OFedED: One-shot Federated Learning with Model Ensemble and Dataset Distillation

OFedED: One-Shot Federated Learning through Model Ensemble and Dataset Distillation

Xuhui Li
Zhengquan Luo*
xuhui.li@mbzuai.ac.ae
zhengquan.luo@mbzuai.ac.ae
MBZUAI
Abu Dhabi, UAE

Zihui Cui
zihui.cui@mbzuai.ac.ae
MBZUAI
Abu Dhabi, UAE

Xin Cao
xin.cao@unsw.edu.au
University of New South Wales
Kensington, Austalia

Zhiqiang Xu†
zhiqiang.xu@mbzuai.ac.ae
MBZUAI
Abu Dhabi, UAE

Abstract

One-shot federated learning (FL) has gained traction due to its communication efficiency and scalability. However, unlike traditional FL, which can frequently align client models through multiple rounds of client training and server aggregation, one-shot FL allows only a single communication round, causing each client to easily overfit its local data and leading to divergent objectives. Without any chance to iteratively correct these biases or mitigate heterogeneity, the aggregated model significantly deviates from the optimum achieved under dataset centralized training. To address this challenge, we propose *OFedED*, a one-shot FL framework that preserves privacy and fully exploits client data by combining local data distillation with server-side ensemble learning. Each client distills its own dataset into an ultracompact coreset that retains essential distributional characteristics; the server aggregates these coresets to guide ensemble training that captures inter-client heterogeneity, harnesses complementary knowledge, corrects local bias, and drives performance close to centralized training. In addition, we theoretically show that, under mild assumptions for local data distillations, the server can simulate a centralized optimization process by finetuning on the aggregated distilled data, effectively bypassing the need for multiple communication rounds, showing that properly distilled data can encode sufficient task-relevant information to support centralized-level optimization. Extensive experiments reveal that OFedED consistently and significantly outperforms SOTA methods, achieving an improvement of up to 9.17% on MNIST and 3.97% on CIFAR-10, the robustness being verified also by experiments using ResNet and various server-client architectures.

CCS Concepts

• **Computing methodologies** → **Machine learning**; **Distributed computing methodologies**.

Keywords

One-shot Federated Learning, Dataset Distillation, Ensemble Modeling

ACM Reference Format:
Xuhui Li, Zhengquan Luo, Zihui Cui, Xin Cao, and Zhiqiang Xu. 2025. OFedED: One-Shot Federated Learning through Model Ensemble and Dataset Distillation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746252.3761309

1 Introduction

Federated learning (FL) [18] enables training of collaborative models among multiple clients while preserving data privacy. However, traditional multi-round FL requires frequent communications between the server and clients, leading to high communication costs and increased exposure to security risks [10, 15, 28]. To mitigate these issues, one-shot FL [7] has been proposed, which completes the entire FL process in a single communication round.

Despite the advantage of communication efficiency, current one-shot FL methods often fail to fully exploit client data. Unlike traditional FL methods, which refine and align models across multiple rounds that benefit from iterative local optimization and server feedback, one-shot FL compresses all client knowledge into a single transmission, without any chance to correct or guide from the server. This inherently limits its ability to achieve distributional alignment or correct local overfitting. Moreover, most methods transmit only model parameters, leaving the question of whether such limited information is sufficient to capture the underlying data distributions and support effective global learning [15]. For instance, DENSE [31] and Co-boosting [4] only transmit model parameters from clients to the server, and then generate synthetic data on the server. Although it reduces both computation on clients and communication, the quality of synthetic data generated on the

*Both authors contributed equally to this research.
†Corresponding author.

CIKM '25, November 10–14, 2025, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761309

server remains unsatisfactory, especially for data with great heterogeneity across clients. The absence of real data during distillation is likely to make it difficult to capture diverse client data distributions. Naturally, the current status of one-shot FL raises the following crucial question:

Would it be possible for one-shot FL to achieve performance close to that under the centralized model training?

To answer the question, we propose *OFedED*, which integrates data distillation with ensemble modeling. In order to capture hierarchical feature structures in client data more effectively and thus improve the representational quality of distilled data, OFedED uses a layer-wise distribution matching scheme to synthesize data on clients. After sending synthetic data to the server, it is crucial to maximize its utility on the server within a single communication round. To this end, we incorporate two key components into OFedED: a class-weighted ensemble model and a class-center loss. The former dynamically adjusts model contributions based on class distributions to effectively mitigate client data heterogeneity, while the latter ensures that server-generated data aligns more closely with synthetic client data to improve representation quality by aligning features with class centers. Since we upload distilled client data to the server, it poses potential issues in data privacy [1]. By adding Gaussian noises to both synthetic client data $\mathcal{S}_k$ and model parameters θ_k, as in Theorem 2, OFedED is guaranteed to satisfy (ϵ, δ)-differential privacy, so that distilled client data remain secure without leakage of sensitive privacy information.

Moreover, with the definition of a distributional distance bound (i.e., Definition 1) that ensures a small feature-level discrepancy, characterized by constant σ_d, between distilled data and the original data. Together with other two mild assumptions (i.e., Assumptions 1 and 2), we establish in Theorem 1 that the expected loss difference between the optimal model trained on aggregated distilled data and that trained on all client data in a centralized setting can be bounded by a small positive constant. This further confirms the potential of OFedED can match centralized performance.

Extensive experiments demonstrate that our OFedED achieves the highest accuracy across various benchmark datasets. In particular, it outperforms SOTA one-shot FL methods by 9.17% on MNIST ($\alpha = 0.05$), 3.97% on CIFAR-10 ($\alpha = 0.3$), and 4.42% on CIFAR-100 ($\alpha = 0.1$), highlighting its effectiveness even in serious heterogeneous settings. We also design experiments with various model architectures to verify OFedED's robustness. The results show that OFedED outperforms competitors on CIFAR-10 ($\alpha = 0.1$) by up to 11.9% with ResNet and by 8.53% with heterogeneous server-client models. This indicates significant performance advantages of our OFedED algorithm across various datasets and heterogeneity levels.

In summary, we make the following main contributions in this work:

- We propose OFedED, a one-shot FL algorithm which integrates dataset distillation with ensemble modeling to leverage distilled client data toward performance close to that by centralized training and robustness across heterogeneous data distributions.
- We propose layer-wise data distillation on clients to better capture client data distributions and also introduce a class-weighted ensemble model along with a class-center loss on the server to effectively utilize distilled client data, aimed at guiding the ensemble model in adapting to distributional differences.
- We provide a comprehensive theoretical analysis, showing that one-shot FL can achieve performance comparable to that by centralized training under mild assumptions on smoothness, convexity, and distributional distance.
- We conducted extensive experiments and demonstrated that OFedED significantly outperforms SOTA one-shot FL methods.

2 Related Work

Dataset distillation. Dataset distillation aims to generate compact synthetic datasets that retain the training utility of large real datasets. The foundational idea, proposed in [24], introduced gradient matching to minimize the difference in model gradients between real and synthetic data. This was extended in [34] through iterative condensation, refining the synthetic data via progressive optimization, and in [32] by incorporating differentiable Siamese augmentation to improve diversity and expressiveness. Trajectory-based methods further advanced the field. Matching Training Trajectories (MTT) [2] aligns the training dynamics of synthetic data with parameter sequences from real-data-trained models. To improve robustness and reduce hyperparameter sensitivity, [17] proposed Automatic Training Trajectories (ATT), which adaptively adjust trajectory lengths to accommodate parameter variation. More recently, distribution matching techniques have emerged. [33] aligns synthetic samples with the feature distribution of real data, while [23] improves upon this by matching layer-wise activations across the network. Together, these methods illustrate a progression from gradient-level alignment to more expressive trajectory and feature-space matching, enabling increasingly effective distillation.

