# Provably Near-Optimal Federated Ensemble Distillation with Negligible Overhead

**Won-Jun Jang** [1] **Hyeon-Seo Park** [1] **Si-Hyeon Lee** [1]

## Abstract

Federated ensemble distillation addresses client heterogeneity by generating pseudo-labels for an unlabeled server dataset based on client predictions and training the server model using the pseudo-labeled dataset. The unlabeled server dataset can either be pre-existing or generated through a data-free approach. The effectiveness of this approach critically depends on the method of assigning weights to client predictions when creating pseudo-labels, especially in highly heterogeneous settings. Inspired by theoretical results from GANs, we propose a provably near-optimal weighting method that leverages client discriminators trained with a server-distributed generator and local datasets. Our experiments on various image classification tasks demonstrate that the proposed method significantly outperforms baselines. Furthermore, we show that the additional communication cost, client-side privacy leakage, and client-side computational overhead introduced by our method are negligible, both in scenarios with and without a pre-existing server dataset.

## 1. Introduction

Federated learning (FL) (McMahan et al., 2017) has received substantial attention in both industry and academia as a promising distributed learning approach. It enables numerous clients to collaboratively train a global model without sharing their private data. A major concern in deploying FL in practice is the severe data heterogeneity across clients. In the real world, it's probable that clients possess non-IID (identical and independently distributed) data distributions. It is known that the data heterogeneity results in unstable convergence and performance degradation (Li et al., 2020b; Wang et al., 2020b; Li & Zhan, 2021; Kairouz et al., 2021; Huang et al., 2023; Karimireddy et al., 2020).

To address the data heterogeneity issue, various approaches have been taken, including regularizing the objectives of the client models (Karimireddy et al., 2020; Li et al., 2020a; Liang et al., 2019; Yao et al., 2021; Mendieta et al., 2022; Son et al., 2024), normalizing features or weights (Dong et al., 2022; Kim et al.), utilizing past round models (Yao et al., 2021; Wang et al., 2023b), sharing feature information (Dai et al., 2023; Yang et al., 2024; Tang et al., 2024; Zhang et al., 2024), introducing personalized layers (Huang et al., 2023), and learning the average input-output relation of client models through ensemble distillation (Chang et al., 2019; Gong et al., 2021; Deng et al., 2023; Sattler et al., 2020; Lin et al., 2020; Cho et al., 2022; Xing et al., 2022; Park et al., 2024; Wang et al., 2023a; Tang et al., 2022; Zhang et al., 2022; 2023a). In particular, the last approach, federated ensemble distillation, has recently gained significant attention for its effectiveness in mitigating data heterogeneity and for its advantage of being effectively applicable to heterogeneous client models. It requires an unlabeled dataset at the server, for which pseudo labels are created based on client predictions. By training on this pseudo-labeled dataset at the server, the server distills the knowledge from the clients. This additional dataset can be either public (Chang et al., 2019; Gong et al., 2021; Deng et al., 2023; Sattler et al., 2020), held only by the server due to its exceptional data collection capability (Lin et al., 2020; Cho et al., 2022; Xing et al., 2022; Park et al., 2024), or obtained through a data-free approach (Wang et al., 2023a; Tang et al., 2022; Zhang et al., 2022; 2023a). Note that the performance of ensemble distillation depends on the quality of the pseudo-labels, which ultimately translates into a problem of appropriately assigning weights to client predictions for each data point, particularly in situations of data heterogeneity. In this research stream of federated ensemble distillation, early studies like FedDF (Lin et al., 2020) applied uniform weighting. Subsequently, algorithms such as Fed-ET (Cho et al., 2022), FedHKT (Deng et al., 2023), FedDS (Park et al., 2024), and DaFKD (Wang et al., 2023a) emerged, which utilize metrics like variance, entropy, and judgement of client discriminator as indicators of confidence in client predictions for weighting. However,

[1]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. Correspondence to: Si-Hyeon Lee <sihyeon@kaist.ac.kr>.

*Proceedings of the $42^{nd}$ International Conference on Machine Learning*, Vancouver, Canada. PMLR 267, 2025. 

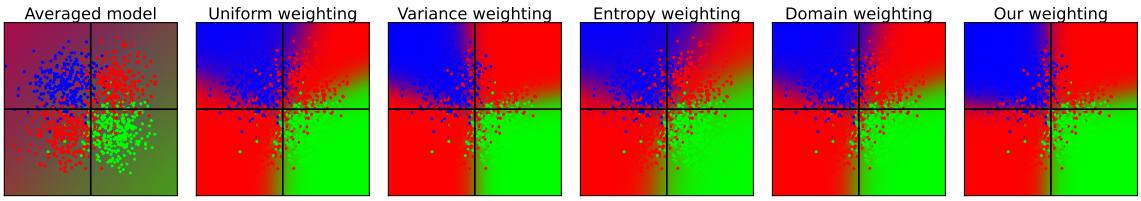



*Figure 1.* A toy example of decision boundaries of aggregated models. Each point represents data, and its color represents the label. The background color represents the decision boundary of each model in the RGB channels. The oracle decision boundary, shown by the black lines, corresponds to the $x$-axis and $y$-axis. For aggregated models, we consider the parameter-averaged model (McMahan et al., 2017) and ensemble-distilled models using uniform weighting (Lin et al., 2020), variance weighting (Cho et al., 2022), entropy weighting (Deng et al., 2023; Park et al., 2024), domain-aware weighting (Wang et al., 2023a), and ours. Detailed settings are provided in Appendix E.1.

analysis regarding the rationale behind optimal weighting remains scarce.

In this paper, we suggest a novel weighting method for federated ensemble distillation that outperforms previous methods (Fig. 1), with theoretically justified optimality based on some results in generative adversarial networks (GANs) (Goodfellow et al., 2014). Our main contributions are summarized in the following:

- We propose **FedGO**: **F**ederated **E**nsemble **D**istillation with **G**AN-based **O**ptimality. Our algorithm incorporates a novel weighting method using the client discriminators that are trained at the clients based on the generator distributed from the server.
- The optimality of our proposed weighting method is theoretically justified. We define an optimal model ensemble and show that a knowledge-distilled model from an optimal model ensemble achieves the optimal performance, within an inherent gap due to the difference between the spanned hypothesis class of ensemble model and the hypothesis class of a single model. Then, based on the theoretical result for vanilla GAN (Goodfellow et al., 2014), we show that our weighting method using client discriminators constitutes an optimal model ensemble.
- We experimentally demonstrate significant improvements of FedGO over existing research both in final performance and convergence speed on multiple image datasets (CIFAR-10/100, ImageNet100). In particular, we demonstrate performance across various scenarios, including cases where the server holds an unlabeled dataset different from the client datasets and where the server does not hold an unlabeled dataset, requiring data-free approaches. Furthermore, we provide a comprehensive analysis of communication cost, privacy leakage, and computational burden for the proposed method, showing that client-side overheads are negligible both in scenarios with and without a pre-existing server dataset.

For ease of reproduction, our code is open-sourced (https://github.com/pupiu45/FedGO).

## 2. System Model and Related Work

**Federated Learning** In federated learning, the goal is to cooperatively train a global model based on data distributed among $K$ clients, by exchanging the models between a server and the clients.

We focus on classification tasks in this paper. Let $\mathcal{X}$ denote the data domain and $y$ denote the labeling function that outputs the label of the data $x \in \mathcal{X}$. A model $f(\cdot;\theta)$ is parameterized by $\theta \in \Theta$ where $\Theta$ is the set of model parameters and $\mathcal{H} = \{h | h(\cdot) = f(\cdot;\theta), \theta \in \Theta\}$ denotes the class of parameterized models. For a distribution $q$ on $\mathcal{X}$, $h_q^*$ denotes the expected loss minimizer on $q$, i.e., $h_q^* \triangleq \arg\min_{h\in\mathcal{H}} \mathcal{L}_q(h)$, where $\mathcal{L}_q(h) = \mathbf{E}_q[l(h(x), y(x))]$ and $l$ is the loss function. Client $k$ possesses a (labeled) dataset $S_k$ of $n_k$ data points, sampled over $\mathcal{X}$ i.i.d. according to distribution $p_k$. Then $p = \sum_{k=1}^K \pi_k \cdot p_k$, where $\pi_k = \frac{n_k}{\sum_{k'=1}^K n_{k'}}$, is the average of client data distribution. The objective of federated learning is given as follows:

$$\min_{h\in\mathcal{H}} \mathcal{L}_p(h) = \min_{h\in\mathcal{H}} \mathbf{E}_p[l(h(x), y(x))] \tag{1}$$

$$= \min_{h\in\mathcal{H}} \sum_{k=1}^K \pi_k \cdot \mathbf{E}_{p_k}[l(h(x), y(x))] = \min_{h\in\mathcal{H}} \sum_{k=1}^K \pi_k \cdot \mathcal{L}_{p_k}(h). \tag{2}$$

In each communication round $t$, a subset $A^t$ of clients downloads the current server model and trains it based on $S_k$ with the objective of minimizing $\mathcal{L}_{p_k}(h)$. Then it sends the trained model to the server. The server aggregates these client models to update the server model. The aforementioned procedure is repeated at the next communication round. For the aggregation of client models at the server, the FedAVG algorithm (McMahan et al., 2017) constructs the server model with parameter $\theta_s^t$ for round $t$ as the average of model parameters $\theta_k^t$ for $k \in A^t$ received in round $t$ (line 7 of Algorithm 1). When the client data distributions are homogeneous, each $p_k$ is same as $p$ and hence $\mathcal{L}_{p_k}$ becomes same as $\mathcal{L}_p$. However, when the client data distributions are heterogeneous, $\mathcal{L}_{p_k}$ and $\mathcal{L}_p$ are not same, leading to a

**Algorithm 1** Federated learning with $K$ clients for $T$ communication rounds, with ensemble distillation exploiting unlabeled dataset on the server. Client $k$ possesses $n_k$ data points, and the fraction $C$ of clients participate in each communication round. $f(\cdot;\theta)$ stands for the model with parameter $\theta$, and $\mu$ stands for the step size.

**Require:** Client labeled dataset $\{S_k\}_{k=1}^K$, server unlabeled dataset $U$
Initialize server model $f(\cdot;\theta_s^0)$ with parameter $\theta_s^0$
**for** communication round $t = 1$ to $T$ **do**
  $A^t \leftarrow$ sample $\lfloor C \cdot K \rfloor$ clients
  **for parallel** client $k \in A^t$ **do**
    $\theta_k^t \leftarrow ClientUpdate(\theta_s^{t-1}, S_k)$ ▷ *Gradient update $\theta_s^{t-1}$ with $S_k$*
  **end for**
  $\theta_s^t \leftarrow \sum_{k\in A^t} \frac{n_k}{\sum_{i\in A^t} n_i} \cdot \theta_k^t$
  **for** server train epoch $e = 1$ to $E_s$ **do**
    **for** unlabeled minibatch $u \in U$ **do**
      $\tilde{y}(u) \leftarrow \sigma(\sum_{k\in A^t} w_k(u) \cdot f(u;\theta_k^t))$ ▷ *Label as a weighted sum of client predictions*
      $\theta_s^t \leftarrow \theta_s^t - \mu \cdot \nabla_{\theta_s^t} \mathrm{KL}(\tilde{y}(u), \sigma(f(u;\theta_s^t)))$ ▷ *Ensemble distillation*
    **end for**
  **end for**
**end for**
**return** $f(\cdot;\theta_s^T)$

significant degradation in the convergence rate of FedAVG to the global optimum (Li et al., 2020b).

In the following, we introduce federated ensemble distillation using an unlabeled dataset on the server to address client data heterogeneity.

**Federated Ensemble Distillation** To address client data heterogeneity, there has been a line of research on federated ensemble distillation using an unlabeled dataset on the server. This unlabeled dataset may either be available from the outset (Lin et al., 2020; Cho et al., 2022; Deng et al., 2023; Park et al., 2024) or produced through a generator trained as a part of FL by taking a data-free approach (Rasouli et al., 2020; Guerraoui et al., 2020; Li et al., 2022; Wang et al., 2023c; Fan & Liu, 2020; Behera et al., 2022; Hardy et al., 2019; Xiong et al., 2023; Zhang et al., 2021; 2023b; Wang et al., 2023a; Zhang et al., 2022; 2023a). With the unlabeled dataset, the server model undergoes additional training to learn the average input-output relationship of client models.

Algorithm 1 describes this federated ensemble distillation, when the client and server model structures are the same. Here $\sigma$ represents the softmax function, and KL denotes the Kullback-Leibler divergence. If the model output already includes the softmax activation, then the softmax function is omitted in lines 10 and 11. After averaging client model parameters in line 7, the performance of the server model depends on the quality of the pseudo-labels, as the server model undergoes additional training with those pseudo-labels. Moreover, the quality of pseudo-labels $\tilde{y}(\cdot)$ relies on designing the weighting function $w_k(\cdot)$, which determines the weighting of client $k$'s output. Therefore, designing a better-performing ensemble distillation during the server update ultimately boils down to designing a better-performing weighting function.

For the weighting function, FedDF (Lin et al., 2020) uses uniform weights for each client, i.e., $w_k(x) = \frac{1}{|A^t|}$ for all $k$ in $A^t$. Subsequently, algorithms assigning higher weights to the outputs of more confident clients have been proposed. In Fed-ET (Cho et al., 2022), higher weights are assigned to models with larger output logit variance, i.e., $w_k(x) = \frac{\mathrm{Var}(f(x;\theta_k^t))}{\sum_{i\in A^t} \mathrm{Var}(f(x;\theta_i^t))}$. FedHKT (Deng et al., 2023) and FedDS (Park et al., 2024) allocate higher weights to models with smaller output softmax entropy, i.e., $w_k(x) = \frac{\exp(-\mathrm{Entropy}(\sigma(f(x;\theta_k^t)))/\tau)}{\sum_{i\in A^t} \exp(-\mathrm{Entropy}(\sigma(f(x;\theta_i^t)))/\tau)}$, where $\tau$ is the temperature parameter. In DaFKD (Wang et al., 2023a), while training a global generator and client discriminators at each round, ensemble distillation is performed on unlabeled dataset generated by the global generator by assigning higher weights to models with larger discriminator outputs, i.e., $w_k(x) = \frac{D_k^t(x)}{\sum_{i\in A^t} D_i^t(x)}$ where $D_k^t$ is the client $k$'s discriminator against the global generator at round $t$.

For theoretical aspects, generalization bounds of an ensemble model are presented in Table 1 for a binary classification task under $\ell_1$ loss. For $n_k = \frac{n}{K}$ for all $k$, a generalization bound for an ensemble model with fixed weights $\alpha_1, ..., \alpha_K$ with $\sum_k \alpha_k = 1$ is given as (1.1) (Lin et al., 2020; Cho et al., 2022), with weight function $w_k(x) = \frac{D_k^t(x)}{\sum_{i\in A^t} D_i^t(x))}$ is given as (1.2) (Wang et al., 2023a), and with our weight function $w_k^*(x)$ described in Section 3.1 is given as (1.3).

