

MODEL-DECOUPLING-BASED FEDERATED LEARNING WITH CONSISTENCY VIA KNOWLEDGE DISTILLATION USING CONDITIONAL GENERATOR

Anonymous authors

Paper under double-blind review

ABSTRACT

Federated Learning (FL) is gaining popularity as a distributed learning framework that only shares model parameters or gradient updates and keeps private data locally. However, FL is at risk of privacy leakage caused by privacy inference attacks. And most existing privacy-preserving mechanisms in FL conflict with achieving high performance and efficiency. Therefore, we propose FedMD-CG, a novel FL method with highly competitive performance and high-level privacy preservation, which decouples each client’s local model into a feature extractor and a classifier, and utilizes a conditional generator instead of the feature extractor to perform server-side model aggregation. To ensure the consistency of local generators and classifiers, FedMD-CG leverages knowledge distillation to train local models and generators at both the latent feature level and the logit level. Also, we construct additional classification losses and design new diversity losses to enhance client-side training. FedMD-CG is robust to data heterogeneity and does not require training extra discriminators (like cGAN). We conduct extensive experiments on various image classification tasks to validate the superiority of FedMD-CG. We provide our code here: <https://anonymous.4open.science/r/FedMD-CG-34E2/>.

1 INTRODUCTION

Many modern real-world applications involve data being dispersed across clients located in different physical locations, such as autonomous driving (Li et al., 2021), medical image analysis (Liu et al., 2021), and IoT (Nguyen et al., 2021). However, various regulation, privacy and security concerns often make it impractical or even impossible to collect these dispersed data into one location for traditional centralized learning (Voigt & Von dem Bussche, 2017). To ameliorate these limitations, Federated Learning (FL) (Li et al., 2020a) has been proposed to enable each client to train a local model using only its own data and share its model parameters or gradient updates with a central server periodically to ensure that each client’s raw data does not leak from the local device.

Despite the success, the vanilla FL (e.g., FedAvg (McMahan et al., 2017) and its variants (Li et al., 2020b; Karimireddy et al., 2020; Luo et al., 2023)) based on sharing complete local model parameters or gradient updates are extremely vulnerable to inference attacks. Several prior arts empirically demonstrate that it is feasible to reconstruct victim clients’ private data from trained and publicly shared parameters and gradient updates (Zhu et al., 2019; Geiping et al., 2020; Haim et al., 2022). Therefore, a variety of efforts have been devoted to reducing the risk of privacy leakage in FL, including homomorphic encryption (HE) (Ma et al., 2022; Zhang et al., 2022b), differential privacy (DP) (Geyer et al., 2017; Cheng et al., 2022) and model decoupling (MD) (Arivazhagan et al., 2019; Liang et al., 2020). In particular, HE achieves high-level privacy protection at the expense of extremely high computation and communication costs, which restricts its deployment in bandwidth-limited and large-model scenarios. DP preserves privacy by perturbing server-side model aggregation or client-side local model update, but this deteriorates the performance of the FL methods. See Appendix A for more related work.

In this paper, we mainly focus on MD, which requires each client to decompose the local model into the base and top layers, and send one of them to the server to reduce the risk of privacy leakage, yet this inevitably results in performance degradation and even privacy exposure. Note that we regard

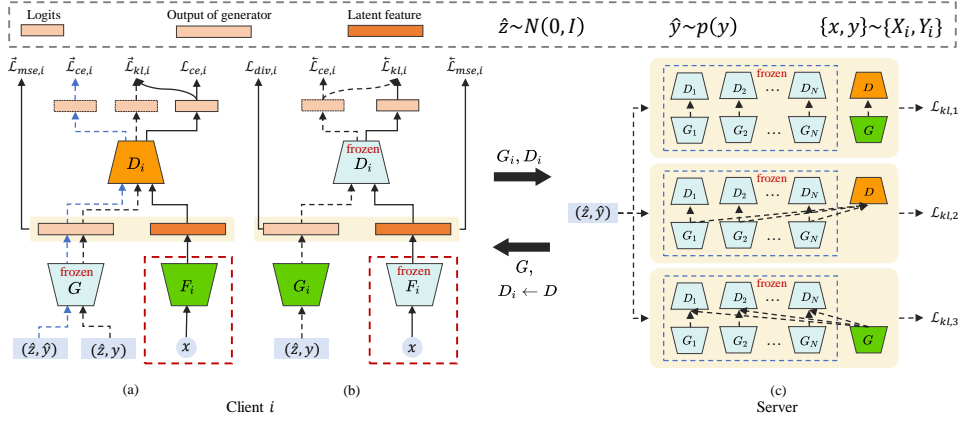


Figure 1: Illustration of FedMD-CG: (a) The local model update distills the experience from the global generator G for augmenting the generalization performance of the local model $[F_i, D_i]$. (b) The local generator update utilizes the trained local model $[F_i, D_i]$ to guide the local generator G_i to mimic latent feature space. Note that G is not involved in client-side training. (c) The server-side data-free KD aggregation takes a crossed manner to achieve as much knowledge transfer as possible. Best viewed in color. Zoom in for details.

the base layers (top layers) as a feature extractor (classifier). Recently, FedCG (Wu et al., 2021) combines FL and conditional generative adversarial network (cGAN) (Mirza & Osindero, 2014) to adversarially train a conditional generator to replace the feature extractor for each client, aiming at achieving competitive performance while maintaining high-level privacy protection. However, we revisit it and observe that the following pitfalls may occur in client-side training. First, knowledge transfer modality at the latent feature level may not be sufficient. Second, additional discriminators need to be trained to satisfy the adversarial training of cGAN. Third, the trained local generator may not match the local classifier, terming their inconsistency. Note that the latent feature denotes the output of the feature extractor.

To this end, we propose a new **F**ederated Learning with **MD** method (dubbed as FedMD-CG), which resorts to knowledge distillation (KD) to train a local **cond**itional **gen**erator for each client to replace the local feature extractor. To be more specific, FedMD-CG works on how to efficiently train the local model and generator on the client side. To achieve this, FedMD-CG utilizes KD to perform knowledge transfer from the global generator to the local model and from the local model to the local generator at the latent feature level and the logit level. Meanwhile, we additionally construct two classification losses to enhance the local model update and the local generator update, respectively. In addition, we devise two novel diversity constraints to ensure the diversity of the local generator outputs. On the server side, FedMD-CG performs aggregation of local generators and classifiers in a crossed data-free KD fashion. The overview of our method is illustrated in Fig. 1.

In a nutshell, the main contributions of this work are as follows: 1) We formulate a novel privacy-preserving FL method FedMD-CG to achieve better generalization performance, via leveraging KD to efficiently transfer knowledge from the global generator to the local model and then from the local model to the local generator. 2) To enhance client-side training, we construct additional classification losses and tailor new diversity constraints. Our method ensures the consistency between trained local generators and classifiers, thereby being robust to data heterogeneity. 3) FedMD-CG performs aggregation in a crossed data-free KD fashion on the server side in order to extract as much knowledge as possible from the local generators and classifiers. 4) We conduct extensive experiments to show that FedMD-CG is highly competitive compared with state-of-the-art baselines w.r.t test performance, convergence speed and privacy protection.

2 PROPOSED METHOD

In this section, we detail the proposed method FedMD-CG. We first define the problem setup and notations for clarity. And then the core modules of FedMD-CG are presented. Moreover, we present pseudocode for FedMD-CG in Appendix B.

Problem Setup and Notations. In this work, we consider supervised federated learning (FL) setting, i.e., the general problem of multi-class classification. To be specific, we focus on the centralized setup that consists of a central server and N clients owning private labeled datasets $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^N$ with $|\mathbf{X}_i| = n_i$, where $\mathbf{X}_i = \{\mathbf{x}_i^b\}_{b=1}^{n_i}$ follows data distribution \mathcal{D}_i over input feature space \mathcal{X}_i , i.e., $\mathbf{x}_i^b \sim \mathcal{D}_i$, and $\mathbf{Y}_i = \{y_i^b\}_{b=1}^{n_i} \subset \mathbb{R}$ denotes the ground-truth labels of \mathbf{X}_i . Notably, we consider the same input feature space, yet the sample distribution may be different among clients, that is, data heterogeneity caused by label distribution skewness (i.e., $\mathcal{X}_i = \mathcal{X}_j$ and $\mathcal{D}_i \neq \mathcal{D}_j, \forall i \neq j, i, j \in [N]$). Besides, each client i holds a local model parameterized by $\theta^i = [\theta_F^i; \theta_D^i]$ comprising two components: the base layers (feature extractor) $F_i : \mathcal{X}_i \rightarrow \mathcal{F}$ parameterized by θ_F^i , and the top layers (classifier) $D_i : \mathcal{F} \rightarrow \mathbb{R}^c$ parameterized by θ_D^i , where $\mathcal{F} \subset \mathbb{R}^p$ is the output space of feature extractor with p dimension, i.e., the latent feature space, and c is the number of classes. For extracting knowledge from clients without accessing any extra data, each client equips with a conditional generator $G : \mathcal{Z} \times \mathcal{Y} \rightarrow \mathbb{R}^p$ parameterized by \mathbf{w} , where $\mathcal{Z} \subset \mathbb{R}^q$ is the multivariate standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathcal{Y} \subset \mathbb{R}^c$ indicates the one-hot vector space of the ground-truth label. We use bold $\mathbf{y} \in \mathcal{Y}$ to denote one-hot vector corresponding to class $y \in \mathbb{R}$. Hereafter, we refer to conditional generator as generator.

2.1 CLIENT-SIDE TWO-STAGE DISTILLATION

The training process for each client i involves two stages: augmenting the local model update with global generator (see Fig. 1 (a)), and guiding the local generator update with trained local model (see Fig. 1 (b)).

Augmenting the local model update with global generator. In the classical local model update, client i leverages the following classification loss to optimize the local model $\theta^i = [\theta_F^i; \theta_D^i]$:

$$\mathcal{L}_{ce,i} = CE(\rho(D_i(F_i(\mathbf{x}))), \mathbf{y}), \quad (1)$$

where ρ is the softmax function and CE is the cross-entropy function. However, $\mathcal{L}_{ce,i}$ has no access to global knowledge in our work, which is embedded in the global generator. To transfer the knowledge of the global generator to the local model efficiently, we construct the following two losses based on KD:

$$\vec{\mathcal{L}}_{mse,i} = \|F_i(\mathbf{x}) - G(\hat{\mathbf{z}}, \mathbf{y})\|^2, \vec{\mathcal{L}}_{kl,i} = KL(\rho(D_i(F_i(\mathbf{x}))) \| \rho(D_i(G(\hat{\mathbf{z}}, \mathbf{y})))), \quad (2)$$

where $\hat{\mathbf{z}}$ is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. $\|\cdot\|^2$ and KL are L_2 -norm function and Kullback-Leibler function, respectively. Specifically, $\vec{\mathcal{L}}_{mse,i}$ utilizes L_2 -norm function to enforce the output of feature extractor $F_i(\mathbf{x})$ to approximate that of global generator $G(\hat{\mathbf{z}}, \mathbf{y})$. After that, client i feeds both $F_i(\mathbf{x})$ and $G(\hat{\mathbf{z}}, \mathbf{y})$ into the classifier to get $D_i(F_i(\mathbf{x}))$ and $D_i(G(\hat{\mathbf{z}}, \mathbf{y}))$. Further, $\vec{\mathcal{L}}_{kl,i}$ harnesses Kullback-Leibler function to make $D_i(F_i(\mathbf{x}))$ close to $D_i(G(\hat{\mathbf{z}}, \mathbf{y}))$.

To further augment the local model update, client i resamples a batch of noisy data to feed the generator and classifier sequentially, and minimizes the following classification loss:

$$\vec{\mathcal{L}}_{ce,i} = CE(\rho(D_i(G(\hat{\mathbf{z}}, \hat{\mathbf{y}}))), \hat{\mathbf{y}}), \quad (3)$$

where $\hat{\mathbf{y}} \sim p(y) \propto \sum_{i \in [N]} n_i^y$, n_i^y denotes the number of samples w.r.t class y on the i -th client.

