Fine-tuning Global Model via Data-Free Knowledge Distillation for Non-IID Federated Learning

Lin Zhang\textsuperscript{1,4*} Li Shen\textsuperscript{2} Liang Ding\textsuperscript{3} Dacheng Tao\textsuperscript{2,3} Ling-Yu Duan\textsuperscript{1,4†}

\textsuperscript{1}Peking University, Beijing, China \textsuperscript{2}JD Explore Academy, Beijing, China \textsuperscript{3}The University of Sydney, Sydney, Australia \textsuperscript{4}Peng Cheng Laboratory, Shenzhen, China

{zhanglin.imre, lingyu}@pku.edu.cn, {mathshenli, dacheng.tao}@gmail.com
ldin3097@sydney.edu.au

Abstract

Federated Learning (FL) is an emerging distributed learning paradigm under privacy constraint. Data heterogeneity is one of the main challenges in FL, which results in slow convergence and degraded performance. Most existing approaches only tackle the heterogeneity challenge by restricting the local model update in client, ignoring the performance drop caused by direct global model aggregation. Instead, we propose a data-free knowledge distillation method to fine-tune the global model in the server (FedFTG), which relieves the issue of direct model aggregation. Concretely, FedFTG explores the input space of local models through a generator, and uses it to transfer the knowledge from local models to the global model. Besides, we propose a hard sample mining scheme to achieve effective knowledge distillation throughout the training. In addition, we develop customized label sampling and class-level ensemble to achieve effective knowledge distillation throughout the training. In addition, we develop customized label sampling and class-level ensemble to achieve maximum utilization of knowledge, which implicitly mitigates the distribution discrepancy across clients. Extensive experiments show that our FedFTG significantly outperforms the state-of-the-art (SOTA) FL algorithms and can serve as a strong plugin for enhancing FedAvg, FedProx, FedDyn, and SCAFFOLD.

1. Introduction

With the explosive growth of data and the strict privacy-protection policy, reckless data transmission and aggregation gradually become unacceptable due to the high bandwidth cost and risk of privacy leakage. Recently, Federated Learning (FL) \cite{30,31} has been proposed to replace the traditional heavily centralized learning paradigm and protect data privacy. It has been successfully applied in real-world tasks, such as smart city \cite{16,34,43}, health care \cite{8,26,27}, etc.

One of the main challenges in FL is the data heterogeneity, i.e., the data in clients are non-identically and independently distributed (Non-IID). It has been verified that the vanilla FL algorithm, FedAvg \cite{30}, leads to drifted local models and forgets the global knowledge catastrophically in this scenario, which further induces degraded performance and slow convergence \cite{14,19,21}. This is because the local model is updated merely with local data, i.e., minimizing the local empirical loss. However, minimizing the local empirical loss is fundamentally inconsistent with minimizing the global empirical loss \cite{1,24,29} in Non-IID FL.

To tackle the data heterogeneity challenge, most existing methods, e.g., FedProx \cite{23}, SCAFFOLD \cite{18}, FedDyn \cite{1}, MOON \cite{22} constrain the direction of local model update to align the local and global optimization objectives. Recently, FedGen \cite{44} learns a lightweight generator to generate pseudo feature and broadcasts it to clients to regulate local training. However, all these methods merely conduct simple model aggregation to get the global model in server, which ignores local knowledge incompatibility and induces knowledge forgetting in the global model. In addition, \cite{37}...
shows that directly aggregating models will largely degrade the performance while fine-tuning can greatly boost the accuracy. These motivate us to fine-tune the aggregated global model in the server with the knowledge in local models. On the other hand, merely aggregating local models in server ignores the server’s rich computing resources that could be potentially utilized to improve the performance of FL, such as the computing source in cross-silo FL [17].

Motivated by these observations, we propose a novel approach that boosts the performance of standard FL by on-the-fly fine-tuning the global model via data-free knowledge distillation (FedFTG), which simultaneously refines the model aggregation procedure and exploits the rich computing power of the server. Concretely, FedFTG models the input space of local models through an auxiliary generator in the server, then generates pseudo data to transfer the knowledge in local models to the global model to improve the performance. To facilitate effective knowledge distillation throughout the training, FedFTG iteratively explores the hard samples in data distribution, which will induce prediction disagreement between local models and global model. Figure 1 compares FedFTG with FedAvg. FedFTG fine-tunes the global model with the hard samples to correct the model shift after model aggregation. The generator and global model are adversarially trained in a data-free manner, thus the whole procedure will not violate the privacy policy in FL. Considering the label distribution shift in data heterogeneity scenario, we further propose customized label sampling and class-level ensemble techniques, which explore the distribution correlation of clients and exploit maximum utilization of knowledge.

FedFTG is orthogonal to several existing local optimizers, such as FedAvg, FedProx, FedDyn, SCAFFOLD and MOON, as it only modifies the procedure of global model aggregation in the server. Consequently, FedFTG can be seamlessly embedded into these local FL optimizers, taking their advantages to further improve the performance of FedFTG. Extensive experiments on various settings verify that FedFTG achieves superior performance compared with state-of-the-art (SOTA) methods.

The main contributions of this work are four-fold:

- We propose FedFTG to fine-tune the global model in server via data-free distillation, which simultaneously enhances the model aggregation step and utilizes the computing power of the server.
- We develop hard sample mining to effectively transfer knowledge to global model. Besides, we propose customized label sampling and class-level ensemble to facilitate maximum utilization of knowledge.
- We demonstrate that FedFTG is orthogonal to exiting local optimizers and can serve as a strong and versatile plugin to enhance the performance of FedAvg, FedProx, FedDyn, SCAFFOLD and MOON.
- We verify the superiority of FedFTG against several SOTA methods for FL, including FedAvg, FedProx, FedDyn, SCAFFOLD, MOON, FedGen and FedDF, with extensive experiments on five benchmarks.

2. Related Work

There exist extensive works on improving the global performance of FL via client selection [4, 7, 15], split learning [11, 39], domain adaptation [26, 33], etc. The readers may refer to monographs [17, 38] and the reference therein to follow up its recent advances. Below, we mainly summarize the most relevant techniques to our work.

