

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

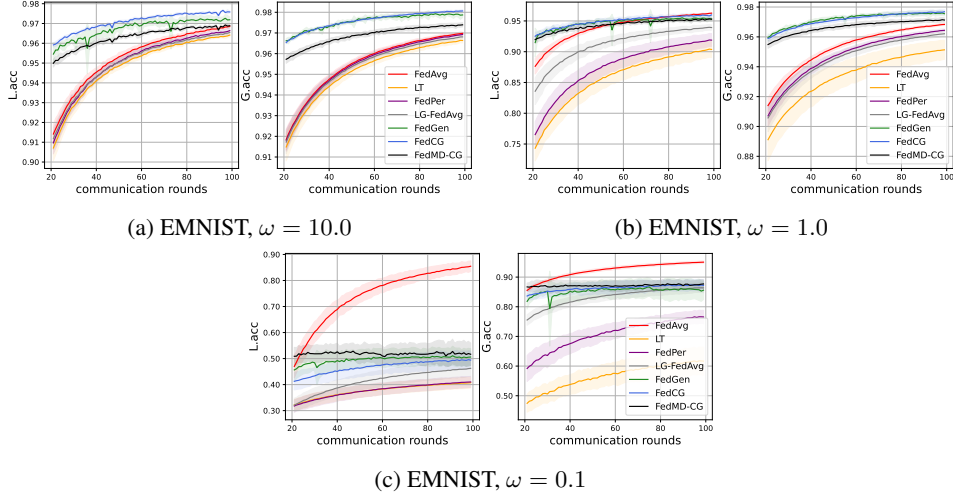


Figure 7: Learning curves for FedMD-CG as well as baselines over EMNIST.

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

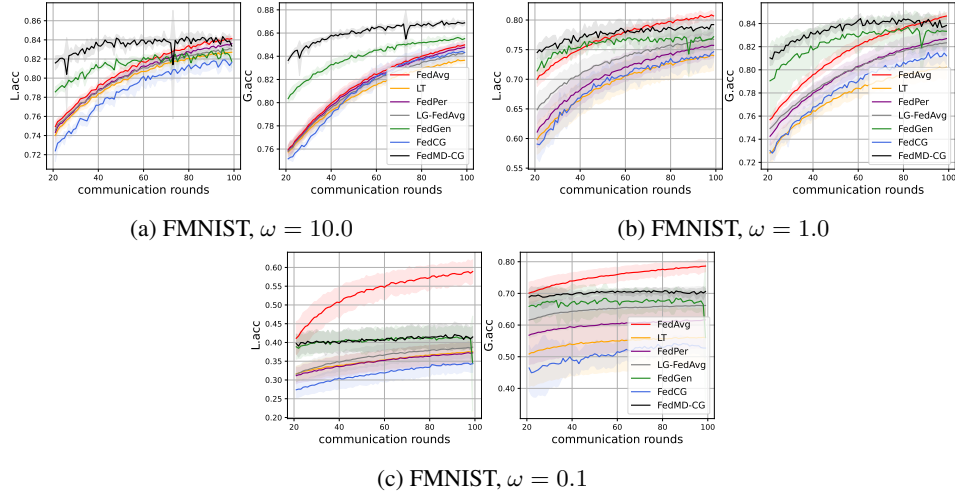


Figure 8: Learning curves for FedMD-CG as well as baselines over FMNIST.

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.

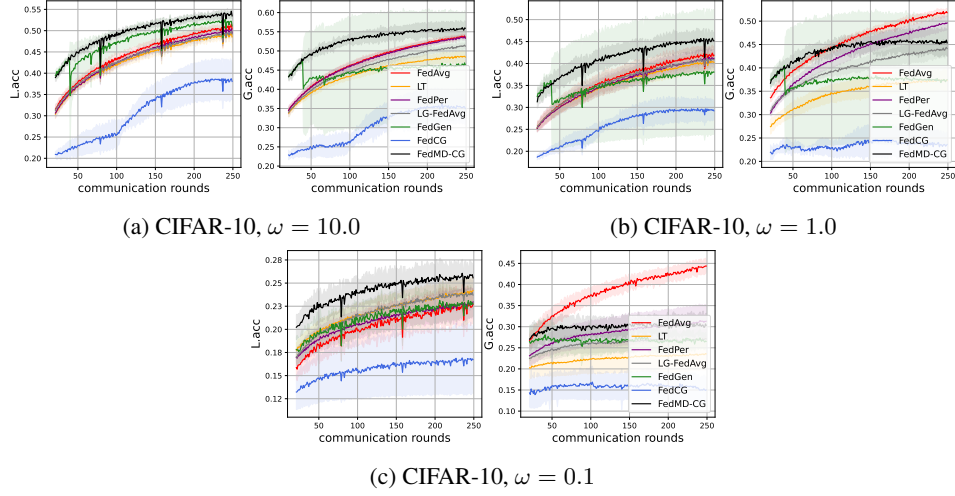


Figure 9: Learning curves for FedMD-CG as well as baselines over over CIFAR-10.