Multi-shot FL with dataset distillation. Dataset distillation has been increasingly used in multi-shot FL to reduce communication costs and enhance generalization under non-i.i.d. client data. [5] proposed replacing model updates with synthetic data transmission, enabling lightweight communication but inheriting the limitations of conventional distillation in handling data imbalance and heterogeneity. To better approximate local objectives, FedDM [25] introduces a bi-level surrogate framework: each client fits a local surrogate to its private loss and generates synthetic data via distribution matching, while the server optimizes a global surrogate constructed from these client-side summaries. DYNAFED [20] takes a trajectory-centric approach, aligning model update paths across clients to capture global dynamics without requiring access to local data or surrogate models, thereby mitigating client drift more effectively.

One-shot FL with dataset distillation. In recent years, notable progress has been made in one-shot FL through dataset distillation. DOSFL [36] pioneered the use of synthetic data transmission instead of model parameters, enabling server-side training without access to raw client data. FedD3 [21] extended this idea by introducing an accuracy gain metric to balance performance and communication cost, though it remained constrained by earlier distillation

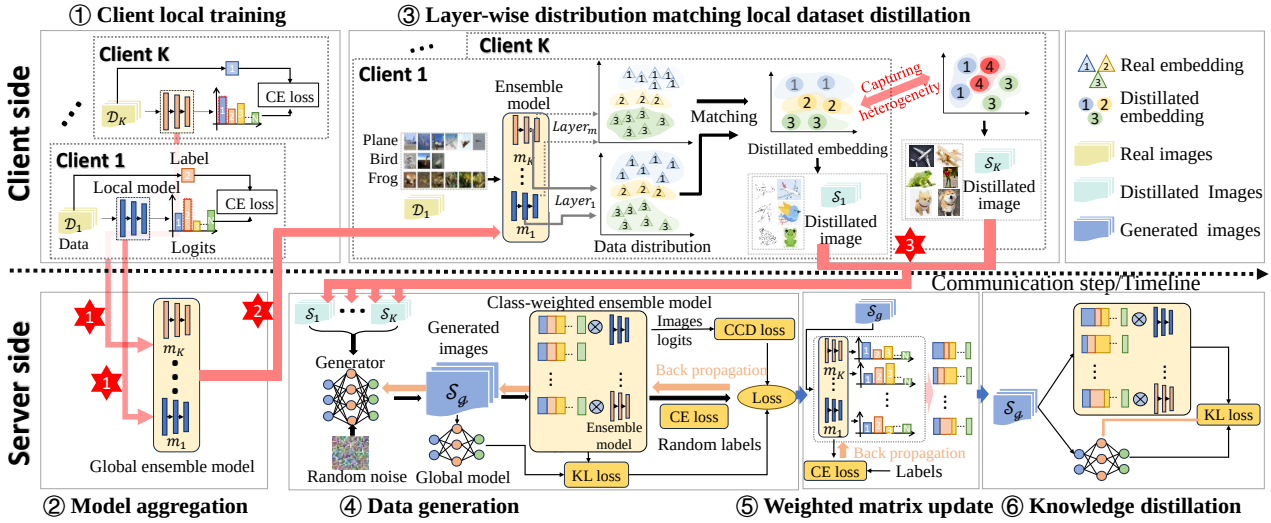


Figure 1: The overall framework involves clients training local models and sending them to the server for aggregation into an ensemble model. This model is redistributed to each client, which then uses layer-wise distribution matching to generate synthetic images and uploads them back to the server. The server leverages these images to guide a generator in creating additional data and trains the global model via knowledge distillation.

techniques. To address heterogeneity, Fed-CVAE [9] employed class-conditional generation using Conditional VAEs, but struggled to produce reliable images. DENSE [31] advanced this by training a synthetic generator with triple loss functions and distillation-based supervision to improve global performance. Building on this, Co-boosting [4] incorporated hard sample generation and ensemble learning to further improve accuracy. Despite these advances, generating high-fidelity synthetic data without client access remains a core challenge for one-shot FL. Recently, diffusion-based OSFL methods have emerged. FedDEO [26] uses compact client-trained descriptions to guide a server-side pre-trained diffusion model for generating client-specific synthetic data, while FedLMG [27] leverages uploaded local models—rather than foundation models—to guide generation, enabling adaptation to heterogeneous clients with minimal computation.

3 Proposed Method

In this section, we propose a more effective one-shot FL framework that can achieve performance close to that under centralized training, as stated in Theorem 1. Our approach, called OFedED, is built upon model ensemble and dataset distillation.

3.1 Framework Overview

To effectively extract information from the client-side training data, we employ layer-wise distribution matching on clients. On the server side, we retain a structure that is largely consistent with previous works [4, 31], while incorporating our improvements to improve model performance.

The overall framework is described in Figure 1 and Algorithm 1. First, each client trains a model using its own local data, and sends the trained model to the server. An ensemble model is then constructed by aggregating the update client models on the server, and download to clients. Next, clients perform layer-wise distribution matching to generate synthetic images, and upload them to the server in order to guide the generator for generating additional images there. Finally, a global model is trained with these generated images which has real data distribution via knowledge distillation [31]. Although our method involves two transmissions within this single round, the process constitutes a single interaction in the one-shot FL sense. Unlike multi-round FL, no repeated exchange of identical messages occurs.

3.2 Client Local Training

The first step of our method follows the traditional one-shot FL paradigm, where each client k trains a local model θ_k on its private dataset $\mathcal{D}_k$. This training is performed independently and only once, using standard empirical risk minimization.

Suppose we have K clients, each client has its own local data $\mathcal{D}_k$, where $\mathcal{D}_k = \left\{\left(x_k^i, y_k^i\right)\right\}_{i=1}^{|\mathcal{D}_k|}$. Clients' model uses their local data to train model parameters θ by solving the following problem:

$$\arg\min_{\theta} \mathcal{L}(\theta) = \sum_{k=1}^{K} p_k \mathcal{L}_k(\theta, \mathcal{D}_k),$$

where $p_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$ is the normalized weight for model combination, and

$$\mathcal{L}_k(\theta, \mathcal{D}_k) = \frac{1}{|\mathcal{D}_k|} \sum_{i=1}^{|\mathcal{D}_k|} \ell_k\left(f_k(x_k^i; \theta), y_k^i\right),$$

where $\mathcal{L}_k(\theta, \mathcal{D}_k)$ is client k's empirical loss, with f_k being its prediction function and $\ell_k(\cdot)$ its training loss, such as cross-entropy loss, on a single data sample. After completion of local training, each client transmits its trained model parameters to the server for aggregation.

Algorithm 1: Training Process of OFedED

Input: Number of clients K; client local models $\{f_k(\cdot)\}_{k=1}^K$; generator $G(\cdot)$ with parameters θ_G, learning rate η_G, and training rounds T_G; global model with parameters $\theta_{\mathcal{S}}$, learning rate $\eta_{\mathcal{S}}$, and training epochs T_s; ensemble weights $\mathbf{W}$ with finetune interval T_f; local training and distribution matching epochs T_l, T_{dm}.