**Federated Optimizer.** The vanilla FL algorithm, i.e. FedAvg [30] periodically aggregates the local models in server and updates the local model with its individual data. FedProx [23] adds a proximal term to the local subproblem to restrict the local update closer to the initial (global) model. SCAFFOLD [18] uses a variance reduction technique to correct the drifted local update. FedDyn [1] modifies the objective of client with linear and quadratic penalty terms to align global and local objectives. In summary, all these methods focus on aligning the local and global model to narrow the distribution drift during the local training without enhancing the global model directly as in FedFTG.

**Knowledge Distillation in Federated Learning.** With the help of an unlabeled dataset, FedDF [25] proposes an ensemble distillation for model fusion, trains the global model using the averaged logits from local models. FedAux [35] finds a model initialization for the local models, and weights the logits from local models using $(\epsilon, \delta)$-differentially private certainty scoring. FedBE [3] generates a series of global models from Bayesian perspective using the local models, then summarizes these models into one global model by ensemble knowledge distillation. All these methods rely on an unlabeled auxiliary dataset in the server, while it is unclear to which extent should the auxiliary dataset be related to training data to guarantee effective knowledge distillation. Though FedDF maintains the auxiliary dataset can be replaced with a pretrained generator, it does not instantiate how to acquire the generator.

**Data-Free Knowledge Distillation (DFKD).** DFKD methods [2, 6] generate pseudo data from a pretrained teacher model, and use them to transfer knowledge of teacher model to another student model. The data is generated by maximizing the response of fake data on teacher model. DeepImpression [32] models the output space of teacher model and recovers the real data by fitting the output space. DeepInversion [41] further optimizes the pseudo data by regularizing the distribution of intermediate feature maps. DAF [2] and DFAD [6] use a generator to generate data efficiently, where DAF optimizes the generator by maximizing the response on prediction and feature level, and DFAD uses an adversarial training scheme to exploit
the knowledge in teacher model effectively.

FedGen also [44] learns a lightweight generator to ensemble knowledge of local models in a data-free manner, but uses the generator to regularize the local training. Besides, we design hard samples mining scheme, customized label sampling, and class-level ensemble to effectively transfer the knowledge from local models to global model in data heterogeneity scenario.

3. Methodology
In this section, we describe the proposed novel federated learning method: FedFTG. In each communication round, FedFTG randomly selects a set of clients and broadcasts the global model to them. Each client initializes the local model using the global model and trains it with a local optimizer. The server collects the local models and aggregates them as a preliminary global model. Instead of broadcasting the aggregated model back to each client directly, FedFTG fine-tunes this preliminary global model in server using the knowledge extracted from local models. Concretely, we develop a data-free knowledge distillation method with hard sample mining to effectively explore and transfer the knowledge to global model. Considering the label distribution shift in clients, we propose customized label sampling and class-level ensemble to facilitate more effective knowledge utilization. Figure 2 visualizes the training procedure on the server, and the corresponding algorithm is summarized in Algorithms 1 & 2. Note that FedFTG is orthogonal to efforts on optimizing local model training, such as SCAFFOLD, FedAvg, FedProx, and FedDyn.

### 3.1. Data-Free Knowledge Distillation With Hard Sample Mining for Global Model Fine-Tuning

Let $\omega$ be the model parameter in the server and clients. In this work, we consider there exist $K$ clients, where

\[ \mathcal{D}_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{N_k} \]

is the dataset individually stored in $k$-th client, $N_k$ is the corresponding number of samples. Generally speaking, federated learning can be formulated as the following problem:

\[
\min_{\omega} \frac{1}{K} \sum_{k=1}^{K} f_k(\omega), \quad f_k(\omega) = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathcal{L}(x_{k,i}, y_{k,i}; \omega),
\]

where $\mathcal{L}$ is the loss function to measure training error, and dataset $\mathcal{D}_k$ for each $k \in \{1, 2, ..., K\}$ could be distributed heterogeneously. Due to the privacy protection constraint in FL, the server can not directly access local data of clients. To solve Eq. (1), for each communication round $t$, existing methods send the global model $\omega$ to a random set of clients $S_t$ and optimize it by $\min_{\omega} f_k(\omega), k \in S_t$. The server collects the local models $\{\omega_k\}_{k \in S_t}$, and aggregates them by averaging the gradients to update the global model $\omega$.

However, the local models are greatly drifted from each other in data heterogeneity scenario. Thus, traditional gra-
By exploring the hard samples in data distribution and fine-tuning the global model, the global model can be gradually corrected during training. Different averaging could lose the knowledge in local models, and the performance of updated global model is much lower than local models [45]. To address this issue, we propose a data-free knowledge distillation method to fine-tune the global model, so that the global model can preserve the knowledge in local models and maintain their performance as much as possible. Concretely, the server maintains a conditional generator $G$ that generates pseudo data to capture the data distribution of clients as follows,

$$\bar{x} = G(z, y; \theta),$$

where $\theta$ is the parameter of $G$, $z \sim \mathcal{N}(0, 1)$ is a standard Gaussian noise, and $y$ is the class label of $\bar{x}$ sampled from predefined distribution $p_i(y)$.

As shown in Figure 2, we then input the pseudo data $\bar{x}$ to the global model to solve the following problem,

$$\min_\omega \mathbb{E}_{z \sim \mathcal{N}(0, 1)} \mathbb{E}_{y \sim p_i(y)} \left[ L_{md} \right] = \min_\omega \mathbb{E}_{z \sim \mathcal{N}(0, 1)} \mathbb{E}_{y \sim p_i(y)} \left[ \sum_{k \in \mathcal{S}_k} \alpha_i^{k,y} \mathcal{L}_{md}^k \right]$$

where $\mathcal{L}_{md}^k$ is the model discrepancy between global model $\omega$ and local model $\omega_k$,

$$\mathcal{L}_{md}^k = D_{KL}(\sigma(D(\bar{x}; \omega)) | | \sigma(D(\bar{x}; \omega_k))),$$

where $D$ is the classifier, $\sigma$ is the softmax function, which will output the prediction score of $\bar{x}$. $D_{KL}$ denotes the Kullback-Leibler divergence. $\alpha_i^{k,y}$ controls the weight of knowledge from different local models during ensemble. By minimizing $L_{md}$, we transfer the knowledge in local models to the global model. In Section 3.2 we will introduce how to acquire $p_i(y)$ and $\alpha_i^{k,y}$ to adapt label distribution shift in data heterogeneity scenario.

