [Salimans et al., 2017] will not be able to learn a reasonable density model. We instead used a Variational Autoencoder [Kingma and Welling, 2013] to approximate the deep generative process. This model is significantly easier to train compared to the much larger generative models, but does not have tractable log-evidence measurement. Instead, models are trained by minimizing the negative Evidence Lower Bound (ELBO).

We filtered each client to contain data only for a single label. Because of the sparseness of the data after filtering, we found that a 2 dimensional latent space was sufficient to compress our data without significant losses. We used a Multivariate Normal distribution for our posterior and prior, and an Independent Bernoulli distribution for our likelihood. The posterior was given a full covariance matrix to account for correlations in the latent variable. All models were trained for 10^4 training steps.

In order to evaluate our models, we used a stochastic approximation to the log-evidence, given by a 1000 sample IWAE [Burda et al., 2015]. IWAE is a lower bound on the Bayesian Evidence that becomes asymptotically tight when computed with a large number of samples. We evaluated the entropy for 100 clients from naturally partitioned, syntactically partitioned, and synthetically partitioned datasets, and computed the average across clients as our estimate for the client data entropy. We find that synthetic partitioning results in an average client entropy of 50 Nats, while Natural partitioning results in clients with only 40 Nats of entropy. Syntactic partitioning falls in between these two, having 45 Nats of entropy.