Combining $\mathcal{L}_{ce,i}$, $\vec{\mathcal{L}}_{ce,i}$, $\vec{\mathcal{L}}_{mse,i}$ and $\vec{\mathcal{L}}_{kl,i}$, the overall objective of the local model update can be formalized as follows:

$$\min_{\theta_F^i, \theta_D^i} \mathbb{E}_{\hat{\mathbf{z}}, \hat{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), p(y)} [\mathcal{L}_{ce,i} + \lambda_1 \vec{\mathcal{L}}_{ce,i} + \lambda_2 \vec{\mathcal{L}}_{mse,i} + \lambda_3 \vec{\mathcal{L}}_{kl,i}], \quad (4)$$

where λ_1 , λ_2 and λ_3 are tunable hyperparameters for balancing different loss items.

Guiding the local generator update with trained local model. After the local model update, we maintain a local generator in client i to extract the knowledge of the trained local model without accessing its private data. Note that the global generator does not replace the local generator in our work to learn the knowledge of the trained local model.

Similar to the manner of augmenting local model update, we utilize KD to construct losses $\overleftarrow{\mathcal{L}}_{kl,i}$ and $\overleftarrow{\mathcal{L}}_{mse,i}$ to transfer the knowledge of the local model to the local generator. $\overleftarrow{\mathcal{L}}_{kl,i}$ and $\overleftarrow{\mathcal{L}}_{mse,i}$ take the

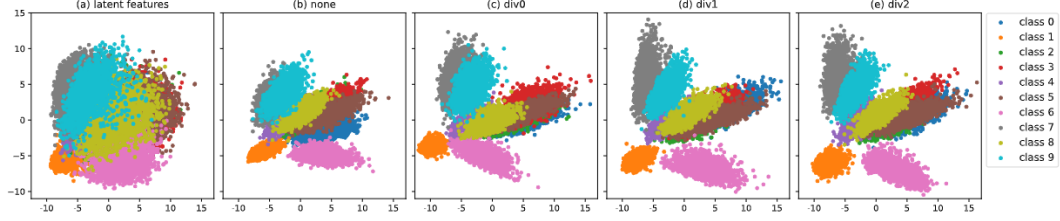


Figure 2: Visualization for output of the generator: The toy example first trains a LeNet (LeCun et al., 1998) as teacher model (T) using the training set of MNIST (LeCun et al., 1998). Then the test set of MNIST is fed to T to get the latent features. And the dimensions of the latent features are reduced by principal component analysis (PCA) (Halko et al., 2011). (a) shows the latent features distribution of T after PCA dimension reduction. Next, we let T guide the training of the generator according to Eq. (9). Similarly, we utilize PCA to perform dimension reduction for the output of the generator. (b) visualizes the output distribution of the generator without diversity constraint. (c), (d) and (e) visualize the output distribution of the generator with \mathcal{L}_{div}^0 , \mathcal{L}_{div}^1 and \mathcal{L}_{div}^2 , respectively.

forms:

$$\overleftarrow{\mathcal{L}}_{mse,i} = \|G_i(\hat{\mathbf{z}}, \mathbf{y}) - F_i(\mathbf{x})\|^2, \overleftarrow{\mathcal{L}}_{kl,i} = KL(\rho(D_i(G_i(\hat{\mathbf{z}}, \mathbf{y}))) \| \rho(D_i(F_i(\mathbf{x}))))). \quad (5)$$

To ensure the fidelity of the output of the local generator G_i , G_i is expected to fit the input space of the local classifier for better knowledge extraction from the local model. Therefore, client i takes the following classification loss $\overleftarrow{\mathcal{L}}_{ce,i}$ to enforce G_i to yield higher prediction on class y :

$$\overleftarrow{\mathcal{L}}_{ce,i} = CE(\rho(D_i(G_i(\hat{\mathbf{z}}, \mathbf{y}))), y). \quad (6)$$

However, if we only optimize $\overleftarrow{\mathcal{L}}_{kl,i}$, $\overleftarrow{\mathcal{L}}_{mse,i}$ and $\overleftarrow{\mathcal{L}}_{ce,i}$ for G_i , it is likely to generate similar outputs for each class with little diversity, which can cause the model collapse of the local generator. To tackle this limitation, the constraint \mathcal{L}_{div}^0 has been added to enhance the output diversity of the generator as follows (Yoo et al., 2019; Zhu et al., 2021; Zhang et al., 2022c):

$$\mathcal{L}_{div}^0 = e^{\frac{1}{B^2} \sum_{j,k \in [B]} (-\|\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_k\|_2 * \|\hat{\mathbf{z}}_j - \hat{\mathbf{z}}_k\|_2)}, \quad (7)$$

where B denotes the batch size and $\hat{\mathbf{f}}_{j/k} = G_i(\hat{\mathbf{z}}_{j/k}, \mathbf{y}_{j/k})$. This constraint treats the noise pair distance $\|\hat{\mathbf{z}}_j - \hat{\mathbf{z}}_k\|_2$ as a weight, which is then multiplied by the corresponding output pair distance $\|\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_k\|_2$ in each batch B , thus imposing more weights on the output pairs whose corresponding noise pairs are more distant. It can be found that this weighting scheme of \mathcal{L}_{div}^0 is label-agnostic. In other words, the weight differences between intra- and inter-class output pairs are not considered in \mathcal{L}_{div}^0 , which may lead to inter-class output pairs being close but intra-class output pairs being distant, thus adversely affecting the performance of G_i . To rectify this issue, we propose two novel diversity constraints, \mathcal{L}_{div}^1 and \mathcal{L}_{div}^2 . In terms of \mathcal{L}_{div}^1 , we simply replace $\|\hat{\mathbf{z}}_j - \hat{\mathbf{z}}_k\|_2$ of Eq. (7) with $\|\hat{\mathbf{z}}_j^y - \hat{\mathbf{z}}_k^y\|_2$, where $\hat{\mathbf{z}}_j^y = [\hat{\mathbf{z}}_j; \mathbf{y}_j]$. Further, we formulate \mathcal{L}_{div}^2 in the following form:

$$\mathcal{L}_{div}^2 = e^{\frac{1}{B^2} \sum_{j,k \in [B]} (-\|\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_k\|_2 * \|\hat{\mathbf{z}}_j - \hat{\mathbf{z}}_k\|_2 * e^{\|\hat{\mathbf{y}}_j - \hat{\mathbf{y}}_k\|_1})}. \quad (8)$$

Compared to \mathcal{L}_{div}^0 , \mathcal{L}_{div}^1 and \mathcal{L}_{div}^2 further differentiate the weights of the generator's intra- and inter-class output pair distances, with more weights applied to the inter-class output pair distances. For brevity, we uniformly denote \mathcal{L}_{div}^0 , \mathcal{L}_{div}^1 and \mathcal{L}_{div}^2 as \mathcal{L}_{div} unless otherwise noted. In Fig. 2, we provide a toy example showing the output distribution of the generator without diversity constraints as well as with different diversity constraints.

We combine $\overleftarrow{\mathcal{L}}_{kl,i}$, $\overleftarrow{\mathcal{L}}_{mse,i}$, $\overleftarrow{\mathcal{L}}_{ce,i}$ and \mathcal{L}_{div} to yield the overall objective of the local generator update for client i is shown below:

$$\min_{\mathbf{w}_i} \mathbb{E}_{\hat{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} [\overleftarrow{\mathcal{L}}_{kl,i} + \lambda_4 \overleftarrow{\mathcal{L}}_{mse,i} + \lambda_5 \overleftarrow{\mathcal{L}}_{ce,i} + \lambda_6 \mathcal{L}_{div,i}], \quad (9)$$

where λ_4 , λ_5 and λ_6 are non-negative hyperparameters. $\mathcal{L}_{div,i}$ denotes the diversity constraint of client i .

2.2 SERVER-SIDE CROSSED DISTILLATION AGGREGATION

After gathering local generators and classifiers uploaded by clients, the server aggregates them as a preliminary global generator and classifier via weighted averaging. However, straightforward average aggregation may counteract the local knowledge from clients. To alleviate this issue, we train the preliminary global generator and classifier via crossed data-free KD to distill as much knowledge as possible from the local generators and classifiers. Fig. 1 (c) shows the distillation schema on the server, where the overall distillation objective consists of three parts: $\mathcal{L}_{kl,1}$, $\mathcal{L}_{kl,2}$ and $\mathcal{L}_{kl,3}$.

Specifically, the server first samples $(\hat{\mathbf{z}}, \hat{\mathbf{y}})$, and feeds it to the local generators $\{G_i\}_{i \in [N]}$ and the global generator G , where $\hat{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\hat{\mathbf{y}} \sim p(y) \propto \sum_{i \in [N]} n_i^y$. Their outputs are then fed into the corresponding classifiers to compute the loss $\mathcal{L}_{kl,1}$:

$$\mathcal{L}_{kl,1} = \sum_{i \in [N]} \tau_{i,\hat{\mathbf{y}}} KL(\rho_g \| \rho_i), \quad (10)$$

where $\rho_g = \rho(D(G(\hat{\mathbf{z}}, \hat{\mathbf{y}})))$, $\rho_i = \rho(D_i(G_i(\hat{\mathbf{z}}, \hat{\mathbf{y}})))$, and $\tau_{i,\hat{\mathbf{y}}} = n_i^{\hat{\mathbf{y}}} / \sum_{j \in [N]} n_j^{\hat{\mathbf{y}}}$. $\mathcal{L}_{kl,1}$ ensures that the ρ_g from the global classifier approximates $\{\rho_i\}_{i \in [N]}$ from the local classifiers. However, simply distilling knowledge by minimizing $\mathcal{L}_{kl,1}$ could be insufficient, since D_i fits $G_i(\hat{\mathbf{z}}, \hat{\mathbf{y}})$ (via optimizing Eq. (9)) but may not fit $G(\hat{\mathbf{z}}, \hat{\mathbf{y}})$ and $G_i(\hat{\mathbf{z}}, \hat{\mathbf{y}})$ fits D_i but may not fit D , such that only partial knowledge from clients can be extracted. Therefore, to address these limitations, we introduce a crossover strategy and formulate two losses $\mathcal{L}_{kl,2}$ and $\mathcal{L}_{kl,3}$ as:

$$\mathcal{L}_{kl,2} = \sum_{i \in [N]} \tau_{i,\hat{\mathbf{y}}} KL(\rho_{ig} \| \rho_i), \mathcal{L}_{kl,3} = \sum_{i \in [N]} \tau_{i,\hat{\mathbf{y}}} KL(\rho_{gi} \| \rho_i), \quad (11)$$

where $\rho_{ig} = \rho(D(G_i(\hat{\mathbf{z}}, \hat{\mathbf{y}})))$ and $\rho_{gi} = \rho(D_i(G(\hat{\mathbf{z}}, \hat{\mathbf{y}})))$.

Further, $\mathcal{L}_{kl,1}$, $\mathcal{L}_{kl,2}$ and $\mathcal{L}_{kl,3}$ form the following overall distillation objective on the server side:

$$\min_{\mathbf{w}, \theta_D} \mathbb{E}_{\hat{\mathbf{z}}, \hat{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), p(y)} [\mathcal{L}_{kl,1} + \mathcal{L}_{kl,2} + \mathcal{L}_{kl,3}]. \quad (12)$$

2.3 DISCUSSION

Privacy. FedMD-CG trains a local generator on each client for replacing the local feature extractor (LFE) by simulating the output vector space of LFE, i.e., the latent feature space, rather than the distribution space of private data. In other words, the local generator captures only high-level feature patterns of the local model, which are incomprehensible to human beings. Also, FedMD-CG requires each client to share its classifier. In our work, the classifier is in the top layers (i.e., fully connected layers) with a high degree of abstraction. As verified by (Yosinski et al., 2014), the lower layer features are more general and higher layer features have larger specificity. This suggests that different inputs to the model can result in the same top-layer activations, making it difficult to reconstruct the original data with the classifier (Wang, 2021). Therefore, FedMD-CG can reduce the risk of privacy leakage, and has the same level of privacy protection as FedCG.

Consistency and Computing cost. FedMD-CG performs client-side knowledge transfer at the latent feature level and the logit level, thus extracting knowledge embedded in the global generator and local model more directly and efficiently than FedCG. Meanwhile, our method guarantees the consistency of the local generator and classifier trained by each client, which may not be satisfied in FedCG. To put it differently, FedMD-CG requires the local generator to generate pseudo-features that the local classifier can significantly distinguish in order to make the generator output more fidelity. According to our experiments, the consistency of FedMD-CG ensures high-quality aggregation on the server side and robustness to data heterogeneity. In addition, FedMD-CG does not employ an additional discriminator to adversarially train the local generator under the cGAN framework independently of the local classifier like FedCG, which reduces the client’s computing cost.