**Data Fidelity and Diversity Constraints.** To better extract knowledge from local models, the pseudo data $\bar{x}$ should fit the input space of local models. Therefore, we use semantic loss $L_{cls}$ to train the generator $G$, which will facilitate the fidelity of pseudo data,

$$\min_\theta \mathbb{E}_{z \sim \mathcal{N}(0, 1)} \mathbb{E}_{y \sim p_i(y)} \left[ L_{cls} \right] = \min_\theta \mathbb{E}_{z \sim \mathcal{N}(0, 1)} \mathbb{E}_{y \sim p_i(y)} \left[ \sum_{k \in \mathcal{S}_k} \alpha_i^{k,y} \mathcal{L}_{cls}^k \right]$$

where $\mathcal{L}_{cls}^k$ is the cross-entropy loss between the prediction of local model on pseudo data $\bar{x}$ and the class label $y$,

$$\mathcal{L}_{cls}^k = \mathcal{L}_{CE}(\sigma(D(\bar{x}; \omega_k)), y),$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss. By minimizing $L_{cls}$, $\bar{x}$ is enforced to yield higher prediction on class $y$, thus it fits the data distribution of class $y$.

Simply using $L_{cls}$ will lead to model collapse of the generator: $G$ outputs the same data for every class. To address this issue, we use diversity loss $L_{dis}$ in [44] to improve the diversity of the generated data,

$$L_{dis} = e^{\frac{1}{N} \sum_{i,j \in \{1, \ldots, Q\}} \left( -\|\bar{x}_i - \bar{x}_j\|_2 \right)^2}$$

where $\bar{x}_i$ is generated using $z_i$. By minimizing $L_{dis}$, the pseudo data will be diverse and scattered in the data space.

**Hard Sample Mining.** Training the generator $G$ using $L_{cls}$ will generate pseudo data $\bar{x}$ with low classification error, which means $\bar{x}$ contains the most discriminative feature of class $y$ and is easy to be classified. However, these naive samples will not cause the prediction disagreement between global model and local models, i.e., $L_{md} = 0$, thus the global model is not optimized during training. As illustrated in Figure 3, the naive samples are already correctly classified by the global model. To effectively exploit the knowledge in local models and transfer them to the global model, we explore the hard samples in data distribution that cause prediction disagreement between local models and global model. Concretely, we adversarially train the generator and the global model with $L_{md}$: (1) the generator is enforced to generate hard samples that maximize $L_{md}$, and (2) the global model is trained to minimize $L_{md}$ using the hard samples. As a result, the global model can be gradually fine-tuned to fit the data distribution as in Figure 3.

To the end, the overall objective of FedFTG in the server is formulated as an adversarial learning scheme,

$$\min_\omega \max_\theta \mathbb{E}_{z \sim \mathcal{N}(0, 1), y \sim p_i(y)} \left[ L_{md} - \lambda_{cls} \mathcal{L}_{cls} - \lambda_{dis} \mathcal{L}_{dis} \right].$$

### 3.2. Adaptation to Label Distribution Shift for Effective Knowledge Distillation

In data heterogeneity scenario, label distributions are different among clients, i.e., $p^i(y) \neq p^j(y)$ for different clients $i$ and $j$. This indicates that: (1) the local dataset $D_k$ of client is class-imbalanced, and the local model trained by $D_k$ contains imbalanced data information; (2) for one class, the importance of knowledge are different among local models of clients. To facilitate more effective knowledge distillation, we propose customized label sampling and class-level ensemble to adapt these two problems respectively.

**Customized Label Sampling.** Typically, dataset in local client is class-imbalanced in data heterogeneity scenario, even having no data for some classes. It has been proved that deep neural networks tend to learn the majority classes and ignore the minority classes [5]. Hence, the data information of minority classes in local models could be wrong and misleading, and the generated pseudo data are invalid to measure the model discrepancy. If uniformly sample the
class label $y$, these invalid data will influence the global model training and induce performance decrease. To mitigate this issue, we customize the sampling probability $p_i(y)$ according to the distribution of whole training data in each round, so that more pseudo data with effective information can be generated,

$$p_i(y) \propto \sum_{k \in S_t} \frac{1}{N_k} \sum_{i=1}^{N_k} E(z_i) \sim D_k[1_{y_i = y}] = \sum_{k \in S_t} n_t^y_k,$$  \hspace{1cm} (9)

where $1_{\text{condition}}$ is 1 if the condition is true and 0 otherwise, $n_t^y_k$ is the instance number of class $y$ in client $k$. According to Eq. (9), the pseudo data of majority classes have high probability to be generated, thus FedFTG can guarantee effective knowledge distillation in data heterogeneity scenario.

Class-Level Ensemble. The widely used ensemble method in knowledge distillation assigns the same weight to the knowledge from different teacher models [25, 44], i.e., $\alpha_t^{k,y} = \frac{1}{|S_t|}$ in Eq. (3) and Eq. (5). Due to the label distribution shift, for one class the importance of knowledge are different among local models. If assigning the same weight to clients, the important knowledge can not be figured out and utilized properly. Therefore, we propose class-level ensemble, which assigns the ensemble weight according to the individual data distribution of clients. Specifically, we compute the weight $\alpha_t^{k,y}$ via the data proportion of class $y$ in client $k$ against the total data in $S_t$,

$$\alpha_t^{k,y} = \frac{n_t^y}{\sum_{i \in S_t} n_t^y},$$  \hspace{1cm} (10)

As a result, the knowledge from clients can be flexibly integrated according to their importance on classes, so that FedFTG can facilitate maximum utilization of knowledge from local models.