*Table 1.* Generalization bound of the ensemble model of ensemble distillation algorithms. For any $\delta \in (0,1)$ and $\sigma > 0$, the generalization bounds hold with probability $1-\delta$. Here, $\hat{p}_k$ is the empirical distribution by sampling $n_k$ data points i.i.d. according to $p_k$, $d_{\mathcal{H}\triangle\mathcal{H}}$ denotes the discrepancy between two distributions, $\lambda_k = \inf_h \mathcal{L}_{p_k}(h) + \mathcal{L}_p(h)$, and $\tau_{\mathcal{H}}$ is growth function bounded by polynomial of the VC-dimension of $\mathcal{H}$.

| Algorithm | Generalization Bound | |
|---|---|---|
| FedDF (Lin et al., 2020), Fed-ET (Cho et al., 2022) | $\mathcal{L}_p(\sum_{k=1}^K \alpha_k h^*_{\hat{p}_k}) \le \sum_{k=1}^K \alpha_k \cdot \left[\mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \frac{1}{2} d_{\mathcal{H}\triangle\mathcal{H}}(p_k, p) + \lambda_k + O\left(\frac{\log(\delta^{-1})}{\sqrt{n_k}}\right)\right]$ | (1.1) |
| DaFKD (Wang et al., 2023a) | $\mathcal{L}_p(\sum_{k=1}^K w_k \cdot h^*_{\hat{p}_k}) \le (K+1) \cdot \sum_{k=1}^K \frac{1}{K} \cdot \left[\mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \sqrt{\frac{\sigma^2 \log \frac{2K}{\delta}}{2n_k}}\right]$ | (1.2) |
| **Ours, Theorem** C.1 | $\mathcal{L}_p(\sum_{k=1}^K w^*_k \cdot h^*_{\hat{p}_k}) \le \sum_{k=1}^K \pi_k \cdot \left[\mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}\right]$ | (1.3) |

These bounds relate the loss of an ensemble model (the LHS of (1.1), (1.2) and (1.3)) to the average empirical loss of client models (the first term in the RHS of (1.1), (1.2) and (1.3)). Note that the bound (1.1) assumes a fixed weight per client irrelevant to data points, hence there is a lack of analysis for assigning varying weights per data point. Furthermore, (1.1) provides a generalization bound that becomes loose as it scales with the average distribution discrepancy between $p$ and $p_k$ across clients. The bound (1.2) assumes a specific weighting function of each data point, but it is too loose because it becomes vacuous as $K$ increases. In contrast, (1.3), which is derived in this paper, provides a refined generalization bound for weighting functions. To the best of our knowledge, this is the tightest bound currently provided, as detailed in Theorem C.1. Furthermore, we also emphasize that the bound (1.3) is independent of client data heterogeneity, as it does not include any client data discrepancy term.

Moreover, in federated ensemble distillation, our ultimate interest is in the loss of the server model, knowledge-distilled from the ensemble model. Note that the hypothesis class of ensemble models is in general larger than that of single models, and hence there exists an inherent gap between the losses of an ensemble model and the knowledge-distilled model. However, the above bounds do not provide an analysis on this gap.

In Section 3.1, we define an optimal model ensemble and show that the server model knowledge-distilled from an optimal model ensemble achieves the optimal loss within the gap arising from the distillation step, which depends on the inherent difference between the hypothesis classes of the server model and the ensemble model, along with the distribution discrepancy between the average client distribution $p$ and the distribution $p_s$ of unlabeled data on the server.

**Generative Adversarial Network** The generative adversarial networks (GANs) are a class of powerful generative models composed of a generator and a discriminator (Goodfellow et al., 2014; Gulrajani et al., 2017; Radford et al., 2016; Chen et al., 2016; Zhu et al., 2017; Choi et al., 2018; Karras et al., 2019). They are trained in an unsupervised learning manner, requiring no class labels. The discriminator aims to distinguish between real images from the dataset and fake images generated by the generator. Meanwhile, the generator strives to produce images that can fool the discriminator.

In Goodfellow et al. (2014), the authors showed that the output of an optimal discriminator against a fixed generator can be expressed in terms of distributions of real and fake images.

**Theorem 2.1.** *(Goodfellow et al., 2014, Proposition 1.) For a fixed generator $G$, let $p_g$ and $p_{data}$ denote the density functions of the generated distribution by $G$ and the real data distribution, respectively. Then the output of an optimal discriminator $D$ for input data $x$ is given as follows:*

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}. \tag{3}$$

Using the above result, we develop a method of assigning weights to client predictions in Section 3.

## 3. Proposed Method

In this section, we propose a weighting method for federated ensemble distillation. First, theoretical results are presented in Section 3.1. In Section 3.1.1, we define an optimal model ensemble and give a bound on the loss of the server model knowledge-distilled from an optimal model ensemble. Next, in Section 3.1.2, we propose a client weighting method to construct an optimal model ensemble, based on Theorem 2.1. In Section 3.2, we introduce our FedGO algorithm, leveraging the theoretical results. We note that a generalization bound of an ensemble model with our proposed weighting method comparable with (1.1) is provided in Appendix C.

### 3.1. Theoretical Results

#### 3.1.1. Ensemble Distillation with Optimal Model Ensemble

We first define an optimal model ensemble.

**Definition 3.1.** For $K$ clients, the ensemble of their models

and weight functions $\{(h_k, w_k)\}_{k=1}^K$ is said to be an optimal model ensemble if the following holds:

$$\mathcal{L}_p\left(\sum_{k=1}^K w_k \cdot h_k\right) = \mathbf{E}_p\left[l\left(\sum_{k=1}^K w_k(x) \cdot h_k(x), y(x)\right)\right] \leq \min_{h\in\mathcal{H}} \mathcal{L}_p(h) = \mathcal{L}_p(h_p^*). \quad (4)$$

We remind that the objective of federated learning is to train a model that minimizes the expected loss over the average client distribution $p$ as shown in (1). If $\{(h_k, w_k)\}_{k=1}^K$ is an optimal model ensemble, its expected loss over $p$ is less than or equal to the minimum expected loss over $p$ achievable by a single model, i.e., $\min_{h\in\mathcal{H}} \mathcal{L}_p(h)$.

However, we cannot guarantee that a knowledge-distilled model from an optimal model ensemble would be optimal, i.e., achieve $\min_{h\in\mathcal{H}} \mathcal{L}_p(h)$, due to the following two reasons: 1) the ensemble model $\sum_{k=1}^K w_k \cdot h_k$ may lie outside the hypothesis class $\mathcal{H}$ of a single model and 2) the distribution used for knowledge distillation (the distribution $p_s$ of unlabeled data on the server) can be different from $p$. In the following theorem, we present a bound on the expected loss over $p$ of a single model by taking into account these factors. For two hypotheses $h, h' \in \mathcal{H}$ and a distribution $q$ over $\mathcal{X}$, the expected difference between $h$ and $h'$ over $q$, denoted $\mathcal{L}_q(h, h')$, is defined as follows:

$$\mathcal{L}_q(h, h') \triangleq \mathbf{E}_q\left[(l(h(x), h'(x))\right]. \quad (5)$$

**Theorem 3.2.** *(Informal) Let* $\bar{\mathcal{H}} \triangleq \{\sum_{k=1}^K w_k \cdot h_k | h_j \in \mathcal{H}, w_j : \mathcal{X} \to [0,1], \sum_{k=1}^K w_k(x) = 1, j = 1, \cdots, K, x \in \mathcal{X}\}$ *be the spanned hypothesis class,* $p_s$ *be a distribution on* $\mathcal{X}$*, and* $\{(h_k, w_k)\}_{k=1}^K$ *be an ensemble of client models and weight functions. Then for any* $h \in \mathcal{H}$*, the following holds:*

$$\mathcal{L}_p(h) \leq \mathcal{L}_p(\sum_{k=1}^K w_k \cdot h_k) + \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k) + \frac{1}{2} d_{\bar{\mathcal{H}}\triangle\bar{\mathcal{H}}}(p, p_s). \quad (6)$$

The formal statement and the proof of the above theorem are in Appendix A. Let us provide a brief sketch of the proof. Utilizing the results from Ben-David et al. (2006) and Crammer et al. (2008), we have $\mathcal{L}_p(h) \leq \mathcal{L}_p(\sum_{k=1}^K w_k \cdot h_k) + \mathcal{L}_p(h, \sum_{k=1}^K w_k \cdot h_k)$. Then from the triangular inequality, we obtain $\mathcal{L}_p(h, \sum_{k=1}^K w_k \cdot h_k) \leq \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k) + |\mathcal{L}_p(h, \sum_{k=1}^K w_k \cdot h_k) - \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k)|$. Now the desired inequality is obtained by applying the results from Ben-David et al., Lemma 3.

From Theorem 3.2, we can ascertain the following. The loss of the server model $h$ is bounded by the sum of three losses: 1) expected loss of the ensemble model over $p$, 2) difference between $h$ and $\sum_{k=1}^K w_k \cdot h_k$ over $p_s$, and 3) the distribution discrepancy between $p$ and $p_s$.

The following corollary is a direct consequence of Theorem 3.2 and Definition 3.1.

**Corollary 3.3.** *(Informal) For an optimal model ensemble* $\{(h_k, w_k)\}_{k=1}^K$*, the following holds for any* $h \in \mathcal{H}$*:*

$$\mathcal{L}_p(h_p^*) \leq \mathcal{L}_p(h) \leq \mathcal{L}_p(h_p^*) + \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k) + \frac{1}{2} d_{\bar{\mathcal{H}}\triangle\bar{\mathcal{H}}}(p, p_s). \quad (7)$$

Corollary 3.3 demonstrates the powerfulness of an optimal model ensemble. If an optimal model ensemble is constituted, the difference between the expected loss of the server model over $p$ and the minimum expected loss $\mathcal{L}_p(h_p^*) = \min_{h\in\mathcal{H}} \mathcal{L}_p(h)$ is bounded by the distillation loss, which depends on the inherent difference between the hypothesis class $\mathcal{H}$ and the spanned hypothesis class $\bar{\mathcal{H}}$, along with the distribution discrepancy between $p$ and $p_s$.

In the next subsection, we propose a weighting method to constitute an optimal model ensemble.

### 3.1.2. Client Weighting for Optimal Model Ensemble

Let us assume that the server has models $\{h_{p_k}^*\}_{k=1}^K$ trained by clients based on their respective data distributions $\{p_k\}_{k=1}^K$. In the following theorem, we present weight functions $\{w_k\}_{k=1}^K$ such that the ensemble of $\{h_{p_k}^*, w_k\}_{k=1}^K$ constitutes an optimal model ensemble.

**Theorem 3.4.** *Let the loss function* $l$ *be convex. Define the client weight functions* $\{w_k^*\}_{k=1}^K$ *as follows:*

$$w_k^*(x) \triangleq \frac{n_k \cdot p_k(x)}{\sum_{i=1}^K n_i \cdot p_i(x)} = \frac{\pi_k \cdot p_k(x)}{\sum_{i=1}^K \pi_i \cdot p_i(x)}. \quad (8)$$

*Then, the ensemble* $\{h_{p_k}^*, w_k^*\}_{k=1}^K$ *is an optimal model ensemble, i.e.,* $\mathcal{L}_p\left(\sum_k w_k^* \cdot h_{p_k}^*\right) \leq \mathcal{L}_p(h_p^*)$.

Theorem 3.4 follows from some manipulations based on the convexity of the loss and the definitions of $w_k^*$'s and $h_{p_k}^*$'s, and its full proof is provided in Appendix B.

Theorem 3.4 demonstrates that for data point $x$, weighting according to each client's proportion of having $x$ constitutes an optimal model ensemble. However, even if weighting each client according to Theorem 3.4 constitues an optimal model ensemble, it is not feasible without knowing the data distribution $p_k$ of each client. Theorem 3.6 addresses this issue based on Theorem 2.1 and provides hints on how to implement an optimal model ensemble.

**Definition 3.5.** (Odds): For $\phi \in (0, 1)$, its odds value $\Phi$ is defined as $\Phi(\phi) = \frac{\phi}{1-\phi}$.

*Table 2.* A comprehensive analysis of additional communication burden, privacy leakage, and computational burden caused by the proposed weighting method, compared to FedAVG.

| Extra Server Dataset | Generator Preparation | Distillation Dataset | Communication Cost | Privacy Leakage | | Client-side Computational Burden |
|---|---|---|---|---|---|---|
| | | | | Server-side | Client-side | |
| S1 | G1 | D1 | Negligible | Non-negligible | Negligible | Negligible |
| S1 | G2 | D1 | Negligible | Non-negligible | Negligible | Negligible |
| S2 | G2 | D2 | Negligible | - | Negligible | Negligible |
| S2 | G3 | D2 | Negligible | - | Negligible | Negligible |

**Theorem 3.6.** *For a fixed generator $G$ with generating distribution $p_g$, let $D_k$ be an optimal discriminator for generator $G$ and client $k$'s distribution $p_k$. Assume that $D_k$ outputs a value over $(0,1)$ using a sigmoid activation function, and let $\Phi_k(x) \triangleq \Phi(D_k(x))$. Then, for $x \in supp(p_g)$, the following holds:*

$$\frac{n_k \cdot \Phi_k(x)}{\sum_{i=1}^K n_i \cdot \Phi_i(x)} = \frac{\pi_k \cdot p_k(x)}{\sum_{i=1}^K \pi_i \cdot p_i(x)} = w_k^*(x). \quad (9)$$

Theorem 3.6 is a direct consequence of Theorem 2.1, because $\Phi_k(x) = \frac{p_k(x)}{p_g(x)}$ from Theorem 2.1. Theorem 3.6 indicates that if the server once receives the optimal discriminators $\{D_k\}_{k=1}^K$ trained by the clients, it can use those discriminators to calculate the weights for optimal model ensemble. Note that the generator $G$ only needs to generate a wide distribution capable of producing sufficiently diverse samples. Therefore, one can use an off-the-shelf generator pretrained on a large dataset.

### 3.2. Proposed Algorithm: FedGO

By leveraging the theoretical results in Section 3.1, we propose FedGO that constitutes an optimal model ensemble and performs knowledge distillation. The main technical novelty of FedGO lies in implementing the optimal weighting function $w_k^*$ using client discriminators, which is a versatile technique that can be integrated to both the following scenarios with/without extra server dataset: (S1) the server holds an extra unlabeled dataset; (S2) the server holds no unlabeled dataset, thus a data-free approach is needed.

For completeness, let us describe how the FedGO algorithm can be adapted depending on the cases (S1) and (S2). FedGO largely consists of two stages: pre-FL and main-FL. In the pre-FL stage, the server and the clients exchange the generator and the discriminators. First, the server obtains a generator through one of the following three methods, and distributes the generator to the clients: (G1) train a generator with an unlabeled dataset on the server, which is possible under (S1); (G2) load an off-the-shelf generator pretrained on a sufficiently rich dataset; or (G3) train a generator through an FL approach, e.g., using FedGAN (Rasouli et al., 2020).

After receiving the generator, each client trains its own discriminator based on its dataset and sends the discriminator to the server.

The main-FL stage operates according to Algorithm 1, except that the server assigns weights for pseudo-labeling according to (9) using the client discriminators. For the server unlabeled dataset $U$ used for distillation, which we call distillation dataset, we consider the following cases: (D1) use the same dataset held by the server, which is possible under (S1); (D2) produce a distillation dataset using the generator.

A comprehensive analysis of additional communication cost, privacy leakage, and computational burden according to the methods for obtaining the generator and distillation set is provided in Table 2, which shows the trade-off among the methods. In particular, an extra dataset at the server makes the communication cost and the client-side privacy and computational burden negligible, at the expense of server-side privacy leakage. In the absence of server dataset, the use of an off-the-shelf generator makes all the burdens negligible, but it can be challenging to secure an off-the-shelf generator whose generation distribution is similar to the client data distribution. Lastly, the data-free approach (G3)+(D2) does not require an extra server dataset or an external generator. While it requires extra communication to prepare the generator, the additional communication burden, privacy leakage and computational burden on the client side remains negligible due to the relatively small number of communication rounds involved, unlike existing data-free methods such as DaFKD (Wang et al., 2023a), which involve both generator and model exchanges in every communication round. By decoupling generator preparation from model training, FedGO with (G3)+(D2) provides a lightweight solution.

A detailed description of FedGO and explanation for Table 2 can be found in Appendices D and G, respectively.

## 4. Experimental Results

In this section, we present the experimental results. All experimental results were obtained using five different random seeds, and the reported results are presented as the mean $\pm$ standard deviation.

