FLBOOST: ON-THE-FLY FINE-TUNING BOOSTS FEDERATED LEARNING VIA DATA-FREE DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) is an emerging distributed learning paradigm for protecting privacy. Data heterogeneity is one of the main challenges in FL, which causes slow convergence and degraded performance. Most existing approaches tackle the heterogeneity challenge by restricting the local model update in client, ignoring the performance drop caused by direct global model aggregation. On the contrary, we propose a new solution: *on-the-fly fine-tuning the global model in server* via data-free distillation to boost its performance, dubbed FLBoost to relieve the issue of direct model aggregation. Specifically, FLBoost adopts an adversarial distillation scheme to continually transfer the knowledge from local models to fine-tune the global model. In addition, focused distillation and attention-based ensemble techniques are developed to balance the extracted pseudo-knowledge to adapt the data heterogeneity scenario, which implicitly mitigates the distribution discrepancy across clients. Extensive experiments show that our FLBoost can achieve superior performance against the state-of-the-art 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) (McMahan et al., 2017) has been proposed to replace the traditional heavily centralized learning paradigm and protect data privacy. The vanilla FL algorithm, i.e., FedAvg (Li et al., 2019), trains a model by periodically aggregating the local model of the client in server and updating the aggregated model in the client with its local data. FL has been successfully applied in real-world tasks, e.g., smart city (Jiang et al., 2020; Zheng et al., 2021), health care (Gao et al., 2019; Lu et al., 2019), and recommender system (Hard et al., 2018; Hartmann et al., 2019).

One of the main challenges in FL is the data heterogeneity. In FL, the local model of the client is updated merely with local data, i.e., by minimizing the local empirical loss. However, minimizing the local empirical loss is fundamentally inconsistent with minimizing the global empirical loss (Acar et al., 2020; Li et al., 2019), which has been verified that local training, such as FedAvg, leads to drifted local models and forgets the global knowledge catastrophically, thus inducing the degraded performance and slow convergence (Hsu et al., 2019; Lee et al., 2021; Khaled et al., 2020).

To tackle the data heterogeneity challenge, most existing methods, e.g., FedProx (Li et al., 2018), SCAFFOLD (Karimireddy et al., 2020), FedDyn (Acar et al., 2020), constrain the direction of local model update to align the local and global optimization objectives. Recently, FedGen (Zhu et al., 2021) learns a lightweight generator to generate pseudo features and broadcasts them to clients to regulate local training. However, all these methods merely conduct simple model aggregation to get the global model in server, which fails to balance the class of local datasets and narrow the distribution discrepancy across clients. In addition, Singh & Jaggi (2020, Section 5.3, Figure 1) show that directly aggregating models will largely degrade the performance while fine-tuning can greatly boost its accuracy, which motivates us to fine-tune the aggregated global model in the server without requiring local or external datasets. 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 source in cross-silo FL (Kairouz et al., 2019).

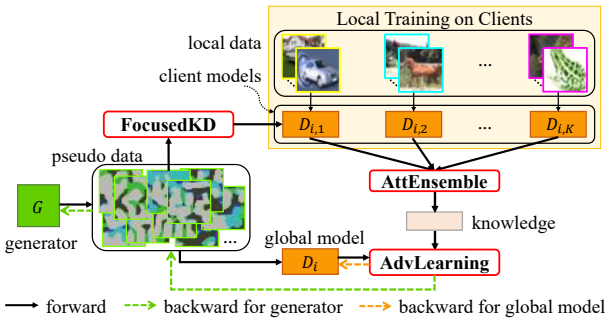


Figure 1: Overview of FLBoost.

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 (DFKD), dubbed FLBoost (see Figure 1), which simultaneously refines the model aggregation procedure and exploits the rich computing power of the sever. In each round, FLBoost first randomly selects a small set of client models trained with local data, which are then transferred to the server and aggregated as a preliminary global model. Instead of broadcasting the aggregated model back to each client directly, FLBoost fine-tunes this preliminary global model in server using the knowledge extracted from local models. Specifically, FLBoost generates a set of pseudo data by an auxiliary generator, which is trained by inversing the local models via DFKD. Since the local data is class-imbalanced in FL, the inversed pseudo data is inaccurate and the extracted knowledge is invalid for the minority classes. We propose a focused knowledge distillation technique (**FocusedKD**), to select the valid pseudo data and control the extent of knowledge distillation on different classes. Due to the discrepancy of data distributions among clients, the extracted knowledge from local models is heterogeneous, and directly assigning the same weights during aggregation may be sub-optimal. We thus propose an attention-based ensemble technique (**AttEnsemble**) to integrate the extracted knowledge and exploit their potential intra information across clients. The processed reliable knowledge is then used to fine-tune the preliminary global model and the generator in an adversarial scheme (**AdvLearning**), which yields the final global model.

We emphasize that FLBoost is orthogonal to several existing local optimizers, such as FedAvg, FedProx, FedDyn, and SCAFFOLD, as it only modifies the procedure of global model aggregation on the server. Consequently, FLBoost can be seamlessly embedded into these local FL optimizers, taking their advantages to further improve the performance of FLBoost. At last, we apply FLBoost to solve the image classification task on CIFAR10 and CIFAR100 datasets with different kinds of heterogeneous dataset partitions. Extensive experiments verify that FLBoost achieves superior performance compared with several state-of-the-art (SOTA) methods including FedAvg, FedProx, FedDyn, SCAFFOLD, FedDF, FedGen. Besides, we demonstrate that FLBoost can serve as a strong plugin to enhance the performance of FedAvg, FedProx, FedDyn, and SCAFFOLD. To the end, the main contributions of this work are four-fold:

- We propose FLBoost 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 sever.
- We adopt an adversarial training scheme, and propose focused knowledge distillation and attention-based ensemble techniques for FLBoost, which can filter the wrongly extracted knowledge with DFKD and exploit the potential intra information among clients.
- We demonstrate that FLBoost is orthogonal to exiting local optimizers and can serve as a strong and versatile plugin to enhance the performance of FedAvg, FedProx, FedDyn and SCAFFOLD.
- We verify the superiority of FLBoost against several SOTA methods for federated learning, including FedAvg, FedProx, FedDyn, SCAFFOLD, FedGen and FedDF, with extensive experiments.

2 Related Work

There exist extensive works focus on improving the global performance of FL via client selection (Fraboni et al., 2021; Chen et al., 2020), split learning (He et al., 2020; Wu & Gong, 2020), domain adaptation (Peterson et al., 2019; Liu et al., 2021), and etc., and improving the local performance of each client via personalized FL (Dinh et al., 2020; Deng et al., 2020). The interested readers may refer to monographs (Kairouz et al., 2019; Wang et al., 2021) and the reference therein to follow up the advance of FL. Below, we mainly summarize the most relevant techniques to our work.

Federated Optimizer. The vanilla FL algorithm, i.e. FedAvg (McMahan et al., 2017) periodically aggregates the local models in server and updates the local model with its individual data. In ad-

dition, FedProx (Li et al., 2018) adds a proximal term to the local subproblem to restrict the local update closer to the initial (global) model. SCAFFOLD (Karimireddy et al., 2020) uses a variance reduction technique to correct the drifted local update. FedDyn (Acar et al., 2020) 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 we do in FLBoost.