Client-side:

LocalUpdate(k):
Initialize θ_k from global model or randomly;
for *epoch* $= 1, \ldots, T_l$ **do**
  for *batch* $(x_i, y_i) \in \mathcal{D}_k$ **do**
    Compute $\mathcal{L}_{CE} = CE(f_k(x_i; \theta_k), y_i)$;
    Update $\theta_k \leftarrow \theta_k - \eta_k \nabla_{\theta_k} \mathcal{L}_{CE}$;
return θ_k;
LDM(k, M_w):
Random initiaize synthetic images $\mathcal{S}_k$;
for *layer* $p = J, J-1, \ldots, 1$ **do**
  for *epoch* $= 1, \ldots, T_{dm}$ **do**
    Update $\mathcal{S}_k$ based on equation Eq. 2
return $\mathcal{S}_k$;

Process:
for *clients* $k = 1, 2, ..., K$ *in parallel* **do**
  $\theta_k \leftarrow$ **LocalUpdate**(k);
Aggregate ensemble model M_w using Eq. (1);
for *client* $k = 1, 2, ..., K$ *in parallel* **do**
  $\mathcal{S}_k \leftarrow$ **LDM**(k, M_w);
Aggregate synthetic dataset $\mathcal{S} = \bigcup_{k=1}^K \mathcal{S}_k$;
Initialize $\theta_G, \theta_S, \mathbf{W}$;
for *epoch* $= 1, \ldots, T_s$ **do**
  Sample batch $\{z_i, y_i\}_{i=1}^b$;
  for $j = 1, \ldots, T_G$ **do**
    Generate $\{x_i\}_{i=1}^b$ using $G(z_i; \theta_G)$;
    Update θ_G: $\theta_G \leftarrow \theta_G - \eta_G \nabla_{\theta_G} \mathcal{L}_{CCD}(x_i, \theta_G)$;
  Update $\theta_{\mathcal{S}}$: $\theta_{\mathcal{S}} \leftarrow \theta_{\mathcal{S}} - \eta_{\mathcal{S}} \nabla_{\theta_{\mathcal{S}}} \mathcal{L}_{\mathrm{KL}}(x_i, \theta_{\mathcal{S}})$;
  if *acc is best* **then**
    Update $\mathbf{W}$: $\mathbf{W}_{t+1} =$ Normalize$(\mathbf{W}^{t-1} - \mu \cdot \mathrm{sign}(\nabla_{\mathbf{W}} \mathcal{L}_{ens}(\mathbf{W})))$;
  Train M_w on $\mathcal{S}$ for T_f epochs:

$$\theta \leftarrow \theta - \eta \nabla_\theta \sum_{(x_i, y_i) \in \mathcal{S}} \mathcal{L}(M_{\mathbf{W}}(x_i; \theta), y_i)$$

3.3 Model Aggregation

After receiving the model parameters from local clients, we construct an ensemble model. Suppose that there are K models from clients, denoted by $f_k(\mathbf{x})$, where $k = 1, 2, ..., K$ and $\mathbf{x}$ are the input feature data. Let:

$$M_w(\mathbf{x}; \{\theta_k\}_{k=1}^K) \triangleq \sum_{k=1}^K w_k f_k(\mathbf{x}; \theta_k), \tag{1}$$

where $f_k(\mathbf{x}; \theta_k)$ denotes the prediction function that outputs the logits (i.e., outputs of the last fully connected layer) of $\mathbf{x}$ given model parameters θ_k, and w_k denotes the weight assigned to each local model $f_k(\mathbf{x}; \theta_k)$ in the global ensemble aggregation of Eq. 1, set as $\frac{1}{K}$ for uniform aggregation, which can form vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_K]$. In the following, we use M_w to denote the weighted ensemble model, and $M_w(\mathbf{x})$ to represent the output logits of the weighted ensemble given $\mathbf{x}$, which corresponds to $M_w(\mathbf{x}; \{\theta_k\}_{k=1}^K)$.

3.4 Local Layer-Wise Distribution Matching

To incorporate global knowledge into local synthetic data generation, we use the ensemble model as a global reference during distribution matching. While each local model reflects its client's specific data distribution, the ensemble captures shared representations across the clients, thus offering a global perspective.

Rather than relying on traditional distribution matching alone, we introduce a more effective approach inspired by FedDM [25], which aligns the synthetic data with the feature representations of the ensemble model. Our key insight is that since the ensemble model approximates centralized knowledge, using it to guide the embedding of synthetic images allows each client to generate data that generalize better across noni.i.d. scenarios.

In this step, we perform client-side distribution matching by minimizing the discrepancy between real and synthetic data embeddings. For a client k with dataset $\mathcal{D}_k$, we use Maximum Mean Discrepancy (MMD) [33] to estimate distributional differences in a lower-dimensional feature space. The objective is to minimize the deviation between the embeddings of the original data $\mathcal{D}_k$ and the synthetic data $\mathcal{S}_k$:

$$\underset{\mathcal{S}_k}{\operatorname{argmin}} \left\| \mathbb{E}[h_\theta(\mathcal{D}_k)] - \mathbb{E}[h_\theta(\mathcal{S}_k)] \right\|,$$

where h_θ is an embedding function that maps data to the feature space, and $\mathcal{S}_k$ is the synthetic dataset for client k.

3.4.1 Generalization error ball with adaptive radius. Inspired by FedDM [25], we introduce a *generalization error ball* with an adaptive radius R_t to guide the selection of h_θ. The radius R_t is computed as the maximum norm of the difference between model updates on any client's synthetic dataset $\mathcal{S}_k$ and the aggregated dataset $\mathcal{S}$ on the server:

$$R_t = \sup_k \left\| \theta_t^{\mathcal{S}_k} - \theta_t^{\mathcal{S}} \right\|,$$

where $\theta_t^{\mathcal{S}_k} = \theta_{t-1} - \eta \nabla \mathcal{L}(\theta_{t-1}; \mathcal{S}_k)$ and $\theta_t^{\mathcal{S}} = \theta_{t-1} - \eta \nabla \mathcal{L}(\theta_{t-1}; \mathcal{S})$ are the model parameters after one gradient step on $\mathcal{S}_k$ and $\mathcal{S}$, respectively.

The generalization error ball $\mathcal{B}(\theta_t; R_t)$, centered on the server's model parameters θ_t with radius R_t, defines a range for sampling embedding functions h_θ. These embeddings guide the distribution matching between the original and distilled data:

$$\underset{\mathcal{S}_k}{\operatorname{argmin}} \sup_{\theta \in \mathcal{B}(\theta_t; R_t)} \left\| \mathbb{E}[h_\theta(\mathcal{D}_k)] - \mathbb{E}[h_\theta(\mathcal{S}_k)] \right\|.$$

3.4.2 Empirical MMD and backward layer-wise alignment with ensemble model. To bridge the gap between local data distributions and global knowledge, we match the feature distributions of each class c between $\mathcal{D}_k$ and $\mathcal{S}_k$ using each client's model:

$$\mathcal{L}_{\mathrm{DM}}^k(\mathcal{S}_k; \mathcal{D}_k) = \sum_{\theta \in M_w} \sum_{c=1}^C \left\| \frac{1}{n_k^c} \sum_{i=1}^{n_k^c} h_\theta(x_k^{c,i}) - \frac{1}{s_k^c} \sum_{i=1}^{s_k^c} h_\theta(\tilde{\mathbf{x}}_k^{c,i}) \right\|, \tag{2}$$

where $\mathbf{x}_k^{c,i}$ and $\tilde{\mathbf{x}}_k^{c,i}$ are the i-th samples of class c in $\mathcal{D}_k$ and $\mathcal{S}_k$, respectively, θ specifies which client model participates in the distribution matching process, n_k^c and s_k^c are the samples number of $\mathcal{D}_k$ and $\mathcal{S}_k$, and C denotes the number of classes in real dataset $\mathcal{D}$.