4. Experiments

In this section, we empirically verify the effectiveness of FedFTG\textsuperscript{1}. We summarize the implementation details in Section 4.1, and compare FedFTG with several SOTA FL algorithms in Section 4.2. Ablation studies are conducted to verify the necessity of each component of FedFTG in Section 4.3. To further validate the effect of FedFTG on real-world FL applications, we evaluate the performance of FedFTG on three real-world datasets in Section 4.4.

4.1. Implementation Details

Baselines. We compare FedFTG against FedAvg [30], FedProx [23], SCAFFOLD [18], FedDyn [1], MOON [22], FedGen [44] and FedDF [25]. Since FedDF does not explain how to obtain the generator, we train it in the same way as FedGen.

Datasets. CIFAR10 and CIFAR100 datasets [20] with heterogeneous dataset partition are used to test the efficacy of FedFTG, which are two difficult tasks in FL scenario and are widely adopted in FL research. Similar to existing works [1, 12, 42], we use Dirichlet distribution $\text{Dir}(\beta)$ on label radios to simulate the non-iid data distribution among clients, where a smaller $\beta$ indicates higher data heterogeneity. During the implementation, we set $\beta = 0.3$ and $\beta = 0.6$.

Network Architecture. For both CIFAR10 and CIFAR100, we employ ResNet18 [13] as the basic backbone. We borrow the generator network architecture from DFAD [6] for FedFTG and FedDF. For FedGen, the network of generator is composed of two embedding layers (for inputs $z$ and $y$, respectively) and two fully-connected (FC) layers with LeakyReLU and BatchNorm layers between them.

Hyperparameters. For all methods, we set the number of local training epoch $E = 5$, communication round $T = 1000$, the client number $K = 100$ with the active fraction $C = 0.1$ (i.e., $|S_t| = 10$). For local training, the batchsize is 50 and the weight decay is $1e^{-3}$. The learning rates for classifier and generator are initialized to be 0.1 and 0.01 respectively, and they are decayed quadratically with weight 0.998. The dimension of $z$ is 100 for CIFAR10 and 256 for CIFAR100. $I_1$, $I_g$, $I_d$ in Algorithm 2 are 10, 1 and 5, respectively. If not specifically declared, we adopt $\lambda_{dis} = 1.0$ and $\lambda_{dis} = 1.0$, and adopt SCAFFOLD as the FL optimizer in FedFTG.

We further provide detailed implementations and extra experiment results in the supplementary material.

4.2. Performance Comparison

Test Accuracy. Table 1 reports the test accuracy of all compared algorithms on CIFAR10 and CIFAR100 datasets. We provide the performance of centralized learning in the first line. All experiments are repeated over 3 random seeds. In Table 1, FedFTG achieves the best performance in all scenarios, surpassing the second one (i.e., SCAFFOLD) by at least 1.5%. FedDF also employs data-free knowledge distillation to improve the global model in server. It outperforms FedAvg and FedProx, and outperforms FedDyn and MOON in some cases, which further validates the superiority of the scheme “fine-tuning the global model in server”. However, it is worse than SCAFFOLD and FedFTG. FedGen yields lower accuracy compared with FedDF and FedFTG, and shows marginal performance gains than FedAvg in some cases. The performance of FedDF and FedGen further verifies the effectiveness of the proposed modules in FedFTG.

Communication Rounds. Table 2 evaluates different FL methods in term of the number of communication rounds to reach target test accuracy (75% and 80% for CIFAR10, 40% and 50% for CIFAR100, respectively). In Ta-

\hspace{1cm}\textsuperscript{1}Code is available at https://github.com/ZhangLin-PKU/FedFTG.
Table 1. Test Accuracy (%) of different FL methods on CIFAR10 and CIFAR100.

|                     | CIFAR10 |               | CIFAR100 |               |
|---------------------|---------|---------------|----------|---------------|
|                     | iid     | β = 0.6       | β = 0.3  | iid           | β = 0.6       | β = 0.3  |
| centralized learning| 92.55 ±0.05 |               |          | 73.98 ±0.26  |               |
| FedAvg              | 83.78 ±0.13 | 82.04 ±0.46   | 79.59 ±1.01 | 50.29 ±0.34  | 50.67 ±0.34  | 50.17 ±0.19 |
| FedProx             | 84.10 ±0.39 | 82.36 ±0.38   | 80.12 ±0.43 | 51.25 ±0.62  | 50.94 ±0.40  | 50.82 ±0.20 |
| FedDyn              | 85.19 ±0.58 | 82.87 ±0.62   | 80.15 ±1.00 | 53.27 ±0.01  | 51.68 ±0.31  | 50.51 ±0.34 |
| MOON                | 84.34 ±0.09 | 82.67 ±0.08   | 80.97 ±0.46 | 52.51 ±0.70  | 52.55 ±0.49  | 51.88 ±0.25 |
| SCAFFOLD            | 85.99 ±0.06 | 84.55 ±0.30   | 82.14 ±1.20 | 53.32 ±0.32  | 53.91 ±0.33  | 54.36 ±0.32 |
| FedGen              | 83.91 ±0.36 | 82.23 ±0.73   | 79.72 ±0.85 | 50.38 ±0.27  | 50.71 ±0.55  | 50.08 ±0.24 |
| FedDF               | 84.47 ±0.20 | 82.92 ±0.64   | 80.97 ±0.74 | 52.12 ±0.15  | 51.36 ±0.02  | 51.26 ±0.09 |
| FedFTG              | 87.34 ±0.16 | 86.06 ±0.19   | 84.38 ±0.49 | 56.94 ±0.19  | 56.49 ±0.55  | 55.96 ±0.39 |

Table 2. Evaluation of different FL methods on CIFAR10 and CIFAR100 (β = 0.3), in terms of the number of communication rounds to reach target test accuracy (acc). Note that we highlight the best and second best results in bold.