Knowledge Distillation in Federated Learning. With the help of an unlabeled dataset, FedDF (Lin et al., 2020) proposes an ensemble distillation for model fusion, trains the global model using the averaged logits from local models. FedAUX (Sattler et al., 2021) finds a model initialization for the local models, and weights the logits from local models using (ε, δ)-differentially private certainty scoring. FedBE (Chen & Chao, 2020) 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 auxiliary dataset in server, which is infeasible for private tasks. Though FedDF maintains the auxiliary dataset can be replaced with a pretrained generator, it does not instantiate how to acquire the generator without auxiliary data.

Data-Free Knowledge Distillation (DFKD). DFKD methods (Chen et al., 2019; Fang et al., 2019) generate pseudo data from a pretrained model (teacher), and use them to transfer knowledge of teacher model to another small model (student). The data is generated by maximizing the response of fake data on teacher model. DeepImpression (Nayak et al., 2019) models the output space of teacher model and recovers the real data by fitting the output space. DeepInversion (Yin et al., 2020) further optimizes the pseudo data by regularizing the distribution of intermediate feature maps. DAFL (Chen et al., 2019) and DFAD (Fang et al., 2019) use a generator to generate data efficiently, where DAFL 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 continually.

In this work, we adopt DFKD to generate pseudo data with an auxiliary generator to boost the performance of global model of FL in server. However, directly applying DFKD technique to FL does not work well in data heterogeneity scenario, since the local models trained with class-imbalanced data contain invalid knowledge for minority classes, and it is hard to integrate the knowledge for multiple local models. Recently, FedGen (Zhu et al., 2021) proposes to learn a lightweight generator to ensemble knowledge of local models in a data-free manner, then delivers it to clients to correct local updates, which is the most related work to our FLBoost. However, FedGen ignores the wrong knowledge in data heterogeneity scenario, which may induce the performance drop during knowledge distillation, while FLBoost proposes FocusedKD to extract effective knowledge from local models. Besides, FLBoost proposes AttEnsemble to derive a maximum utility of local models.

3 FLBOOST: ON-THE-FLY FINE-TUNE THE GLOBAL MODEL

In this section, we describe the proposed novel federated learning paradigm: FLBoost. We firstly introduce the proposed FLBoost algorithm in Section 3.1, and then we introduce the two key techniques in FLBoost: focused knowledge distillation (FocusedKD) and attention-based ensemble training method (AttEnsemble) in Section 3.2 and Section 3.3, respectively.

3.1 OVERVIEW OF FLBOOST

Let ω 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 the number of samples. Generally speaking, federated learning can be formulated as the following optimization:

$$\min_{\omega} \frac{1}{K} \sum_{k=1}^{K} f_k(\omega), \text{ with } f_k(\omega) = \frac{1}{N_k} \sum_{i=1}^{N_k} f(\omega, x_{k,i}, y_{k,i}), \tag{1}$$

where f is the loss function to measure training error in FL, and the dataset $\mathcal{D}_k$ for $k = 1, 2, \ldots, K$ could be distributed heterogeneously. Below, we first describe the procedure of FLBoost for solving problem (1) in Algorithm 1, which is composed of two sub-algorithms, ClientUpdate and ServerUpdate for updating the network parameters in local models and global model, respectively.

Algorithm 1 FLBoost: On-the-fly fine-tune the global model

Input: T: communication round; $\mathcal{D}_k$: the dataset of k-th client; C: the fraction of active client in each round.

initialize model parameters ω and θ, initialize state s for server and state $s_1 \sim s_K$ for all clients
for $t = 1, ..., T$ **do**
  $S_t \leftarrow$ (random set of $\lceil C \cdot K \rceil$ clients);
  for $k \in S_t$ **in parallel do**
    $\omega_k, s_k, \Delta s_k \leftarrow \text{ClientUpdate}(\omega, D_k, s_k, s)$ ▷ FedAvg, FedProx, FedDyn, and SCAFFOLD
  end for
  $\omega, \theta, s \leftarrow \text{ServerUpdate}(\omega, \theta, s, \{\omega_k\}_{k\in S_t}, \{\Delta s_k\}_{k\in S_t})$ ▷ see Algorithm 2
end for

In each communication round, FLBoost 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. We emphasize that FLBoost is orthogonal to efforts on optimizing local model training, such as SCAFFOLD, FedAvg, FedProx, and FedDyn, which can serve as the role of ClientUpdate. In Section 4.1, we display the performance of FLBoost incorporated with above mentioned local optimizers, where significant performance improvements have been observed. Moreover, for any local optimizer, FLBoost only needs to additionally transmit the statistics of training data for clients in S_t in each round (detailed in Section 3.2 and 3.3), which causes negligibly extra transmission cost on these local optimizers. Without loss of generality, we use SCAFFOLD in FLBoost as an instance, where a state s is used to mitigate the client-drift (see Algorithm 3 in Appendix A.1).

Followed by ClientUpdate step, the ServerUpdate in Algorithm 2 updates the global model using the uploaded local models. However, the vanilla model aggregation in existing works will largely degrade performance in data heterogeneity scenario, as demonstrated by Lee et al. (2021). Instead, we replace the hand-crafted global model update with a learning procedure, so that the global model can preserve the knowledge from local models and maintain their performance as much as possible. We achieve this by fine-tuning the global model using the local models, under the help of an auxiliary generator. The overall objective on fine-tuning the global model is formulated as an adversarial learning scheme:

Algorithm 2 ServerUpdate, round t

Input: I: iteration of the training procedure in server; I_g, I_d: inner iteration of training the generator and the global model; η_g: the global step-size; ω, θ, s, $\{\omega_k\}_{k\in S}$, $\{\Delta s_k\}_{k\in S}$, λ_{cls}, λ_r.

$\Delta\omega = \frac{1}{|S_t|}\sum_{k\in S_t}(\omega_k - \omega)$, $\Delta s = \frac{1}{K}\sum_{k\in S_t} s_k$
$\omega \leftarrow \omega + \eta_g \nabla\omega$, $s \leftarrow s + \Delta s$
compute $p_t(y)$ according to Eq. (5)
for $i = 1, ..., I$ **do**
  $(Z, Y) \leftarrow$ (sample a batch of $z \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and $y \sim p_t(y)$)
  compute $\alpha_t^{k,y}$ for $\forall k \in S_t, \forall y \in Y$ according to Eq. (6)
  for $j = 1, ..., I_g$ **do**
    update θ with the loss Eq. (2) on samples (Z, Y)
  end for
  for $j = 1, ..., I_d$ **do**
    update ω with loss $\frac{1}{|Y|}\sum_{(z,y)\in(Z,Y)} \mathcal{L}_{md}$
  end for
end for
return ω, θ, s

$$(\omega, \theta) = \arg\min_{\omega}\max_{\theta} \mathbb{E}_{z\sim\mathcal{N}(\mathbf{0},\mathbf{1}), y\sim p_t(y)} \left[\mathcal{L}_{md}(\omega, \theta) - \lambda_{cls}\mathcal{L}_{cls}(\theta) - \lambda_r\mathcal{L}_r(\theta)\right], \tag{2}$$

$$with \quad \mathcal{L}_{md}(\omega, \theta) = \sum_{k\in S_t} \alpha_t^{k,y} D_{KL}(D(G(z, y; \theta); \omega) || D(G(z, y; \theta); \omega_k)), \tag{3}$$

$$\mathcal{L}_{cls}(\theta) = \sum_{k\in S_t} \alpha_t^{k,y} \mathcal{L}_{CE}(D(G(z, y; \theta); \omega_k), y), \ \mathcal{L}_r(\theta) = R(G(z, y; \theta), z), \tag{4}$$

where D parameterized with w and w_k are the global classifier and uploaded local classifier in the sever, respectively; G is the generator parameterized with θ in server, whose inputs are a random noise data z and a class label y, and output is the pseudo data of class y; $p_t(y)$ is the sampling probability of the class y; $\alpha_t^{k,y}$ controls the weight of local models during ensemble.