### 4.1. Experimental Setting

**Datasets and FL Setup** We employed datasets CIFAR-10/100 (Krizhevsky, 2009) (MIT license) and downsam-

*Table 3.* Server test accuracy (%) of our FedGO and baselines on three image datasets at the 100-th communication round. A smaller $\alpha$ indicates higher heterogeneity.

| | CIFAR-10 | | CIFAR-100 | | ImageNet100 | |
|---|---|---|---|---|---|---|
| | $\alpha = 0.1$ | $\alpha = 0.05$ | $\alpha = 0.1$ | $\alpha = 0.05$ | $\alpha = 0.1$ | $\alpha = 0.05$ |
| Central Training | 85.33±0.25 | | 51.72±0.65 | | 43.20±1.00 | |
| FedAVG | 58.65±5.75 | 46.61±8.54 | 38.93±0.74 | 36.66±0.97 | 29.44±0.41 | 27.58±0.88 |
| FedProx | 64.69±2.15 | 55.56±9.86 | 38.21±0.95 | 34.44±1.26 | 29.96±0.66 | 26.99±0.97 |
| SCAFFOLD | 61.20±3.98 | 50.10±10.00 | 38.15±0.80 | 36.14±1.06 | 29.13±0.79 | 27.08±0.69 |
| FedDisco | 56.78±7.22 | 48.08±8.35 | 38.81±1.02 | 36.86±0.88 | 29.69±0.66 | 27.54±0.51 |
| FedUV | 62.58 ± 4.83 | 53.80 ± 5.68 | 38.84 ± 0.79 | 36.17 ± 1.24 | 30.09 ± 1.09 | 27.32 ± 0.65 |
| FedTGP | 61.16 ± 6.98 | 61.51 ± 7.78 | 39.58 ± 0.10 | 36.56 ± 0.11 | 29.21 ± 1.13 | 26.34 ± 1.02 |
| FedDF | 71.56±5.09 | 59.53±9.88 | 42.74±1.22 | 37.18±1.03 | 33.48±1.00 | 30.94±1.60 |
| FedGKD$^+$ | 72.59±4.10 | 59.96±8.60 | 43.35±1.14 | 40.47±1.00 | 34.10±0.67 | 31.42±0.93 |
| DaFKD | 71.52±5.56 | 67.51±10.77 | 44.12±2.25 | 39.50±0.85 | 33.34±0.69 | 31.59±1.46 |
| **FedGO (ours)** | **79.62**±4.36 | **72.35**±9.01 | **44.66**±1.27 | **41.04**±0.99 | **34.20**±0.71 | **31.70**±1.55 |

*Table 4.* The number of communication rounds to achieve a test accuracy of at least $\text{Acc}_{\text{target}}$.

| | CIFAR-10 | | CIFAR-100 | | ImageNet100 | |
|---|---|---|---|---|---|---|
| | $\alpha = 0.1$ | $\alpha = 0.05$ | $\alpha = 0.1$ | $\alpha = 0.05$ | $\alpha = 0.1$ | $\alpha = 0.05$ |
| $\text{Acc}_{\text{target}}$ | 60% | 45% | 35% | 35% | 25% | 25% |
| FedAVG | 65.6±22.8 | 47.4±14.9 | 42.4±12.8 | 76.0±8.5 | 22.2±3.1 | 43.8±7.3 |
| FedProx | 38.0±9.1 | 33.0±12.7 | 45.6±5.9 | 86.0±11.8 | 20.8±3.8 | 47.6±5.8 |
| SCAFFOLD | 53.2±14.6 | 45.8±19.7 | 47.4±4.2 | 76.2±9.0 | 22.2±2.2 | 47.8±8.4 |
| FedDisco | 63.4±27.8 | 44.0±11.6 | 44.6±4.5 | 76.2±9.3 | 21.8±3.2 | 51.2±5.6 |
| FedUV | 45.8 ± 11.6 | 43.0 ± 22.8 | 43.0 ± 5.7 | 45.4 ± 5.28 | 20.4 ± 2.2 | 47.0 ± 6.3 |
| FedTGP | 58.6 ± 16.1 | 40.4 ± 4.5 | 63.6 ± 17.5 | 69.6 ± 3.9 | 22.8 ± 2.5 | 44.2 ± 1.2 |
| FedDF | 5.4±1.4 | 6.0±1.5 | 15.2±5.7 | 78.0±23.8 | 9.4±1.9 | 22.0±5.7 |
| FedGKD$^+$ | 5.6±1.6 | 4.2±1.2 | 12.6±3.3 | 39.8±19.6 | 9.0±1.4 | 14.8±2.5 |
| DaFKD | 5.6±1.4 | 3.0±0.6 | 13.4±5.4 | 50.2±27.9 | 9.0±2.8 | 15.6±4.1 |
| **FedGO (ours)** | **3.0**±0.9 | **2.0**±0.6 | **11.0**±2.1 | **25.4**±9.1 | **8.4**±1.0 | **12.6**±1.6 |

pled ImageNet100 (ImageNet100 dataset; Chrabaszcz et al., 2017). Unless specified otherwise, the entire client dataset corresponds to half of the specified client dataset (half for each class), and each client dataset is sampled from the entire client dataset according to Dirichlet($\alpha$), akin to setups in Lin et al. (2020); Cho et al. (2022). $\alpha$ is set to 0.1 and 0.05 to represent data-heterogeneous scenarios. The server dataset corresponds to half of the specified server dataset (half for each class) without labels. If not otherwise specified, the server dataset and the client datasets partition the same dataset disjointly. We considered 20 and 100 clients (20 clients if not specified otherwise), assuming that 40% of the clients participate in each communication round.

**Models and Baselines** For architecture, we employed ResNet-18 (He et al., 2016) with batch normalization layers (Ioffe & Szegedy, 2015). For baselines, we considered the vanilla FedAVG (McMahan et al., 2017), FedProx (Li et al., 2020a), SCAFFOLD (Karimireddy et al., 2020), FedDisco (Ye et al., 2023), FedUV (Son et al., 2024) and FedTGP (Zhang et al., 2024) that do not perform ensemble distillation, FedDF (Lin et al., 2020), FedGKD$^+$ (Yao et al., 2021) and DaFKD (Wang et al., 2023a) that incorporate ensemble distillation. For comparison with other weighting methods, we considered the variance-based weighting method of Cho et al. (2022), the entropy-based methods of Deng et al. (2023) and Park et al. (2024), and the domain-aware method of Wang et al. (2023a), described in Section 2. As an upper bound of the performance, we also compared with central training that trains the server model directly using the entire client dataset. FedGO and DaFKD require image generators and discriminators. For the generator, we considered the three approaches (G1), (G2), and (G3) in Section 3.2. For (G1) and (G3), we adopted the model architecture and training method proposed in WGAN-GP (Gulrajani et al., 2017). For (G2), we employed StyleGAN-XL (Sauer et al., 2022), pretrained on ImageNet (Krizhevsky et al., 2012). Unless specified otherwise, we assume (G1). For discriminators, we utilized a 4-layer CNN. More experimental details are provided in Appendix E.2.

### 4.2. Results

**Test Accuracy and Convergence Speed** Table 3 shows the test accuracy of the server model and Table 4 presents

*Table 5.* Server test accuracy (%) of our FedGO with a generator trained with the unlabeled dataset on the server (Scratch) and with an off-the-shelf generator pretrained on ImageNet (Pretrained) on three image datasets with $\alpha = 0.05$.

| | CIFAR-10 | | CIFAR-100 | | ImageNet100 | |
|---|---|---|---|---|---|---|
| Generator | Scratch | Pretrained | Scratch | Pretrained | Scratch | Pretrained |
| Accuracy | 72.35±9.01 | **74.40**±6.97 | **41.04**±0.99 | **41.04**±0.79 | 31.70±1.55 | **32.72**±0.18 |

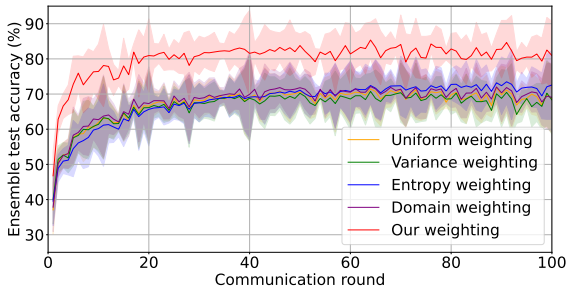
the communication rounds required for the server model to achieve target accuracy ($\text{Acc}_{\text{target}}$) for the first time, for the baselines and FedGO, on CIFAR-10/100 and ImageNet100 datasets. Our FedGO algorithm exhibits the smallest performance gap from the central training and the fastest convergence speed across all the datasets and data heterogeneity settings.

For CIFAR-10 with $\alpha = 0.1$, our FedGO algorithm shows a performance improvement of over 7%p compared to the baselines. However, we observe a diminishing gain for CIFAR-100 and ImageNet100. We argue in Appendix F.2 that this is not due to the marginal improvement in FedGO's ensemble performance, but rather due to larger distillation loss as the server model more struggles to keep up with the performance of the ensemble model.

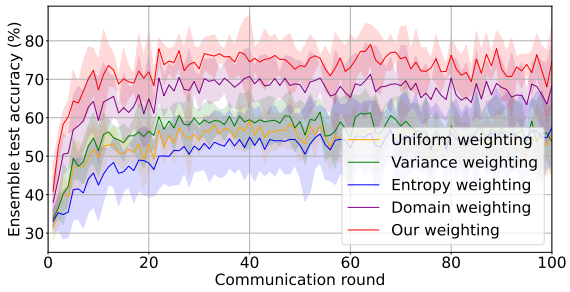
**Comparison of Weighting Methods** Figure 2 shows the ensemble test accuracy along with communication rounds on the CIFAR-10 dataset, according to weighting methods. We evaluated ensemble test accuracy to compare the efficacy of each method in generating pseudo-labels. For the baseline weighting methods, we considered the uniform (Lin et al., 2020), the variance-based (Cho et al., 2022), the entropy-based (Deng et al., 2023; Park et al., 2024), and the domain-aware (Wang et al., 2023a) methods. For fair comparison, all the baselines follow the same steps except the weighting methods. The effectiveness of our weighting method is demonstrated by its ensemble test accuracy outperforming all the other weighting methods over all communication rounds.

**FedGO with a Pretrained Generator** If there exists a pretrained generator capable of generating sufficiently diverse data, the server can distribute the pretrained generator to clients instead of training a generator from scratch using the server's unlabeled dataset, which corresponds to the case (G2) in Section 3.2. This approach has the advantage of saving the server's computing resources required for training a generator.

Table 5 reports the performance of FedGO for various datasets with $\alpha = 0.05$, when using a generator trained with the server's unlabeled dataset versus using a generator pretrained on ImageNet (Krizhevsky et al., 2012). We observe that utilizing the pretrained generator results in superior performance on CIFAR-10 and ImageNet100, whereas it remains the same for CIFAR-100. A key factor contributing




(a) $\alpha = 0.1$




(b) $\alpha = 0.05$

*Figure 2.* Ensemble test accuracy (%) of FedGO and other baseline weighting methods over communication rounds on CIFAR-10 with $\alpha = 0.1$ and $\alpha = 0.05$.

to performance enhancement seems to be the larger model structure of the pretrained generator and its training with a richer dataset. This enhances the generalization performance of client discriminators, enabling optimal weighting even for test data. However, since the assumption of Theorem 3.6 does not hold for $x \in \text{supp}(p) \setminus \text{supp}(p_g)$, the portion of data for which an optimal weighting is guaranteed decreases as the portion of $p$'s support not covered by $p_g$ increases, potentially leading to performance degradation. We note that ImageNet100 is a subset of ImageNet, and ImageNet includes the classes of CIFAR-10 except deer. However, there are several classes of CIFAR-100 not included in ImageNet, which could possibly result in no performance gain.

**More Results** In Appendix F, we provide more experimental results. We report ensemble test accuracy of the baselines and FedGO, demonstrating a larger improvement compared to test accuracy. We also provide results for scenarios involving larger clients or different model structures, cases where the server dataset is different from the client

datasets, and data-free approaches when no server dataset is available, showing significant performance gains over the baselines. Additionally, we report the performance of FedGO with a reduced server dataset and various discriminator training epochs, showing that even with only 20% of the server dataset, FedGO achieves a performance gain of 15%p over FedAVG. Furthermore, FedGO outperforms the baselines even with significantly fewer discriminator training epochs. Finally, we demonstrate the robustness of FedGO under adversarial conditions where a subset of clients behave in a Byzantine manner, showing that our method maintains superior performance even when up to 50% of the client discriminators are compromised.

In Appendix G, a comprehensive analysis of communication costs, privacy, and computational costs for FedGO and baselines is provided.

## 5. Conclusion

We proposed the FedGO algorithm, which effectively addresses the challenge of client data heterogeneity. Our algorithm was proposed based on theoretical analysis of optimal ensemble distillation, and various experimental results demonstrated its high performance and fast convergence rate under various scenarios with and without extra server dataset. The limitation of our work is provided in Appendix H.

## Acknowledgements

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) through the Next Generation Semantic Communication Network Research Center Grant (RS-2024-00398948) funded by the Korean Government (MSIT), and in part by the IITP through 6G · Cloud Research and Education Open Hub Grant (RS-2024-00428780) funded by the Korea government (MSIT).

## Impact Statement

In this work, we proposed a federated learning algorithm that demonstrates strong performance in scenarios where client data is heterogeneous. This capability makes our approach highly effective for distributed learning in many practical situations, where data across different clients can vary significantly. By efficiently handling such data diversity, our algorithm holds the potential to enhance the applicability and robustness of federated learning systems in real-world applications.

## References

Adlam, B., Weill, C., and Kapoor, A. Investigating under and overfitting in wasserstein generative adversarial networks. *arXiv preprint arXiv:1910.14137*, 2019.

Behera, M. R., Upadhyay, S., Shetty, S., Priyadarshini, S., Patel, P., and Lee, K. F. Fedsyn: Synthetic data generation using federated learning. *arXiv preprint arXiv:2203.05931*, 2022.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. https://www.alexkulesza.com/pubs/adapt_mlj10.pdf.

Ben-David, S., Blitzer, J., Crammer, K., Pereira, and Fernando. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Chang, H., Shejwalkar, V., Shokri, R., and Houmansadr, A. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

Cho, Y. J., Manoel, A., Joshi, G., Sim, R., and Dimitriadis, D. Heterogeneous ensemble knowledge transfer for training large models in federated learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI) Main Track*, 2022.

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8789–8797, 2018.

Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of ImageNet as an alternative to the CIFAR datasets. *CoRR*, 2017.

Crammer, K., Kearns, M., and Wortman, J. Learning from multiple sources. *Journal of Machine Learning Research*, 9(8), 2008.

Dai, Y., Chen, Z., Li, J., Heinecke, S., Sun, L., and Xu, R. Tackling data heterogeneity in federated learning with class prototypes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7314–7322, 2023.

Deng, Y., Ren, J., Tang, C., Lyu, F., Liu, Y., and Zhang, Y. A hierarchical knowledge transfer framework for heterogeneous federated learning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pp. 1–10. IEEE, 2023.

Dong, X., Zhang, S. Q., Li, A., and Kung, H. Spherefed: Hyperspherical federated learning. In *European Conference on Computer Vision*, pp. 165–184. Springer, 2022.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pp. 265–284. Springer, 2006.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Fan, C. and Liu, P. Federated generative adversarial learning. In *Pattern Recognition and Computer Vision: Third Chinese Conference, PRCV 2020, Nanjing, China, October 16–18, 2020, Proceedings, Part III 3*, pp. 3–15. Springer, 2020.

Gong, X., Sharma, A., Karanam, S., Wu, Z., Chen, T., Doermann, D., and Innanje, A. Ensemble attention distillation for privacy-preserving federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15076–15086, 2021.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Guerraoui, R., Guirguis, A., Kermarrec, A.-M., and Merrer, E. L. Fegan: Scaling distributed gans. In *Proceedings of the 21st International Middleware Conference*, pp. 193–206, 2020.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Hardy, C., Le Merrer, E., and Sericola, B. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. In *2019 IEEE international parallel and distributed processing symposium (IPDPS)*, pp. 866–877. IEEE, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hinton, G., sh Srivastava, N., and Swersky, K. Neural networks for machine learning. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Huang, W., Shi, Y., Cai, Z., and Suzuki, T. Understanding convergence and generalization in federated learning through feature learning theory. In *The Twelfth International Conference on Learning Representations*, 2023.