|                     | CIFAR10 |               | CIFAR100 |               |
|---------------------|---------|---------------|----------|---------------|
|                     | acc = 75% | acc = 80%   | acc = 50% | acc = 50%   |
| FedAvg              | 153.67 ±20.33 | 425.33 ±61.67 | 86.67 ±6.33 | 713.67 ±191.33 |
| FedProx             | 143.67 ±0.33 | 391.67 ±13.33 | 86.00 ±1.00 | 529.00 ±36.00 |
| FedDyn              | 90.67 ±2.33 | 183.67 ±23.33 | 64.00 ±8.00 | 239.33 ±15.67 |
| MOON                | 128.00 ±10.00 | 347.00 ±24.00 | 79.67 ±2.33 | 376.00 ±29.00 |
| SCAFFOLD            | 100.33 ±14.67 | 212.00 ±24.00 | 58.33 ±3.67 | 185.67 ±0.33 |
| FedGen              | 140.00 ±4.00 | 406.67 ±29.33 | 95.00 ±1.00 | 684.00 ±92.00 |
| FedDF               | 132.67 ±11.33 | 329.00 ±42.00 | 94.50 ±1.50 | 452.00 ±5.00 |
| FedFTG              | 92.67 ±14.33 | 188.67 ±31.33 | 57.00 ±1.00 | 166.33 ±10.67 |

Figure 4. Learning Curve of (a) CIFAR10 and (b) CIFAR100 in 1000 communication rounds (β = 0.3).

Figure 5. (a) Test accuracy w.r.t. data heterogeneity. (b) Test accuracy w.r.t. fraction C of active clients in each round (β = 0.3). All experiments are conducted on CIFAR10.

Data heterogeneity and Partial Client Participant. Figure 5(a) displays the test accuracy on different β values. In this figure, FedFTG achieves the best accuracy on all settings, which validates that FedFTG is effective in various data heterogeneity scenarios. Besides, FedFTG gains more accuracy improvement in extreme data heterogeneity scenario β = 0.2. In addition, as the degree of data heterogeneity increases i.e., β increases, the accuracy of each method is ascending. Figure 5(b) displays the test accuracy of FL methods with different fractions of active clients in each communication round. FedFTG also yields the best performance in this figure. Besides, the more clients involved in communication, the higher accuracy will be achieved.

Orthogonality of FedFTG with existing FL optimizers. Table 3 provides the performance of FedFTG using FedAvg, FedProx, FedDyn, SCAFFOLD and MOON opti-
Box plot w.r.t. Box plot w.r.t. Box plot w.r.t.

0.81 0.81 0.82
0.82 0.82 0.82
0.83 0.83 0.84
0.84 0.86 0.86
0.86 0.87 0.87

4.3. Ablation Study

FedDyn+FedFTG as lo-
This validates the effectiveness and the orthogonality of
The experiments are conducted on CIFAR10, β = 0.3.

Table 4. Impact of the each components in FedFTG. The ex-

| Method | Accuracy (%) |
|--------|--------------|
| baseline | FedFTG 83.43±0.10 |
| module | -hsm 82.49±0.42 |
| -cls 82.39±0.22 |
| -abe 82.40±0.21 |
| -hsm&cls 82.18±0.16 |
| -hsm&abe 82.15±0.14 |
| -cls&abe 82.11±0.28 |
| -hsm&cls&abe 81.98±0.14 |
| loss | -Lcls 82.50±0.55 |
| -Ldis 82.52±0.35 |
| -Lcls&Ldis 82.12±0.16 |
| DKL ← Lmax 10.17±0.26 |

FedAvg+FedFTG 83.82±0.31 82.27±0.67
FedProx+FedFTG 84.06±0.32 82.21±0.46
FedDyn+FedFTG 83.17±0.50 81.43±0.18
MOON+FedFTG 83.81±0.45 82.19±0.91
SCAFFOLD+FedFTG 86.06±0.19 84.38±0.49
FedDyn+FedFTG 79.00±3.00 168.00±13.00
MOON+FedFTG 114.33±6.67 276.00±12.00
SCAFFOLD+FedFTG 92.67±14.33 188.67±31.33
FedAvg+FedFTG 122.00±4.00 279.33±17.67
FedProx+FedFTG 117.33±6.67 278.67±25.33
FedDyn+FedFTG 168.00±13.00
MOON+FedFTG 276.00±12.00
SCAFFOLD+FedFTG 188.67±31.33

FedGen ad-

Figure 6. Performance of FedFTG using different hyperparameters (a) λcls, (b) λdis, (c) dimension d of noise z on CIFAR10 with β = 0.3.

Sensors. In Table 3, SCAFFOLD+FedFTG yields the best test accuracy among all the optimizers. FedDyn+FedFTG performs better than SCAFFOLD+FedFTG in terms of the round number to reach the target accuracy. This is consistent with the results in Table 2, where FedDyn requires fewer rounds than SCAFFOLD. Comparing Table 3 with Tables 1 and 2, we notice that for any FL optimizer, its performance can be largely boosted by using FedFTG. This validates the effectiveness and the orthogonality of FedFTG. Besides, simply using FedAVG+FedFTG as local optimizer already exceeds the other methods in Table 1 except SCAFFOLD.

4.3. Ablation Study

Necessity of each component in FedFTG. Table 4 displays the test accuracy of FedFTG after discarding some modules and losses, trained with 500 communication rounds on CIFAR10, β = 0.3. Here hsm, cls and abe represent the hard sampling mining, customized label sampling and class-level ensemble, respectively. We can see that removing any module leads to worse and unstable performance, i.e., lower accuracy and larger confidence interval. In addition, their joint absence can cause a further decrease on accuracy. On the other hand, a similar tendency is observed for the losses: the absence of single loss will lead to performance decrease, and removing multiple losses will enlarge the decrease. It should be noticed that, if replacing the KL divergence with Mean Average Square (Lmse) to measure the model discrepancy, the model will collapse, which leads to severe performance degradation.

Robustness of FedFTG on hyperparameters. To measure the influence of hyperparameter selection, we select λcls and λdis from [0.5, 0.75, 1.0, 1.25, 1.5] and select the dimension d of noise data z in [50, 100, 150, 200, 250]. Figure 6 illustrates the test accuracy in term of the box plot, where FedFTG achieves similar performance among all the choices. Besides, the worst accuracy in Figure 6(a)-(b) is better than the best of previous works in Table 1. This indicates that FedFTG is not sensitive to the selection of hyperparameter in a large range.