During training, the generator first generates pseudo data by $G(z, y; \theta)$. Then the pseudo data is input to global model $D(\cdot; \omega)$ and local models $D(\cdot; \omega_k), \forall k \in S_t$. The global model is updated by minimizing the model discrepancy $\mathcal{L}_{md}$ compared to local models, while the generator is updated by maximizing $\mathcal{L}_{md} - \lambda_{cls}\mathcal{L}_{cls} - \lambda_r\mathcal{L}_r$ to generate high-quality data. In implementation, we adopt stochastic gradient descent-ascent algorithm to solve Problem (2) as illustrated in Algorithm 2.

In Eq. (2), $\mathcal{L}_{md}$ measures the model discrepancy between global model ω and local model ω_k by Kullback-Leibler divergence (D_{KL}), as in Eq. (3). It is used to train the adversarial variables: (1)

train θ to generate hard samples that enlarge the model discrepancy; (2) train ω to minimize the discrepancy using the hard samples. As a result, FLBoost can continually exploit the knowledge in $\{\omega_k\}_{k\in S_t}$ and transfer it to ω. $\mathcal{L}_{cls}$ is used to facilitate the fidelity of the generated data. In Eq. 4, $\mathcal{L}_{cls}$ is formulated as cross-entropy loss ($\mathcal{L}_{CE}$) between the prediction of local model on generated data (i.e., $D(G(z,y;\theta);\omega_k)$) and the desired class label y. By minimizing $\mathcal{L}_{cls}$ with θ, $G(z,y;\theta)$ are enforced to yield higher prediction on class y to fit the data distribution of y. In addition, simply using $\mathcal{L}_{cls}$ will lead to model collapse of generator: G outputs the same data for every class. To mitigate this issue, we use the diversity loss $\mathcal{L}_r$ in Zhu et al. (2021) to improve the diversity of the generated data. By unifying $\mathcal{L}_{md}$, $\mathcal{L}_{cls}$ and $\mathcal{L}_r$, the pseudo data generated in FLBoost can fit the distribution of training data with high-quality.

To facilitate an effective knowledge transfer in data heterogeneity scenario, we propose focused knowledge distillation and attention-based ensemble techniques, which customizes the sampling probability $p_t(y)$ and the ensemble weight $\alpha_t^{k,y}$ according to the data distributions of clients in each round. In the following two subsections, we will introduce these methods separately.

3.2 Focused Knowledge Distillation

Typically, datasets in local clients are class-imbalanced in data heterogeneity scenario, even have no data for some classes. It has been proved that deep neural networks tend to learn majority classes and ignore the minority classes (Fang et al., 2021). Figure 2 illustrates the accuracy of model trained by class-imbalanced CIFAR10 data, and the quality of pseudo data inversed by the model. The data quality is displayed in terms of percentage of pseudo data that correctly classified by a well-trained classifier (trained on all data in CIFAR10 with 81.38% test accuracy). We can see that the model tends to learn majority classes and yields extremely low even zero accuracy for minority classes (classes 6,9,8). Moreover, the quality of pseudo data is highly related to original data distribution. For the minority classes, the pseudo data accuracy is less than 10%, which means it is even worse than random data. We further explore the influence of class-imbalanced data in Appendix A.2.

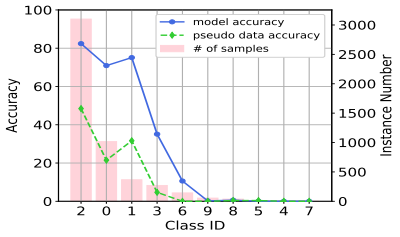


Figure 2: Correlation of model accuracy, distilled pseudo data accuracy and the amount of data on each class.

Based on above observation, the data information of minority classes in local models could be wrong and misleading. Hence, the pseudo data generated from local models are ambiguous for the minority classes. If uniformly sample the class label y and generating data by $G(z,y;\theta)$, the wrong knowledge will be propagated to global model and induce performance decrease. To mitigate this issue, we propose FocusedKD to focus the data generation and knowledge distillation on the majority classes and filter the minority classes. As a result, FLBoost can guarantee an effective knowledge distillation in data heterogeneity scenario. Specifically, we customize the sampling probability $p_t(y)$ according to the distribution of whole training data in each round,

$$p_t(y) \propto \sum_{k\in S_t}\sum_{i=1}^{N_k} \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_k}\left[\mathbf{1}_{y_i=y}\right] = \sum_{k\in S_t} n_k^y, \tag{5}$$

where $\mathbf{1}_{\text{condition}}$ is 1 if the condition is true and 0 otherwise, n_k^y is the data number of class y in client k. Substituting Eq. (5) to Eq. (2), the pseudo data of minority classes have low probability to be generated, thus the extracted knowledge are more reliable.

FocusedKD uses a shared sampling probability $p_t(y)$ to generate pseudo data and input to *all local models*, as the input of teacher models should be the same in ensemble KD. However, $p_t(y)$ may not be consistent with the individual data distribution of clients, thus the extracted knowledge may be invalid for some clients. In Section 3.3, we propose a re-weight mechanism to mitigate this problem.

3.3 Attention-Based Ensemble

The typical ensemble method in KD assigns same weight to the knowledge from different teacher models (Zhu et al., 2021; Lin et al., 2020). However, the data distributions of clients are different in data heterogeneity scenario, which means the knowledge in local models is heterogeneous.

Consequently, for one class the importance of knowledge are different among clients. If assigning same weight to clients, the important knowledge can not be figured out and utilized properly. We therefore propose an attention-based ensemble method (AttEnsemble), which assigns the ensemble weight via the individual data distribution of clients. Specifically, for the knowledge distillation in Eq. (3) and the data inversion in Eq. (4), we use $\alpha_t^{k,y}$ to weight the importance of model w_k on class y, where $\alpha_t^{k,y}$ is defined by the data proportion of class y in client k against total data in S_t,

$$\mathcal{L}_{md} = \sum_{k \in S_t} \alpha_t^{k,y} D_{KL}^k, \mathcal{L}_{cls} = \sum_{k \in S_t} \alpha_t^{k,y} \mathcal{L}_{CE}^k, \ \ with \ \ \alpha_t^{k,y} = n_k^y / \sum_{i \in S_t} n_i^y, \tag{6}$$

where $\mathcal{L}_{CE}^k = \mathcal{L}_{CE}(D(G(z, y; \theta); \omega_k), y)$ is the cross-entropy loss of client k, and D_{KL}^k is similar. As a result, the knowledge from clients can be flexibly integrated according to their importance on classes, so that FLBoost can facilitate a maximum utilization of knowledge from local models. Note that, compared with merging the outputs of local models as used in FedDF and FedGen, i.e., $\mathcal{L}_{cls} = \mathcal{L}_{CE}(\frac{1}{|S_t|} \sum_{k \in S_t} D(G(z, y; \theta); \omega_k), y)$, the knowledge is disentangled in Eq. (6), thus the knowledge of clients will not pollute each other in FLBoost.