ImageNet100 dataset. https://www.kaggle.com/datasets/ambityga/imagenet100.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pp. 5132–5143. PMLR, 2020.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Kifer, D., Ben-David, S., and Gehrke, J. Detecting change in data streams. In *VLDB*, volume 4, pp. 180–191. Toronto, Canada, 2004.

Kim, S., Lee, G., Oh, J., and Yun, S.-Y. FedFN: Feature normalization for alleviating data heterogeneity problem in federated learning. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Li, H., Cai, Z., Wang, J., Tang, J., Ding, W., Lin, C.-T., and Shi, Y. Fedtp: Federated learning by transformer personalization. *IEEE transactions on neural networks and learning systems*, 2023.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020a.

Li, W., Chen, J., Wang, Z., Shen, Z., Ma, C., and Cui, X. Ifl-gan: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedAvg on Non-IID data. In *International Conference on Learning Representations*, 2020b.

Li, X.-C. and Zhan, D.-C. Fedrs: Federated learning with restricted softmax for label distribution non-iid data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 995–1005, 2021.

Liang, X., Shen, S., Liu, J., Pan, Z., Chen, E., and Cheng, Y. Variance reduced local sgd with lower communication complexity. *arXiv preprint arXiv:1912.12844*, 2019.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33: 2351–2363, 2020.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2022.

Marfoq, O., Neglia, G., Vidal, R., and Kameni, L. Personalized federated learning through local memorization. In *International Conference on Machine Learning*, pp. 15070–15092. PMLR, 2022.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Mendieta, M., Yang, T., Wang, P., Lee, M., Ding, Z., and Chen, C. Local learning matters: Rethinking data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8397–8406, 2022.

Park, J.-M., Jang, W.-J., Oh, T.-H., and Lee, S.-H. Overcoming client data deficiency in federated learning by exploiting unlabeled data on the server. *IEEE Access*, 2024.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06434.

Rasouli, M., Sun, T., and Rajagopal, R. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228*, 2020.

Sattler, F., Marban, A., Rischke, R., and Samek, W. Communication-efficient federated distillation. *arXiv preprint arXiv:2012.00632*, 2020.

Sauer, A., Schwarz, K., and Geiger, A. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Shalev-Shwartz, S. and Ben-David, S. Understanding machine learning: From theory to algorithms. https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

Son, H. M., Kim, M.-H., Chung, T.-M., Huang, C., and Liu, X. Feduv: uniformity and variance for heterogeneous federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5863–5872, 2024.

Tang, Z., Zhang, Y., Shi, S., He, X., Han, B., and Chu, X. Virtual homogeneity learning: Defending against data heterogeneity in federated learning. In *International Conference on Machine Learning*, pp. 21111–21132. PMLR, 2022.

Tang, Z., Zhang, Y., Shi, S., Tian, X., Liu, T., Han, B., and Chu, X. FedImpro: Measuring and improving client update in federated learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020a.

Wang, H., Li, Y., Xu, W., Li, R., Zhan, Y., and Zeng, Z. Dafkd: Domain-aware federated knowledge distillation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 20412–20421, 2023a.

Wang, H., Xu, H., Li, Y., Xu, Y., Li, R., and Zhang, T. FedCDA: Federated learning with cross-rounds divergence-aware aggregation. In *The Twelfth International Conference on Learning Representations*, 2023b.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020b.

Wang, J., Xie, G., Huang, Y., Lyu, J., Zheng, F., Zheng, Y., and Jin, Y. FedMed-GAN: Federated domain translation on unsupervised cross-modality brain image synthesis. *Neurocomputing*, 546:126282, 2023c.

Wang, Y., Zhang, J., and Wang, Y. Do generated data always help contrastive learning? In *The Twelfth International Conference on Learning Representations*, 2024.

Xing, H., Xiao, Z., Qu, R., Zhu, Z., and Zhao, B. An efficient federated distillation learning system for multitask time series classification. *IEEE Transactions on Instrumentation and Measurement*, 71:1–12, 2022.

Xiong, Z., Li, W., and Cai, Z. Federated generative model on multi-source heterogeneous data in iot. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10537–10545, 2023.

Yang, C., Shen, Y., Xu, Y., Zhao, D., Dai, B., and Zhou, B. Improving gans with a dynamic discriminator. *Advances in Neural Information Processing Systems*, 35:15093–15104, 2022.

Yang, Z., Zhang, Y., Zheng, Y., Tian, X., Peng, H., Liu, T., and Han, B. FedFed: Feature distillation against data heterogeneity in federated learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Yao, D., Pan, W., Dai, Y., Wan, Y., Ding, X., Jin, H., Xu, Z., and Sun, L. Local-global knowledge distillation in heterogeneous federated learning with non-iid data. *arXiv preprint arXiv:2107.00051*, 2021.

Ye, R., Xu, M., Wang, J., Xu, C., Chen, S., and Wang, Y. FedDisco: Federated learning with discrepancy-aware collaboration. In *International Conference on Machine Learning*, pp. 39879–39902. PMLR, 2023.

Yoon, Y., Hu, D., Weissburg, I., Qin, Y., and Jeong, H. Redifine: Reusable diffusion finetuning for mitigating degradation in the chain of diffusion. *arXiv preprint arXiv:2407.17493*, 2024.

Zhang, J., Guo, S., Guo, J., Zeng, D., Zhou, J., and Zomaya, A. Y. Towards data-independent knowledge transfer in model-heterogeneous federated learning. *IEEE Transactions on Computers*, 72(10):2888–2901, 2023a.

Zhang, J., Zhao, L., Yu, K., Min, G., Al-Dubai, A. Y., and Zomaya, A. Y. A novel federated learning scheme for generative adversarial networks. *IEEE Transactions on Mobile Computing*, 2023b.

Zhang, J., Liu, Y., Hua, Y., and Cao, J. Fedtgp: Trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 16768–16776, 2024.

Zhang, L., Wu, D., and Yuan, X. Fedzkt: Zero-shot knowledge transfer towards resource-constrained federated learning with heterogeneous on-device models. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pp. 928–938. IEEE, 2022.

Zhang, X., Zhu, X., Wang, J., Bao, W., and Yang, L. T. Dance: Distributed generative adversarial networks with communication compression. *ACM Transactions on Internet Technology (TOIT)*, 22(2):1–32, 2021.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

## A. Formal Statement and Proof of Theorem 3.2

In addition to the setups and definitions introduced in Section 2, we assume binary classification task, i.e., $y(x) \in [0,1]$ and $h(x) \in \{0,1\}$, coupled with $\ell_1$ loss.

We first present some definitions and a lemma.

**Definition A.1.** (Kifer et al., 2004, Definition 1) For two distributions $q$ and $q'$ over a domain $\mathcal{X}$, let $\mathcal{H}$ denote a hypothesis class on $\mathcal{X}$ and $I(h)$ for $h \in \mathcal{H}$ denote the set $\{x \in \mathcal{X} : h(x) = 1\}$. The $\mathcal{H}$-divergence between $q$ and $q'$ is

$$d_{\mathcal{H}}(q, q') = 2 \sup_{h \in \mathcal{H}} |Pr_{x \sim q}[I(h)] - Pr_{x \sim q'}[I(h)]|. \tag{10}$$

**Definition A.2.** For a hypothesis space $\mathcal{H}$, the symmetric difference hypothesis space $\mathcal{H}\triangle\mathcal{H}$ is the set of hypotheses

$$g \in \mathcal{H}\triangle\mathcal{H} \Leftrightarrow g(x) = h(x) \oplus h'(x) \text{ for some } h, h' \in \mathcal{H} \tag{11}$$

where $\oplus$ is the XOR function.

**Lemma A.3.** *For hypotheses $h, h' \in \mathcal{H}$ and distributions $q, q'$ on $\mathcal{X}$, we have*

$$|\mathcal{L}_q(h, h') - \mathcal{L}_{q'}(h, h')| \leq \frac{1}{2} d_{\mathcal{H}\triangle\mathcal{H}}(q, q'). \tag{12}$$

*Proof.* By the definition of $\mathcal{H}\triangle\mathcal{H}$-distance, we have

$$d_{\mathcal{H}\triangle\mathcal{H}}(q, q') = 2 \sup_{h \in \mathcal{H}} |Pr_{x \sim q}[h(x) \neq h'(x)] - Pr_{x \sim q'}[h(x) \neq h'(x)]| \tag{13}$$

$$= 2 \sup_{h \in \mathcal{H}} |\mathcal{L}_q(h, h') - \mathcal{L}_{q'}(h, h')| \tag{14}$$

$$\geq 2|\mathcal{L}_q(h, h') - \mathcal{L}_{q'}(h, h')|, \tag{15}$$

which completes the proof. □

Now we are ready to present the formal statement and proof of Theorem 3.2.

**Theorem A.1.** *For binary classification task with $\ell_1$ loss, consider hypothesis class $\mathcal{H}$ such that $h \in \mathcal{H}$ outputs 0 or 1 and its spanned hypothesis class $\bar{\mathcal{H}} \triangleq \{\sum_{k=1}^K w_k \cdot h_k | h_k \in \mathcal{H}, w_k : \mathcal{X} \to [0,1]$ for all $k = 1, ..., K, \sum_{k=1}^K w_k = 1\}$. For any $h \in \mathcal{H}$ and $(\sum_{k=1}^K w_k \cdot h_k) \in \bar{\mathcal{H}}$, the following holds:*

$$\mathcal{L}_p(h) \leq \mathcal{L}_p(\sum_{k=1}^K w_k \cdot h_k) + \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k) + \frac{1}{2} d_{\bar{\mathcal{H}}\triangle\bar{\mathcal{H}}}(p, p_s). \tag{16}$$

*Proof.* We have

$$\mathcal{L}_p(h) = \mathbf{E}_p[l(h(x), y(x)] \tag{17}$$

$$\leq \mathbf{E}_p[l(h(x), (\sum_{k=1}^K w_k \cdot h_k)(x)] + \mathbf{E}_p[l((\sum_{k=1}^K w_k \cdot h_k)(x), y(x)] \tag{18}$$

$$= \mathcal{L}_p(\sum_{k=1}^K w_k \cdot h_k) + \mathcal{L}_p(h, \sum_{k=1}^K w_k \cdot h_k) \tag{19}$$

by triangle inequality (Ben-David et al., 2006; Crammer et al., 2008).

Since $A \leq B + |A - B|$, letting $A = \mathcal{L}_p(h, \sum_{k=1}^K w_k \cdot h_k)$, $B = \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k)$, the RHS of (19) is upper-bounded by

$$\mathcal{L}_p(\sum_{k=1}^K w_k \cdot h_k) + \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k) + |\mathcal{L}_p(h, \sum_{k=1}^K w_k \cdot h_k) - \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k)| \tag{20}$$

$$= \mathcal{L}_p(\sum_{k=1}^K w_k \cdot h_k) + \mathcal{L}_{p_s}(h, \sum_{k=1}^K w_k \cdot h_k) + \frac{1}{2} d_{\bar{\mathcal{H}}\triangle\bar{\mathcal{H}}}(p, p_s) \tag{21}$$

by the definition of $d_{\bar{\mathcal{H}}\triangle\bar{\mathcal{H}}}$ and Lemma A.3. Thus, we prove Theorem A.1. □

## B. Proof of Theorem 3.4

Before we start the proof of Theorem 3.4, we present the following lemma with the setups and definitions introduced in Section 2.

**Lemma B.1.** *Let the loss function $l$ be convex and $\{h_k\}_{k=1}^K \subset \mathcal{H}$. For the weight functions $\{w_k^*\}_{k=1}^K$ defined in Theorem 3.4, the following holds:*

$$\mathcal{L}_p(\textstyle\sum_k w_k^* \cdot h_k) \leq \sum_k \pi_k \cdot \mathcal{L}_{p_k}(h_k). \tag{22}$$

*Proof.* Note that

$$\begin{aligned}
\mathcal{L}_p(\textstyle\sum_k w_k^* \cdot h_k) &= \mathbb{E}_{x\sim p}\left[l\left(\sum_k w_k^*(x)\cdot h_k(x), y(x)\right)\right] && (23)\\
&= \int l\left(\textstyle\sum_k w_k^*(x)\cdot h_k(x), y(x)\right)\cdot p(x)dx && (24)\\
&= \int l\left(\textstyle\sum_k w_k^*(x)\cdot h_k(x), y(x)\right)\cdot \sum_j \pi_j \cdot p_j(x)dx && (25)\\
&= \int l\left(\textstyle\sum_k w_k^*(x)\cdot h_k(x), \sum_k w_k^*(x)\cdot y(x)\right)\cdot \sum_j \pi_j \cdot p_j(x)dx && (26)\\
&\leq \int \left(\textstyle\sum_k w_k^*(x)\cdot l\left(h_k(x), y(x)\right)\right)\cdot \sum_j \pi_j \cdot p_j(x)dx && (27)\\
&= \int \textstyle\sum_k \frac{\pi_k(x)\cdot p_k(x)}{\sum_{i=1}^K \pi_i\cdot p_i(x)}\cdot l\left(h_k(x), y(x)\right)\cdot \sum_j \pi_j \cdot p_j(x)dx && (28)\\
&= \sum_k \int \pi_k \cdot p_k(x)\cdot l\left(h_k(x), y(x)\right)dx && (29)\\
&= \sum_k \pi_k \cdot \int l\left(h_k(x), y(x)\right)\cdot p_k(x)dx && (30)\\
&= \sum_k \pi_k \cdot \mathcal{L}_{p_k}(h_k), && (31)
\end{aligned}$$

where (27) holds due to the convexity of loss function $l(\cdot,\cdot)$. This completes the proof. □

Now we present the proof of Theorem 3.4.

*Proof.* For $h \in \mathcal{H}$, we have

$$\begin{aligned}
\mathcal{L}_p(h) &= \mathbb{E}_{x\sim p}\left[l\left(h(x), y(x)\right)\right] && (32)\\
&= \int l\left(h(x), y(x)\right)\cdot p(x)dx && (33)\\
&= \int l\left(h(x), y(x)\right)\cdot \textstyle\sum_k \pi_k \cdot p_k(x)dx && (34)\\
&= \textstyle\sum_k \pi_k \cdot \int \left[l\left(h(x), y(x)\right)\right]\cdot p_k(x)dx && (35)\\
&= \textstyle\sum_k \pi_k \cdot \mathcal{L}_{p_k}(h) && (36)\\
&\geq \textstyle\sum_k \pi_k \cdot \mathcal{L}_{p_k}(h_{p_k}^*). && (37)
\end{aligned}$$

Hence, it suffices to show that

$$\mathcal{L}_p(\textstyle\sum_k w_k^* \cdot h_{p_k}^*) \leq \sum_k \pi_k \cdot \mathcal{L}_{p_k}(h_{p_k}^*), \tag{38}$$

and this is the direct result of Lemma B.1 with $\{h_k\}_{k=1}^K = \{h_{p_k}^*\}_{k=1}^K$. □

## C. Generalization Bound with Empirical Loss Minimizer

In this section, we present the generalization loss bound of the ensemble of empirical loss minimizers of clients with our weighting method.