Table 1: Test performance (%) comparison between FedMD-CG and baselines over different datasets. Note that $L.acc$ and $G.acc$ denote *local test accuracy* and *global test accuracy*, respectively.

Alg.s	EMNIST				FMNIST				CIFAR-10			
	$\omega = 1.0$		$\omega = 0.1$		$\omega = 1.0$		$\omega = 0.1$		$\omega = 10.0$		$\omega = 1.0$	
	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
FedAvg	96.29±0.06	96.83±0.07	85.65±2.33	95.06±0.46	80.99±0.81	84.77±0.30	59.29±3.19	78.91±2.12	51.68±0.53	54.18±0.52	42.84±2.03	52.39±0.58
LT	90.48±1.46	95.17±0.70	40.77±2.34	62.08±5.11	74.19±2.92	80.32±1.02	37.71±2.99	56.34±10.42	49.83±0.88	48.99±1.35	40.92±2.25	37.92±2.24
FedPer	91.96±1.20	96.45±0.10	41.08±2.40	76.90±2.34	75.83±2.42	82.94±1.11	37.39±3.17	61.88±9.80	50.72±0.63	53.87±0.58	41.80±2.15	49.83±1.66
LG-FedAvg	94.01±0.53	96.22±0.19	46.29±3.39	86.48±1.75	77.03±1.94	82.55±0.46	38.89±3.18	66.35±6.65	50.11±0.80	51.80±0.67	41.49±2.56	44.59±1.88
FedGen	95.62±0.38	97.64±0.17	51.29±4.01	87.69±2.50	77.88±3.10	83.81±1.95	41.96±3.40	68.05±3.96	52.94±2.38	48.49±3.13	38.13±4.89	40.85±3.87
FedCG	96.06±0.33	97.70±0.16	49.91±3.83	87.66±2.08	74.92±2.11	81.74±0.81	34.97±2.55	54.61±2.67	39.39±5.23	37.06±4.35	30.44±3.30	26.79±2.82
FedMD-CG	95.45±0.25	97.18±0.17	54.45±3.56	87.87±1.64	79.00±1.43	84.47±0.38	42.55±3.68	71.09±1.01	54.82±0.79	55.18±1.75	46.30±2.24	47.56±2.21

3 EXPERIMENTS

3.1 IMPLEMENTATION SETTINGS

Datasets. We perform our experiments on three public datasets EMNIST (Cohen et al., 2017), Fashion-MNIST (Xiao et al., 2017) (FMNIST in short in this paper), and CIFAR-10 (Krizhevsky et al., 2009). Following existing works (Zhang et al., 2022c; Acar et al., 2021; Zhu et al., 2021), we use Dirichlet process $Dp(\omega)$ to strictly partition the training set of each dataset across clients. Notably, a smaller ω corresponds to higher data heterogeneity. We set $\omega \in \{0.1, 1.0, 10.0\}$ in our experiments.

Backbone Architectures and Baselines. Throughout all our experiments, we deploy LeNet-5 (LeCun et al., 1998) as the backbone network with two convolutional layers (i.e., feature extractor) and three fully connected layers (i.e., classifier). Similarly, we employ three fully connected layers with BatchNorm as the generator for each client and adjust its output dimension to match that of the corresponding feature extractor. We select five FL methods most relevant to our work as baselines for comparison, including FedAvg (McMahan et al., 2017), FedPer (Arivazhagan et al., 2019), LG-FedAvg (Liang et al., 2020), FedGen¹ (Zhu et al., 2021) and FedCG (Wu et al., 2021). Moreover, we consider the baseline that trains a local model for each client, without any sharing. We call it Local Training (LT for short). For fairness, FedGen shares clients’ classifiers with the server. In particular, we treat the client’s classifier whose output dimension is set to 1 as the discriminator of cGAN in FedCG.

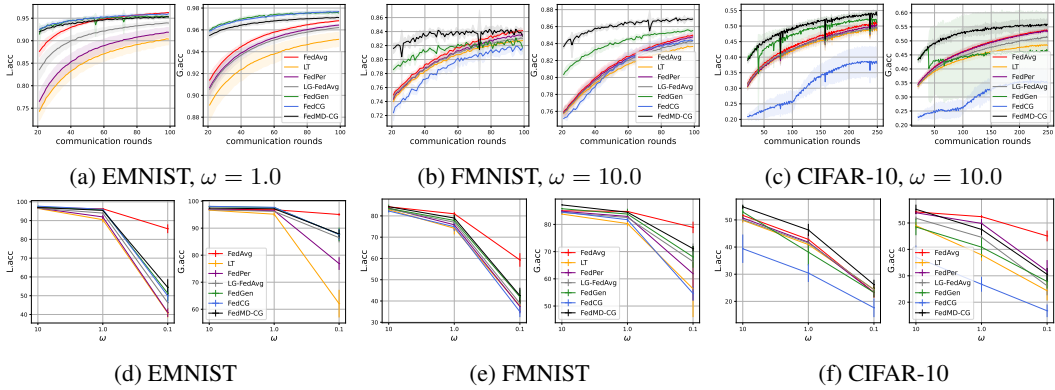


Figure 3: (a)-(c) are learning curves selected from FedMD-CG as well as baselines over different datasets. (d)-(f) show test performance (%) w.r.t data heterogeneity over each dataset.

Evaluation Metrics. We use the test set of each dataset to evaluate the test performance of different FL methods. 1) *Local test accuracy*. We randomly and evenly distribute the test set to each client and harness the test set on each client to verify the performance of local models. 2) *Global test accuracy*. We construct a virtual global model to evaluate the global performance of different FL methods via utilizing the original test set. As with FedAvg, this virtual global model is obtained by uploading all local models to the server for weighted average. 3) *Peak signal-to-noise ratio (PSNR)*. Consistent

¹In this paper, we consider FedGen with partial parameter sharing.

with FedCG (Wu et al., 2021), we also consider the server is malicious, which uses DLG attack (Zhu et al., 2019) to recover the original data from victim clients. We employ PSNR to measure the quality of the recovered images, thus evaluating the privacy-preserving capability of different FL methods. To ensure reliability, we report the average for each experiment over 5 different random seeds. Due to the space limitations, we relegate full experimental settings and results to Appendix E.

3.2 RESULTS COMPARISON

Overview test performance comparison. As shown in Table 1, FedAvg achieves the best test performance while FedMD-CG achieves the second-best test performance in most cases of EMNIST and FMNIST. FedAvg’s test performance benefits from the fact that the server can collect complete local models from clients and then obtain the real global model to ensure remarkable test performance. In most cases, the test performance of LT is worse than that of other methods since no information is shared among clients, inevitably causing over-fitting and poor generalization to new samples. LG-FedAvg consistently outperforms FedPer w.r.t the local test accuracy, indicating that personalized classifiers can mitigate the sacrifice of local model performance when the feature extractor has several convolutional layers. Meanwhile, FedMD-CG achieves the optimal local test accuracy on CIFAR-10. We conjecture that the simple model average aggregation in FedAvg may counteract the personalized knowledge from clients, thus adversely affecting local models’ performance in difficult classification tasks. Further, Fig. 3 (b)-(e) demonstrate that there is an overwhelming advantage of FedMD-CG over baselines in terms of learning efficiency during the early stages of training. Particularly, the local learning efficiency of FedMD-CG consistently outperforms that of baselines on FMNIST with $\omega = 10.0$ and CIFAR-10. Fig. 3 (f)-(h) reveal the impact of data heterogeneity on test performance for the methods. It can be observed that the test performance of all methods deteriorates as ω decreases. In most cases, FedMD-CG dominates the baselines that share only part of the model in terms of the local test accuracy. Also, FedMD-CG uniformly surpasses baselines w.r.t the local test accuracy over varying ω on CIFAR-10. This indicates that our method is robust to data heterogeneity.

Privacy comparison. Here, we compare the privacy-preserving ability of FedMD-CG with other baselines under DLG attack. It is worth noting that PSNR measures the similarity between the original image and the restored image. A larger PSNR value indicates a higher similarity between the images. As observed in Table 2, while FedAvg achieves excellent test performance (see Table 1), it scores the highest PSNR value across all datasets, which seriously threatens clients’ private information. Also, Fig. 4 illustrates that the DLG attack is able to reconstruct the image very close to the original image in FedAvg. According to Table 1 and Fig. 4, it is noticed that the strategy in FedPer to share clients’ feature extractors should be prohibited, as it neither enables competitive test performance nor protects clients’ privacy. On the other hand, LG-FedAvg, FedCG and FedMD-CG can effectively prevent the privacy leakage of clients due to the low PSNR values. Despite the small performance gap between FedMD-CG and LG-FedAvg w.r.t. PSNR, FedMD-CG can significantly outperform LG-FedAvg in terms of

Table 2: Comparison of FedMD-CG and baselines in terms of PSNR (dB) ($\omega = 10.0$). Note that both FedGen and LG-FedAvg upload local classifiers to the server, and their privacy-preserving capabilities are intuitively the same, so we only report the PSNR of LG-FedAvg.

Alg.s	EMNIST	FMNIST	CIFAR-10
FedAvg	24.54±0.15	22.63±0.61	29.22±1.56
FedPer	23.55±0.52	19.66±1.03	14.84±2.61
LG-FedAvg	6.78±0.09	6.33±1.32	8.75±1.03
FedCG	7.05±0.63	6.98±1.57	9.87±1.83
FedMD-CG	6.95±0.31	7.02±1.22	9.69±1.04

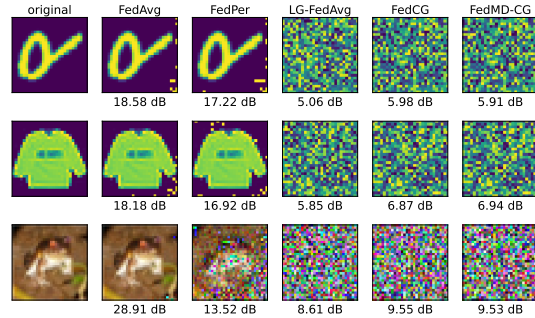


Figure 4: Image reconstruction with DLG attack in FedMD-CG and baselines. From the first to the last row, the images are selected from EMNIST, FMNIST and CIFAR-10 respectively. PSNR (dB) is reported under each recovered image.

Table 3: Test performance (%) comparison FedMD-CG and FedCG with different server-side aggregation manners over different datasets. Note that *AVE_agg* and *AVE_agg** denote weighted average aggregation. Specifically, FedMD-CG with *AVE_agg* transfers the knowledge from the global generator to local models at both the latent feature level and the logit level, whereas FedMD-CG with *AVE_agg** transfers the knowledge from the global generator to local models only at the latent feature level. Also, *KD_agg* and *KDC_agg* denote the server-side aggregation manners from FedCG and FedMD-CG, respectively.

Alg.s	Agg.	EMNIST, $\omega = 0.1$		FMNIST, $\omega = 0.1$		CIFAR-10, $\omega = 1.0$	
		<i>L.acc</i>	<i>G.acc</i>	<i>L.acc</i>	<i>G.acc</i>	<i>L.acc</i>	<i>G.acc</i>
FedCG	<i>AVE_agg*</i>	50.55 \pm 4.32	86.74 \pm 1.44	39.46 \pm 3.40	67.41 \pm 3.59	41.85 \pm 2.48	44.98 \pm 1.79
	<i>KD_agg</i>	49.91 \pm 3.83	87.66 \pm 2.08	34.97 \pm 2.55	54.61 \pm 2.67	30.44 \pm 3.30	26.79 \pm 2.82
	<i>KDC_agg</i>	39.65 \pm 4.67	82.32 \pm 4.66	37.23 \pm 2.54	62.89 \pm 7.01	28.87 \pm 0.90	25.92 \pm 1.53
FedMD-CG	<i>AVE_agg*</i>	51.36 \pm 3.63	86.88 \pm 1.53	40.55 \pm 3.55	67.34 \pm 5.41	43.24 \pm 2.32	45.10 \pm 1.44
	<i>AVE_agg</i>	52.62 \pm 3.74	86.92 \pm 1.35	41.44 \pm 2.98	67.88 \pm 6.07	45.16 \pm 2.35	46.72 \pm 2.32
	<i>KD_agg</i>	53.14 \pm 4.73	83.86 \pm 2.10	41.79 \pm 3.54	64.68 \pm 4.31	45.12 \pm 2.30	46.98 \pm 2.60
	<i>KDC_agg</i>	54.45\pm3.56	87.87\pm1.64	42.55\pm3.68	71.09\pm1.01	46.30\pm2.24	47.56\pm2.21

test performance (see Table 1). This indicates that approximating the local feature extractor with a generator not only has little privacy leakage risk but also improves performance.