To end this section, we compare FLBoost with the recently proposed FedGen (Zhu et al., 2021) which also guide the federated learning with DFKD, to further display the advantages of FLBoost:

- FLBoost and FedGen use the pseudo data to train the global model and local model respectively. However, the local model is originally trained by real and correct data, and additional knowledge transfer will disturb the local model optimization instead, as illustrated in Section 4.1.
- FLBoost trains the generator and global model with an adversarial training process, so that the knowledge can be continually exploited and distilled from local model, whereas FedGen trains the generator and local model in one-shot mode.
- FLBoost customizes the sampling probability and ensemble weight according to the data distribution in each round, while FedGen uses uniform probability for every class and uniform weight for every client. As a result, FLBoost can facilitate an effective knowledge transfer.
- FLBoost combines $\mathcal{L}_{md}$, $\mathcal{L}_{cls}$ and $\mathcal{L}_r$ to improve the quality of generated data as well as exploit the knowledge in local models, while FedGen only trains the generator by minimizing $\mathcal{L}_{cls}$.

4 Experiments

In this section, we empirically verify the efficacy of FLBoost. We summarize the implementation details in Section 4.1, and compare FLBoost with several SOTA FL algorithms in Section 4.2. Ablation studies are conducted to verify the necessity of each component of FLBoost in Section 4.3.

4.1 Implementation Details

Baselines. We compare FLBoost against FedAvg, FedProx, SCAFFOLD, FedDyn, FedGen and FedDF. For a fair comparison, we further derive a variant of FedGen, denoted FedGen*, which generates data in the input space instead of feature space in FedGen. Besides, since FedDF does not explain how to obtain the generator, we train the generator in the same way as FedGen.

Datasets. CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009) with heterogeneous dataset partition are used to test the efficacy of FLBoost, which are two difficult tasks in FL scenario and are widely adopts in FL research. Similar to existing works (Acar et al., 2020; Yurochkin et al., 2019), we use Dirichlet distribution $\mathbf{Dir}(\alpha)$ on label radios to simulate the non-iid data distribution among clients, where a smaller α indicates higher data heterogeneity. During the implementation, we set $\alpha = 0.3$ and $\alpha = 0.6$, respectively.

Network Architecture. For both CIFAR10 and CIFAR100 datasets, we employ ResNet18 (He et al., 2016) as the basic backbone. We borrow the generator network architecture from DFAD (Fang et al., 2019) for FLBoost, FedDF and FedGen*. For FedGen, the generator network architecture is composed of two embedding layers (for inputs z and y, respectively) and two fully-connected 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., $|S_t| = 10$).

Table 1: Test Accuracy of different FL methods on CIFAR10 and CIFAR100.

	CIFAR10		CIFAR100	
	$\alpha = 0.6$	$\alpha = 0.3$	$\alpha = 0.6$	$\alpha = 0.3$
FedAvg	82.04±0.46	79.59±1.01	50.67±0.34	50.17±0.19
FedProx	82.36±0.38	80.12±0.43	50.94±0.40	50.82±0.20
FedDyn	82.87±0.62	80.15±1.00	51.68±0.31	50.51±0.34
SCAFFOLD	84.55±0.30	82.14±1.20	53.91±0.33	54.36±0.32
FedGen	82.23±0.73	79.72±0.85	50.71±0.55	50.08±0.24
FedGen*	80.80±0.21	78.92±1.16	48.68±0.26	48.04±0.19
FedDF	82.92±0.64	80.97±0.74	51.36±0.02	51.26±0.09
FLBoost	**86.06**±0.19	**84.38**±0.49	**56.49**±0.55	**55.96**±0.39

For local training, the batchsize is 50 and the weight decay is $1e-3$. The learning rate 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, I_g, I_d in Algorithm 1 are 10, 1 and 5, respectively. If not specifically declared, we adopt $\lambda_{cls} = 1.0$ and $\lambda_r = 1.0$, and adopt SCAFFOLD as the FL optimizer in FLBoost.

We further provide detailed implementations such as descriptions of baselines, heterogeneity of datasets, topology of network architecture and the settings of hyperparameters in the Appendix A.

4.2 Comparison of FLBoost With Existing FL Methods

Test Accuracy. Table 1 reports the test accuracy of all compared algorithms on CIFAR10 and CIFAR100 datasets. All experiments are repeated over 3 random seeds. In Table 1, FLBoost achieves the best performance in all scenarios, surpassing the second one (i.e., SCAFFOLD) by at least 1.5%. FedDF also employs DFKD to improve the global model in server. It outperforms the FedAvg, FedProx and FedDyn, which further validates the superiority of the scheme "fine-tuning the global model via data-free knowledge distillation". However, it is worse than SCAFFOLD and FLBoost, which verifies the effectiveness of FocusedKD and AttEnsemble techniques in FLBoost.

FedGen yields lower accuracy compared with FedDF and FLBoost, and shows marginal performance gains than FedAvg in some cases. To evaluate the key component of FedGen that most influences the performance, we evaluate the performance of FedGen*, which generates pseudo data in the input space (same as FLBoost) instead of feature space. From Table 1, we see that FedGen* is worse than FedGen, which means the reason that FedGen is worse than FLBoost is not the feature-level generation of FedGen. Thus, we conclude the possible reason is that FedGen does not filter the extracted knowledge from local models. The wrong knowledge will influence the local training, since local client merely holds a small amount of real dataset.

Communication Rounds. Table 2 evaluates different FL methods in term of the number of communication rounds to reach target test accuracy ($acc = 75\%$ and $acc = 80\%$ for CIFAR10, $acc = 40\%$ and $acc = 50\%$ for CIFAR100, respectively). In Table 2, FBLoost achieves the second best and the best results on CIFAR10 and CIFAR100, respectively. Besides, FLBoost reduces the round number required by its FL optimizer (SCAFFOLD) in all scenarios. For CIFAR10, although FedDyn uses fewer rounds to achieve the target accuracy, its final accuracy is much worse than FLBoost, as displayed in Table 1. Below, we provide the results of using FedDyn as the optimizer of FLBoost, and the derived method FedDyn+FLBoost requires fewer rounds to reach target accuracy than FedDyn.

Orthogonality of FLBoost with existing FL optimizers. Table 3 provides the results of FLBoost, FedGen and FedDF using different FL optimizers. First, we evaluate *the performance of FLBoost using FedAvg, FedProx and FedDyn optimizers*. In Table 3, SCAFFOLD+FLBoost yields the best test accuracy among all the optimizers. FedDyn+FLBoost performs better than SCAFFOLD+FLBoost 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 FLBoost than without using it. This validates the effectiveness and the orthogonality of FLBoost. Second, we also *replace the optimiziers in FedGen and FedDF with SCAFFOLD*, and compare them with SCAFFOLD+FLBoost to further illustrate the advantage of FLBoost. Though FedGen and FedDF achieve higher accuracy by using SCAFFOLD optimizer, they are still worse than FLBoost.