**Theorem C.1.** *For binary classification task with $\ell_1$ loss, the following holds for our weighting function $\{w_k^*\}_{k=1}^K$ defined in Theorem 3.4:*

$$\mathcal{L}_p(\sum_{k=1}^K w_k^* \cdot h_{\hat{p}_k}^*) \leq \sum_{k=1}^K \pi_k \cdot \left[\mathcal{L}_{\hat{p}_k}(h_{\hat{p}_k}^*) + \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}\right] \tag{39}$$

$$\leq \mathcal{L}_{\hat{p}}(h_{\hat{p}}^*) + \sum_{k=1}^K \pi_k \cdot \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\cdot\sqrt{2n_k}}, \tag{40}$$

*where $\hat{p}_k$ is the empirical distribution by sampling $n_k$ data points i.i.d. according to $p_k$, $\hat{p} = \sum_{k=1}^K \pi_k \cdot \hat{p}_k$, and $\tau_{\mathcal{H}}$ is growth function bounded by polynomial of the VC-dimension of $\mathcal{H}$.*

Compared to (1.2), we can see that the ensemble of empirical loss minimizers with our weighting method has a tighter generalization bound without the factor of $(K+1)$.

Before we prove Theorem C.1, we present the following theorem.

**Theorem C.2.** *(Shalev-Shwartz & Ben-David, Theorem 6.11) Let $\mathcal{H}$ be a hypothesiss class and let $\tau_{\mathcal{H}}$ be its growth function. Then, for every distribution $q$ on $\mathcal{X}$ and every $\delta \in (0,1)$, with probability of at least $1-\delta$ over the $m$ i.i.d. choice of $S \sim q^m$ with its empirical distribution $\hat{q}$, we have*

$$|\mathcal{L}_q(h) - \mathcal{L}_{\hat{q}}(h)| \leq \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2m))}}{\delta\sqrt{2m}}. \tag{41}$$

We also present the bound of growth function $\tau_{\mathcal{H}}$.

**Lemma C.1.** *(Shalev-Shwartz & Ben-David, Lemma 6.10) Let $\mathcal{H}$ be a hypothesis class with VC-dimension of $\mathcal{H}$ is smaller than $d$, i.e. $VCDim(\mathcal{H}) \leq d < \infty$. Then, for all $m$, $\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^d \binom{m}{i}$. In particular, if $m > d+1$, then $\tau_{\mathcal{H}}(m) \leq (em/d)^d$, where $e$ is Euler's number.*

Now we present the proof of Theorem C.1.

*Proof.* By the result of Lemma B.1 with $\{h_k\}_{k=1}^K = \{h_{\hat{p}_k}^*\}_{k=1}^K$, we can derive

$$\mathcal{L}_p(\textstyle\sum_k w_k^* \cdot h_{\hat{p}_k}^*) \leq \sum_k \pi_k \cdot \mathcal{L}_{p_k}(h_{\hat{p}_k}^*). \tag{42}$$

Also we note that $S_k \sim p_k^{n_k}$. We can derive the following inequality for $k = 1, ..., K$ using Theorem C.2. With probability at least of $1-(\delta/K)$,

$$\mathcal{L}_{p_k}(h_{\hat{p}_k}^*) \leq \mathcal{L}_{\hat{p}_k}(h_{\hat{p}_k}^*) + \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}. \tag{43}$$

where $\tau_{\mathcal{H}}$ is growth function bounded by polynomial of the VC-dimension of $\mathcal{H}$.

By the union bound, we have

$$P\left[\bigcap_{k=1}^{K}\left(\mathcal{L}_{p_k}(h^*_{\hat{p}_k}) \le \mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}\right)\right] \tag{44}$$

$$= 1 - P\left[\bigcup_{k=1}^{K}\left(\mathcal{L}_{p_k}(h^*_{\hat{p}_k}) \ge \mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}\right)\right] \tag{45}$$

$$\ge 1 - \sum_{k=1}^{K} P\left[\left(\mathcal{L}_{p_k}(h^*_{\hat{p}_k}) \ge \mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}\right)\right] \tag{46}$$

$$\ge 1 - \sum_{k=1}^{K}(\delta/K) \tag{47}$$

$$\ge 1 - \delta. \tag{48}$$

Hence, with probability at least $1-\delta$, following inequality holds for all $k = 1, .., K$:

$$\mathcal{L}_{p_k}(h^*_{\hat{p}_k}) \le \mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}. \tag{49}$$

Furthermore, by definition of $\hat{p}$,

$$\mathcal{L}_{\hat{p}}(h^*_{\hat{p}}) = \textstyle\sum_k \pi_k \cdot \mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}}) \tag{50}$$

$$\ge \textstyle\sum_k \pi_k \cdot \mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}). \tag{51}$$

By combining the above results, with probability of at least $1-\delta$, we have

$$\mathcal{L}_p(\textstyle\sum_{k=1}^{K} w^*_k \cdot h^*_{\hat{p}_k}) \le \textstyle\sum_k \pi_k \cdot \mathcal{L}_{p_k}(h^*_{\hat{p}_k}) \tag{52}$$

$$\le \sum_{k=1}^{K} \pi_k \cdot \left[\mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}\right] \tag{53}$$

$$\le \sum_{k=1}^{K} \left[\pi_k \cdot \mathcal{L}_{\hat{p}_k}(h^*_{\hat{p}_k}) + \pi_k \cdot \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}\right] \tag{54}$$

$$\le \mathcal{L}_{\hat{p}}(h^*_{\hat{p}}) + \sum_{k=1}^{K} \pi_k \cdot \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2n_k))}}{(\delta/K)\sqrt{2n_k}}. \tag{55}$$

This completes the proof. □

## D. Description of FedGO

Figure 3 illustrates the operation of FedGO. Algorithm 2 presents a pseudo-code of FedGO. For training discriminators, each client optimizes the GAN loss with respect to its labeled dataset. A pseudo-code of client discriminator update is provided in Algorithm 3.

## E. Experimental Details

All experiments were conducted in Python 3.8.12 environment using a 64-core Intel 2.90GHz Xeon Gold 6226R CPU with 512GB memory, and an RTX 3090 GPU. We also implemented the algorithms using PyTorch with version 1.11.0.

### E.1. Detailed Experimental Setting and Analysis of Toy Example (Figure 1)

For the toy example in Figure 1, the dataset is generated from a mixture of four Gaussian distributions, each with a variance of 3. The top row of Figure 4 shows the global data distribution and the datasets held by four clients. Each point represents

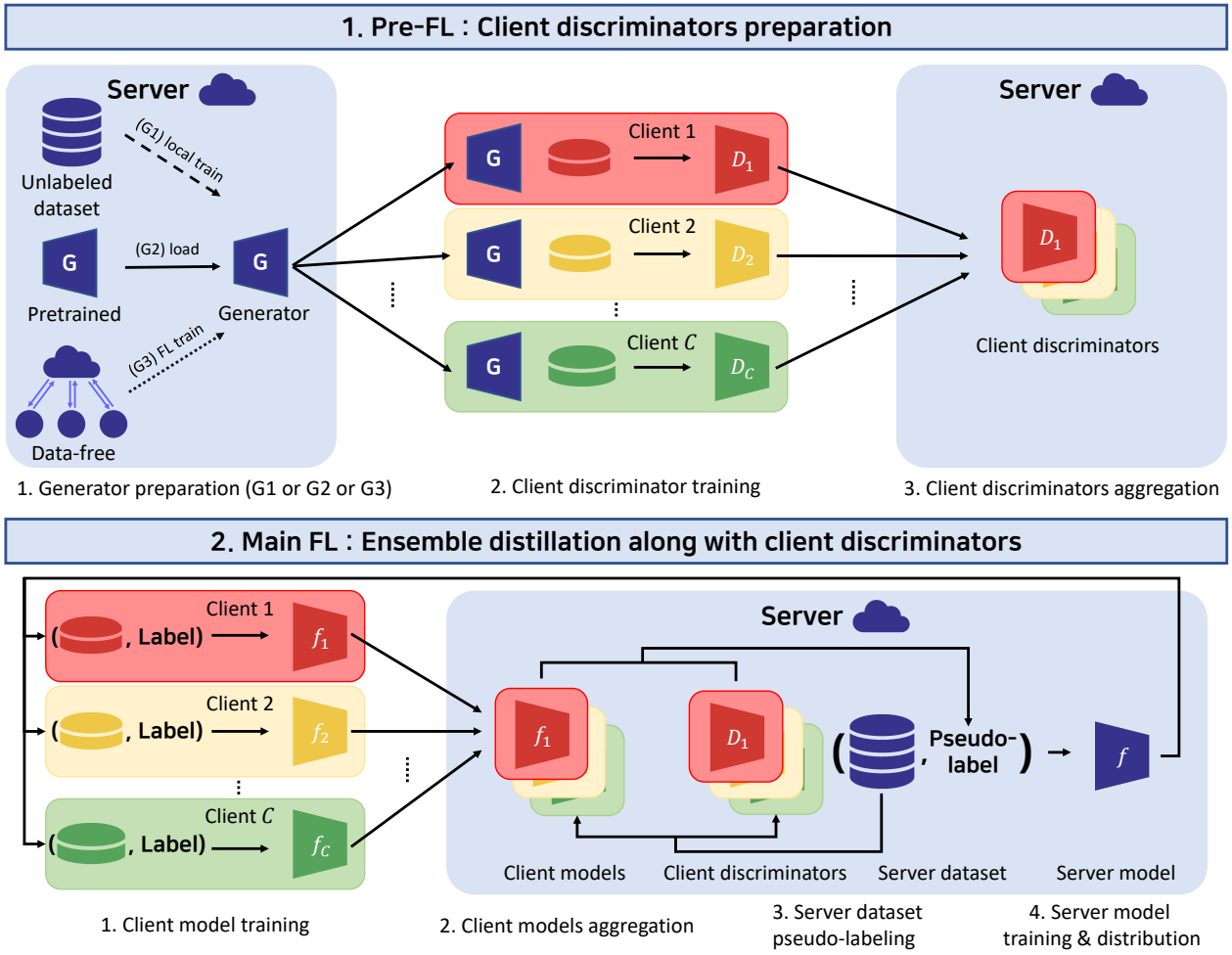



*Figure 3.* Illustration of our FedGO algorithm.

data, with its color indicating the class label: data from Gaussians with means at (4, 4) and (-4, -4) are labeled as Red, data from the Gaussian with mean at (-4, 4) as Blue, and data from the Gaussian with mean at (4, -4) as Green. Each Gaussian provides 300 data samples. Each client holds 90% of data from the Gaussian whose mean is in a certain quadrant (the 3rd, 4th, 2nd, 1st quadrants for Clients 1, 2, 3, and 4, respectively), and the remaining 10% from Gaussians with means in the other quadrants. The clients' global dataset comprises 1200 samples, with 300 from each Gaussian. The server unlabeled dataset comprises 300 data, uniformly distributed on the square $[-12, 12] \times [-12, 12]$.

Each client trains a 3-layer MLP classifier for 2 epochs using its dataset, and a 3-layer discriminator for 1 epoch using its dataset as real dataset and server dataset as fake dataset. We used Adam (Kingma & Ba, 2015) with learning rate 0.001 and $(\beta_1, \beta_2) = (0.9, 0.999)$ for classifier optimizer, and RMSprop (Hinton et al.) with learning rate 0.00005 for discriminator optimizer. Also we used a batch size of 64 for both.

The bottom row of Figure 4 (same as Figure 1) illustrates the decision boundaries of server models. The leftmost plot is from the model with averaged client model parameters, while the remaining plots are from the server models trained via ensemble distillation for 2 epochs using pseudo-labeled dataset: the global dataset is pseudo-labeled using uniform weighting (Lin et al., 2020), variance weighting (Cho et al., 2022), entropy weighting (Deng et al., 2023; Park et al., 2024), domain-aware weighting (Wang et al., 2023a) , and our weighting method. The background color indicates the decision boundary in RGB channels. Given the Gaussian distributions, the optimal decision rule is red in the 1st and 3rd quadrants, blue in the 2nd quadrant, and green in the 4th quadrant. Thus, the oracle decision boundary aligns with the x-axis and y-axis, depicted by black lines.

**Algorithm 2** FedGO algorithm with $K$ clients for $T$ communication rounds. $f(\cdot;\theta)$ stands for the model with parameter $\theta$, $\mu$ stands for the step size, and $\Phi_k(x)$ stands for the odds value of $D_k(x)$.

**Require:** Client labeled dataset $\{S_k\}_{k=1}^K$
  Initialize server model $f(\cdot;\theta_s^0)$ with parameter $\theta_s^0$
  Prepare generate $G$ and unlabeled dataset $U$ ▷ *By one of the methods in Table 2*
  **for parallel** client $k \in \{1, 2, ..., K\}$ **do**
    $D_k \leftarrow DiscriminatorUpdate(G, S_k)$ ▷ *Detailed in Algorithm 3*
  **end for**
  **for** communication round $t = 1$ to $T$ **do**
    $A^t \leftarrow$ sample $\lfloor C \cdot K \rfloor$ clients
    **for parallel** client $k \in A^t$ **do**
      $\theta_k^t \leftarrow ClientUpdate(\theta_s^{t-1}, S_k)$ ▷ *Gradient update $\theta_s^{t-1}$ with $S_k$*
    **end for**
    $\theta_s^t \leftarrow \sum_{k\in A^t} \frac{n_k}{\sum_{i\in A^t} n_i} \cdot \theta_k^t$
    **for** server train epoch $e = 1$ to $E_s$ **do**
      **for** unlabeled minibatch $u \in U$ **do**
        $\tilde{y}(u) \leftarrow \sigma(\sum_{k\in A^t} w_k^*(u) \cdot f(u;\theta_k^t))$ ▷ $w_k^*(u) = \frac{n_k \cdot \Phi_k(u)}{\sum_{i\in A^t} n_i \cdot \Phi_i(u)}$
        $\theta_s^t \leftarrow \theta_s^t - \mu \cdot \nabla_{\theta_s^t} \mathrm{KL}(\tilde{y}(u), \sigma(f(u;\theta_s^t)))$
      **end for**
    **end for**
  **end for**
  **return** $f(\cdot;\theta_s^T)$

**Algorithm 3** Discriminator update for $E_d$ epochs. $\mu_d$ stands for the step size and $D(\cdot;\theta)$ is the parameterized discriminator with parameter $\theta$.

**Require:** Generator $G$, labeled dataset $S$
  Initialize discriminator model $D(\cdot;\theta_d^0)$ with parameter $\theta_d^0$
  **for** epoch $e = 1$ to $E_d$ **do**
    $\theta_d^e \leftarrow \theta_d^{e-1}$
    **for** minibatch $m \in S$ **do**
      $(x_{real}, y) \leftarrow$ (images, labels) pair of minimatch $m$
      $x_{fake} \leftarrow$ generated images by generator $G$
      $Loss_{GAN}(D(\cdot;\theta_d^e)) \leftarrow \log(D(x_{real};\theta_d^e) + \log(1 - D(x_{fake};\theta_d^e))$
      $\theta_d^e \leftarrow \theta_d^e - \mu_d \nabla_{\theta_d^e} Loss_{GAN}(D(\cdot;\theta_d^e))$ ▷ *Update with gradient for vanilla GAN loss*
    **end for**
  **end for**
  **return** $D(\cdot;\theta_d^{E_d}))$

The averaged parameter model exhibits a blurred decision boundary compared to models trained via ensemble distillation. Furthermore, among the models with ensemble distillation, the decision boundary of the model trained via our weighting method is closest to the oracle decision boundary.

### E.2. Detailed Experimental Settings for Image Classification Tasks

**Hyperparameter Tuning** We identified the best-performing hyperparameters on CIFAR-100 with Dirichlet $\alpha = 0.05$ and used the same values for other settings. During the ensemble distillation process, we trained both clients and server with the Adam optimizer (Kingma & Ba, 2015) at a learning rate of 0.001 with batch size 64, without weight decay. The $(\beta_1, \beta_2)$ parameters for Adam were set to (0.9, 0.999). Additionally, we applied cosine annealing (Loshchilov & Hutter, 2022) to decay the server learning rate until the final communication round $T = 100$ as in Lin et al. (2020), except for the results of F.3 and F.5.