To further exploit data features, we implement a *backward layer-wise feature alignment*. Unlike previous methods that perform distribution matching only at the last layer (before the classifier), we extend it to intermediate layers. The embedding function h_θ comprises multiple layers $\{v_1, v_2, \ldots, v_J\}$. We sequentially align feature distributions starting from the last layer to the first, allowing deeper layers to guide earlier ones in capturing richer feature distributions. For a layer $p \in \{J, J-1, \ldots, 1\}$, the alignment objective is:

$$\mathcal{L}_{\mathrm{DM}}^{k,p}(\mathcal{S}_k; \mathcal{D}_k) = \sum_{q=p}^{J} \mathcal{L}_{\mathrm{MMD}}^{k,v_q}(\mathcal{S}_k; \mathcal{D}_k).$$

This backward alignment ensures that deeper layers guide earlier ones, improving the extraction of feature distributions from each client's data.

3.5 Server-Side Data Generation

After receiving the class-wise synthetic datasets $\{\mathcal{S}_k\}_{k=1}^K$ from all clients, we aggregate them to form a global synthetic dataset $\mathcal{S}$. However, the number of synthetic samples is limited due to privacy and communication concerns. This dataset alone may not suffice to train a robust student model through knowledge distillation.

Following the approach introduced in prior works such as DENSE [31], we incorporate an additional generator-based image expansion step. In this step, a generator $G(\cdot)$ with parameter θ_G is trained to generate images based on M_w, which effectively enhances the training set. The following loss is used to make sure the generated dataset can be classified correctly:

$$\mathcal{L}_{\mathrm{CE}}(\hat{\mathbf{x}}_g; \theta_G) \triangleq CE(M_w(\hat{\mathbf{x}}_g), y),$$

where $CE(\cdot,\cdot)$ denotes the cross-entropy function, $\hat{\mathbf{x}}_g$ represents the image generated from standard Gaussian noise z for each epoch, and $\widetilde{\mathcal{S}} = \{(\hat{\mathbf{x}}_g, y)\}$ is the set of all the generated images, produced by a generator on the server, paired with labels that are sampled uniformly at random.

In addition, to further enhance the effectiveness of data generation, we introduce some key components into our framework. First, to address the challenge of class imbalance commonly observed in federated settings, we propose a class-weighted ensemble model that adjusts the influence of each client based on class-wise contributions. Second, to better transfer local data characteristics from the uploaded synthetic images to the newly generated ones, we introduce a class-center distance loss, which encourages the generator to preserve class-level feature semantics during image generation.

Class-weighted ensemble model. The weighted ensemble model was proposed in [4] to better capture the heterogeneity of data across different clients. However, experiments revealed that this weighted ensemble model did not improve the performance of the ensemble model and even caused a slight decline in some cases (see Figure 2(a) for more details). Based on this observation, we propose a refined class-weighted ensemble model:

$$\min_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{w}}(\boldsymbol{w}) \triangleq \frac{1}{|\widetilde{\mathcal{S}}|} \sum_{\{\mathbf{x}_g, y_g\} \in \widetilde{\mathcal{S}}} \ell_{CE}\Big(\sum_{k=1}^{K}\sum_{c=1}^{C} w_{k,c} f_k(\mathbf{x}_g; \boldsymbol{\theta}_k), y_g\Big),$$

where $w_{k,c}$ indicates the k-th model and c-th class weight. Our class-weighted ensemble model assigns a distinct weight for each model–class pair. This design upgrades the weight vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_K]$ to a matrix $\boldsymbol{W} \in \mathbb{R}^{K\times C}$, enabling the ensemble to better handle class imbalance by allowing per-class adjustment of model contributions. The weight list for class c is:

$$\boldsymbol{w} = [w_{1,c}, w_{2,c}, \ldots, w_{K,c}].$$

For each sample $(\mathbf{x}_g, y_g)$, the weight column $[w_{1,y_g}, w_{2,y_g}, \ldots, w_{K,y_g}]$ is selected to compute the weighted model prediction for class y_g. We use $M_{\mathbf{W}}$ to represent the class-weighted ensemble model.

We update w after processing each batch of synthesized data by adjusting it in the direction of the gradient with a fixed step size μ [6] as below, which is consistent with Co-boosting [4].

$$\mathbf{W}^t = \mathrm{Normalize}(\mathbf{W}^{t-1} - \mu \cdot \mathrm{sign}(\nabla_{\mathbf{W}} \mathcal{L}_{ens}(\mathbf{W}))),$$

where Normalize means that each w_k is confined to be in $[0, 1]$. $\nabla_{\mathbf{W}} \mathcal{L}_{ens}(\mathbf{W})$ is the loss computed over the ensemble model with respect to the weight matrix $\mathbf{W}$. By introducing a detailed weight matrix that assigns class-specific weights to each model, the class-weighted ensemble can better address class imbalance and data heterogeneity. Moreover, this ensemble serves as an alternative representation of global distributional knowledge, enabling more effective integration of information across clients, thereby improving the model performance. (see figure 2(a)).

Class-center distance loss. In [31], the performance of the generator is limited by the absence of client data during generation [16]. To address this issue, we incorporate class-center loss into the design of our loss functions, with the aim of ensuring that the generated images produced by the generator on the server exhibit distributions similar to those of the original data[1].

On the server, we generate data from random noise at each epoch, under the guidance of a loss function designed based on synthetic data to learn the true data distribution from clients.

Let $\mathcal{X}_c$ be the synthetic dataset of all samples of class c, and $\mu_c = \frac{1}{|\mathcal{X}_c|} \sum_{\mathbf{x}\in\mathcal{X}_c} \mathbf{x}$, where μ_c is the class c's center. Define

$$\mathcal{L}_{CCD}(\mathbf{x}_g; \theta_G) = \sum_{\mathbf{x}_g \in \widetilde{\mathcal{S}}} \frac{||M_{\mathbf{W}}(\mathbf{x}_g) - M_{\mathbf{W}}(\mu_c)||_2}{\sum_{c=1}^{C} ||M_{\mathbf{W}}(\mathbf{x}_g) - M_{\mathbf{W}}(\mu_c)||_2},$$

where $\widetilde{\mathcal{S}}$ is the generated dataset. By aligning the synthetic data with the class centers in the feature space, the CCD loss ensures that the generated data maintain intra-class coherence and inter-class separability, which is critical for the subsequent distillation process.

Other Strategies. To better exploit the utility of synthetic images, we use them to adaptively guide the weight updates of the ensemble model. At each iteration t, the ensemble weights $\mathbf{W}_t$ are updated based on the predictive precision of the model over synthetic data. Formally, the update rule is defined as:

$$\mathbf{W}_{t+1} = \begin{cases} \mathrm{Normalize}(\mathbf{W}^{t-1} - \mu \cdot \mathrm{sign}(\nabla_{\mathbf{W}} \mathcal{L}_{ens}(\mathbf{W}))), & \textit{acc is best}, \\ \mathbf{W}_t, & \textit{otherwise}. \end{cases}$$

By doing this, the ensemble model incrementally learns to prioritize information that enhances its performance on these synthetic examples.

[1] We observed in our experiments that the Batch Normalization (BN) loss has minimal impact on the performance of the generator, and thus is not included in our approach.