Table 2: Evaluation of different FL methods on CIFAR10 and CIFAR100, 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, respectively.

CIFAR10	$\alpha = 0.6$		$\alpha = 0.3$	
	$acc = 75$	$acc = 80$	$acc = 75$	$acc = 80$
FedAvg	104.33±6.67	270.67±13.33	153.67±20.33	425.33±61.67
FedProx	109.67±8.33	263.0±27.0	143.67±0.33	391.67±13.33
FedDyn	**72.67**±7.33	**133.33**±28.67	**90.67**±2.33	**183.67**±23.33
SCAFFOLD	77.00±3.00	161.00±8.00	100.33±14.67	212.00±24.00
FedGen	114.00±8.00	284.33±30.67	140.00±4.00	406.67±29.33
FedGen*	140.33±13.67	402.33±4.67	223.67±52.33	666.00±191.00
FedDF	97.67±8.33	246.33±24.67	132.67±11.33	329.00±42.00
FLBoost	***73.67***±4.33	***143.33***±5.67	***92.67***±14.33	***188.67***±31.33
CIFAR100	$\alpha = 0.6$		$\alpha = 0.3$	
	$acc = 40$	$acc = 50$	$acc = 40$	$acc = 50$
FedAvg	81.67±2.33	563.67±163.33	86.67±6.33	713.67±191.33
FedProx	81.67±11.33	476.00±199.00	86.00±1.00	529.00±36.00
FedDyn	***56.00***±6.00	213.67±6.33	64.00±8.00	239.33±15.67
SCAFFOLD	61.67±7.33	***186.33***±10.67	***58.33***±3.67	***185.67***±0.33
FedGen	82.00±5.00	571.33±78.67	95.00±1.00	684.00±92.00
FedGen*	95.67±5.33	>1000	100.33±7.67	>1000
FedDF	90.00±6.00	445.00±42.00	94.50±1.50	452.00±5.00
FLBoost	**55.00**±3.00	**152.33**±10.67	**57.00**±1.00	**166.33**±10.67

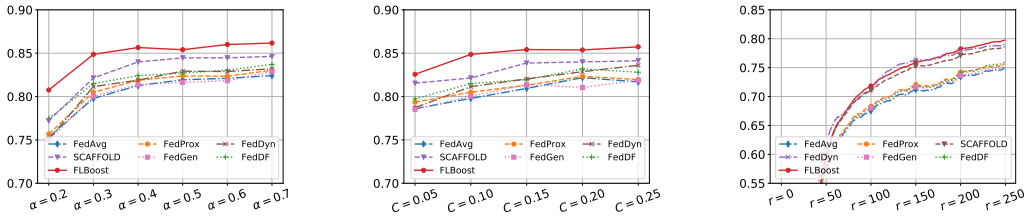
Table 3: The impact of FL optimizer on FLBoost and other methods, and the impact of feature-level generation on FLBoost. We display the test accuracy on CIFAR10, $\alpha = 0.3$ and 0.6, and the round number to reach the target accuracy ($acc = 75\%$ and $acc = 80\%$) when $\alpha = 0.3$.

	Accuracy		Round ($\alpha = 0.3$)	
	$\alpha = 0.6$	$\alpha = 0.3$	$acc = 75$	$acc = 80$
SCAFFOLD+FLBoost *(baseline)*	**86.06**±0.19	**84.38**±0.49	92.67±14.33	188.67±31.33
FedAvg+FLBoost	83.82±0.31	82.27±0.67	122.00±4.00	279.33±17.67
FedProx+FLBoost	84.06±0.32	82.21±0.46	117.33±6.67	278.67±25.33
FedDyn+FLBoost	83.17±0.50	81.43±0.18	**79.00**±3.00	**168.00**±13.00
SCAFFOLD+FedGen	84.62±0.16	82.20±0.62	104.67±7.33	221.67±14.33
SCAFFOLD+FedGen*	82.96±0.13	81.27±0.73	169.33±14.67	415.67±71.00
SCAFFOLD+FedDF	85.40±0.12	83.58±0.45	103.33±8.67	201.00±22.00
SCAFFOLD+FLBoost-f	84.67±0.35	82.76±0.81	103.00±11.00	225.00±29.00

Data heterogeneity and Partial Client Participant. Figure 3(a) displays the test accuracy of compared FL methods on different α values. In this figure, FLBoost achieves the best accuracy on all settings, which validates that FLBoost is effective in various data heterogeneity scenarios. Besides, FLBoost gains more accuracy improvement in extreme data heterogeneity scenario ($\alpha = 0.2$). In addition, as the degree of data heterogeneity decreases (α increases), the accuracy of each method is ascending. Figure 3(b) displays the test accuracy of FL methods with different numbers of active clients participating in each communication round. FLBoost also yields the best performance in this figure. Besides, the more client involved in communication, the higher accuracy will be achieved. Figure 3(c) displays the learning curve of different methods in the first 250 rounds. Though FedDyn has faster accuracy improvement rate in the beginning, its improvement trend is gradually slowing down as the round increases and its accuracy is falling behind FLBoost after 150 rounds.

4.3 Ablation Study

Comments on feature-level pseudo data. Like FedGen, in Table 3 we provide the results of FLBoost using feature-level pseudo data, denote FLBoost-f. In Section 4.2 we conclude that for FedGen, feature-level generation is better then input-level generation (FedGen*). But here we draw a completely different conclusion for FLBoost: though FLBoost-f still exceeds the other methods in Table 1 (including FedGen), it suffers significant performance drop compared with the original FLBoost, which indicates the input-level generation is more effective for FLBoost. Besides, the rounds to reach target accuracy 75% and 80% are greatly increased in FLBoost-f, and even exceeds FedDF



(a) Test accuracy w.r.t. α (b) Test accuracy w.r.t. C (c) Learning Curve

Figure 3: (a) Test accuracy w.r.t. data heterogeneity. (b) Test accuracy w.r.t. fraction C of active clients in each round. (c) Learning Curve of the first 250 rounds. All experiments are conducted on CIFAR10. For (b) and (c), $\alpha = 0.3$ is applied.

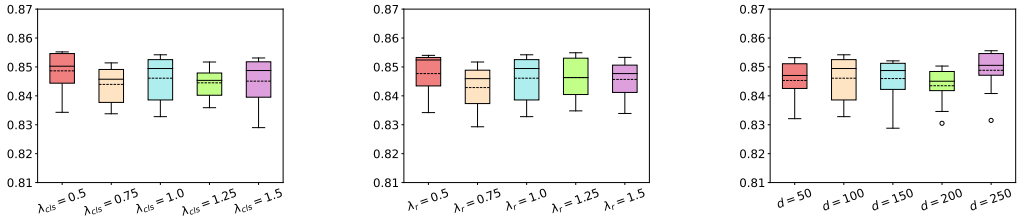
in Table 2 when target accuracy is 75%. Presumably this is because FLBoost-f can only fine-tune the last few layers of the global model, so the effect of knowledge transfer is limited.