Comments on feature-level pseudo data. FedGen advises that the data in feature space are more compact than in input space, thus it generates pseudo data in feature-level and fine-tunes the last few FC layers of federated model. Motivated by this, we compare the performance of FedFTG
Table 5. The impact of feature-level generation (F) and input-level generation (I) on FedFTG. The experiments are conducted on CIFAR10, $\beta = 0.3$ and 0.6.

| Method   | MIO-TCD (Accuracy %) | CompCar (Accuracy %) | Tiny-ImageNet (Accuracy %) |
|----------|----------------------|----------------------|-----------------------------|
| FedAvg   | 89.63 ± 1.06         | 43.34 ± 2.93         | 34.68 ± 0.67                |
| FedProx  | 89.69 ± 1.00         | 44.07 ± 3.41         | 35.39 ± 0.54                |
| FedDyn   | 90.47 ± 0.99         | 50.46 ± 2.57         | 41.77 ± 0.28                |
| SCAFFOLD | 89.88 ± 1.11         | 48.64 ± 3.46         | 38.80 ± 0.18                |
| FedGen   | 89.85 ± 1.03         | 45.96 ± 4.18         | 35.44 ± 0.35                |
| FedDF    | 90.01 ± 0.70         | 47.31 ± 3.47         | 36.19 ± 0.40                |
| **FedFTG** | **91.16 ± 0.92** | **51.85 ± 3.46** | **42.23 ± 0.22** |

Table 6. Test accuracy (%) on real-world datasets MIP-TCD, CompCar and Tiny-ImageNet.

| Method   | MIO-TCD | CompCar | Tiny-ImageNet |
|----------|---------|---------|---------------|
| FedAvg   | 89.63±1.06 | 43.34±2.93 | 34.68±0.67   |
| FedProx  | 89.69±1.00 | 44.07±3.41 | 35.39±0.54   |
| FedDyn   | 90.47±0.99 | 50.46±2.57 | 41.77±0.28   |
| SCAFFOLD | 89.88±1.11 | 48.64±3.46 | 38.80±0.18   |
| FedGen   | 89.85±1.03 | 45.96±4.18 | 35.44±0.35   |
| FedDF    | 90.01±0.70 | 47.31±3.47 | 36.19±0.40   |
| **FedFTG** | **91.16±0.92** | **51.85±3.46** | **42.23±0.22** |

using feature-level generation (F) and input-level generation (I) in Table 5. Though FedFTG(F) still exceeds the other methods in Table 1, it suffers significant performance drop compared with FedFTG(I), which indicates the input-level generation is more effective for FedFTG. This is because FedFTG(F) only fine-tunes the last few layers of the global model, so the effect of knowledge transfer is limited.

4.4. Experiments on Real-World Datasets

In this section, we test the performance of FedFTG on more challenging real-world datasets - vehicle classification datasets MIO-TCD [28] and CompCar [40], and large-scale image classification dataset Tiny-ImageNet[2]. To better validate the effectiveness of FedFTG, we use the surveillance subset of CompCar, of which the images are collected by surveillance cameras. For MIO-TCD and Tiny-ImageNet, we assign the training data to 100 clients, while for CompCar the client number is 50. $\beta$ of Dirichlet distribution is 0.6 for all these datasets. The images of MIO-TCD and CompCar are resized to 112 × 112 before training, and we adopt a deeper generator for them. The communication round is 50, 100 and 1000 for MIO-TCD, CompCar and Tiny-ImageNet respectively. The other settings are the same as in Section 4.1. Experiment results are presented in Table 6.

From this table, we find that FedFTG consistently outperforms the other methods in all scenarios, which verifies the effectiveness of FedFTG in real-world FL applications. FedDF and FedGen also adopt data-free knowledge generation to improve the federated model. Though they yield higher performance than FedAvg and FedProx, FedFTG exceeds them by 1% ~ 6%. This further validates the effectiveness of the proposed modules in FedFTG.

5. Discussion

**Privacy issue.** Since FedFTG recovers the training data of clients in server, it may violate the privacy regulation in FL. However, according to our observation, the pseudo data only captures the high-level feature pattern of real data, which cannot be understood by human beings (see Figure 2). Besides, as the generator is trained by all local models, the pseudo data tend to show shared features of data in clients, which means the attribute of individual data will not be revealed. Uploading label statistics of data in clients may also leak privacy. One optional solution is adding noise to label statistics. According to our experiments, when the noise ratio is less than 10%, its influence on performance is less than 0.1% on CIFAR10, $\beta = 0.3$ setting.

**Communication cost.** Compared with other methods, FedFTG only need to additionally transmit the label statistics of training data (i.e., $\{n_t^k, y_t^k\}_{k \in S_t, y_t \in [1, ..., M]}$, $M$ is the class number), which induces negligibly extra transmission cost. If the training data keep the same during training, the label statistics can be reported to server before training, thus no extra transmission cost will be introduced.

**Limitations.** The main limitation of this work mainly exists in computation efficiency. As FedFTG additionally trains the global model apart from local training, it will make the whole training time longer than the other methods. In our experiment, FedFTG requires about double the time of FedAVG in each communication round. Besides, as the global model training is conducted in the server, FedFTG is more applicable to the cross-silo FL applications as defined in [17], where the server can be organizations that own sufficient computation source.

6. Conclusion

In this paper we propose a new data-free knowledge distillation method FedFTG to fine-tune the global model and to boost the performance of federated learning. A hard sampling scheme is proposed to effectively explore the knowledge in local models and transfer it to global model. Facing the label distribution shift in data heterogeneity scenario, we propose customized label sampling and class-level ensemble to derive maximum utilization of knowledge. Extensive experiments on five benchmarks validate the effectiveness of the proposed FedFTG.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62088102, and in part by the PKU-NTU Joint Research Institute (JRI) sponsored by a donation from the Ng Teng Fong Charitable Foundation, and in part by Science and Technology Innovation 2030 – “Brain Science and Brain-like Research” Major Project (No. 2021ZD0201402 and No. 2021ZD0201405).

2https://www.kaggle.com/c/tiny-imagenet
benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing*, 27(10):5129–5141, 2018.