Since each model within the ensemble is only exposed to data from a single client, we mitigate data heterogeneity by training the ensemble model on synthetic images. Due to the limited size of the synthetic dataset, this training is completed in a small number T_f of epochs, e.g., $T_f = 10$. Given the synthetic dataset $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{n}$, the training process minimizes:

$$\arg\min_{\theta} \mathcal{L}_{\text{ens}}(\theta) = \sum_{(x_i,y_i)\in\mathcal{S}} \mathcal{L}(M_{\mathbf{W}}(x_i;\theta), y_i).$$

These two strategies ensure that the ensemble model not only adapts dynamically to synthetic examples but also learns a more comprehensive representation of the global data distribution.

3.6 Server-side Knowledge Distillation

The server then uses both generated images and updated ensemble model $M_{\mathbf{W}}$ to optimize parameter $\theta_{\mathcal{S}}$ of the global model by minimizing:

$$\mathcal{L}_{\text{KL}}(\hat{\mathbf{x}}_g;\theta_{\mathcal{S}}) \triangleq KL(M_{\mathbf{W}}(\hat{\mathbf{x}}_g), f_S(\hat{\mathbf{x}}_g;\theta_{\mathcal{S}})),$$

where $KL(\cdot,\cdot)$ denotes the Kullback-Leibler (KL) divergence loss, and $f_{\mathcal{S}}(\hat{\mathbf{x}}_g;\theta_{\mathcal{S}})$ represents the global model.

This centralized process enables the server to integrate the shared knowledge distilled from all clients. In our framework, the use of a well-trained ensemble model, combined with high-quality generated images, greatly enhances this process, leading to a more accurate and generalizable global model that better captures the underlying data distribution across heterogeneous clients.

4 Theoretical Insight

We now conduct a comprehensive theoretical analysis of one-shot FL to answer the crucial question raised in Section 1. Meanwhile, we also consider aspects such as differential privacy (DP), whose theoretical guarantees remain under-explored. Before presenting our main theoretical results, we list the necessary assumptions and definition below.

Assumption 1 (Smooth and convex). *The local objective function $\mathcal{L}_k(\cdot)$ and the embedding functions $h_\theta(\cdot)$ for any client $k = 1,\ldots,K$ are L-smooth and μ-strongly convex.*

Assumption 2 (Bounded gradient). *For each client, the norm of the gradient is bounded:*

$$\|\nabla\mathcal{L}_k(\theta)\| \le \beta, \quad \forall\theta\in\mathbb{R}^d, \quad \forall k\in\{1,2,\ldots,K\},$$

where β is a positive constant.

Definition 1 (Bounded distributional distance). *For any embedding model with parameters randomly sampled from a generalization error ball centered at θ_t with radius R, the synthetic dataset $\mathcal{S}_k$ and given dataset $\mathcal{D}_k$ on the k-th client are assigned to the feature space by the embedding model. The expected feature distance between them is bounded by a small constant $\sigma_d > 0$ for any sampled θ:*

$$\begin{aligned} &\|\mathbb{E}[h_\theta(\mathcal{D}_k)] - \mathbb{E}[h_\theta(\mathcal{S}_k)]\| \le \sigma_d, \\ &\forall\theta\in\mathcal{B}(\theta_t,R), \quad \forall k\in\{1,2,\ldots,K\} \end{aligned} \tag{3}$$

Assumption 1 guarantees that the local objective functions exhibit smoothness and strong convexity, which are critical properties for ensuring stable convergence during global optimization. Assumption 2 enforces an upper bound on the gradient norms, ensuring that the optimization process remains controlled and prevents divergence. Definition 1 describes a desired property about distribution matching: the synthetic images should approximate the original data distribution, ensuring they faithfully represent client data with minimal distortion.

4.1 Effectiveness Guarantee

Under the above assumption, we can conclude the following theorem:

Theorem 1 (Effectiveness). *Let $\mathcal{L}(\theta^*_{\mathcal{S}};\mathcal{D})$ denote the loss of the model with optimal parameters $\theta^*_{\mathcal{S}}$ trained on the aggregated distilled dataset $\mathcal{S}$, evaluated on the original dataset $\mathcal{D}$, and $\theta^*_{\mathcal{D}}$ present the optimal parameters trained on $\mathcal{D}$. With distributional distance bound $\sigma_d = \sqrt{2\mu^2\epsilon'/L}$ for small positive constant ϵ', it holds that:*

$$\mathbb{E}\Big[\big\|\mathcal{L}(\theta^*_{\mathcal{S}};\mathcal{D}) - \mathcal{L}(\theta^*_{\mathcal{D}};\mathcal{D})\big\|\Big] \le \varepsilon, \tag{4}$$

where $\varepsilon = \frac{L\sigma_d^2}{2\mu^2} \le \epsilon'$ is a sufficiently small positive constant.

Theorem 1 provides the theoretical basis for one-shot FL by showing that training on the aggregated distilled dataset leads to a loss value closely approximating that of centralized training on the full dataset. This result validates the use of distilled data as a reliable proxy for raw data, establishing that a single-round client-server interaction can suffice to recover near-centralized performance. In particular, it holds regardless of the heterogeneity of the client data.

Proof sketch. To establish the effectiveness of our method, we show that the global model trained on the distilled dataset $\mathcal{S}$ generalizes well to the original dataset $\mathcal{D}$. First, we prove that the client-side model θ_t converges to the optimal parameter $\theta^*_{\mathcal{S}_k}$ under standard assumptions of L-smoothness and μ-strong convexity. If gradient descent with learning rate $\eta < \frac{2}{L}$, the convergence is linear:

$$\mathcal{L}(\theta_t) - \mathcal{L}(\theta^*_{\mathcal{S}_k}) \le (1-\eta\mu)^t \cdot \Big(\mathcal{L}(\theta_0) - \mathcal{L}(\theta^*_{\mathcal{S}_k})\Big).$$

Second, to minimize the gap between θ^*_s (trained on $\mathcal{S}$) and θ^*_d (trained on $\mathcal{D}$), we formulate a loss alignment objective:

$$\min_{\theta} \mathbb{E}_{x\sim\mathcal{D}_k}\|h_\theta(x) - h_{\text{ens}}(x)\|^2,$$

where h_{ens} is the ensemble model that better reflects the global data distribution than any single client. Under bounded gradient discrepancy $\|\nabla\mathcal{L}(\theta;\mathcal{D}) - \nabla\mathcal{L}(\theta;\mathcal{S})\| \le \sigma_d$, and strong convexity, we obtain:

$$\|\theta^*_s - \theta^*_d\| \le \frac{\sigma_d}{\mu},$$

which leads to the final generalization bound:

$$\mathbb{E}\left[\|\mathcal{L}(\theta^*_s;\mathcal{D}) - \mathcal{L}(\theta^*_d;\mathcal{D})\|\right] \le \frac{L\sigma_d^2}{2\mu^2}.$$

By setting $\sigma_d = \sqrt{\frac{2\mu^2\epsilon'}{L}}$, we conclude:

$$\mathbb{E}\left[\|\mathcal{L}(\theta^*_s;\mathcal{D}) - \mathcal{L}(\theta^*_d;\mathcal{D})\|\right] \le \epsilon'.$$

This shows that training on the aggregated distilled dataset provides a close approximation to the optimal global solution.

4.2 Privacy Protection

To ensure that our algorithm is privacy-preserving, we introduce differential privacy by adding Gaussian noise to both model parameters and synthetic client images. The following theorem shows that our algorithm satisfies the (ϵ, δ)-differential privacy.