Necessity of each component in FLBoost. Table 4 displays the test accuracy of FLBoost after discarding some modules and losses, trained with 500 communication rounds on CIFAR10, $\alpha = 0.3$. Here "FKD", "AEN" and "ATT" represent the modules FocusedKD, AttEnsemble and AdvLearning. We can see that removing each 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 tend 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 Kullback-Leibler divergence with Mean Average Square to measure the model discrepancy ($\mathcal{L}_{mse}$), the model will collapse, which leads to severe performance degradation.

Method		Accuracy
baseline	FLBoost	83.43±0.10
module	- FKD	82.39±0.22
	- AEN	82.40±0.21
	- ADV	82.49±0.42
	- FKD - AEN	82.11±0.28
	- FKD - ADV	82.18±0.16
	- AEN - ADV	82.15±0.14
	- all	81.98±0.14
loss	- $\mathcal{L}_{cls}$	82.50±0.55
	- $\mathcal{L}_{dis}$	82.52±0.35
	- $\mathcal{L}_{cls}$- $\mathcal{L}_{dis}$	82.32±0.16
	$D_{KL} \leftarrow \mathcal{L}_{mse}$	10.17±0.26

Table 4: Impact of the each components in FLBoost (CIFAR10, $\alpha = 0.3$).

Robustness of FLBoost on hyperparameters. To measure the influence of hyperparameter selection, we select λ_{cls} and λ_r 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 4 illustrates the test accuracy in term of the box plot, where FLBoost achieves similar performance among all the choices. Besides, the worst accuracy in Figure 4(a)-(b) is better than the best of previous works in Table 1. This indicates that FLBoost is not sensitive to the selection of hyperparameter in a large range.



(a) Box plot w.r.t. λ_{cls} (b) Box plot w.r.t. λ_r (b) Box plot w.r.t. d

Figure 4: The performance of FLBoost using different hyperparameters (a) λ_{cls}, (b) λ_r and (c) the dimension d of noise z on CIFAR10 with $\alpha = 0.3$.

5 Conclusion

In this paper we propose FLBoost method, which fine-tunes the global model via data-free knowledge distillation, to boost the performance of federated learning. Facing the problem of knowledge heterogeneity in this scenario, we propose FocusedKD to filter invalid knowledge in local models, and design AttEnsemble to derive maximum utility when integrating the knowledge. Extensive experiments validate the efficacy and orthogonality of FLBoost.

Reproducibility Statement. Here we introduce supplementary experiment details for easy reproduction. In experiments, each data is normalized with mean and standard deviation (std) before training. For CIFAR10, the mean are [*0.491, 0.482, 0.447*] and the std are [*0.247, 0.243, 0.262*] for R, G, B channels respectively. For CIFAR100, the mean are [*0.5071, 0.4867, 0.4408*] and the std are [*0.2675, 0.2565, 0.2761*] for R, G, B channels respectively. No data augmentation technique is used for both datasets, such as random crop, random flip, etc. We use official resnet18 implementation[1] in PyTorch without any modification. The code of FLBoost will be released upon the acceptance.

REFERENCES

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

Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3514–3522, 2019.

Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

Wenlin Chen, Samuel Horvath, and Peter Richtarik. Optimal client sampling for federated learning. *arXiv preprint arXiv:2010.13723*, 2020.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*, 2020.

Cong Fang, Hangfeng He, Qi Long, and Weijie J Su. Layer-peeled model: Toward understanding well-trained deep neural networks. *arXiv preprint arXiv:2101.12699*, 2021.

Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019.

Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. *arXiv preprint arXiv:2105.05883*, 2021.

Dashan Gao, Ce Ju, Xiguang Wei, Yang Liu, Tianjian Chen, and Qiang Yang. Hhhfl: Hierarchical heterogeneous horizontal federated learning for electroencephalography. *arXiv preprint arXiv:1909.05784*, 2019.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Florian Hartmann, Sunah Suh, Arkadiusz Komarzewski, Tim D Smith, and Ilana Segall. Federated learning for ranking browser history suggestions. *arXiv preprint arXiv:1911.11807*, 2019.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *arXiv preprint arXiv:2007.14513*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[1] https://pytorch.org/vision/stable/_modules/torchvision/models/resnet.html

Ji Chu Jiang, Burak Kantarci, Sema Oktug, and Tolga Soyata. Federated learning in smart city sensing: Challenges and opportunities. *Sensors*, 20(21):6230, 2020.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pp. 4519–4529. PMLR, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Gihun Lee, Yongjin Shin, Minchan Jeong, and Se-Young Yun. Preservation of the global knowledge by not-true self knowledge distillation in federated learning. *arXiv preprint arXiv:2106.03097*, 2021.

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

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2019.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *arXiv preprint arXiv:2006.07242*, 2020.

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

Songtao Lu, Yawen Zhang, Yunlong Wang, and Christina Mack. Learn electronic health records by fully decentralized federated learning. *arXiv preprint arXiv:1912.01792*, 2019.

Zhiming Luo, Frederic Branchaud-Charron, Carl Lemaire, Janusz Konrad, Shaozi Li, Akshaya Mishra, Andrew Achkar, Justin Eichel, and Pierre-Marc Jodoin. Mio-tcd: A new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing*, 27 (10):5129–5141, 2018.

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

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*, pp. 4743–4751. PMLR, 2019.

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

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

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

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

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

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

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

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 deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021.

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

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*, pp. 1–28, 2021.

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, pp. 12878–12889. PMLR, 2021.

Weiming Zhuang, Xin Gan, Yonggang Wen, Xuesen Zhang, Shuai Zhang, and Shuai Yi. Towards unsupervised domain adaptation for deep face recognition under privacy constraints via federated learning. *arXiv preprint arXiv:2105.07606*, 2021.

A APPENDIX

A.1 ALGORITHM OF CLIENT UPDATE IN FLBOOST

Algorithm 3 illustrates the algorithm of ClientUpdate in FLBoost. Here we instantiate ClientUpdate as SCAFFOLD (Karimireddy et al., 2020). For a better readability, we present its iterate procedure in Algorithm 3. We first initialize the local model ω_k with glocal model ω. Then ω_k is trained using local dataset $\mathcal{D}_k$. Two states s and s_k are used to correct the client-drift caused by heterogeneous data. The state s for server estimates the update direction for global model ω, and the state s_k for client k estimates the update direction for local model ω_k. Their difference $s - s_k$ is then an estimate of the client-drift and is used to correct the local update, as shown in line 4. The readers can refer to SCAFFOLD (Karimireddy et al., 2020) for detailed explanation of the states. Finally, we update the state s_k, and compute the difference Δs_k before and after the update, which will be used to update the state s in server (see Algorithm 2).

Algorithm 3 ClientUpdate, round t

Input: E: local iterations; η_l: local setp-size; k: the client id; ω: global model; $\mathcal{D}_k$: local dataset; s_k: state of client k; s: state of server.