For the client and server classifier training epochs, we performed a grid search to find the optimal number of training

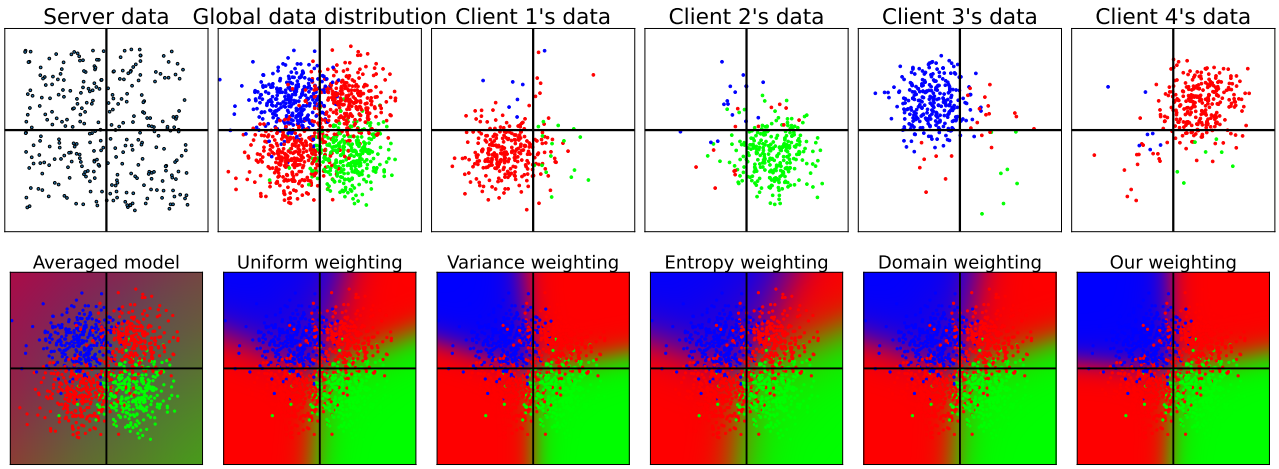



*Figure 4.* Top row represents the server and clients' datasets. Bottom row, showing the decision boundaries of the aggregated models, is the same as Figure 1 and copied here for ease of analysis.

epochs. The initial grid was {5, 10, 30, 50}, and the experiments were conducted with 30 client epochs and 10 server epochs ($E_s = 10$) for CIFAR-10/100. To leverage the increased number of steps due to the additional number of data, experiments on ImageNet100 were conducted with 10 client classifier epochs and 3 server classifier epochs ($E_s = 3$).

To train the generator utilized by our FedGO from scratch, we trained the WGAN-GP model following the training method proposed in Gulrajani et al. (2017). The generator and discriminator of WGAN-GP were trained using the Adam optimizer with a learning rate of 0.0002 and $(\beta_1, \beta_2) = (0, 0.9)$. The training was conducted with a batch size of 64 until the generator completed 100,000 gradient steps. The generator was updated every 5 steps of the discriminator, and a gradient penalty coefficient $\lambda$ of 10 was used.

When training a generator in a data-free setting, i.e., the case (G3), we utilized the FedGAN (Rasouli et al., 2020) algorithm. The generator was trained for only $T' = 5$ communication rounds, with each local generator and discriminator trained for 3 epochs per round.

For the client discriminator, we adopted the hyperparameters from https://github.com/Ksuryateja/DCGAN-MNIST-pytorch/blob/master/gan_mnist.py and trained it with a batch size of 64 for 30 epochs for CIFAR-10/100, and for 10 epochs for ImageNet100. The optimizer Adam was used with a learning rate of 0.0002, and $(\beta_1, \beta_2) = (0.5, 0.999)$.

FedProx (Li et al., 2020a) introduces a proximal term to the client training loss, which helps to address heterogeneity by penalizing large deviations from the server model. The proximal term is multiplied by a coefficient $\mu$ and added to the primary objective loss. We performed a grid search to tune the value of $\mu$ from {0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001}, and chose the best value $\mu = 0.001$.

FedDisco (Ye et al., 2023) determines client parameter aggregation coefficient by leveraging both the dataset size and the discrepancy between local and global category distributions. Here, $a$ controls the influence of the discrepancy, while $b$ adjusts the baseline aggregation coefficient. Following the FedDisco paper, we adopted the best-performing parameters $a = 0.5$ and $b = 0.1$, which were tuned for optimal performance for CIFAR-10.

FedUV (Son et al., 2024) introduces two regularization terms—classifier variance and encoder representation uniformity—to emulate IID conditions in federated learning, mitigating local bias in non-IID settings. The loss function combines the standard classification loss with the uniformity term (weighted by $\lambda$) and the variance term (weighted by $\alpha$). By following Son et al. (2024), we set the hyperparameters $\lambda = 0.5$ and $\alpha = 2.5$ as they achieved the best trade-off between generalization and personalization across clients.

FedTGP (Zhang et al., 2024) proposes Trainable Global Prototypes (TGP) with Adaptive-Margin-Enhanced Contrastive Learning to address data heterogeneity by improving class separability and semantic alignment among clients. The model trains prototypes using contrastive learning with a temperature-scaled margin. By following Zhang et al. (2024), for the TGP model, we used a batch size of 10, trained for 1,000 epochs with a learning rate of 0.005 and a margin parameter $\tau$ =

100. The contrastive loss was weighted by $\lambda = 0.001$ to balance its influence with the main task objective.

FedHKT (Deng et al., 2023) and FedDS (Park et al., 2024) introduce a temperature parameter $\tau > 0$, which allows client weights to approach uniform weighting as $\tau$ increases. By following Deng et al. (2023), we set $\tau = 1$.

FedGKD (Yao et al., 2021) introduces an additional buffer of length $M$ on the server, where the server model is stored after each round. The server then creates an additional model with averaged parameters from the models stored in the buffer and sends this model to the clients each round. Each client uses a temperature parameter $\tau$ to compute the knowledge distillation loss on the received additional model, multiplies this loss by $\gamma/2$, and adds it to the primary objective loss. Consequently, it is necessary to tune three additional hyperparameters: $M$, $\tau$, and $\gamma$. We conducted a grid search with $M$ and $\tau$ in $\{1, 3, 5, 10\}$ and $\gamma$ in $\{0.1, 0.05, 0.01, 0.005, 0.001\}$. The best performing parameters were $M = 5$, $\tau = 3$, and $\gamma = 0.001$.

Similar to our FedGO, DaFKD (Wang et al., 2023a) utilizes discriminators to implement client weighting function. However, unlike FedGO, DaFKD trains the generator and discriminators collaboratively. To focus on the weighting method, the domain-aware weighting method in Figure 2 is implemented by only modifying the weighting step in our FedGO algorithm.

**Model Implementation** We used ResNet-18 (He et al., 2016) as the classification model, following the implementation from https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py. Additionally, our FedGO requires extra generator and discriminator models. When training the generator from scratch, we utilized the WGAN-GP model as proposed in Gulrajani et al. (2017), following its official open-source implementation[1]. We re-implemented this code in PyTorch for our experiments. For a pretrained off-the-shelf generator, we utilized StyleGAN-XL (Sauer et al., 2022) model pretrained on ImageNet (Krizhevsky et al., 2012) with resolution of 32×32. We downloaded the model parameters from https://github.com/autonomousvision/stylegan-xl and implemented the model using these parameters. For the client discriminator, we adopted a simple 4-layer CNN discriminator, following the implementation from https://github.com/Ksuryateja/DCGAN-MNIST-pytorch/blob/master/gan_mnist.py. To address the widely known overfitting issue of the discriminator (Adlam et al., 2019; Yang et al., 2022) and the resulting dominance of client weights, we employed a composition of two sigmoid activations for the discriminator output. This ensures that the odds value $\Phi_k$ for client $k$'s discriminator $D_k$ is constrained between 1 and $e$.

FedTGP (Zhang et al., 2024) utilizes an additional TGP model. We implemented TGP model following its official open-source implementation from https://github.com/TsingZ0/FedTGP.

**Heterogeneous Client Data Split** To introduce non-iid distributions among client datasets, we ensured that each client's distribution follows a Dirichlet distribution Dir($\alpha$), similar as in Lin et al. (2020); Wang et al. (2020a); Marfoq et al. (2022); Li et al. (2023). As the parameter $\alpha$ increases, each client tends to have a more homogeneous distribution, whereas smaller $\alpha$ values result in increased data heterogeneity among clients. We conducted experiments for each dataset with $\alpha$ values of 0.1 and 0.05. The number of data samples that each client has per class for CIFAR-10/100 datasets with $\alpha$ values of 0.1 and 0.05 is illustrated in Figures 5 and 6. It's worth noting that ImageNet100 also has 100 classes, so the trends observed in CIFAR-100 would likely align with those in ImageNet100. We can observe that when $\alpha = 0.05$, the difference in the number of data samples per class for each client is more pronounced compared to when $\alpha = 0.1$. This results in more skewed distributions for individual clients.

**Details for Dataset** We normalized the pixel values of all image datasets to fall within the range $[-1, 1]$, ensuring that the generated data also has pixel values within this range. Additionally, for both the training datasets of clients and the server's unlabeled dataset, we conducted further data augmentation using PyTorch's random horizontal flip.

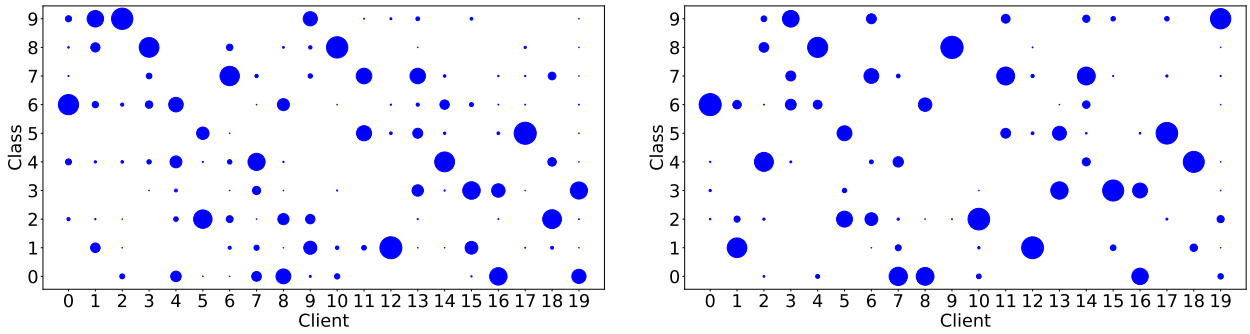
**Selection of $\text{Acc}_{\text{target}}$** We used the highest multiple of 5 of the test accuracy (%) achieved by the FedAVG algorithm within 100 rounds for all five different random seeds as $\text{Acc}_{\text{target}}$ for Table 4.

## F. Additional Experimental Results

### F.1. Results with 100 Clients

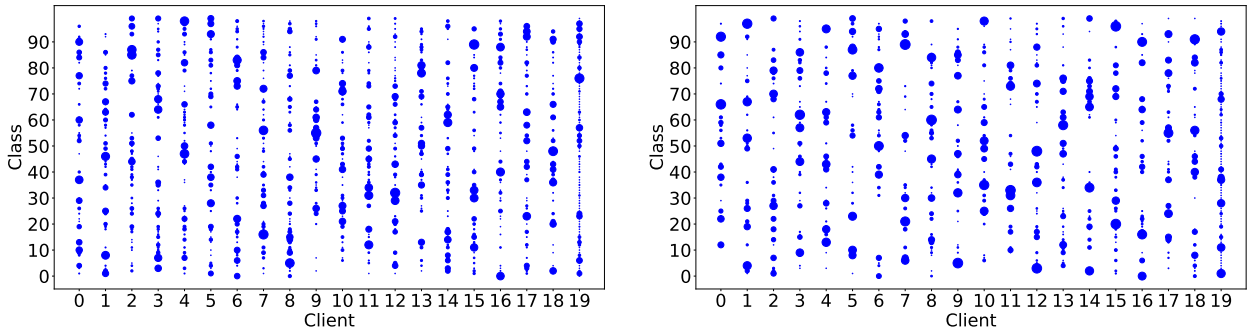
Figure 7 shows (a) the test accuracy of the server model, (b) the test accuracy of the ensemble model, and (c) the test loss of the ensemble model during the training process for $K = 100$ clients on CIFAR-10 dataset with $\alpha = 0.05$. The latter

[1] https://github.com/igul222/improved_wgan_training




(a) $\alpha = 0.1$ (b) $\alpha = 0.05$

*Figure 5.* Client data split for CIFAR-10 with $\alpha = 0.1, 0.05$.




(a) $\alpha = 0.1$ (b) $\alpha = 0.05$

*Figure 6.* Client data split for CIFAR-100 with $\alpha = 0.1, 0.05$.

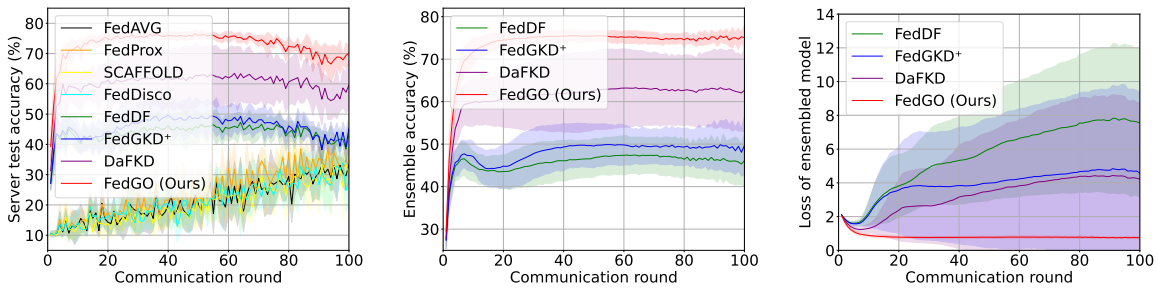
two measures were evaluated only for algorithms incorporating ensemble distillation. FedGO achieves the test accuracy of 69.52%, which is slightly lower than 72.35% with 20 clients (Table 3). In comparison, FedAVG, FedProx, FedDF, FedGKD$^+$, and DaFKD show significant performance drops to 33.40%, 35.07%, 44.36%, 45.44%, and 59.62%, respectively. This demonstrates that even in settings with a large number of clients, FedGO exhibits robust performance compared to the baselines.

In terms of the test accuracy and the test loss of the ensemble model, FedGO consistently demonstrates superior performance across all rounds compared to the baseline algorithms. Furthermore, unlike the baseline algorithms, whose test loss initially decreases but then becomes unstable and increases from early rounds, FedGO's loss converges with small deviation.

### F.2. Ensemble Test Accuracy Comparison and Analysis

Figure 8 shows the ensemble test accuracy on the server's unlabeled dataset during the training process for our FedGO algorithm and the baseline ensemble algorithms: FedDF, FedGKD$^+$, and DaFKD. It demonstrates that using pseudo-labels generated by theoretically guaranteed weighting methods allows the server to achieve higher final performance and faster convergence.

However, in Table 3 of the paper, the performance gap between our method and the baselines on CIFAR-100 and ImageNet100 was not as large as that on CIFAR-10. We infer the reason from Theorem 3.2. The second term on the RHS of Theorem 3.2 can be interpreted as the distillation loss due to the difference between the hypothesis class and the spanned hypothesis class. Even if our ensemble is close to optimal, the knowledge-distilled server model may not follow the performance of the ensemble if it hard for a single model to learn the pseudo-labels, and we conjectures that it becomes harder as the number of classes increases.




(a) Server test accuracy (b) Ensemble test accuracy (c) Ensemble test loss

*Figure 7.* Server test accuracy (%), test accuracy of the ensemble model (%), and test loss of the ensemble model of our FedGO and baselines for 100 clients on CIFAR-10 dataset with $\alpha = 0.05$.