Theorem 2 (Privacy). *There exist constants c_1 and c_2 such that given sampling probability q, total number of update steps of DP-SGD T, learning rate η and gradient clipping norm τ, for any $0 < \epsilon < c_1 q^2 T$, OFedED is (ϵ, δ)-differential private by respectively adding Gaussian noise to synthetic images $\mathcal{S}_k$ and model parameters θ_k with the following noise standard deviations:*

$$\sigma_s \geq 2\eta\tau\sqrt{\frac{2\log(1.25/\delta)}{\epsilon}}, \quad \sigma_\theta \geq c_2 L \frac{q\sqrt{T\log(1/\delta)}}{\epsilon}. \tag{5}$$

Differential privacy guarantee. In OFedED, to ensure (ϵ, δ)-differential privacy (DP), we analyze two components: (1) the training of the client model and (2) the generation of synthetic data $\mathcal{S}_k$ by distribution matching.

Client model training: Each client applies per-sample gradient clipping with threshold τ and adds Gaussian noise with variance σ^2 during local training. The weight update becomes:

$$\theta_k \leftarrow \theta_k - \eta \cdot \frac{1}{|\mathcal{D}_k|} \sum_{i \in \mathcal{D}_k} \text{clip}_\tau(\nabla \ell(f_k(x_i), y_i)) + \mathcal{N}(0, \sigma^2 \mathbf{I}).$$

The ℓ_2 sensitivity of the clipped gradient is bounded by 2τ, and thus, under the Gaussian mechanism, the update satisfies (ϵ, δ)-DP if:

$$\sigma \geq 2\eta\tau\sqrt{\frac{2\log(1.25/\delta)}{\epsilon}}.$$

Synthetic data generation: We train the synthetic set $\mathcal{S}_k$ using DP-SGD with batch sampling ratio q over T steps. Following [1], the training process satisfies (ϵ, δ)-DP when the Gaussian noise σ_s satisfies:

$$\sigma_s \geq c_2 L \frac{q\sqrt{T\log(1/\delta)}}{\epsilon},$$

where c_2 is a constant determined by the model's Lipschitz bound and gradient clipping norm. Since model updates and synthetic data generation follow independent computational flows, and client contributions are independent, we apply the parallel composition theorem to prove that the entire OFedED procedure satisfies (ϵ, δ)-DP.

5 Experiments

To evaluate the performance of our proposed OFedED algorithm, we conduct a series of experiments in the setting given below.

5.1 Experimental Setting

Datasets. We conduct experiments on four real-world image datasets commonly used in the FL literature: MNIST [12], CIFAR-10 [11], CIFAR-100 [11], and SVHN [19]. To simulate statistical heterogeneity [35], we generate disjoint non-IID client training datasets using Dirichlet distribution [14, 30]. Specifically, we sample a nonnegative vector $p_k \sim Dir(\alpha)$ with component p_k^c representing the proportion of data for class c allocated to client k. The parameter α determines the degree of statistical imbalance, with smaller values of α leading to more uneven label distributions across clients [9].

Baseline model. We compare OFedED with FedAvg [18], DENSE [31], Co-boosting [4], and FedDF [16] for experiments. The ensemble-based FL learning method DAFL [3], ADI [29], the model-centric method MAEcho [22], and the diffusion-based method FedLMG [27] are also included for comparison.

Configuration. We use the same setting for clients' local training and the auxiliary generator as Co-boosting. Our experiments are carried out on the basis of 10 clients. Furthermore, we performed experiments with α values in $\{0.3, 0.1, 0.05\}$, where smaller values indicate greater heterogeneity, and used a convolutional network of 3 layers for CNN. For the 3-layer CNN, each of the first two layers is trained for 100 iterations in distribution matching, while the final layer is trained for 200 iterations. All distilled data are initialized randomly. We use $T_s = 200$ global training epochs on the server and $T_G = 30$ training rounds for the generator. Unless otherwise specified, all experiments are conducted without adding Gaussian noise and on non-i.i.d. data to simulate real-world scenarios.

5.2 Results and Analysis

The experimental results are reported in Table 1, which provides a detailed comparison of FL methods at varying levels of heterogeneity (α) and datasets. It is evident that our method, OFedED, consistently outperforms baselines, particularly in scenarios of high client heterogeneity, i.e., low values α. For example, on MNIST with $\alpha = 0.05$, OFedED achieves 85.38% accuracy, significantly higher than the second best performing method, FedLMG (76.21%). Similar trends are observed in CIFAR-10 and CIFAR-100, where OFedED exceeds baselines by a substantial margin. In particular, on CIFAR-10 with $\alpha = 0.3$, OFedED achieves 65.28%, compared to Co-Boosting's 57.19%, and on CIFAR-100 with $\alpha = 0.1$, OFedED reaches 24.41%, slightly outperforming FedDF's 24.07%. In the SVHN dataset, OFedED performs well when $\alpha = 0.05$, which is slightly higher than DENSE and Co-boosting with $\alpha = 0.3$ and $\alpha = 0.1$. We notice that the performance of Co-Boosting declines, compared to that of the original paper, where the output of Co-Boosting had taken advantage of synthetic data from multiple runs of the algorithm[2].

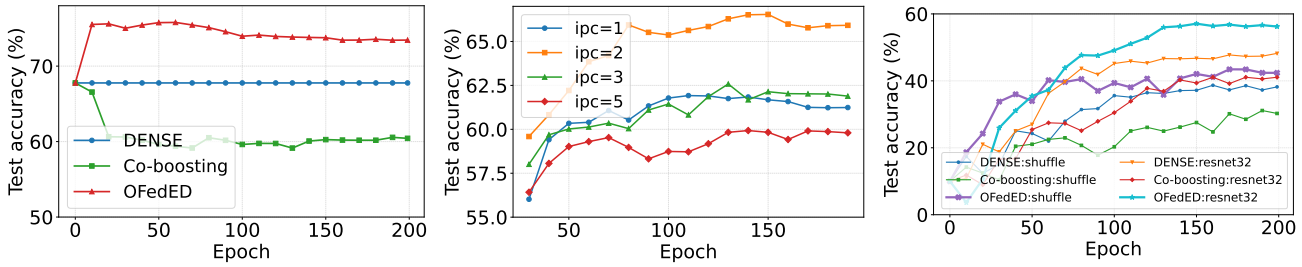
Optimization stability. Figure 2(a) shows that Co-Boosting struggles to achieve substantial accuracy gains in some cases, demonstrating a stagnation in performance after initial epochs. This means that the addition of model weights to the ensemble model, although theoretically effective in addressing the issue of client heterogeneity, adversely impacts the performance of the ensemble model in practice. In contrast, our method, OFedED, consistently improves the ensemble model's performance. The accuracy increases steadily, surpassing both DENSE and Co-Boosting.

Heterogeneity robustness. To further evaluate the robustness of OFedED, we conduct experiments on the CIFAR-10 dataset using the ResNet18 model. Table 2 shows the result of our experiments, where "local" means the ensemble model as the unweighted aggregation of locally trained models, i.e., Eq. (1) with $w_k = \frac{1}{K}$. We observe that in highly heterogeneous scenarios, our method significantly improves model performance. In contrast, in low-heterogeneity settings, all methods perform notably worse than

[2]This trick is disabled across all algorithms in our experiments.