1: initialize local model $\omega_k \leftarrow \omega$
2: **for** $i = 1, ..., E$ **do**
3: compute the mini-batch gradient $g(\omega_k)$
4: $\omega_k \leftarrow \omega_k - \eta_l(g(\omega_k) - s_k + s)$
5: **end for**
6: $s_k \leftarrow s_k - s + \frac{1}{K\eta_l}(\omega - \omega_k)$, $\Delta s_k = -s + \frac{1}{K\eta_l}(\omega - \omega_k)$
7: return ω_k, s_k, Δs_k

A.2 EXPLORATION OF LONG-TAIL PROBLEM

To explore the influence of long-tailed data on model performance, we train the model with multiple imbalanced subsets of CIFAR10, which have different degrees of imbalance. Here the subset is generated by Dirichlet distribution $\mathbf{Dir}(\alpha)$, where a smaller α indicates more imbalanced training data. The data number of each subset is 5000, and the architecture of model is resnet34 (He et al., 2016). The results are illustrated in Figure 5. 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 imbalanced 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, which indicates that the model tends to learn majority data from imbalanced training data and ignore the long-tailed classes. The model achieves high accuracy on imbalanced test data, as the model only learns the majority classes, and these classes also dominate the imbalanced test data. However, for the total test data that contains balanced data for ever class, the model yields lower accuracy, as it can not correctly predict the data of minority classes.

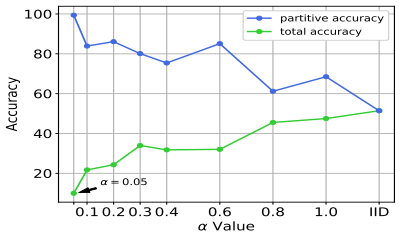


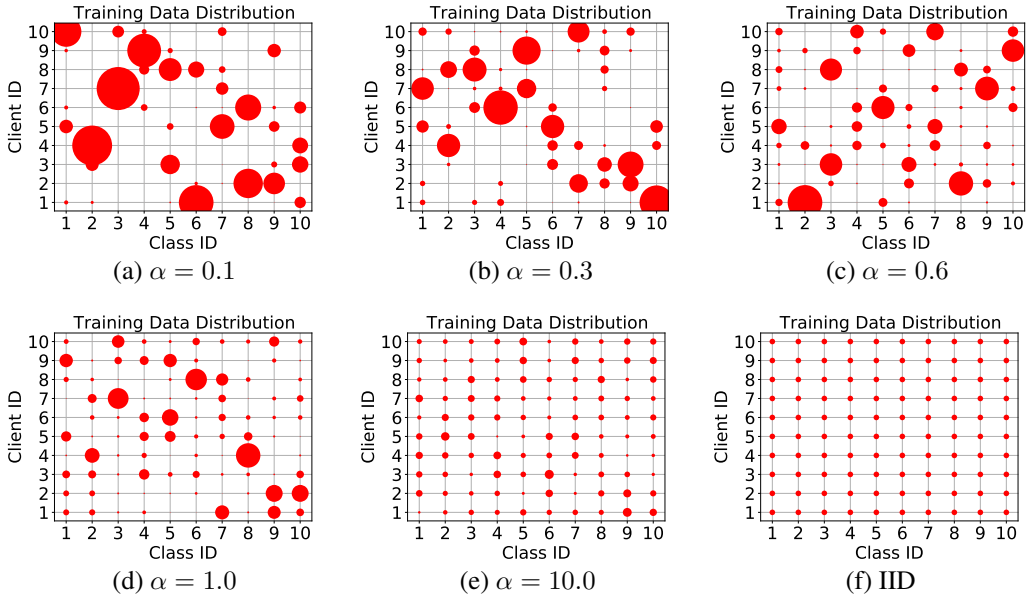
Figure 5: Test accuracy of model trained on class-imbalanced dataset.

A.3 VISUALIZATION OF DATA HETEROGENEITY AMONG CLIENTS

In Figure 6, we figure out the data distributions of clients that generated by Dirichlet distribution $\mathbf{Dir}(\alpha)$ with different α as well as IID data distributions. For each α value, we display the data distributions of 10 clients. In Figure 6, the data distributions of clients are significantly different when α is small, and the client even has no data for some classes. When α grows, the data is distributed more evenly in each client, and the discrepancy of data distributions among clients becomes smaller.

A.4 DETAILED ARCHITECTURE OF GENERATOR

Table 5 lists the architectures of generator for FLBoost, FedDF and FedGen used in Section 4. Here, d is the dimension of noise data z, and it is 100 and 256 for CIFAR10 and CIFAR100, respectively.



(a) $\alpha = 0.1$ (b) $\alpha = 0.3$ (c) $\alpha = 0.6$

(d) $\alpha = 1.0$ (e) $\alpha = 10.0$ (f) IID

Figure 6: Visualization of the number of samples per class allocated to each clients (indicated by dot size), for different α values of Dirichlet distribution.

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 on both generators. Note that in Table 5(b) the output of generator is 512-dimensional, as the input of the last FC layer in ResNet18 is 512-dimensional. For the other classifiers, the dimensions are adjusted accordingly.

Table 5: The architecture of generator used in Section 4.

(a) Generator for FLBoost and FedDF

$z \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ $m = \text{Map}(y) \in \mathbb{R}^M, y \in [1, ..., M]$
FC(z) $\rightarrow$ 4096 FC(m) $\rightarrow$ 4096
Concat $\rightarrow$ 8192
Reshape, BN $\rightarrow 128 \times 8 \times 8$
Conv2D, BN, LeakyReLU $\rightarrow 128 \times 8 \times 8$
Upsampling $\rightarrow 128 \times 16 \times 16$
Conv2D, BN, LeakyReLU $\rightarrow 64 \times 16 \times 16$
Upsampling $\rightarrow 64 \times 32 \times 32$
Conv2D, Tanh $\rightarrow 64 \times 32 \times 32$

(b) Generator for FedGen

$z \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ $m = \text{Map}(y) \in \mathbb{R}^M, y \in [1, ..., M]$
FC(z) $\rightarrow$ 4096 FC(m) $\rightarrow$ 4096
Concat, BN $\rightarrow$ 8192
FC, BN, LeakyReLU $\rightarrow$ 8192
FC $\rightarrow$ 512

A.5 Baseline Description

In the following, we introduce the baselines compared in the experiments.

- **FedAvg** (McMahan et al., 2017) is the first effective *Federated Learning* optimizer to learn a shared model across multiple mobile devices, which contain rich data that is privacy sensitive and large in quantity. It leaves the training data distributed on mobile devices, and learns a shared model by aggregating locally computed updates. The whole training constantly happens between a server and the devices for multiple communication rounds. In each round, the server first sends the global model to devices. Then each device initializes the local model by global model and trains it using local dataset. Finally, the devices send the local models back to server, and the server aggregates new global model by averaging the local models according to the data proportion. Note that in each round, only a fraction of clients will join the training, because of the condition of network connection, the availability of devices, etc. Though the success of FedAvg on protecting privacy and reducing the data transmission cost, it shows degraded performance and slow convergence when the data is non-identically distributed across devices.

- **FedProx** (Li et al., 2018) is a generalization and re-parametrization of FedAvg to improve the convergence in data heterogeneity scenario. It adds a proximal term to restrict the local updates to be closer to the initial (global) model, which is formulated as the L_2 norm between the parameters of global and local models. As a result, the impact of variable local updates is limited, and the global model is optimized with higher stability and faster convergence. Besides, it allows for variable amounts of update steps to be performed locally across devices based on their available systems resources.