Comparison between FedCG and FedMD-CG.

From Table 1 and Fig. 3, the test performance of FedCG is worse than that of FedMD-CG in most cases, even worse than other baselines on CIFAR-10 and FMNIST ($\omega = 0.1$). We speculate that this attributes to the way FedCG transfers knowledge from the global generator to the local model and the inconsistency between the local generator and classifier in each client. We next perform extensive experiments to verify our statement, as shown in Table 3 and Fig. 5. From Table 3, FedMD-CG with *AVE_agg** consistently surpasses FedCG with *AVE_agg** in terms of the local test accuracy. The main reason is that FedMD-CG enables the local generators

to extract the knowledge of the local models more effectively, which results in higher-quality trained local generators. Also, the test performance of FedMD-CG with *AVE_agg* uniformly leads that of FedMD-CG with *AVE_agg**, suggesting the insufficiency of knowledge transfer from the global generator to local models only at the latent feature level. In addition, we compare the efficacy of different server-side aggregation manners. Concretely, the test performance of FedMD-CG with *KDC_agg* consistently outperforms FedMD-CG with *KD_agg*, indicating that *KDC_agg* is more effective in transferring knowledge from local generators and classifiers to the global generator and classifier. However, *KD_agg* and *KDC_agg* significantly deteriorate the test performance of FedCG. We conjecture that the output of the local generator does not match the local classifier’s, that is, the local classifier cannot effectively distinguish the output of the local generator, resulting in the degraded performance of FedCG. As shown in Fig. 5, G+D loss of FedCG is consistently larger than that of FedMD-CG and does not converge. In general, the inconsistency of the local generator and classifier in each client may impede the server-side knowledge distillation aggregation training, resulting in poor performance of FedCG.

3.3 ABLATION STUDY

Necessity of losses in client-side for FedMD-CG. We look into the test performance of FedMD-CG on CIFAR-10 with $\omega = 10.0$ after discarding some losses in Eqs. (4) and (9), respectively, as shown in Table 4. We can see that removing any loss leads to worse performance, i.e., lower local test accuracy and global test accuracy. Also, their joint absence can cause further degradation of test performance. A trend in losses is observed that the absence of a single loss leads to a drop in test performance, while the removal of multiple losses enlarges the drop. In addition, it should be noted that dropping multiple losses in the local generator update leads to more severe test performance degradation compared to the local model update. This shows that well-trained local generators can

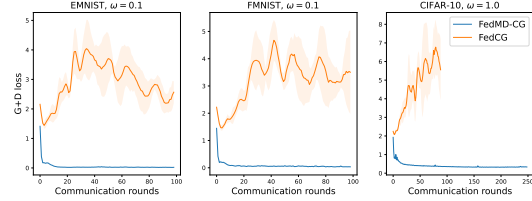


Figure 5: The consistency comparison between local generators and classifiers for FedCG and FedMD-CG w.r.t. *AVE_agg**. G+D loss denotes the classification loss of the local classifier on the output of the local generator.

Table 4: Impact of each loss for client-side training over CIFAR-10 with $\omega = 10.0$. Note that L.M.U and L.G.U denote the local model update and the local generator update, respectively. Also, we omit the subscript i of each loss for client i .

FedMD-CG (baseline)					
L_{acc}			G_{acc}		
54.82±0.79			55.18±1.75		
L.M.U	L_{acc}	G_{acc}	L.G.U	L_{acc}	G_{acc}
$\rightarrow \mathcal{L}_{ce}$	51.53±1.04	52.52±1.37	$\leftarrow \mathcal{L}_{mse}$	52.73±1.15	53.19±1.72
$\rightarrow \mathcal{L}_{mse}$	53.07±0.97	53.73±1.99	$\leftarrow \mathcal{L}_{ce}$	53.89±0.89	52.88±2.09
$\rightarrow \mathcal{L}_{kl}$	53.46±0.94	53.34±1.74	$\leftarrow \mathcal{L}_{div}$	52.66±0.77	53.11±1.72
$\rightarrow \mathcal{L}_{ce}, \rightarrow \mathcal{L}_{mse}$	51.02±0.52	52.96±1.01	$\leftarrow \mathcal{L}_{mse}, \leftarrow \mathcal{L}_{ce}$	46.94±1.33	49.37±1.53
$\rightarrow \mathcal{L}_{ce}, \rightarrow \mathcal{L}_{kl}$	51.55±0.60	52.81±1.22	$\leftarrow \mathcal{L}_{mse}, \leftarrow \mathcal{L}_{div}$	47.80±0.30	50.15±1.24
$\rightarrow \mathcal{L}_{mse}, \rightarrow \mathcal{L}_{kl}$	52.64±0.40	53.46±1.25	$\leftarrow \mathcal{L}_{ce}, \leftarrow \mathcal{L}_{div}$	48.02±0.31	50.41±1.61
$\rightarrow \mathcal{L}_{ce}, \rightarrow \mathcal{L}_{mse}, \rightarrow \mathcal{L}_{kl}$	50.27±0.41	49.55±1.28	$\leftarrow \mathcal{L}_{mse}, \leftarrow \mathcal{L}_{ce}, \leftarrow \mathcal{L}_{div}$	44.33±1.38	47.66±1.35

effectively boost the performance of our method, while under-trained local generators hinder the training of models.

Impacts of diversity constraints. We also explore the effect of different diversity constraints on FedMD-CG. Note that we omit the subscript i of diversity loss for client i . From Table 5, FedMD-CG with \mathcal{L}_{div}^1 and \mathcal{L}_{div}^2 beats FedMD-CG with \mathcal{L}_{div}^0 w.r.t. the test performance in most case. Also, \mathcal{L}_{div}^1 and \mathcal{L}_{div}^2 uniformly trump \mathcal{L}_{div}^0 in terms of the local test accuracy. Consequently, an empirical finding can be derived that imposing more weight on the inter-class output pair distance of the local generator boosts the local models’ performance.

Robustness of FedMD-CG against hyperparameters. We investigate the test performance of FedMD-CG with varying hyperparameters over FMNIST. We set $\omega = 1.0$ and select $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ and λ_6 from $[0.25, 0.5, 0.75, 1.0, 1.25, 1.5]$. Fig. 6 shows the test performance using the box plot, where FedMD-CG exemplifies similar test performance for non-zero selection of hyperparameters. Notably, for a single loss, the effect of non-zero varying hyperparameters on the local test accuracy of FedMD-CG is slight. This indicates that FedMD-CG is insensitive to the choice of non-zero hyperparameters over a large range for a single loss.

Table 5: Test performance (%) comparison among different diversity constraints.

Div. con.	EMNIST, $\omega = 0.1$		FMNIST, $\omega = 1.0$		CIFAR-10, $\omega = 10.0$	
	L_{acc}	G_{acc}	L_{acc}	G_{acc}	L_{acc}	G_{acc}
\mathcal{L}_{div}^0	53.09±4.27	88.85±1.31	78.58±1.58	84.69±0.46	54.24±0.72	54.78±1.88
\mathcal{L}_{div}^1	53.65±4.12	88.51±0.80	79.03±1.52	84.73±0.49	54.81±0.71	54.90±1.73
\mathcal{L}_{div}^2	54.45±3.56	87.87±1.64	79.00±1.43	84.47±0.38	54.82±0.79	55.18±1.75

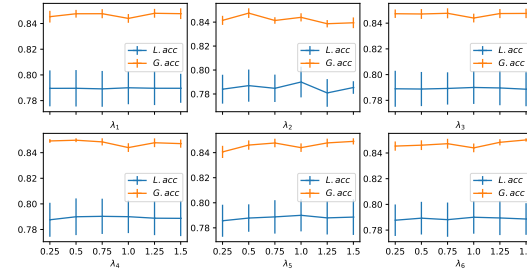


Figure 6: Test performance of FedMD-CG using varying hyperparameters on FMNIST with $\omega = 1.0$.

4 CONCLUSIONS

In this paper, we propose a novel FL method FedMD-CG, which achieves high competitive performance and high-level privacy preservation. Specifically, FedMD-CG decomposes each client’s local model into a feature extractor and a classifier, and utilizes a conditional generator instead of the feature extractor to perform server-side model aggregation. Meanwhile, our method taps KD to train local models and generators at the latent feature level and the logit level, thereby ensuring the consistency of local generators and classifiers. Also, we construct additional classification losses and craft new diversity losses to enhance client-side training. On the server side, FedMD-CG aggregates trained local generators and classifiers in a crossed data-free KD manner. Finally, we conduct extensive experiments to verify the superiority of FedMD-CG. Due to space constraints, we discuss in detail the **limitations** and **broadier impacts** of our work in Appendixes F and G, respectively.

REFERENCES

- Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *ICLR*, 2021.
- Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.
- Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3514–3522, 2019.
- Anda Cheng, Peisong Wang, Xi Sheryl Zhang, and Jian Cheng. Differentially private federated learning with local regularization and sparsification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10122–10131, 2022.
- Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.
- Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.
- Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019.
- Xiuwen Fang and Mang Ye. Robust federated learning with noisy and heterogeneous clients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10072–10081, 2022.
- Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.
- Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.
- Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.
- Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyang Wu, Terrence Chen, David Doermann, and Arun Innanje. Preserving privacy in federated learning with ensemble cross-domain knowledge distillation. *AAAI*, 2022.
- Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *Advances in Neural Information Processing Systems*, 35: 22911–22924, 2022.
- Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2): 217–288, 2011.
- Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.
- Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In *CVPR*, pp. 10143–10153, 2022.
- Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.
- Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

- Jinkyu Kim, Geeho Kim, and Bohyung Han. Multi-level branched regularization for federated learning. In *International Conference on Machine Learning*, pp. 11058–11073, 2022.
- Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.
- Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020a.
- Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020b.
- Yijing Li, Xiaofeng Tao, Xuefei Zhang, Junjie Liu, and Jin Xu. Privacy-preserved federated learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):8423–8434, 2021.
- Zhuohang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. Auditing privacy defenses in federated learning via generative gradient leakage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10132–10142, 2022.
- Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.
- Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *NIPS*, 33:2351–2363, 2020.
- Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1013–1023, 2021.
- Kangyang Luo, Xiang Li, Yunshi Lan, and Ming Gao. Gradma: A gradient-memory-based accelerated federated learning with alleviated catastrophic forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3708–3717, 2023.
- Jing Ma, Si-Ahmed Naas, Stephan Sigg, and Xixiang Lyu. Privacy-preserving federated learning based on multi-key homomorphic encryption. *International Journal of Intelligent Systems*, 2022.
- Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.
- Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.
- Yiqing Shen, Yuyin Zhou, and Lequan Yu. Cd2-pfed: Cyclic distillation-guided channel decoupling for model personalization in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10041–10050, 2022.
- Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed.*, Cham: Springer International Publishing, 10(3152676):10–5555, 2017.
- Zi Wang. Data-free knowledge distillation with soft targeted transfer set synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10245–10253, 2021.

- Yuezhou Wu, Yan Kang, Jiahuan Luo, Yuanqin He, and Qiang Yang. Fedcg: Leverage conditional gan for protecting privacy and maintaining competitive performance in federated learning. *arXiv preprint arXiv:2111.08211*, 2021.
- Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4133–4141, 2017.
- Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.
- Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. *Advances in Neural Information Processing Systems*, 32, 2019.
- Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.
- Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.
- Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. Dense: Data-free one-shot federated learning. In *Advances in Neural Information Processing Systems*, 2022a.
- Li Zhang, Jianbo Xu, Pandi Vijayakumar, Pradip Kumar Sharma, and Uttam Ghosh. Homomorphic encryption-based privacy-preserving federated learning in iot-enabled healthcare system. *IEEE Transactions on Network Science and Engineering*, 2022b.
- Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *CVPR*, pp. 10174–10183, 2022c.
- Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.
- Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *ICML*, pp. 12878–12889, 2021.