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.

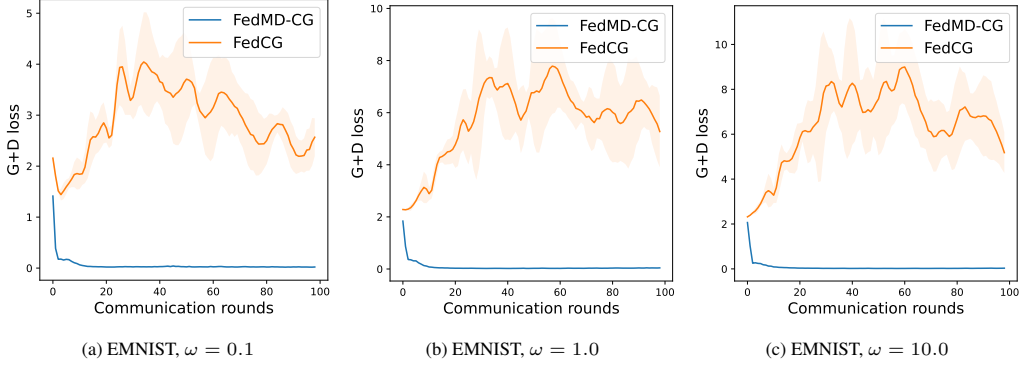


Figure 10: The consistency comparison between local generators and classifiers for FedCG and FedMD-CG w.r.t. AVE_agg^* over EMNIST. G+D loss denotes the classification loss of the local classifier on the output of the local generator.

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

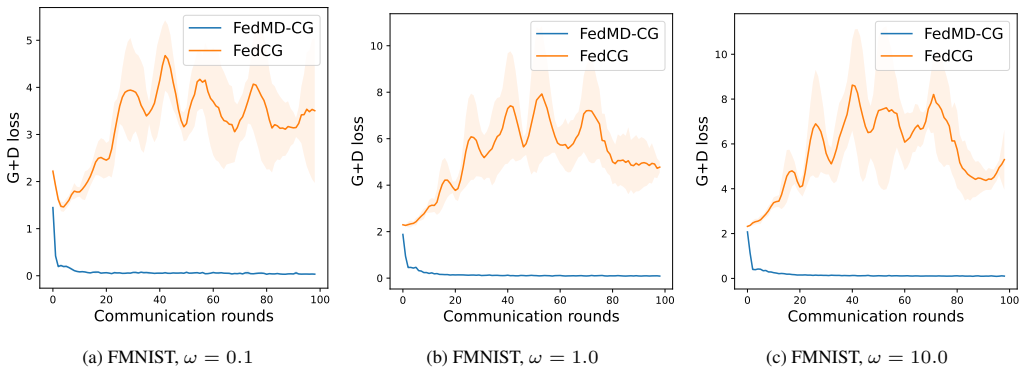


Figure 11: The consistency comparison between local generators and classifiers for FedCG and FedMD-CG w.r.t. AVE_agg^* over FMNIST. G+D loss denotes the classification loss of the local classifier on the output of the local generator.

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

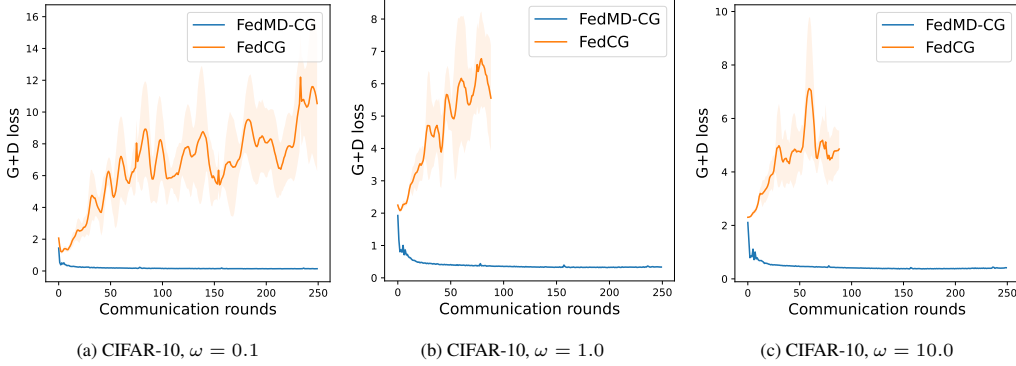


Figure 12: The consistency comparison between local generators and classifiers for FedCG and FedMD-CG w.r.t. AVE_agg^* over CIFAR-10. G+D loss denotes the classification loss of the local classifier on the output of the local generator.