[29] Grigory Malinovskiy, Dmitry Kovalev, Elnur Gasanov, Laurent Condat, and Peter Richtarik. From local sgd to local fixed-point methods for federated learning. In *International Conference on Machine Learning*, pages 6692–6701. PMLR, 2020.

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. pages 1273–1282. PMLR, 2017.

[31] Jed Mills, Jia Hu, and Geyong Min. Communication-efficient federated learning for wireless edge intelligence in iot. *IEEE Internet of Things Journal*, 7(7):5986–5994, 2019.

[32] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In *International Conference on Machine Learning*, pages 4743–4751. PMLR, 2019.

[33] Daniel Peterson, Pallika Kanani, and Virendra J Marathe. Private federated learning with domain adaptation. *arXiv preprint arXiv:1912.06733*, 2019.

[34] Basheer Qolomany, Kashif Ahmad, Ala Al-Fuqaha, and Junaid Qadir. Particle swarm optimized federated learning for industrial iot and smart city services. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[35] Felix Sattler, Tim Korjakow, Roman Rischke, and Wojciech Samek. Fedaux: Leveraging unlabeled auxiliary data in federated learning. *arXiv preprint arXiv:2102.02514*, 2021.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33, 2020.

[38] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avetissian, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[39] Guile Wu and Shaogang Gong. Decentralised learning from independent multi-domain labels for person re-identification. *arXiv preprint arXiv:2006.04150*, 2020.

[40] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3973–3981, 2015.

[41] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deep-inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

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

[43] Zhaohua Zheng, Yize Zhou, Yilong Sun, Zhang Wang, Boyi Liu, and Keqiu Li. Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges. *Connection Science*, pages 1–28, 2021.

[44] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12878–12889. PMLR, 2021.

[45] Weiming Zhuang, Yonggang Wen, Xuesen Zhang, Xin Gan, Daiying Yin, Dongzhan Zhou, Shuai Zhang, and Shuai Yi. Performance optimization of federated person re-identification via benchmark analysis. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 955–963, 2020.
7. Supplementary

7.1. Exploration of Long-Tail problem

To explore the influence of long-tailed data on model performance, we train models using multiple subsets of CIFAR10, which have different degrees of imbalance. Here the subset is generated by Dirichlet distribution $\text{Dir}(\beta)$, where a smaller $\beta$ indicates more imbalanced data. The data number of each subset is 5000, and the architecture of the model is ResNet34 [13]. The results are illustrated in Figure 7. Here, the curves in green and blue are the test accuracy on total test data and partitive test data respectively, where the distribution of partitive test data is the same as the distribution of training data. We can see there is a performance gap between two curves, and the gap becomes larger when the degree of imbalance is increased. This is because the model only learns the majority classes, and these classes also dominate the partitive test data, thus the model achieves high accuracy on partitive test data; whereas for the total test data that contains balanced data for every class, the model can not correctly predict the data of minority classes, the model tends to learn majority classes and yields extremely low even zero accuracies for minority classes 7,10 and 9. Moreover, the quality of pseudo data is highly related to original data distribution. For the minority classes, the test accuracy of the pseudo data is less than 10%, i.e., the quality of the pseudo data is even worse than random noise. This indicates that the pseudo data generated via biased model could be invalid to conduct knowledge transfer, which motivates as to customize the sample probability of label during data generation to facilitate effective knowledge transfer.

7.2. Visualization of Data Heterogeneity

In Figure 9, we figure out the data distributions of clients that generated by Dirichlet distribution $\text{Dir}(\beta)$ with different $\beta$, as well as IID data distributions. For each $\beta$ value, we display the data distributions of 10 clients. In Figure 9, the data distributions of clients are significantly different when $\beta$ is small, and the client even has no data for some classes. When $\beta$ grows, the data is distributed more evenly in each client, and the discrepancy of data distributions among clients becomes smaller.

7.3. Detailed Hyperparameters

Here we introduce the setting of hyperparameters for baselines during experiments. For FedProx, the proximal regularization parameter $\mu$ is $1e-4$. $\alpha$ in FedDyn is $1e-2$. We set the local update round in SCAFFOLD following [1], which is 50 according to our experiment setting. Following [22], we set $\tau = 0.5$, tune $\mu$ from $\{0.1, 1, 5\}$ and report the best result. For FedGen and FedDF, the learning rate for the generator is the same as FedFTG, i.e., it is initialized as 0.01 and is decayed quadratically with weight 0.998. As ResNet18 only has one fully-connected layer, $l$ in FedGen is $L - 1$, where $L$ is the total layer number.

7.4. Detailed Architecture of Generator

Table 7 lists the architectures of generators for FedFTG, FedDF and FedGen used in Section 4.1 ~ Section 4.3. Here, $d$ is the dimension of noise data $z$, and it is 100 and 256 for CIFAR10 and CIFAR100, respectively. $M$ is the class number of datasets, and it is 10 and 100 for CIFAR10 and CIFAR100 respectively. The inplace of LeakReLU is 0.2 here. Note that in Table 7(b) the output of generator is 512-dimensional, as the input of the last FC layer in ResNet18 is 512-dimensional. If using the other classifiers, the dimension of the generator’s output should be adjusted accordingly.

Table 8 lists the architectures of generators used in Section 4.4. Here $d = 256$ for all the datasets MIO-TCD.
Figure 9. Visualization of the instance number per class allocated to each clients (indicated by dot size), for different $\beta$ values of Dirichlet distribution $\text{Dir}(\beta)$.

Table 7. The architectures of generators used in Section 4.1 ~ Section 4.3.