To support our hypothesis, we show the minimum of mean distillation loss for five different random seeds during 100 round of communication rounds in Table 6. The distillation loss increases progressively from CIFAR-10 to CIFAR-100 to ImageNet100. In addition, the distillation loss is higher for $\alpha = 0.05$ than for $\alpha = 0.1$, which explains why the gap from the central training is larger for $\alpha = 0.05$.

*Table 6.* Minimum mean distillation loss of FedGO on three image datasets with $\alpha = 0.1, 0.05$.

| Dataset | CIFAR-10 | CIFAR-100 | ImageNet100 |
|---|---|---|---|
| $\alpha = 0.1$ | 0.175 | 0.237 | 0.363 |
| $\alpha = 0.05$ | 0.266 | 0.348 | 0.539 |

### F.3. Ensemble Distillation with a Different Server Dataset

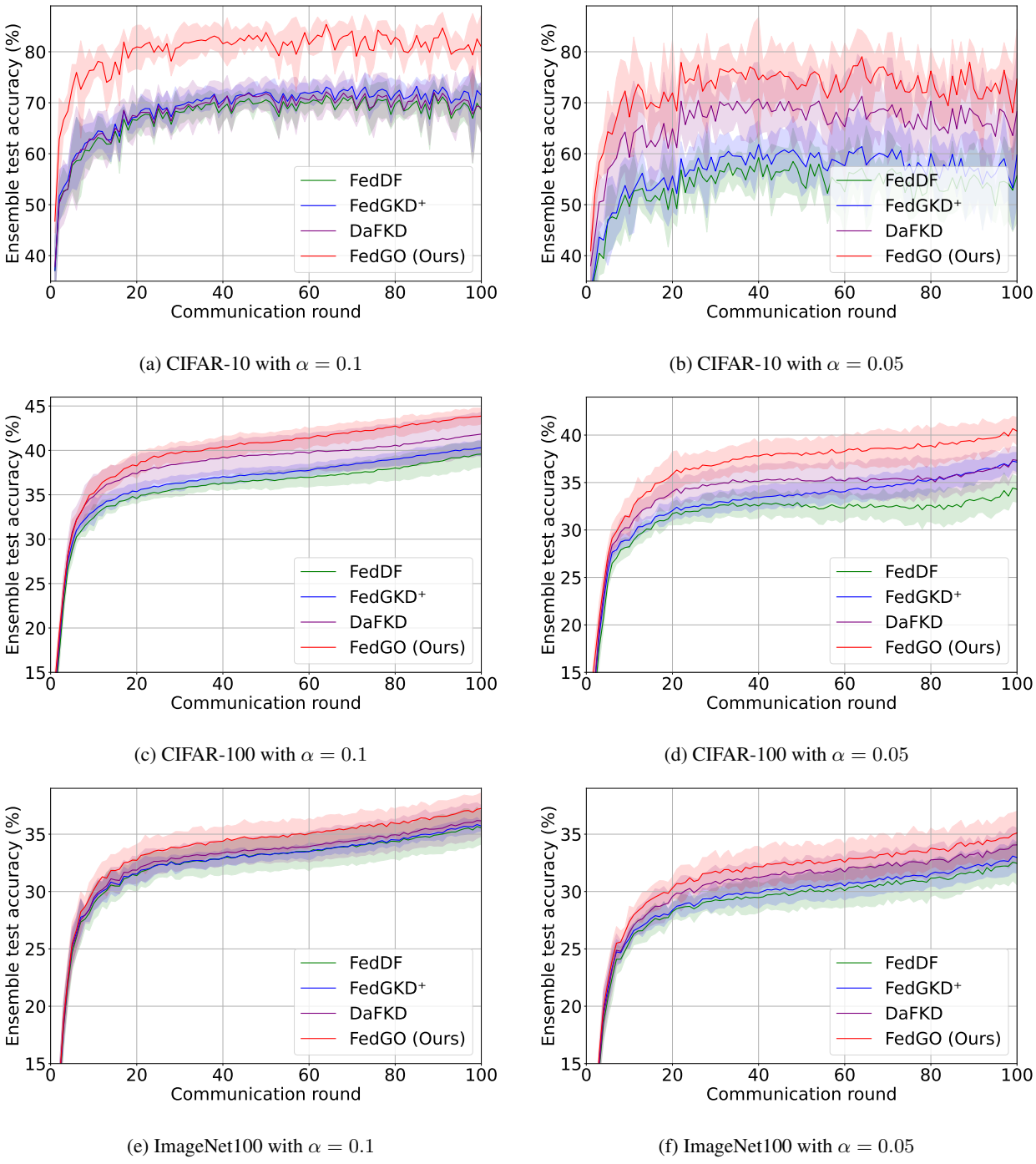
Our theoretical justification of constituting an optimal ensemble in Corollary 3.3 allows heterogeneity between the server data distribution $p_s$ and the client average distribution $p$. To demonstrate the effectiveness of FedGO when $p_s \neq p$ which makes more sense in practice, we report the results when clients have the half of the CIFAR-10 dataset and the server has the half of the CIFAR-100 (unlabeled) dataset, in Table 7. The experimental results demonstrate that ensemble distillation even with heterogeneous server dataset is helpful in improving the performance. Furthermore, by employing optimal model ensemble, our FedGO algorithm, with theoretical performance guarantee, shows improvement over FedDF and DaFKD.

*Table 7.* Server test accuracy (%) and ensemble test accuracy (%) of our FedGO and baselines with heterogeneous server dataset: CIFAR-10 for client dataset and CIFAR-100 for server's unlabeled dataset.

| | | FedAVG | FedDF | DaFKD | **FedGO (ours)** |
|---|---|---|---|---|---|
| $\alpha = 0.1$ | Server test accuracy | 58.65±5.75 | 59.89±1.88 | 60.84±2.65 | **60.92**±1.95 |
| | Ensemble test accuracy | - | 62.62±0.90 | 63.88 ± 2.02 | **64.23**±1.29 |
| $\alpha = 0.05$ | Server test accuracy | 46.61±8.54 | 49.21±4.48 | 52.31±4.26 | **52.89**±3.47 |
| | Ensemble test accuracy | - | 56.06±207 | 59.30 ±1.33 | **60.43** ±0.56 |

### F.4. Results with Alternative Model Architectures

In the main paper, we conducted experiments with ResNet-18 model structure. In this subsection, we present the results with VGG11 (Simonyan & Zisserman, 2015) (with BatchNorm Layers (Ioffe & Szegedy, 2015)) and ResNet-50 models. For VGG11, both the client and server models are trained using SGD with a learning rate of 0.01 and momentum of 0.9, and all the other settings including hyperparameters are kept identical to those in the main paper. We implemented VGG11 based on https://github.com/chengyangfu/pytorch-vgg-cifar10. For ResNet-50, all the settings including optimizer and




(a) CIFAR-10 with $\alpha = 0.1$

(b) CIFAR-10 with $\alpha = 0.05$

(c) CIFAR-100 with $\alpha = 0.1$

(d) CIFAR-100 with $\alpha = 0.05$

(e) ImageNet100 with $\alpha = 0.1$

(f) ImageNet100 with $\alpha = 0.05$

*Figure 8.* Ensemble test accuracy (%) of FedGO and baselines over communication rounds on three image datasets with $\alpha = 0.1, 0.05$.

hyperparameters are set to the same as the main paper. Table 8 presents the server test accuracy of FedGO and baseline algorithms with the aforementioned model structures on CIFAR-10 with $\alpha = 0.1$ after 100 communication rounds.

We can see that our FedGO algorithm consistently achieves performance gains over FedDF and FedGKD$^+$ across different model structures.

*Table 8.* Server test accuracy (%) of central training, FedDF, FedGKD$^+$ and FedGO on CIFAR-10 with $\alpha = 0.1$ after 100 communication rounds, when utilizing VGG11 and ResNet-50.

| | VGG11 | ResNet-50 |
|---|---|---|
| Central training | $83.27 \pm 0.60$ | $85.12 \pm 0.44$ |
| FedDF | $68.59 \pm 4.65$ | $65.21 \pm 4.62$ |
| FedGKD$^+$ | $67.81 \pm 3.60$ | $66.21 \pm 3.01$ |
| **FedGO (ours)** | $72.53 \pm 4.10$ | $75.52 \pm 4.30$ |

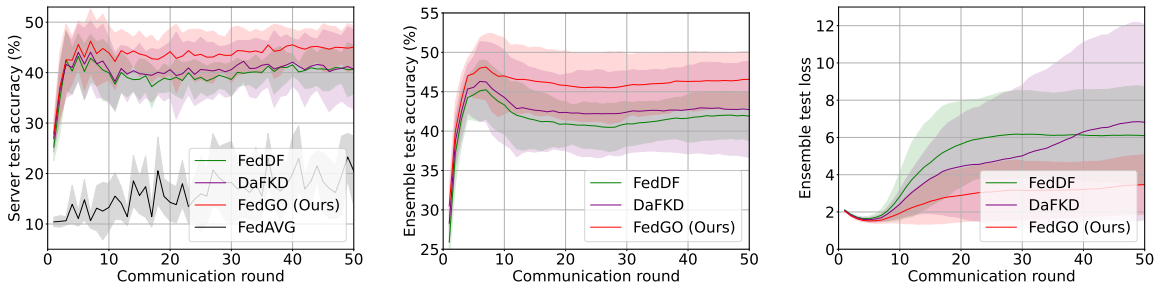
### F.5. Data-free FedGO

In practice, the server may have no extra dataset. In this case, we first prepare a generator and then generate a distillation dataset using the generator. The generator can either be an off-the-shelf pretrained model or trained through an FL approach (Rasouli et al., 2020; Guerraoui et al., 2020; Li et al., 2022; Wang et al., 2023c; Fan & Liu, 2020; Behera et al., 2022; Hardy et al., 2019; Xiong et al., 2023; Zhang et al., 2021; 2023b), corresponding to the 3rd and 4th scenarios in Table 2 in our main paper, respectively.

Figure 9 and Table 9 present the results for the two data-free approaches with 100 clients on CIFAR-10 dataset. We employed styleGAN (Karras et al., 2019) pretrained with ImageNet dataset for the off-the-shelf generator, and applied the FedGAN algorithm (Rasouli et al., 2020) for training a generator. For both approaches, our FedGO shows performance gains in server test accuracy, ensemble test accuracy, and ensemble test loss compared to the baseline algorithms, including the uniform weighting of FedDF (Lin et al., 2020) and the domain-aware weighting of DaFKD (Wang et al., 2023a). Moreover, we observe that ensemble test accuracy is higher than server test accuracy, which can be attributed to the increased discrepancy between the distillation dataset distribution $p_s$ and the average client distribution $p$. This discrepancy amplifies the gap between the ensemble model's loss and the server model's loss, as analyzed in Theorem 3.2 of the main paper.

In both data-free approaches, we have $p_s = p_g$, under which our weighting method is optimal for $\forall x \in p_s = p_g$ from Theorem 3.6 in our main paper. Note that the distance between $p$ and $p_g = p_s$ for the generator trained from FedGAN is smaller than that for the off-the-shelf generator. Consequently, despite using a simpler generator trained on a smaller dataset, we observe that the performance of FedGO using the generator trained from FedGAN is slightly better than that using the off-the-shelf generator.

Finally, we can observe performance degradation compared to the case where the distillation is performed on a real dataset. This can be attributed to the naive reuse of generated images, which has been identified as a cause of performance degradation (Yoon et al., 2024; Wang et al., 2024). An interesting future work would be on improving the performance of knowledge distillation using generated images. Still, experimental results demonstrate that ensemble distillation is beneficial in improving performance even with generated images. Furthermore, by employing an optimal model ensemble, our FedGO shows improvement over FedDF and DaFKD.




(a) Server test accuracy (b) Ensemble test accuracy (c) Ensemble test loss

*Figure 9.* Test accuracy of server model (%), ensemble test accuracy (%), and test loss of ensemble model of the data-free FedGO with an off-the-shelf generator (the case (G2)+(D2) of Table 2) and baselines with 100 clients on the CIFAR-10 dataset with $\alpha = 0.05$.

*Table 9.* Test accuracy of server model (%), ensemble test accuracy (%), and test loss of ensemble model of the data-free FedGO at 100 communication round, with a generator trained from FedGAN (the case (G3)+(D2) of Table 2) and baselines with 100 clients on the CIFAR-10 dataset with $\alpha = 0.05$.

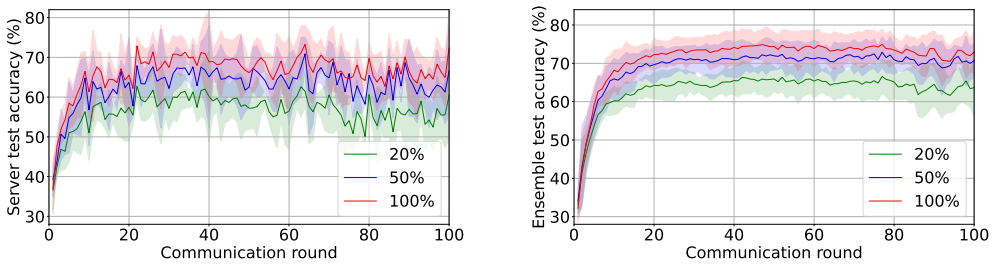
| Method | FedAVG | FedProx | SCAFFOLD | FedDisco | FedDF | DaFKD | **FedGO (ours)** |
|---|---|---|---|---|---|---|---|
| Server Test Accuracy | 18.12±6.50 | 21.22±10.03 | 16.42±4.69 | 21.11±7.33 | 25.81±5.35 | 23.71±4.99 | **27.26**±2.32 |
| Ensemble Test Accuracy | - | - | - | - | 37.01±5.22 | 35.07±5.40 | **38.84**±2.45 |
| Ensemble Test Loss | - | - | - | - | 1.76±0.09 | 1.84±0.15 | **1.69**±0.06 |

## F.6. Impact of Server Model Hyperparameters on Performance

### F.6.1. Amount of Unlabeled Data

Figure 10 shows the test accuracy of the server model and the test accuracy of the ensemble model during the training process for our FedGO algorithm. We conducted experiments by reducing the server dataset size to 50% and 20% of the size assumed in our main CIFAR-10 experiments. For these experiments, the server epochs were adjusted to ensure the same number of gradient steps: doubled for 50% and quintupled for 20%, while keeping other hyperparameters the same.

Figure 10 demonstrates that when the server dataset size decreases, the test accuracy of the ensemble model remains nearly consistent, while that of the server model decreases. This suggests that even with pseudo-labels of similar quality, the performance of the server model can decline as the server dataset size decreases. This can be interpreted as the server model becoming more prone to overfitting as the distillation dataset becomes smaller (Hinton, 2015). Note that FedGO has the performance improvement of about 15% over FedAVG even with only 20% of the dataset, which corresponds to 20% of the total client dataset size.




(a) Server test accuracy

(b) Ensemble test accuracy

*Figure 10.* Server test accuracy (%) and ensemble test accuracy (%) of our FedGO on the CIFAR-10 dataset with $\alpha = 0.05$, according to the size of the unlabeled dataset at the server. In the legend, $X\%$ means that the size of the unlabeled dataset at the server is reduced to $X\%$ of the size assumed in our main CIFAR-10 setting.

### F.6.2. Server Model Training Epochs

Table 10 shows the impact of server model training epochs on FedGO's performance on CIFAR-10 with $\alpha = 0.1$ after 100 communication rounds. Using 5 epochs outperforms 1 epoch, with minimal performance differences beyond 5 epochs. Notably, even with only 1 epoch, FedGO significantly outperforms all the baselines trained with 10 server epochs in Table 3.