Dataset	α	FedAvg	FedDF	F-DAFL	F-ADI	MAEcho	FedLMG	DENSE	Co-Boosting	OFedED
MNIST	0.05	55.38±0.98	58.05±1.51	29.51±3.30	67.97±0.23	26.85±0.80	76.21±2.13	40.52±0.10	64.09±0.15	**85.38±0.09**
	0.1	54.95±0.82	73.88±2.34	65.50±3.28	73.56±2.97	73.70±0.25	91.91±1.90	64.23±0.18	79.74±0.22	**95.81±0.11**
	0.3	61.96±0.82	90±0.78	79.98±2.25	89.63±0.68	86.36±0.21	95.95±2.08	97.12±0.08	95.85±0.14	**98.54±0.05**
CIFAR-10	0.05	28.18±2.51	27.66±0.33	23.51±0.80	23.71±0.71	14.67±1.01	36.60±2.30	24.24±3.10	33.15±1.22	**39.28±0.64**
	0.1	35.95±1.80	31.05±0.10	30.06±2.93	35.64±0.52	42.60±0.53	47.19±2.65	38.52±2.05	48.82±0.71	**51.16±1.43**
	0.3	41.90±2.37	44.83±1.44	34.96±1.87	54.38±0.57	50.14±0.50	64.54±0.66	56.31±1.46	57.19±1.19	**65.28±1.53**
CIFAR-100	0.05	5.28±0.92	**18.34±0.26**	5.76±0.05	5.38±0.19	2.87±1.20	13.72±1.89	11.67±1.13	12.11±1.68	17.95±0.56
	0.1	6.89±1.70	24.07±0.16	8.08±0.90	9.08±0.21	3.80±1.05	19.61±0.86	15.25±2.76	16.20±1.59	**24.41±0.48**
	0.3	8.89±2.08	28.1±0.15	15.66±3.18	16.87±0.27	3.80±1.50	26.65±0.67	23.96±1.93	22.82±1.07	**28.49±0.24**
SVHN	0.05	33.44±1.67	44.56±2.24	20.06±0.60	26.04±0.53	37.84±0.60	53.76±0.97	23.45±0.25	48.32±2.21	**72.97±1.04**
	0.1	39.29±1.96	57.13±3.38	54.02±2.32	47.81±0.30	66.23±0.36	64.03±3.22	58.70±0.30	75.10±1.77	**75.77±1.23**
	0.3	62.30±3.12	77.11±1.75	83.35±1.55	80.26±0.34	66.64±0.33	81.94±2.65	84.28±0.36	83.39±0.46	**84.98±0.39**

Table 1: Comparison of FL methods on various datasets with different levels of heterogeneity (α).



(a) Ensemble model accuracy **(b) Comparison on different IPC** **(c) Comparison on different models**

Figure 2: Comparisons of ensemble models, different images per class (IPC), and different knowledge distillation models. Experiments conducted on cifar10, α=0.3.

local. This is because almost all clients have access to all classes' training images and can thus train models close to centralized training, whereas our method relies on synthetic images, which still exhibit a gap compared to the real images.

Impact of IPC. Figure 2(b) shows that when the IPC (Images Per Class) is set to 2, meaning that each client generates 200 synthetic images (number of classes * number of client users * IPC) in the cifar10 data set, the model achieves its best performance. However, when the IPC increases or decreases, the performance of the model is affected to some extent, which is consistent with previous studies [8, 13]. This might be because traditional dataset distillation methods tend to overly capture common patterns by optimizing for dominant, simpler features, leading to synthetic datasets that overlook rare, complex patterns essential for comprehensive model generalization.

Model Robustness. To assess the robustness of our method across model architectures, we conduct additional experiments on CIFAR-10 using different student models. As shown in Figure 2(c), our method consistently outperforms DENSE and Co-Boosting under knowledge distillation settings. Specifically, the teacher model is fixed as a CNN, while the student model varies between ResNet32 and ShuffleNet, demonstrating that our approach maintains strong performance across diverse architectures.

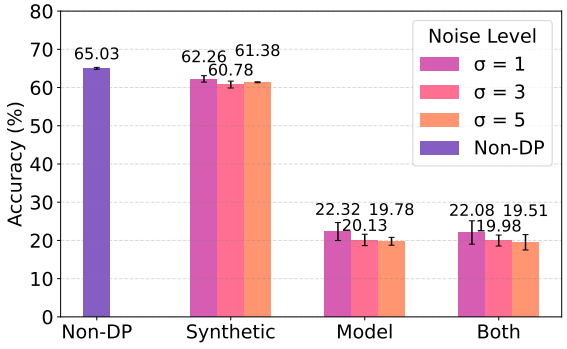


Figure 3: Comparison on differential privacy

Privacy guarantee. Figure 3 demonstrates the impact of adding Gaussian noise with varying levels ($\sigma = 1, 3, 5$) to synthetic images, model gradients, or both during training under differential privacy constraints. The results indicate that all scenarios experience a performance decline to varying degrees, with the severity dependent on the noise level and its application. Adding noise to synthetic images moderately reduces performance, as it impairs the quality of the data used for training. Since we primarily use synthetic images for calculating the class center loss with generated images, this can partially mitigate the impact of noise. When noise is applied

α	Local	FedDF	F-ADI	F-DAFL	DENSE	Co-boosting	OFedED
0.05	21.50	19.20±0.57	17.41±0.54	12.72±1.17	18.14±0.27	31.40±0.36	**42.43±0.63**
0.1	49.27	22.59±1.31	20.50±1.40	34.33±0.39	39.94±0.41	46.07±0.30	**61.17±1.18**
0.3	**74.80**	29.55±0.75	24.08±1.39	60.94±0.40	59.65±0.02	65.66±0.40	63.03±0.41

Table 2: Performance comparison across different methods for varying α. Local refers to ensemble model in Eq. (1) with $w_k = \frac{1}{K}$.

to model gradients, the performance drops significantly, reflecting the critical role of accurate gradients in effective optimization. The combination of noise in both synthetic images and gradients results in the most significant decline, showcasing the compounded effect of reduced data quality and optimization accuracy.

Loss	Ens	Others	CIFAR-10	SVHN
			57.19±1.19	83.39±0.46
✓			60.04±0.61	83.58±0.52
	✓		63.09±0.44	83.55±0.10
		✓	57.80±0.06	83.34±0.20
✓	✓		65.13±0.25	84.53±0.79
✓		✓	62.89±0.64	83.63±0.22
	✓	✓	64.88±0.32	84.16±0.61
✓	✓	✓	**65.28±1.53**	**84.98±0.39**

Table 3: Ablations on different components of our method. "Loss" for class-center distance loss. "Ens" for class-weighted ensemble model. "Others" for synthetic images guide and finetune strategy.

Efficiency. To validate the efficiency of our method, we compare OFedED with FedAvg under the same performance. As shown in Table 4, this comparison clearly highlights the communication advantage of one-shot FL. Specifically, FedAvg requires 10 rounds of communication, each involving 100 local training epochs, leading to a total client-side cost of $10 \times 100 \times 0.792 = 792$ GFLOPs. In contrast, OFedED completes the training in a single round, with 100 local training epochs and 400 lightweight distribution matching epochs, resulting in only $100 \times 0.792 + 400 \times 0.132 = 132$ GFLOPs. This demonstrates that OFedED not only inherits the communication efficiency of one-shot FL, but also reduces the overall client computation.