- **FedDyn** (Acar et al., 2020) points out a fundamental dilemma in data heterogeneity scenario: the minima of local empirical loss on clients are inconsistent with those of the global empirical loss. Motivated by this, it proposes a dynamic regularizer to align the solutions of global and local. Specifically, it adds a linear penalty term and a quadratic penalty term for each device at each round, whose minima is consistent with the global stationary point. Here, the linear penalty term is formulated as inner product between current local model and the gradient of previous local model, and the quadratic penalty term is formulated as the L_2 norm between the parameters of global and local models. It further provides a theoretical analysis and demonstrates the convergence of the local models with a rate of $O(\frac{1}{T})$, where T is the number of rounds communicated.

- **SCAFFOLD** (Karimireddy et al., 2020) proves that the data heterogeneity could introduce a *drift* in the updates of each client, which will result in slow and unstable convergence. To mitigate this issue, it maintains a state for each client (client control variate s_k) and for the server (server control variate s), which is an estimate of the update direction for the local model and for the global model. The difference $s - s_k$ is then an estimate of the client-drift and is used to correct the local update. In each round, the server sends the state s together with global model to the participating clients, and each client makes use of s to correct the local update. After the local training, the client updates the state s_k using the gradient of local model, and sends back a variant of s_k to server, which will be used to update the server state s.

- **FedGen** (Zhu et al., 2021) utilizes the advantages of data-free knowledge distillation to eliminate the client-drift in data heterogeneity scenario. Specifically, it learns a lightweight generator in server to ensemble data information in a data-free manner, and uses it to regulate local training through the learned knowledge. In each round, the generator in server is sent to each participating client. Then it generates a set of pseudo data and trains the local model together with the local data by minimizing the prediction error. After the clients upload the updated local models to server, the server aggregated a new global model, and trains the generator using all local models via data-free knowledge distillation technique.

- **FedDF** (Lin et al., 2020) investigates a flexible model aggregation scheme in server. Compared with the aforementioned methods that mainly restrict the local model update to align the global and local optimization objectives, FedDF explores a new solution for FL in data heterogeneity scenario. In FedDF, an ensemble knowledge distillation method is proposed for model aggregation, which fine-tune the global model using the ensemble knowledge in local models. The knowledge is extracted through an unlabeled dataset in server, which may be infeasible in real-world tasks.

A.6 Detailed Hyperparameters

Here we introduce the setting of hyperparameters for baselines during experiments. For FedProx, the proximal regularization parameter μ is $1e-4$. α in FedDyn is $1e-2$. We set the local update round in SCAFFOLD following Acar et al. (2020), which is 50 according to our experiment setting. For FedGen and FedDF, the learning rate for generator is the same as FLBoost, 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.

A.7 Performance of FLBoost on Other Network Architectures

Table 6 displays the test accuracy when adopting VGG11 (Simonyan & Zisserman, 2014) and ResNet34 (He et al., 2016) as the classifier. Tables 6 demonstrates FLBoost yields the best performance compared with baselines, which further validates the efficacy of FLBoost on various architectures of deep neural networks.

Table 6: Test Accuracy using VGG11 and ResNet34 on CIFAR10 with $\alpha = 0.3$.

	VGG11	ResNet34
FedAvg	82.05±0.59	80.48±0.89
FedProx	82.10±0.53	81.02±0.53
FedDyn	85.38±0.44	81.13±1.11
SCAFFOLD	86.78±0.37	83.31±0.71
FedGen	84.38±0.56	80.72±0.44
FedDF	84.71±0.78	81.20±0.46
FLBoost	**87.46**±0.49	**85.00**±0.45

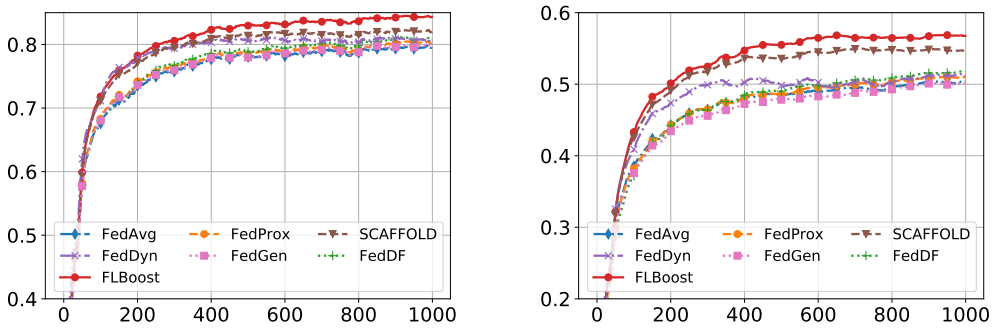
A.8 Discussion

Privacy issue. Since FLBoost trains a generator to recover the training data of clients, it may violate the privacy protection regulation in FL. However, according to our observation, the generator can only captures the high-level feature pattern of training, which can not be understand by human beings (see Figure 1). Besides, as the generator is trained by all local models, the generated data tend to show shared features of data, which means the attribute of individual data will not be revealed. In addition, according to Yin et al. (2020) and Yin et al. (2021), the local model itself can be utilized to recover the training data, without using an auxiliary generator.

On the other hand, uploading data statistics of clients to server may also leak privacy information. One optional solution to alleviate the risk of it is using partial local data during local training, as the global model training in FLBoost is based on the *statistics of data involved in current round*. Using this strategy, the clients need to send the statistics of involved data in each round, rather than upload the statistics of total data once before training. If client designs the data selection properly, the server and data stealer will not know the precise data statistics. Note that partial local training is adopted and tolerated in many methods, such as SCAFFOLD, FedProx and FedFR (Zhuang et al., 2021). Besides, it is common in real-world FL tasks due to the system heterogeneity of clients.

Communication cost. FLBoost only need to additionally transmit the statistics of training data in each round (i.e., $n_t^{k,y}, \forall k \in S_t, \forall y \in [1, .., M]$, M the class number), which induces negligibly extra transmission cost. If the training data and the local training epoch of clients keep the same during training, the statistics of every client can be reported to server before training, so that no extra transmission cost will be induced.

A.9 Learning Curve on the Whole Training Process



(a) Learning curve of CIFAR10.

(b) Learning curve of CIFAR100.

Figure 7: Learning Curves of CIFAR10 and CIFAR100 on 1000 communication rounds ($\alpha = 0.3$).

Figure 7 illustrates the learning curve of CIFAR10 and CIFAR100 on the whole 1000 communication rounds, which corresponds to the results in Table 2. In this figure, FLBoost exceeds the other methods in about 150 and 100 rounds respectively, and it achieves distinct performance gain after 1000 rounds.

A.10 EXPERIMENTS ON MORE CHALLENGING DATASETS

Table 7: 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
FLBoost	91.16±0.92	51.85±3.46	42.23±0.22

In this section, we test the performance of FLBoost on more challenging real-world datasets - vehicle classification datasets MIOTCD (Luo et al., 2018) and CompCar (Yang et al., 2015), and large-scale image classification dataset Tiny-ImageNet[2]. To better validate the efficacy of FLBoost in real-world scenario, 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. α 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. We display the experiment results in Table 7.

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

[2]https://www.kaggle.com/c/tiny-imagenet