APPENDIX

A RELATED WORK

Privacy Preservation in FL. FL (McMahan et al., 2017; Li et al., 2020b) has emerged as a de facto machine learning area and received rapidly increasing research interest from the community. One of the primary attractions of FL is that it provides basic-level data privacy and security, as clients jointly train a global model by sharing model parameters or gradient updates without exposing their private data. Yet, for such “naked” FL methods that do not provide any formal or provable privacy guarantees, some inference attacks (e.g., model-inversion (Yin et al., 2020; Haim et al., 2022) and deep leakage from gradients (DLG) (Zhu et al., 2019; Geiping et al., 2020)) can easily extract sensitive information and even recover the original data from the trained model parameters or gradient updates without any information assistance. To reduce the risk of privacy leakage, FL often combines homomorphic encryption (HE) (Gentry, 2009) and differential privacy (DP) (Dwork et al., 2006). However, HE is computationally inefficient, while DP can deteriorate the training performance. Meanwhile, there exists an alternative line of FL methods (Arivazhagan et al., 2019; Liang et al., 2020; Shen et al., 2022), which focuses on decomposing a model into private and public layers and sharing the public layers to boost the privacy of FL at the cost of inevitable performance drop. Particularly, FedPer (Arivazhagan et al., 2019) splits a model into base and top layers. Each client uploads the base layers and hides the top layers from the server. Whereas, LG-FedAvg (Liang et al., 2020) shares the top layers while keeps the base layers localized.

Knowledge Distillation in FL. The main insight of KD is to extract knowledge from one or more teacher models to a student model via learning their soft predictions, attention maps or intermediate (latent) features (Hinton et al., 2015; Zagoruyko & Komodakis, 2016; Yim et al., 2017). FL with KD has recently emerged as effective methods for dealing with real-world tasks. For example, FedMLB (Kim et al., 2022) and RHFL (Fang & Ye, 2022) mitigate the fall of performance caused by data heterogeneity (i.e., non-IID). FedMD (Li & Wang, 2019), FedDP (Lin et al., 2020) and FCCL (Huang et al., 2022) are able to perform FL with heterogeneous local models across clients. In addition, FedGen (Zhu et al., 2021) and FedKD (Gong et al., 2022) facilitate privacy-preserving of FL.

Conditional Generator in FL. The objects mimicked by a conditional generator in FL can be roughly divided into two categories: clients’ raw data and models’ latent features. For **the former**, FAug (Jeong et al., 2018) requires each client to collectively train a cGAN to augment its local data yielding an IID dataset. DENSE (Zhang et al., 2022a) and FedFTG (Zhang et al., 2022c) employ data-free KD to train a conditional generator on the server, thus training and fine-tuning the global model, respectively. Note that data-free KD is a promising approach to transfer knowledge from the teacher model to another student model without any real data (Chen et al., 2019; Fang et al., 2019). The mentioned FL methods utilize a conditional generator to enhance the generalization performance of global models under heterogeneity or communication cost constraints, but they are highly susceptible to privacy attacks or even violate the key privacy assumptions of FL. Alternatively, the conditional generator can also be used as an attack tool to reconstruct the private data of the victim clients on a malicious server (Li et al., 2022). For **the latter**, FedGen (Zhu et al., 2021) uploads the last layer of the local models to the server and trains a global conditional generator using data-free KD to boost the local model update of each client. FedCG (Wu et al., 2021) integrates cGAN into FL aiming to harness a conditional generator to replace the local feature extractor and upload it to the server together with the local classifier, thus maintaining high-level privacy protection.

B PSEUDOCODE

In this section, we detail the pseudocode of FedMD-CG in Algorithm 1.

C ALGORITHM DESCRIPTION

Algorithm 1 summarizes the training procedure of FedMD-CG. Concretely, starting from the local model update, clients first sample two mini-batch data $\{x_b, \hat{z}_b, y_b\}_{b=1}^B$ and $\{\hat{z}_b, \hat{y}_b\}_{b=1}^B$ to perform the local model update (lines 9-13), and then train the local generator with re-sampled mini-batch

Algorithm 1 FedMD-CG

```

1: Input: communication round  $R$ , client number  $N$ , label distribution  $p(y)$ , client-side training
   step  $I_c$ , client-side learning rates  $\eta_c^l, \eta_c^\omega$ , client-side label counter  $\{c_i\}_{i \in [N]}$ , server-side training
   step  $I_s$ , server-side learning rate  $\eta_s$ , batch size  $B$ , hyperparameters  $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6$ .
2: Initialize  $\mathbf{w}^i$  and  $\boldsymbol{\theta}^i = [\boldsymbol{\theta}_F^i, \boldsymbol{\theta}_D^i]$  on the client  $i$ .
3: Initialize  $[\mathbf{w}, \boldsymbol{\theta}_D]$  on the server.
4: for  $r = 1, \dots, R$  do
5:   Server broadcasts  $(\mathbf{w}, \boldsymbol{\theta}_D)$  and  $p(y)$  to the clients.
6:   On clients:
7:   for  $i \in [N]$  parallel do
8:      $\boldsymbol{\theta}_D^i = \boldsymbol{\theta}_D$ 
9:     for  $\tau = 1, \dots, I_c$  do
10:      Sample  $\{\mathbf{x}_b, \hat{\mathbf{z}}_b, y_b\}_{b=1}^B$  and resample  $\{\hat{\mathbf{z}}_b, \hat{y}_b\}_{b=1}^B$ , where  $\{\mathbf{x}_b, y_b\} \sim \{\mathbf{X}_i, \mathbf{Y}_i\}$ ,  $\hat{\mathbf{z}}_b \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and  $\hat{y}_b \sim p(y)$ .
11:      Update label counter  $c_i$ .
12:      Update  $\boldsymbol{\theta}^i$  with  $\eta_c^l$  according to Eq. (4).
13:    end for
14:    for  $\tau = 1, \dots, I_c$  do
15:      Sample  $\{\mathbf{x}_b, \hat{\mathbf{z}}_b, y_b\}_{b=1}^B$ , where  $\{\mathbf{x}_b, y_b\} \sim \{\mathbf{X}_i, \mathbf{Y}_i\}$  and  $\hat{\mathbf{z}}_b \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ .
16:      Update  $\mathbf{w}^i$  with  $\eta_c^\omega$  according to Eq. (9).
17:    end for
18:    Upload  $[\mathbf{w}^i, \boldsymbol{\theta}_D^i]$  and  $c_i$  to the server.
19:   On server:
20:    $[\mathbf{w}, \boldsymbol{\theta}_D] \leftarrow \sum_{i \in [N]} \frac{n_i}{\sum_{j \in [N]} n_j} [\mathbf{w}^i, \boldsymbol{\theta}_D^i]$  and update  $p(y)$  based on  $\{c_i\}_{i \in [N]}$ .
21:   for  $\tau = 1, \dots, I_s$  do
22:     Sample  $\{\hat{\mathbf{z}}_b, \hat{y}_b\}_{b=1}^B$ , where  $\hat{\mathbf{z}}_b \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and  $\hat{y}_b \sim p(y)$ .
23:     Update  $[\mathbf{w}, \boldsymbol{\theta}_D]$  with  $\eta_s$  according to Eq. (12).
24:   end for
25: end for

```

data $\{\mathbf{x}_b, \hat{\mathbf{z}}_b, y_b\}_{b=1}^B$ (lines 14-17). The trained local generators and classifiers are sent to the server. The server subsequently aggregates these generators and classifiers by simple model averaging to form the preliminary global generator and classifier, and then trains the global generator and classifier by using sampled data $\{\hat{\mathbf{z}}_b, \hat{y}_b\}_{b=1}^B$ (lines 21-24). Notably, the global generator is under-trained at the early stages of training, which may mislead the local model training. Therefore, during the training phase, λ_3, λ_2 and λ_1 are first initialized to 0 and increase to pre-defined values with the increase of communication round. Readers are referred to the experimental section for more details.

D COMPUTING DEVICES AND PLATFORMS

- OS: Ubuntu 18.04.3 LTS
- CPU: Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz
- CPU Memory: 256 GB.
- GPU: NVIDIA Tesla V100 PCIe
- GPU Memory: 32GB
- Programming platform: Python 3.7.4
- Deep learning platform: PyTorch 1.9.0

E FULL EXPERIMENTS

E.1 FULL EXPERIMENTAL SETTING

Datasets. We perform our experiments on three public datasets EMNIST (Cohen et al., 2017), Fashion-MNIST (Xiao et al., 2017) (FMNIST in short in this paper), and CIFAR-10 (Krizhevsky et al., 2009). Following existing works (Zhang et al., 2022c; Acar et al., 2021; Zhu et al., 2021), we use Dirichlet process $Dp(\omega)$ to strictly partition the training set of each dataset across clients. Notably, a smaller ω corresponds to higher data heterogeneity. We set $\omega \in \{0.1, 1.0, 10.0\}$ in our experiments.

Backbone Architectures and Baselines. Throughout all our experiments, we deploy LeNet-5 (LeCun et al., 1998) as the backbone network with two convolutional layers (i.e., feature extractor) and three fully connected layers (i.e., classifier). Similarly, we employ three fully connected layers with BatchNorm as the generator for each client and adjust its output dimension to match that of the corresponding feature extractor. We select five FL methods most relevant to our work as baselines for comparison, including FedAvg (McMahan et al., 2017), FedPer (Arivazhagan et al., 2019), LG-FedAvg (Liang et al., 2020), FedGen (Zhu et al., 2021) and FedCG (Wu et al., 2021). Moreover, we consider the baseline that trains a local model for each client, without any sharing. We call it Local Training (LT for short). For fairness, FedGen shares clients’ classifiers with the server. In particular, we treat the client’s classifier whose output dimension is set to 1 as the discriminator of cGAN in FedCG.

Configurations. For EMNIST (FMNIST), we set communication round $R = 100$ (100) and client number $N = 20$ (10). And we set $R = 250$ and $N = 10$ for CIFAR-10. We adopt client-side training step $I_c = 20$ and server-side training step $I_s = 50$. For client-side training, SGD and Adam are applied to optimize the local models and generators, respectively. The learning rate η_c^l for SGD is searched over the range of $\{0.01, 0.05, 0.08\}$ and the best one is picked. And we set $\eta_c^\omega = 0.0003$ for Adam. For server-side training, the Adam optimizer with $\eta_s = 0.0003$ is used to update the global generator and classifier. For all update steps, we set batch size B to 64 and weight decay to $1e - 4$. For FedMD-CG, we set the diversity constraint to \mathcal{L}_{div}^2 unless otherwise specified. For the hyperparameters used to balance different loss items, all are set to 1 unless otherwise specified. Particularly, in the local model update, we initialize λ_3, λ_2 and λ_1 to 0 and increase their values to pre-defined values with the increase of communication round to avoid the misleading caused by the under-trained global generator. We set the parameter values to be incremented by $\lambda = \lambda^{pre}((r - 1)/R)^d$, where λ^{pre} is a pre-defined value and d controls how fast the parameter increases. We set $d = 1$. The dimension of $\hat{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is 128 for all datasets.

E.2 FULL EXPERIMENTAL RESULTS

Table 6: Test performance (%) comparison between FedMD-CG and baselines over EMNIST. Note that $L.acc$ and $G.acc$ denote *local test accuracy* and *global test accuracy*, respectively.