(a) Generator for FedFTG and FedDF
\[
\begin{align*}
    z &\in \mathbb{R}^d \sim \mathcal{N}(0, 1) \\
    m &= \text{Map}(y) \in \mathbb{R}^M, y \in [1, ..., M] \\
    FC(z) &\to 4096 \\
    FC(m) &\to 4096 \\
    \text{Concat} &\to 8192 \\
    \text{Reshape, BN} &\to 128 \times 8 \times 8 \\
    \text{Conv2D, BN, LeakyReLU} &\to 128 \times 8 \times 8 \\
    \text{Upsampling} &\to 128 \times 16 \times 16 \\
    \text{Conv2D, BN, LeakyReLU} &\to 64 \times 16 \times 16 \\
    \text{Upsampling} &\to 64 \times 32 \times 32 \\
    \text{Conv2D, Tanh} &\to 3 \times 32 \times 32
\end{align*}
\]

(b) Generator for FedGen
\[
\begin{align*}
    z &\in \mathbb{R}^d \sim \mathcal{N}(0, 1) \\
    m &= \text{Map}(y) \in \mathbb{R}^M, y \in [1, ..., M] \\
    FC(z) &\to 4096 \\
    FC(m) &\to 4096 \\
    \text{Concat, BN} &\to 8192 \\
    \text{FC, BN, LeakyReLU} &\to 8192 \\
    \text{FC} &\to 512
\end{align*}
\]

CompCar and Tiny-ImageNet. $s$ is the image size, and $s = 112, 112, 64$ for MIO-TCD, CompCar and Tiny-ImageNet respectively. Note that for the experiments of VGG11 in Table 3 in the main paper, we also adopt these two generators for FedDF, FedGen and FedFTG.

7.5. Supplementary Experiment Results

Table 9 illustrates the communication rounds of different methods to reach the target test accuracy (75% and 80% for CIFAR10, 40% and 50% for CIFAR100) when $\beta = 0.6$, which is a supplement to Table 2 in the main paper. Same as Table 2, FedFTG achieves the second best and the best convergence for CIFAR10 and CIFAR100 respectively. Besides, it greatly reduces the round numbers required by its FL optimizer SCAFFOLD. Table 10 displays the test accuracy when adopting VGG11 [36] and ResNet34 [13] as the classifier. In this table, FedFTG yields the best performance in all scenarios, which validates the effectiveness of FedFTG on various architectures of deep neural network.
Table 8. The architectures of generators used in Section 4.4.

(a) Generator for FedFTG and FedDF
\[
z \in \mathbb{R}^d \sim \mathcal{N}(0, 1)
\]
\[
m = \text{Map}(y) \in \mathbb{R}^M, y \in [1, ..., M]
\]
\[
\text{FC}(z) \rightarrow s^2
\]
\[
\text{FC}(m) \rightarrow s^2
\]
\[
\text{Concat} \rightarrow 2s^2
\]
\[
\text{Reshape, BN} \rightarrow 512 \times (s/16) \times (s/16)
\]
\[
\text{Conv2D, LeakyReLU} \rightarrow 256 \times (s/8) \times (s/8)
\]
\[
\text{Conv2D, LeakyReLU} \rightarrow 128 \times (s/4) \times (s/4)
\]
\[
\text{Conv2D, LeakyReLU} \rightarrow 64 \times (s/2) \times (s/2)
\]
\[
\text{Conv2D, LeakyReLU} \rightarrow 64 \times s \times s
\]
\[
\text{Conv2D, Tanh} \rightarrow 3 \times s \times s
\]

(b) Generator for FedGen
\[
z \in \mathbb{R}^d \sim \mathcal{N}(0, 1)
\]
\[
m = \text{Map}(y) \in \mathbb{R}^M, y \in [1, ..., M]
\]
\[
\text{FC}(z) \rightarrow s^2
\]
\[
\text{FC}(m) \rightarrow s^2
\]
\[
\text{Concat, BN} \rightarrow 2s^2
\]
\[
\text{Conv2D, LeakyReLU} \rightarrow s^2
\]
\[
\text{Conv2D, LeakyReLU} \rightarrow s^2
\]
\[
\text{Conv2D} \rightarrow 512
\]

Table 9. Evaluation of different FL methods on CIFAR10 and CIFAR100 ($\beta = 0.6$), in terms of the number of communication rounds to reach target test accuracy ($\text{acc}$). Note that we highlight the best and second best results in bold.

| Method | CIFAR10 $\text{acc} = 75\%$ | CIFAR10 $\text{acc} = 80\%$ | CIFAR100 $\text{acc} = 40\%$ | CIFAR100 $\text{acc} = 50\%$ |
|--------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| FedAvg | 104.33±6.67                 | 270.67±13.33                | 81.67±2.33                  | 563.67±163.33               |
| FedProx| 109.67±8.33                 | 263.0±27.0                  | 81.67±11.33                 | 476.0±199.0                 |
| MOON   | 102.67±1.33                 | 252.33±32.67                | 83.67±3.33                  | 354.0±21.00                 |
| FedDyn | 72.67±7.33                  | **133.33±28.67**            | 61.67±7.33                  | **186.33±10.67**            |
| SCAFFOLD| 77.00±3.00                 | 161.00±8.00                 | 82.00±5.00                  | 571.33±78.67                |
| FedGen | 97.67±8.33                  | 246.33±24.67                | 90.00±6.00                  | 445.00±42.00                |
| FedDF  | **73.67±4.33**              | **143.33±5.67**             | **55.00±3.00**              | **152.33±10.67**            |

Table 10. Test Accuracy (%) of different methods on CIFAR10 using VGG11 and ResNet34 networks ($\beta = 0.3$).

| Method | VGG11 $\text{acc} = 75\%$ | VGG11 $\text{acc} = 80\%$ | ResNet34 $\text{acc} = 75\%$ | ResNet34 $\text{acc} = 80\%$ |
|--------|---------------------------|---------------------------|-----------------------------|-----------------------------|
| FedAvg | 82.05±0.59                | 80.48±0.89                | 81.02±0.53                  | 81.13±1.11                  |
| FedProx | 82.10±0.53                | 81.02±0.53                | 81.13±1.11                  | 81.15±0.46                  |
| MOON   | 83.69±0.76                | 81.5±0.46                 | 83.31±0.71                  | 80.72±0.44                  |
| SCAFFOLD| 86.78±0.37                | 83.31±0.71                | 84.38±0.56                  | 80.72±0.44                  |
| FedGen | 84.71±0.78                | 81.20±0.46                | 84.38±0.56                  | 81.20±0.46                  |
| FedFTG | **87.46±0.49**            | **85.00±0.45**            | **85.00±0.45**              | **85.00±0.45**              |