*Table 10.* Server test accuracy (%) and ensemble test accuracy (%) of FedGO on CIFAR-10 with $\alpha = 0.1$ after 100 communication rounds, according to the number of server model training epochs.

| Epoch | 1 | 5 | 10 | 20 |
|---|---|---|---|---|
| Server Test Accuracy | 74.03±6.41 | 79.56±5.30 | 79.62±4.36 | 78.32±5.13 |
| Ensemble Test Accuracy | 77.16±0.88 | 80.97±0.87 | 81.56±0.48 | 81.39±0.75 |

#### F.6.3. SERVER MODEL LEARNING RATE DECAY

In the main paper, we used cosine learning rate decay by following the experimental setting of FedDF. As shown in Table 11, the absence of learning rate decay results in further performance improvement. Specifically, an ensemble test accuracy of 85.20% is achieved, which is comparable to the central training model's accuracy of 85.33%, demonstrating the effectiveness of our provably near-optimal weighting method.

*Table 11.* Server test accuracy (%) and ensemble test accuracy (%) of FedGO on CIFAR-10 with $\alpha = 0.1$ after 100 communication rounds, with and without learning rate decay during server model training.

| | FedGO | |
|---|---|---|
| | with LR decay | without LR decay |
| Server Test Accuracy | 79.62±4.36 | 80.18±2.16 |
| Ensemble Test Accuracy | 81.56±0.48 | 85.20±1.33 |

### F.7. Impact of Generator and Discriminator Quality on Performance

#### F.7.1. GENERATOR TRAINING STEPS

Table 12 shows the performance of our FedGO with varying generator training steps (100,000 in the main setup) alongside baseline algorithms after 50 communication rounds, while keeping all other settings unchanged from the main setup. FedGO with the generator trained for 25,000 steps performs better than that with the randomly initialized generator (0 steps), with little performance improvement beyond 25,000 steps. Remarkably, even a randomly initialized generator outperforms FedDF with uniform weighting and achieves performance comparable to DaFKD with a generator trained for 100,000 steps.

*Table 12.* Server test accuracy (%) and ensemble test accuracy (%) of FedGO on CIFAR-10 with $\alpha = 0.1$ after 50 communication rounds, according to the number of generator training steps.

| | FedDF | DaFKD | **FedGO (ours)** | | | | |
|---|---|---|---|---|---|---|---|
| Generator Training Steps | - | 100,000 | 0 | 25,000 | 50,000 | 75,000 | 100,000 |
| Server Test Accuracy | $70.18 \pm 2.56$ | $71.42 \pm 3.11$ | $71.12 \pm 2.07$ | $76.74 \pm 3.16$ | $78.43 \pm 0.99$ | $78.89 \pm 1.55$ | $78.24 \pm 1.61$ |
| Ensemble Test Accuracy | $73.55 \pm 2.41$ | $74.54 \pm 2.80$ | $74.88 \pm 1.63$ | $79.12 \pm 1.97$ | $80.72 \pm 0.75$ | $80.87 \pm 0.98$ | $80.82 \pm 0.82$ |

#### F.7.2. DISCRIMINATOR TRAINING EPOCHS

Table 13 shows the final performance of the FedGO algorithm for different numbers of discriminator training epochs on CIFAR-10 with $\alpha = 0.05$. It can be seen that training the discriminator more times results in better final performance. Additionally, we note that among the baselines in Table 3 and Figure 2, except DaFKD which originally trains the generator and discriminators at each round, the highest performance is achieved by the variance weighting method, with the test accuracy of 67.51±10.77%, indicating that there is a performance gain from the FedGO algorithm with just 5 epochs of discriminator training.

*Table 13.* Server test accuracy (%) of FedGO on CIFAR-10 with $\alpha = 0.05$ at the 100-th communication round, according to the number of discriminator training epochs at the clients.

| Epoch | 1 | 5 | 10 | 30 | 50 |
|---|---|---|---|---|---|
| Accuracy | 63.96±9.03 | 71.38±7.76 | 70.84±8.88 | 72.35±9.01 | **76.92**±5.08 |

#### F.7.3. DISCRIMINATOR ARCHITECTURES

Table 14 presents the number of parameters, the number of FLOPs required for the forward computation, and the performance of FedGO on CIFAR-10 with $\alpha = 0.1$ at the 100-th communication round, when the following three different client discriminator structures are used:

- CNN: The baseline architecture used in the main setting. It consists of four convolutional layers.
- CNN+MLP: A variation of the CNN architecture, where the last two convolutional layers in the CNN are replaced by a single multi-layer perceptron (MLP) layer, resulting in a three-layer shallow network.
- ResNet: A deeper architecture based on ResNet-8, an 8-layer residual network.

*Table 14.* Server test accuracy (%) of FedGO on CIFAR-10 with $\alpha = 0.1$ at the 100-th communication round along with the number of parameters and the number of FLOPs for the forward computation, according to different client discriminator structures.

| | FedGO | | |
|---|---|---|---|
| Discriminator Structure | CNN | CNN+MLP | ResNet |
| Number of Parameters | 662,528 | 142,336 | 1,230,528 |
| FLOPs | 17.6 MFLOPs | 9.18 MFLOPs | 51.1 MFLOPs |
| Server Test Accuracy | 79.62±4.36 | 79.71±4.71 | 78.73±5.03 |

Table 14 shows almost identical performances regardless of client discriminator architectures, demonstrating the robustness of FedGO to the discriminator architecture. In particular, the CNN+MLP discriminator, which has less than a quarter of the parameters and around the half of the FLOPs compared to the original CNN structure, achieves similar performance.

### F.8. Impact of Byzantine Clients

We have conducted additional experiments in which 5 and 10 out of 20 clients were Byzantine, outputting only the maximum value for the discriminator. As shown in Table 15, the classification accuracy on CIFAR-10 without any Byzantine clients was 72.35% with a standard deviation of 9.01. When 5 clients were Byzantine, the accuracy dropped to 69.75%, and further decreased to 66.38% when 10 clients were Byzantine. Despite this challenging scenario where up to half of the clients were compromised, our method still significantly outperformed all baseline approaches that do not incorporate a discriminator, demonstrating its strong robustness against adversarial behavior.

*Table 15.* Server test accuracy (%) of FedGO on CIFAR-10 with $\alpha = 0.05$ at the 100-th communication round along with the number of Byzantine clients.

| Byzantine Clients | Accuracy (%) |
|---|---|
| 0 | 72.35±9.01 |
| 5 | 69.75±5.05 |
| 10 | 66.38±4.97 |

## G. Comprehensive Analysis of Communication, Privacy, and Computational Complexity

Let us provide a detailed explanation for Table 2. If the server dataset is available from the outset (first two rows in Table 2), we only need one-shot communication of generator (from the server to the clients) and discriminators (from the clients to the server). Hence, additional communication burden and client-side privacy leakage are negligible. In particular, for our experiments, the parameters of the ResNet-18 classifier are approximately 90MB when stored as a PyTorch state_dict. In comparison, the generator and discriminator models are 4.61MB and 2.53MB, respectively. Over 100 communication rounds, during which ResNet-18 is transmitted repeatedly, the additional communication burden introduced by FedGO is nearly negligible. However, the server dataset is used for distillation for each communication round, incurring non-negligible privacy leakage on the server side. If there is no server dataset (last two rows in Table 2), there is no additional privacy leakage on the server side. If we use a pretrained generator instead (third row), additional communication burden, client-side privacy leakage and computational burden are negligible, but it is challenging in general to secure an off-the-shelf generator which generates data with a distribution similar to the client data distribution. To train a generator through FL (last row), multiple rounds of GAN exchanges between the server and clients are required; however, since our FedGO requires a

small number of GAN exchanges in pre-FL, compared to the number of model exchanges in main FL, the additional communication burden, client-side privacy leakage, and computational burden are still negligible.

In the following, we provide a quantitative analysis of additional privacy leakage of FedGO compared to FedAVG, and an explicit comparison of computational cost for FedGO and baselines.

### G.1. Privacy Analysis

For privacy measure, we consider local differential privacy (LDP) (Dwork et al., 2006) which is widely accepted both in academia and industry. Note that when the data is provided $n$ times by independently applying an LDP mechanism with privacy budget $\epsilon$ for each provision, the total privacy budget becomes $n\epsilon$ from the parallel composition result (Dwork et al., 2014).

Let $T$ denote the total number of communication rounds in the main-FL stage. For the case (G3) in Section 3.2, let $T'$ denote the total number of communication rounds required to train a GAN in the pre-FL stage, which is 1/20 of $T$ in our experiment. For simplicity, we assume that every client participates in FL for each communication round. Let $\epsilon_M, \epsilon_D$, and $\epsilon_G$ denote the privacy budgets of LDP mechanisms applied to the classifier, discriminator, generator sent from each client at each communication round, respectively. Let $\hat{\epsilon}_M$ and $\hat{\epsilon}_G$ denote the privacy budgets of LDP mechanisms applied to the classifier and the generator sent from the server when the server uses its own dataset in case of (S1) for training the generator and for distillation, respectively.

Table 16 shows the client-side and the server-side total privacy leakage of FedAVG and FedGO under various scenarios. For FedAVG, each client provides the classifier $T$ times, and hence the client-side total privacy leakage becomes $T \cdot \epsilon_M$. Let us first analyze the additional client-side privacy leakage of FedGO under various scenarios. For FedGO with the method (G1)+(D1), (G2)+(D1), or (G2)+(D2), the client sends its discriminator only once, incurring extra privacy leakage of $\epsilon_D$, which is negligible with large $T$. For FedGO with (G3)+(D2), the clients need to send the discriminator and the generator for $T'$ times to train a GAN in the pre-FL stage, leading to additional privacy leakage of $T' \cdot (\epsilon_D + \epsilon_G)$; however, since $T'$ is only 1/20 of $T$ in our experiment, the additional privacy leakage is negligible. Next, server-side privacy issues arise only when the server has its own dataset. If the server trains the generator from its dataset and provides it to the clients for the case of (G1), it yields the privacy leakage of $\hat{\epsilon}_G$. In addition, if the server uses its dataset for distillation and applies an LDP mechanism with privacy budget $\hat{\epsilon}_M$ to the classifier for each communication round for the case of (D1), it results in a non-negligible amount of additional privacy leakage $T \cdot \hat{\epsilon}_M$.

We consider this as a future work direction: developing methods for training ensemble distillation while preserving LDP constraints in classifiers or discriminators.

*Table 16.* Quantitative analysis of the client-side and the server-side total privacy leakage of FedAVG and FedGO under various scenarios.

| | | Client-side | Server-side |
|---|---|---|---|
| FedAVG | | $T \cdot \epsilon_M$ | – |
| FedGO | (G1)+(D1) | $T \cdot \epsilon_M + \epsilon_D$ | $\hat{\epsilon}_G + T \cdot \hat{\epsilon}_M$ |
| | (G2)+(D1) | $T \cdot \epsilon_M + \epsilon_D$ | $T \cdot \hat{\epsilon}_M$ |
| | (G2)+(D2) | $T \cdot \epsilon_M + \epsilon_D$ | – |
| | (G3)+(D2) | $T' \cdot (\epsilon_D + \epsilon_G) + T \cdot \epsilon_M + \epsilon_D$ | – |

### G.2. Computational Cost Comparison

Table 17 shows the floating point operations (FLOPs) during CIFAR-10 training for the baselines and FedGO with the four scenarios described in Table 2. 1 MFLOP represents $10^6$ FLOPs.

First, on the client side, the computational cost for FedGO with (G1) or (G2) is comparable to that of FedAVG and FedDF, which only optimize the client's vanilla supervised loss. The cost is roughly half of the cost of FedGKD$^+$, which includes a regularization term in the client objective. This reduction is because the client only needs to train the discriminator only once during the pre-FL stage. In each round, FedAVG and FedDF compute 4.17e+7 MFLOPs per client update, whereas the computational cost for training a client's discriminator is 3.29e+7 MFLOPs—less than the cost for one round of classifier training. The additional computation cost for FedGO with (G1) or (G2) is therefore minimal, especially considering its

fast convergence speed.[2] Note that the client-side computational cost of FedProx is same as that of FedAVG because the proximal term computation is negligible. For FedGO with (G3), clients train the generator using an FL approach during the pre-FL stage. Although this process adds computational overhead to the client side, the number of FL rounds required for training the generator is relatively small—only 1/20 of the rounds needed for the main FL stage in our experiment. Consequently, this results in only a slight increase in the overall computational cost compared to FedAVG. In contrast, data-free FL algorithms like DaFKD (Wang et al., 2023a) exchange both the generator and model in every communication round, resulting in significant overhead associated with utilizing the generator when many rounds for model training are needed. By decoupling generator preparation from model training, FedGO with (G3)+(D2) avoids this issue, reducing the additional overhead.

Next, on the server side, in FedGO with (G1)+(D1), training a generator using server dataset involves significant additional computation compared to FedDF due to the 100,000 steps required for training a ResNet-based generator and discriminator. However, given that federated learning typically involves a server with ample resources and clients with limited computational resources, this increase in server-side computation is more affordable in practice, compared to increasing the computational burden on clients. Furthermore, while the computation for training the generator is irrelevant to the number of clients, the computation required for pseudo-labeling scales linearly with the number of clients. Note that the total computational cost in Table 17 assumes 20 clients. In real-world scenarios, where 100+ clients may participate in FL, the relative proportion of the computational cost for training the generator will decrease.

Note that using an off-the-shelf generator reduces the additional server-side computational cost of FedGO. FedGO with (G2)+(D1) requires approximately 2% more computation than FedDF, but achieves a significant performance gain of about 13%p on CIFAR-10 with $\alpha = 0.05$ in Table 3. The reason why FedGO with (G2)+(D2) has a higher server-side computational cost compared to FedGO with (G2)+(D1) is that FedGO with (G2)+(D2) generates distillation dataset using a heavy generator, StyleGAN.

FedGO with (G3)+(D2) also generates distillation dataset using a generator but the generator used here is lighter than StyleGAN generator used in (G2)+(D2). The computational cost for the generation of distillation dataset in FedGO with (G3)+(D2) is 2.11e+7 MFLOPs which is negligible compared to the computational cost for pseudo-labelling and ensemble distillation which is 5.07e+10 MFLOPs. On the other hand, note that the computational cost of FedGO with (G3)+(D2) is slightly lower than DaFKD while both train a generator using an FL approach. The reduction mainly comes from the difference that FedGO with (G3)+(D2) generates a distillation dataset only once after the training of the generator, while DaFKD updates the distillation dataset in every communication round. Finally, note that the server-side computational cost of FedGO with (G3)+(D2) is comparable to the case (G2)+(D1), even though (G3) trains a generator through an FL approach. This is because the server's role is limited to averaging the clients' generator and discriminator, incurring negligible additional computational cost on the server side.

*Table 17.* The number of MFLOPs for training our FedGO and baselines on CIFAR-10 for 100 communication rounds.

| | FedAVG | FedProx | FedDF | FedGKD$^+$ | DaFKD | FedGO (ours) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | (G1)+(D1) | (G2)+(D1) | (G2)+(D2) | (G3)+(D2) |
| Client-side | 3.33e+10 | 3.33e+10 | 3.33e+10 | 6.67e+10 | 8.81e+11 | 3.40e+10 | 3.40e+10 | 3.40e+10 | 3.52e+10 |
| Server-side | 7.82e+3 | 7.82e+3 | 5.00e+10 | 5.00e+10 | 5.28e+10 | 1.39e+11 | 5.07e+10 | 6.01e+10 | 5.07e+10 |
| Total | 3.33e+10 | 3.33e+10 | 8.33e+10 | 1.17e+11 | 9.34e+11 | 1.73e+11 | 8.47e+10 | 9.41e+10 | 8.59e+10 |

## H. Limitation

Although our FedGO algorithm can be extended to model heterogeneous scenarios as in FedDF, we found it challenging to define an optimal model ensemble for multiple hypothesis classes. Consequently, it appears difficult to apply the results of Theorem 3.2 and Corollary 3.3 in such cases.

[2]We note that the computational cost exceeds 2%, rather than being below 1%, because in each round, only $C = 0.4$ proportion of clients are sampled to participate in federated learning, rather than full client participation.