Alg.s	EMNIST					
	$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
FedAvg	96.88±0.08	97.00±0.08	96.29±0.06	96.83±0.07	85.65±2.33	95.06±0.46
LT	96.45±0.08	96.66±0.13	90.48±1.46	95.17±0.70	40.77±2.34	62.08±5.11
FedPer	96.69±0.10	96.94±0.08	91.96±1.20	96.45±0.10	41.08±2.40	76.90±2.34
LG-FedAvg	96.57±0.11	96.84±0.10	94.01±0.53	96.22±0.19	46.29±3.39	86.48±1.75
FedGen	97.34±0.16	97.97±0.09	95.62±0.38	97.64±0.17	51.29±4.01	87.69±2.50
FedCG	97.67±0.03	98.08±0.07	96.06±0.33	97.70±0.16	49.91±3.83	87.66±2.08
FedMD-CG	96.97±0.05	97.41±0.11	95.45±0.25	97.18±0.17	54.45±3.56	87.87±1.64

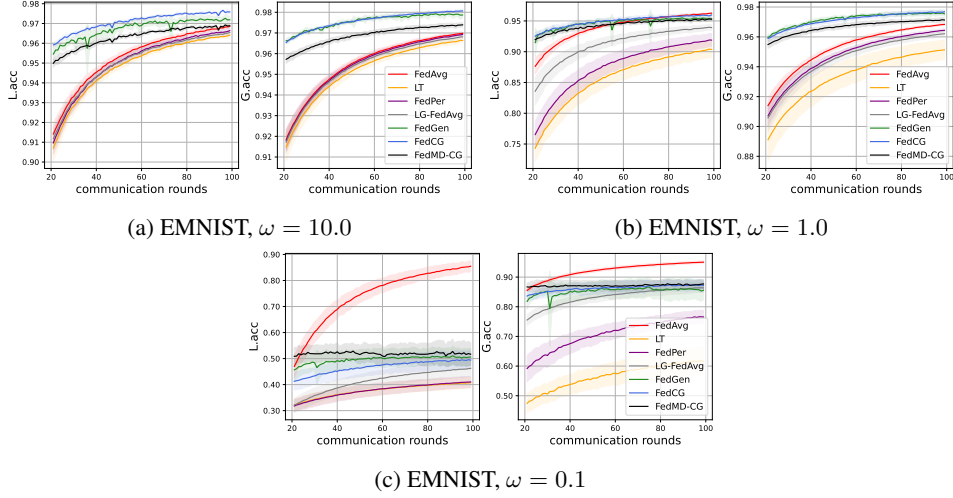


Figure 7: Learning curves for FedMD-CG as well as baselines over EMNIST.

Table 7: Test performance (%) comparison between FedMD-CG and baselines over FMNIST. Note that $L.acc$ and $G.acc$ denote *local test accuracy* and *global test accuracy*, respectively.

Alg.s	FMNIST					
	$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
FedAvg	84.28 \pm 0.24	85.02 \pm 0.29	80.99\pm0.81	84.77\pm0.30	59.29\pm3.19	78.91\pm2.12
LT	82.89 \pm 0.29	83.80 \pm 0.23	74.19 \pm 2.92	80.32 \pm 1.02	37.71 \pm 2.99	56.34 \pm 10.42
FedPer	83.68 \pm 0.21	84.83 \pm 0.27	75.83 \pm 2.42	82.94 \pm 1.11	37.39 \pm 3.17	61.88 \pm 9.80
LG-FedAvg	83.25 \pm 0.24	84.39 \pm 0.19	77.03 \pm 1.94	82.55 \pm 0.46	38.89 \pm 3.18	66.35 \pm 6.65
FedGen	83.52 \pm 1.76	85.77 \pm 0.12	77.88 \pm 3.10	83.81 \pm 1.95	41.96 \pm 3.40	68.05 \pm 3.96
FedCG	82.29 \pm 0.63	84.58 \pm 0.77	74.92 \pm 2.11	81.74 \pm 0.81	34.97 \pm 2.55	54.61 \pm 2.67
FedMD-CG	84.32\pm0.28	87.18\pm0.08	79.00 \pm 1.43	84.47 \pm 0.38	42.55 \pm 3.68	71.09 \pm 1.01

F LIMITATIONS

In the field of Federated Learning (FL), there are many trade-offs, including utility, privacy protection, computational efficiency and communication cost, etc. It is well known that trying to develop a universal FL method that can address all problems is extremely challenging. In this work, we work on improving privacy leakage defects in FL while maintaining robust model performance. Next, we discuss some of the limitations of FedMD-CG.

Computational Efficiency, Communication Cost and Utility. We acknowledge that deploying FedMD-CG in a real-world FL application requires clients to have more hardware and computational resources to train generators and local models as compared to FedAvg. Specifically, compared with FedAvg or MD-based methods (e.g., FedPer and LG-FedAvg), the training time of FedMD-CG will be longer, as it needs to additionally train the generator on the clients. In our experiments, FedMD-CG takes two to three times longer to run per communication round than they do. Moreover, compared to FedAvg or MD-based methods, FedMD-CG requires an additional vector of label statistics to be transmitted (see line 18 in Algorithm 1). However, the communication cost of this vector is negligible compared to that of the model. We also acknowledge that FedMD-CG still has room for improvement in model performance. Table 1 shows that FedAvg achieves the optimal model performance in many scenarios, so it is an attractive topic in the field of FL to achieve comparable test performance levels to FedAvg with high-level privacy protection. Meanwhile, there is a trade-off between the capacity of the generator and the communication cost.

Privacy Protection. Since FedMD-CG trains a local generator on each client for replacing the local feature extractor (LFE) by simulating the output vector space of LFE, i.e., the latent feature

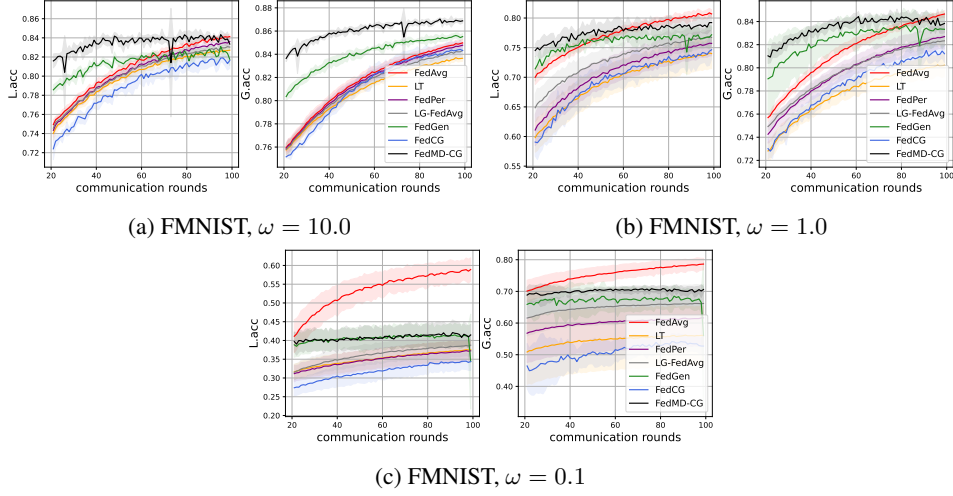


Figure 8: Learning curves for FedMD-CG as well as baselines over FMNIST.

Table 8: Test performance (%) comparison between FedMD-CG and baselines over CIFAR-10. Note that $L.acc$ and $G.acc$ denote *local test accuracy* and *global test accuracy*, respectively.

Alg.s	CIFAR-10					
	$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
FedAvg	51.68 \pm 0.53	54.18 \pm 0.52	42.84 \pm 2.03	52.39\pm0.58	23.28 \pm 1.94	45.08\pm2.03
LT	49.83 \pm 0.88	48.99 \pm 1.35	40.92 \pm 2.25	37.92 \pm 2.24	24.42 \pm 1.47	24.33 \pm 3.66
FedPer	50.72 \pm 0.63	53.87 \pm 0.58	41.80 \pm 2.15	49.83 \pm 1.66	23.26 \pm 1.74	31.74 \pm 4.21
LG-FedAvg	50.11 \pm 0.80	51.80 \pm 0.67	41.49 \pm 2.56	44.59 \pm 1.88	24.28 \pm 1.76	26.21 \pm 3.59
FedGen	52.94 \pm 2.38	48.49 \pm 3.13	38.13 \pm 4.89	40.85 \pm 3.87	23.33 \pm 1.86	27.84 \pm 2.67
FedCG	39.39 \pm 5.23	37.06 \pm 4.35	30.44 \pm 3.30	26.79 \pm 2.82	17.65 \pm 3.45	16.75 \pm 2.40
FedMD-CG	54.82\pm0.79	55.18\pm1.75	46.30\pm2.24	47.56 \pm 2.21	26.18\pm1.79	30.55 \pm 2.42

space, rather than the distribution space of private data, it provides high-level privacy protection. Also, FedMD-CG requires clients to upload the label statistics of the data, which also is at risk of compromising privacy. In addition, we argue that the **theoretical guarantee** for privacy protection is crucial. However, it's worth noting that even in exist well-known MD-based federated learning efforts (Wu et al., 2021; Zhu et al., 2021; Arivazhagan et al., 2019; Liang et al., 2020), as well as federated learning methods with the help of the generator (Wu et al., 2021; Zhu et al., 2021; Zhang et al., 2022c;a), comprehensive theoretical analysis concerning the privacy guarantees (or privacy disclosure) is often absent. Given the lack of suitable theoretical frameworks, we concentrated on robust empirical validation, showcasing our method (FedMD-CG). Our results, we believe, robustly demonstrate our method's utility. We intend to delve deeper into theoretical aspects in future work.

G BROADER IMPACTS

We work on how to improve privacy leakage defects in FL while maintaining robust model performance. Our work points out the pitfalls of the existing method FedCG. First, knowledge transfer modality at the latent feature level may not be sufficient. Second, additional discriminators need to be trained to satisfy the adversarial training of cGAN. Third, the trained local generator may not match the local classifier, terming their inconsistency. Our proposed FedMD-CG can deal with the said issues well. FedMD-CG exemplifies potential positive impacts on society, enabling models with superior performance while ensuring high-level privacy protection in real-world FL applications. Meanwhile, FedMD-CG may have negative social impacts related to high resource consumption.

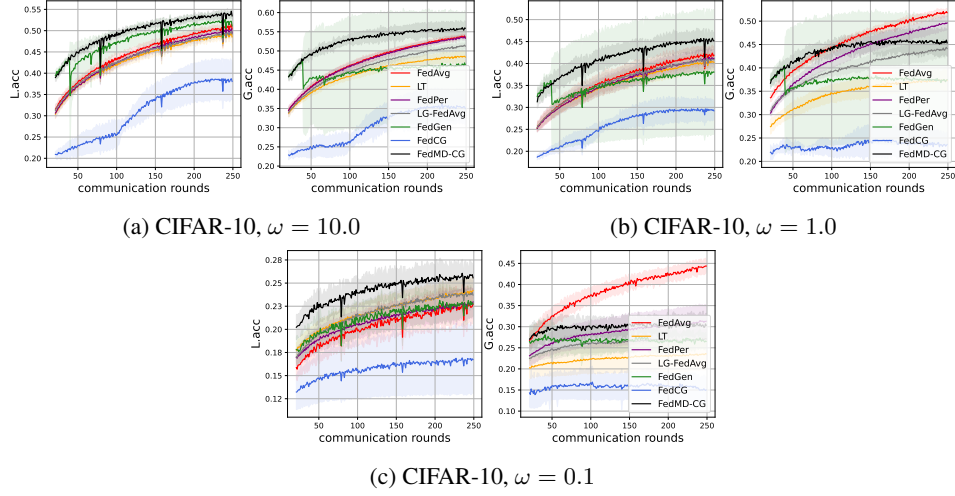


Figure 9: Learning curves for FedMD-CG as well as baselines over over CIFAR-10.

Table 9: Test performance (%) comparison between FedMD-CG and FedCG with different server-side aggregation manners over EMNIST. Note that *AVE_agg* and *AVE_agg** denote weighted average aggregation. Specifically, FedMD-CG with *AVE_agg* transfers the knowledge from the global generator to local models at both the latent feature level and the logit level, whereas FedMD-CG with *AVE_agg** transfers the knowledge from the global generator to local models only at the latent feature level. Also, *KD_agg* and *KDC_agg* denote the server-side aggregation manners from FedCG and FedMD-CG, respectively.

Alg.s	Agg.	EMNIST					
		$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
		<i>L.acc</i>	<i>G.acc</i>	<i>L.acc</i>	<i>G.acc</i>	<i>L.acc</i>	<i>G.acc</i>
FedCG	<i>AVE_agg*</i>	97.26 \pm 0.02	98.00 \pm 0.04	95.05 \pm 0.17	97.05 \pm 0.24	50.55 \pm 4.32	86.74 \pm 1.44
	<i>KD_agg</i>	97.67 \pm 0.03	98.08 \pm 0.07	96.06\pm0.33	97.70 \pm 0.16	49.91 \pm 3.83	87.66 \pm 2.08
	<i>KDC_agg</i>	96.02 \pm 0.27	96.78 \pm 0.28	93.17 \pm 0.63	96.31 \pm 0.27	39.65 \pm 4.67	82.32 \pm 4.66
FedMD-CG	<i>AVE_agg*</i>	97.29 \pm 0.01	98.19 \pm 0.03	95.12 \pm 0.46	97.21 \pm 0.21	51.36 \pm 3.63	86.88 \pm 1.53
	<i>AVE_agg</i>	97.74\pm0.05	98.36\pm0.04	95.51 \pm 0.25	97.86\pm0.08	52.62 \pm 3.74	86.95 \pm 1.35
	<i>KD_agg</i>	96.69 \pm 0.11	97.19 \pm 0.10	94.70 \pm 0.44	96.72 \pm 0.22	53.14 \pm 4.73	83.86 \pm 2.10
	<i>KDC_agg</i>	96.97 \pm 0.05	97.41 \pm 0.11	95.45 \pm 0.25	97.18 \pm 0.17	54.45\pm3.56	87.87\pm1.64

FedMD-CG-based FL systems require more client-side power resources to train the generator and local model. FedMD-CG does not involve social ethics.