We further compare OFedED with other one-shot FL baselines such as DENSE and Co-Boosting in Table 5. Compared to large models such as ResNet-18, the communication cost of transmitting synthetic samples (e.g., only 22 MB at IPC = 10) in OFedED is significantly lower. In addition to this efficiency, OFedED achieves comparable performance while reducing server-side computation by more than 38% in some cases, demonstrating its ability to effectively capture and leverage client-side knowledge.

Ablation study. Table 3 presents an ablation study on CIFAR-10 and SVHN, $\alpha = 0.3$, which analyzes the impact of key components in our method. Our goal is to transfer knowledge from local client data to the global model in one-shot FL setting, and each component contributes uniquely to this pipeline. The ensemble model proves critical: removing it leads to a significant drop in accuracy, confirming its effectiveness in aggregating diverse client knowledge

Method	TCC (GFLOPs)	US-CNN (MB)	US-Res (MB)
FedAvg	792	120	4470
OFedED	132	34	469

Table 4: Computation and communication comparison with FedAvg. TCC: Total Client Computation, US-CNN/Res: Upload Size for CNN/ResNet.

Method	TCC (GFLOPs)	US-CNN (MB)	US-Res (MB)	SC (GFLOPs)	Red. (%)
DENSE	79.2	12	447	42180	38.3
Co-Boosting	79.2	12	447	43450	40.1
OFedED	132	34	469	26040	–

Table 5: Comparison of one-shot FL methods. TCC: Total Client Computation, US: Upload Size, SC: Server Computation, Red.: Reduction over baseline.

while addressing class imbalance. The class-center distance loss ensures that distilled representations effectively propagate to larger server-generated images, thereby enhancing generalization. These enriched images, when used in the guide-and-finetune stage, further refine global predictions and improve feature alignment. Overall, all variants of our method outperform Co-Boosting, showcasing the robustness of our method in effectively leveraging client-side information while mitigating distributional shifts.

6 Conclusion

Our work is driven by the challenge of reaching data-centralized training performance under single communication round constraint of one-shot FL, especially with serious client heterogeneity. To see this, we proposed OFedED, a one-shot FL framework that combines local dataset distillation with class-weighted ensemble modeling. Extensive experiments on MNIST, CIFAR-10, CIFAR-100, and SVHN demonstrate that OFedED consistently outperforms existing one-shot FL methods in heterogeneous or different model structure settings. Theoretical analysis in this work supports the validity of training on aggregated distilled data as a surrogate for centralized training. In addition, OFedED maintains competitive performance under differential privacy constraints, highlighting its robustness and practicality for real-world application.

Limitations and future work . We notice that OFedED incurs an overall increase in computational costs, highlighting the need for future work on efficiency optimization. Moreover, although our current evaluation focuses on CNN-based models, extending OFedED to large-scale architectures such as Transformers presents a promising research direction, especially given their growing relevance in federated applications.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. 2022. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4750–4759.

[3] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. 2019. Data-free learning of student networks. In *Proceedings of the IEEE/CVF international conference on computer vision*. 3514–3522.

[4] Rong Dai, Yonggang Zhang, Ang Li, Tongliang Liu, Xun Yang, and Bo Han. 2024. Enhancing One-Shot Federated Learning Through Data and Ensemble Co-Boosting. *arXiv preprint arXiv:2402.15070* (2024).

[5] Jack Goetz and Ambuj Tewari. 2020. Federated learning via synthetic data. *arXiv preprint arXiv:2008.04489* (2020).

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[7] Neel Guha, Ameet Talwalkar, and Virginia Smith. 2019. One-shot federated learning. *arXiv preprint arXiv:1902.11175* (2019).

[8] Ziyao Guo, Kai Wang, George Cazenavette, Hui Li, Kaipeng Zhang, and Yang You. 2023. Towards lossless dataset distillation via difficulty-aligned trajectory matching. *arXiv preprint arXiv:2310.05773* (2023).

[9] Clare Elizabeth Heinbaugh, Emilio Luz-Ricca, and Huajie Shao. 2023. Data-free one-shot federated learning under very high statistical heterogeneity. In *The Eleventh International Conference on Learning Representations*.

[10] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning* 14, 1–2 (2021), 1–210.

[11] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[13] Yongmin Lee and Hye Won Chung. 2024. SelMatch: Effectively scaling up dataset distillation via selection-based initialization and partial updates by trajectory matching. In *Forty-first International Conference on Machine Learning*.

[14] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 965–978.

[15] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine* 37, 3 (2020), 50–60.

[16] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems* 33 (2020), 2351–2363.

[17] Dai Liu, Jindong Gu, Hu Cao, Carsten Trinitis, and Martin Schulz. 2024. Dataset distillation by automatic training trajectories. *arXiv preprint arXiv:2407.14245* (2024).

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[19] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, Vol. 2011. Granada, 4.

[20] Renjie Pi, Weizhong Zhang, Yueqi Xie, Jiahui Gao, Xiaoyu Wang, Sunghun Kim, and Qifeng Chen. 2023. Dynafed: Tackling client data heterogeneity with global dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12177–12186.

[21] Rui Song, Dai Liu, Dave Zhenyu Chen, Andreas Festag, Carsten Trinitis, Martin Schulz, and Alois Knoll. 2023. Federated learning via decentralized dataset distillation in resource-constrained edge environments. In *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–10.

[22] Shangchao Su, Bin Li, and Xiangyang Xue. 2023. One-shot federated learning without server-side training. *Neural Networks* 164 (2023), 203–215.

[23] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. 2022. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12196–12205.

[24] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. 2018. Dataset distillation. *arXiv preprint arXiv:1811.10959* (2018).

[25] Yuanhao Xiong, Ruochen Wang, Minhao Cheng, Felix Yu, and Cho-Jui Hsieh. 2023. Feddm: Iterative distribution matching for communication-efficient federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16323–16332.

[26] Mingzhao Yang, Shangchao Su, Bin Li, and Xiangyang Xue. 2024. Feddeo: Description-enhanced one-shot federated learning with diffusion models. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 6666–6675.

[27] Mingzhao Yang, Shangchao Su, Bin Li, and Xiangyang Xue. 2025. One-Shot Heterogeneous Federated Learning with Local Model-Guided Diffusion Models. arXiv:2311.08870 [cs.CV] https://arxiv.org/abs/2311.08870

[28] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. 2021. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16337–16346.

[29] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8715–8724.

[30] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*. PMLR, 7252–7261.

[31] Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. 2022. Dense: Data-free one-shot federated learning. *Advances in Neural Information Processing Systems* 35 (2022), 21414–21428.

[32] Bo Zhao and Hakan Bilen. 2021. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*. PMLR, 12674–12685.

[33] Bo Zhao and Hakan Bilen. 2023. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 6514–6523.

[34] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929* (2020).

[35] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).

[36] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. 2020. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999* (2020).

GenAI Usage Disclosure

In this work, we used the following generative AI tools:

- **Grammarly**: We used Grammarly (an AI-powered writing assistant) to check grammar and improve readability in this paper. All content was ultimately reviewed and approved by the authors.
- **ChatGPT-4o**:
  - **code**: During the coding stage, ChatGPT-4o was used to assist in debugging by providing suggestions for correcting coding errors. All implementations were ultimately written, tested, and verified by the author.
  - **writing**: ChatGPT-4o was used exclusively during the final stage of manuscript preparation to assist in polishing the English descriptions within the experimental analysis section. All generated suggestions were carefully reviewed, edited, and adapted by the authors to ensure technical accuracy and align with the original intent. Other sections of the paper were not generated or rewritten using AI tools.

No AI tool was used to generate core content or conclusions.