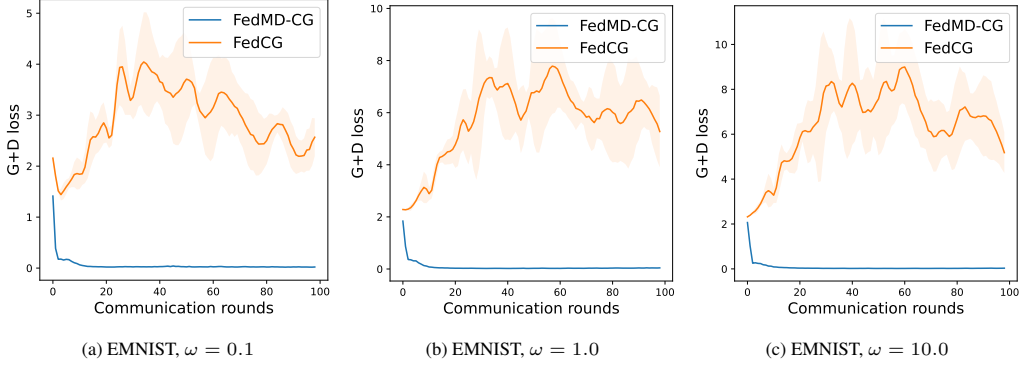


Figure 10: The consistency comparison between local generators and classifiers for FedCG and FedMD-CG w.r.t. AVE_agg^* over EMNIST. G+D loss denotes the classification loss of the local classifier on the output of the local generator.

Table 10: Test performance (%) comparison between FedMD-CG and FedCG with different server-side aggregation manners over FMNIST. Note that AVE_agg and AVE_agg^* denote weighted average aggregation. Specifically, FedMD-CG with AVE_agg transfers the knowledge from the global generator to local models at both the latent feature level and the logit level, whereas FedMD-CG with AVE_agg^* transfers the knowledge from the global generator to local models only at the latent feature level. Also, KD_agg and KDC_agg denote the server-side aggregation manners from FedCG and FedMD-CG, respectively.

Alg.s	Agg.	FMNIST					
		$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
		$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
FedCG	AVE_agg^*	83.08 \pm 0.20	85.13 \pm 0.23	77.53 \pm 1.53	82.87 \pm 0.31	39.46 \pm 3.40	67.41 \pm 3.59
	KD_agg	82.29 \pm 0.63	84.58 \pm 0.77	74.92 \pm 2.11	81.74 \pm 0.81	34.97 \pm 2.55	54.61 \pm 2.67
	KDC_agg	83.28 \pm 3.49	85.52 \pm 3.37	75.64 \pm 2.26	82.63 \pm 0.48	37.23 \pm 2.54	62.89 \pm 7.01
FedMD-CG	AVE_agg^*	83.32 \pm 0.14	85.88 \pm 0.17	78.01 \pm 1.14	83.79 \pm 0.73	40.55 \pm 3.55	67.34 \pm 5.41
	AVE_agg	83.78 \pm 0.19	86.57 \pm 0.11	78.94 \pm 1.43	84.91\pm0.48	41.44 \pm 2.98	67.88 \pm 6.07
	KD_agg	83.07 \pm 0.23	85.65 \pm 0.14	78.08 \pm 1.51	83.89 \pm 0.67	41.79 \pm 3.54	64.68 \pm 4.31
	KDC_agg	84.32\pm0.18	87.18\pm0.13	79.00\pm1.43	84.47 \pm 0.38	42.55\pm3.68	71.09\pm1.01

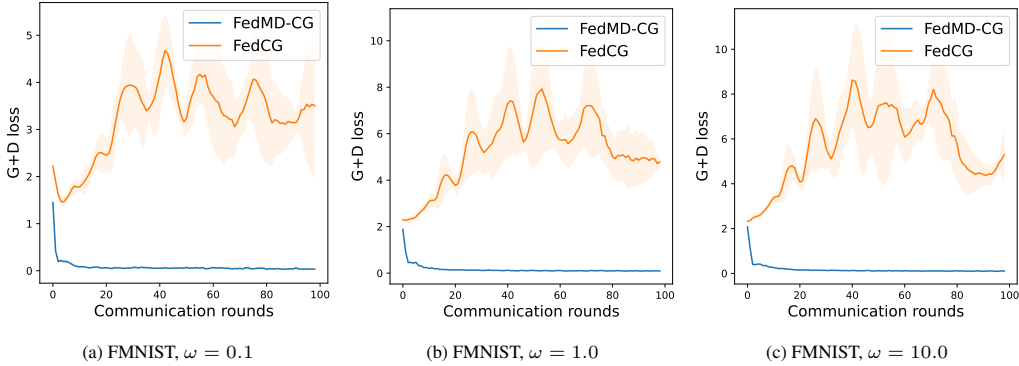


Figure 11: The consistency comparison between local generators and classifiers for FedCG and FedMD-CG w.r.t. AVE_agg^* over FMNIST. G+D loss denotes the classification loss of the local classifier on the output of the local generator.

Table 11: Test performance (%) comparison between FedMD-CG and FedCG with different server-side aggregation manners over CIFAR-10. Note that AVE_agg and AVE_agg^* denote weighted average aggregation. Specifically, FedMD-CG with AVE_agg transfers the knowledge from the global generator to local models at both the latent feature level and the logit level, whereas FedMD-CG with AVE_agg^* transfers the knowledge from the global generator to local models only at the latent feature level. Also, KD_agg and KDC_agg denote the server-side aggregation manners from FedCG and FedMD-CG, respectively.

Alg.s	Agg.	CIFAR-10					
		$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
		$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
FedCG	AVE_agg^*	48.88 \pm 0.19	51.47 \pm 1.16	41.85 \pm 2.48	44.98 \pm 1.79	24.78 \pm 1.46	28.35 \pm 2.33
	KD_agg	39.39 \pm 5.23	37.06 \pm 4.35	30.44 \pm 3.30	26.79 \pm 2.82	17.65 \pm 3.45	16.75 \pm 2.40
	KDC_agg	28.70 \pm 2.44	29.14 \pm 0.54	28.87 \pm 0.90	25.92 \pm 1.53	21.63 \pm 2.02	26.44 \pm 3.91
FedMD-CG	AVE_agg^*	51.34 \pm 0.80	52.71 \pm 1.73	43.24 \pm 2.32	45.10 \pm 2.32	25.80 \pm 1.72	30.27 \pm 2.65
	AVE_agg	52.34 \pm 0.80	53.71 \pm 1.73	45.16 \pm 2.35	46.72 \pm 2.32	25.81 \pm 1.82	30.70 \pm 2.12
	KD_agg	52.48 \pm 1.03	53.71 \pm 1.65	45.12 \pm 2.30	46.98 \pm 2.60	26.06 \pm 1.71	31.04\pm2.60
	KDC_agg	54.82\pm0.79	55.18\pm1.75	46.30\pm2.24	47.56\pm2.21	26.18\pm1.79	30.55 \pm 2.42

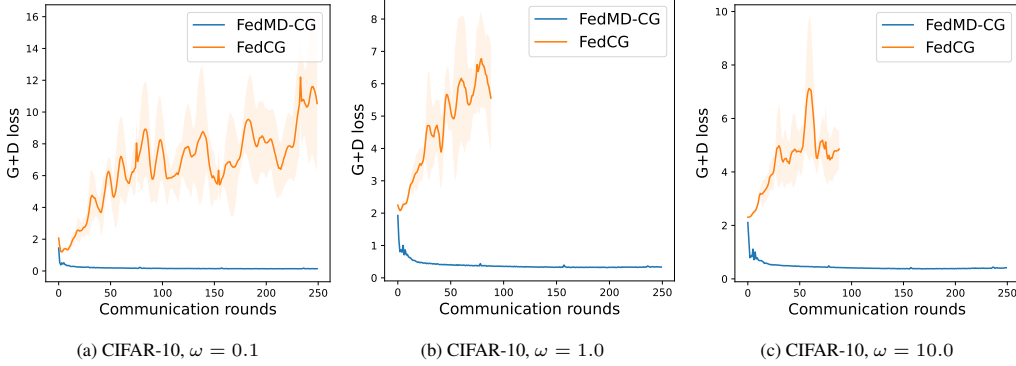


Figure 12: The consistency comparison between local generators and classifiers for FedCG and FedMD-CG w.r.t. AVE_agg^* over CIFAR-10. G+D loss denotes the classification loss of the local classifier on the output of the local generator.

Table 12: Impact of each loss for client-side training in FedMD-CG over EMNIST with $\omega = 0.1$. Note that L.M.U and L.G.U denote the local model update and the local generator update, respectively. Also, we omit the subscript i of each loss for client i .

FedMD-CG (baseline)					
$L.acc$			$G.acc$		
54.45\pm3.56			87.87\pm1.64		
L.M.U	$L.acc$	$G.acc$	L.G.U	$L.acc$	$G.acc$
$\vec{\mathcal{L}}_{ce}$	52.51 \pm 3.61	85.44 \pm 1.40	$\overleftarrow{\mathcal{L}}_{mse}$	49.33 \pm 3.27	81.09 \pm 4.49
$\vec{\mathcal{L}}_{mse}$	52.85 \pm 3.17	86.47 \pm 1.54	$\overleftarrow{\mathcal{L}}_{ce}$	51.87 \pm 2.67	82.97 \pm 1.54
$\vec{\mathcal{L}}_{kl}$	52.31 \pm 2.29	84.70 \pm 1.03	$\overleftarrow{\mathcal{L}}_{div}$	52.17 \pm 3.68	87.86 \pm 2.15
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{mse}$	51.55 \pm 3.42	84.35 \pm 1.20	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{ce}$	47.81 \pm 2.95	80.66 \pm 3.45
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{kl}$	50.42 \pm 3.43	82.59 \pm 1.37	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{div}$	47.12 \pm 2.65	80.49 \pm 2.05
$\vec{\mathcal{L}}_{mse}, \vec{\mathcal{L}}_{kl}$	51.34 \pm 3.16	85.63 \pm 1.07	$\overleftarrow{\mathcal{L}}_{ce}, \overleftarrow{\mathcal{L}}_{div}$	50.32 \pm 3.35	82.09 \pm 2.38
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{mse}, \vec{\mathcal{L}}_{kl}$	46.53 \pm 4.77	84.51 \pm 2.24	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{ce}, \overleftarrow{\mathcal{L}}_{div}$	40.12 \pm 2.15	61.53 \pm 3.55

Table 13: Impact of each loss for client-side training in FedMD-CG over FMNIST with $\omega = 1.0$. Note that L.M.U and L.G.U denote the local model update and the local generator update, respectively. Also, we omit the subscript i of each loss for client i .

FedMD-CG (baseline)					
$L.acc$			$G.acc$		
79.00±1.43			84.47±0.38		
L.M.U	$L.acc$	$G.acc$	L.G.U	$L.acc$	$G.acc$
$\vec{\mathcal{L}}_{ce}$	78.61±1.63	84.67±0.32	$\overleftarrow{\mathcal{L}}_{mse}$	77.98±1.78	84.90±0.40
$\vec{\mathcal{L}}_{mse}$	77.55±1.37	83.67±0.41	$\overleftarrow{\mathcal{L}}_{ce}$	78.43±1.57	83.82±0.59
$\vec{\mathcal{L}}_{kl}$	78.02±1.34	84.16±0.28	$\overleftarrow{\mathcal{L}}_{div}$	77.87±1.49	83.84±0.41
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{mse}$	77.02±1.79	82.73±0.24	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{ce}$	77.71±1.78	84.35±0.41
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{kl}$	77.91±1.57	83.64±0.29	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{div}$	76.93±1.54	82.58±0.53
$\vec{\mathcal{L}}_{mse}, \vec{\mathcal{L}}_{kl}$	76.92±1.66	82.66±0.35	$\overleftarrow{\mathcal{L}}_{ce}, \overleftarrow{\mathcal{L}}_{div}$	77.61±2.00	84.50±0.35
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{mse}, \vec{\mathcal{L}}_{kl}$	75.34±1.42	81.46±0.98	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{ce}, \overleftarrow{\mathcal{L}}_{div}$	72.26±1.85	79.41±0.15

Table 14: Impact of each loss for client-side training in FedMD-CG over CIFAR-10 with $\omega = 10.0$. Note that L.M.U and L.G.U denote the local model update and the local generator update, respectively. Also, we omit the subscript i of each loss for client i .

FedMD-CG (baseline)					
$L.acc$			$G.acc$		
54.82±0.79			55.18±1.75		
L.M.U	$L.acc$	$G.acc$	L.G.U	$L.acc$	$G.acc$
$\vec{\mathcal{L}}_{ce}$	51.53±1.04	52.52±1.37	$\overleftarrow{\mathcal{L}}_{mse}$	52.73±1.15	53.19±1.72
$\vec{\mathcal{L}}_{mse}$	53.07±0.97	53.73±1.99	$\overleftarrow{\mathcal{L}}_{ce}$	53.89±0.89	52.88±2.09
$\vec{\mathcal{L}}_{kl}$	53.46±0.94	53.34±1.74	$\overleftarrow{\mathcal{L}}_{div}$	52.66±0.77	53.11±1.72
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{mse}$	51.02±0.52	52.96±1.01	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{ce}$	46.94±1.33	49.37±1.53
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{kl}$	51.55±0.60	52.81±1.22	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{div}$	47.80±0.30	50.15±1.24
$\vec{\mathcal{L}}_{mse}, \vec{\mathcal{L}}_{kl}$	52.64±0.40	53.46±1.25	$\overleftarrow{\mathcal{L}}_{ce}, \overleftarrow{\mathcal{L}}_{div}$	48.02±0.31	50.41±1.61
$\vec{\mathcal{L}}_{ce}, \vec{\mathcal{L}}_{mse}, \vec{\mathcal{L}}_{kl}$	50.27±0.41	49.55±1.28	$\overleftarrow{\mathcal{L}}_{mse}, \overleftarrow{\mathcal{L}}_{ce}, \overleftarrow{\mathcal{L}}_{div}$	44.33±1.38	47.66±1.35

Table 15: Test performance (%) comparison among different diversity constraints used by FedMD-CG over EMNIST. Note that we omit the subscript i of diversity loss for client i .

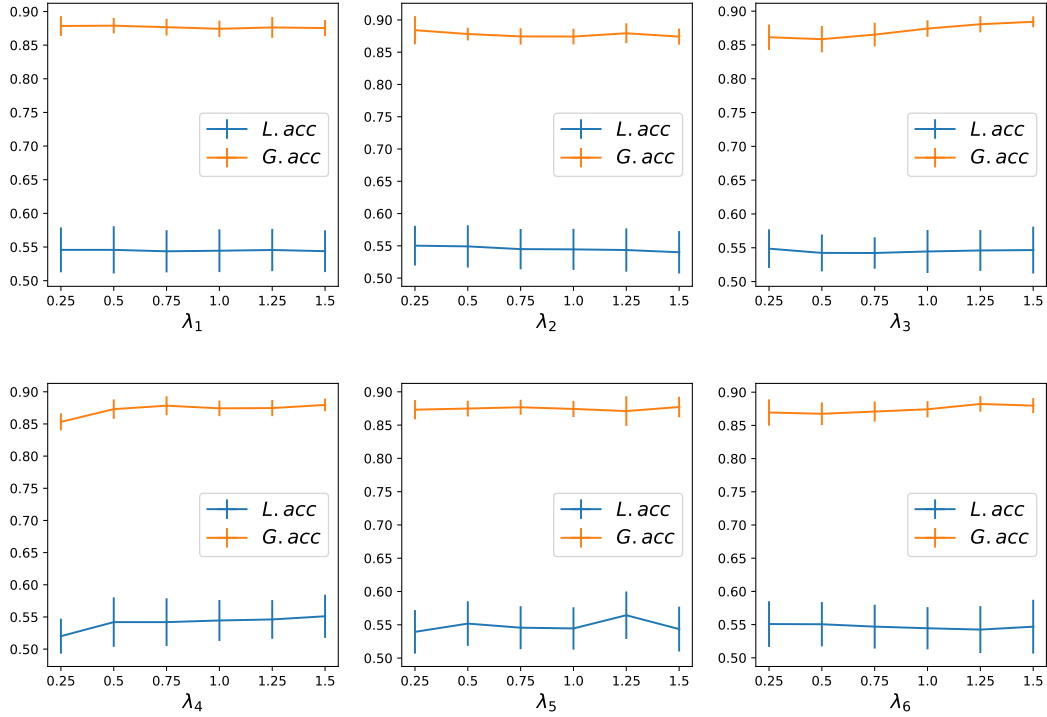
Div. con.	EMNIST					
	$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
\mathcal{L}_{div}^0	96.96±0.06	97.41±0.11	95.39±0.21	97.17±0.17	53.09±4.27	88.85±1.31
\mathcal{L}_{div}^1	96.98±0.05	97.40±0.11	95.43±0.27	97.16±0.19	53.65±4.12	88.51±0.80
\mathcal{L}_{div}^2	96.97±0.05	97.41±0.11	95.45±0.25	97.18±0.17	54.45±3.56	87.87±1.64

Table 16: Test performance (%) comparison among different diversity constraints used by FedMD-CG over FMNIST. Note that we omit the subscript i of diversity loss for client i .

Div. con.	FMNIST					
	$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
\mathcal{L}_{div}^0	83.93 \pm 0.20	87.79\pm0.17	78.58 \pm 1.58	84.69 \pm 0.46	42.20 \pm 3.42	70.47 \pm 1.79
\mathcal{L}_{div}^1	84.10 \pm 0.16	87.60 \pm 0.12	79.03\pm1.52	84.73\pm0.49	42.25 \pm 3.36	70.28 \pm 1.26
\mathcal{L}_{div}^2	84.32\pm0.18	87.18 \pm 0.13	79.00 \pm 1.43	84.47 \pm 0.38	42.55\pm3.68	71.09\pm1.01

Table 17: Test performance (%) comparison among different diversity constraints used by FedMD-CG over CIFAR-10. Note that we omit the subscript i of diversity loss for client i .

Div. con.	CIFAR-10					
	$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
\mathcal{L}_{div}^0	54.24 \pm 0.72	54.78 \pm 1.88	46.36 \pm 2.36	47.20 \pm 2.15	26.04 \pm 1.76	30.62 \pm 2.56
\mathcal{L}_{div}^1	54.81 \pm 0.71	54.90 \pm 1.73	46.38\pm2.37	47.53 \pm 2.14	26.14 \pm 1.80	30.70\pm2.55
\mathcal{L}_{div}^2	54.82\pm0.79	55.18\pm1.75	46.30 \pm 2.24	47.56\pm2.21	26.18\pm1.79	30.55 \pm 2.42

Figure 13: Test performance of FedMD-CG using varying hyperparameters on EMNIST with $\omega = 0.1$

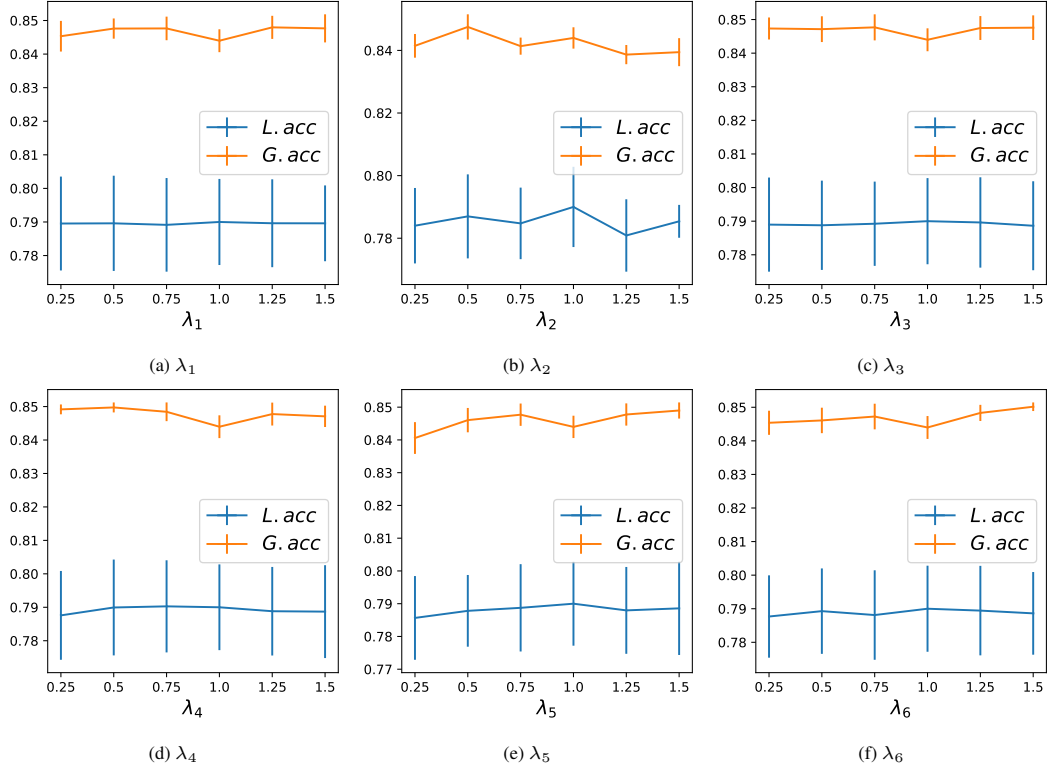


Figure 14: Test performance of FedMD-CG using varying hyperparameters on FMNIST with $\omega = 1.0$.

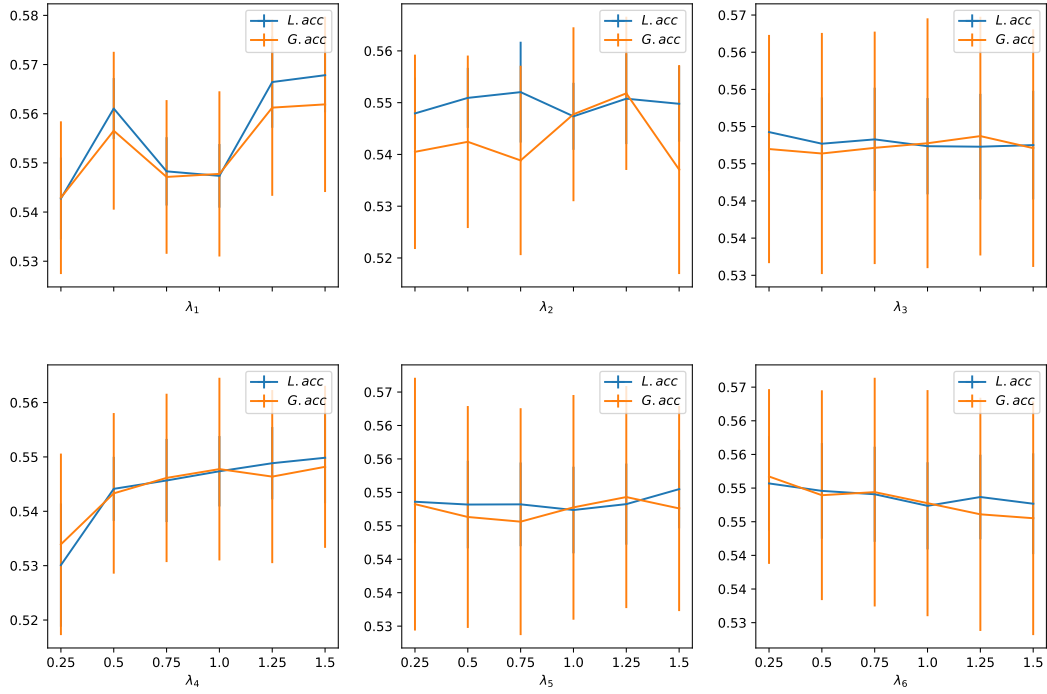


Figure 15: Test performance of FedMD-CG using varying hyperparameters on CIFAR-10 with $\omega = 10.0$.