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	$L.acc$	$G.acc$
$\rightarrow \mathcal{L}_{ce}$	52.51 \pm 3.61	85.44 \pm 1.40	$\leftarrow \mathcal{L}_{mse}$	49.33 \pm 3.27	81.09 \pm 4.49
$\rightarrow \mathcal{L}_{mse}$	52.85 \pm 3.17	86.47 \pm 1.54	$\leftarrow \mathcal{L}_{ce}$	51.87 \pm 2.67	82.97 \pm 1.54
$\rightarrow \mathcal{L}_{kl}$	52.31 \pm 2.29	84.70 \pm 1.03	$\leftarrow \mathcal{L}_{div}$	52.17 \pm 3.68	87.86 \pm 2.15
$\rightarrow \mathcal{L}_{ce}, \rightarrow \mathcal{L}_{mse}$	51.55 \pm 3.42	84.35 \pm 1.20	$\leftarrow \mathcal{L}_{mse}, \leftarrow \mathcal{L}_{ce}$	47.81 \pm 2.95	80.66 \pm 3.45
$\rightarrow \mathcal{L}_{ce}, \rightarrow \mathcal{L}_{kl}$	50.42 \pm 3.43	82.59 \pm 1.37	$\leftarrow \mathcal{L}_{mse}, \leftarrow \mathcal{L}_{div}$	47.12 \pm 2.65	80.49 \pm 2.05
$\rightarrow \mathcal{L}_{mse}, \rightarrow \mathcal{L}_{kl}$	51.34 \pm 3.16	85.63 \pm 1.07	$\leftarrow \mathcal{L}_{ce}, \leftarrow \mathcal{L}_{div}$	50.32 \pm 3.35	82.09 \pm 2.38
$\rightarrow \mathcal{L}_{ce}, \rightarrow \mathcal{L}_{mse}, \rightarrow \mathcal{L}_{kl}$	46.53 \pm 4.77	84.51 \pm 2.24	$\leftarrow \mathcal{L}_{mse}, \leftarrow \mathcal{L}_{ce}, \leftarrow \mathcal{L}_{div}$	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 .

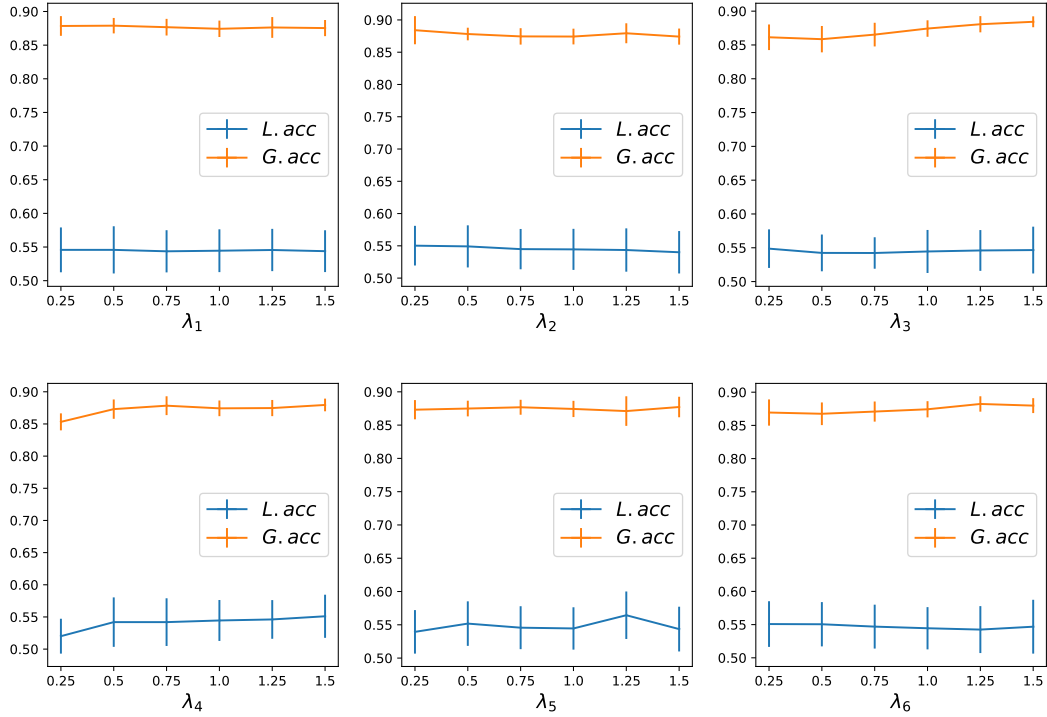
Div. con.	EMNIST					
	$\omega = 10.0$		$\omega = 1.0$		$\omega = 0.1$	
	$L.acc$	$G.acc$	$L.acc$	$G.acc$	$L.acc$	$G.acc$
\mathcal{L}_{div}^0	96.96±0.06	97.41±0.11	95.39±0.21	97.17±0.17	53.09±4.27	88.85±1.31
\mathcal{L}_{div}^1	96.98±0.05	97.40±0.11	95.43±0.27	97.16±0.19	53.65±4.12	88.51±0.80
\mathcal{L}_{div}^2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

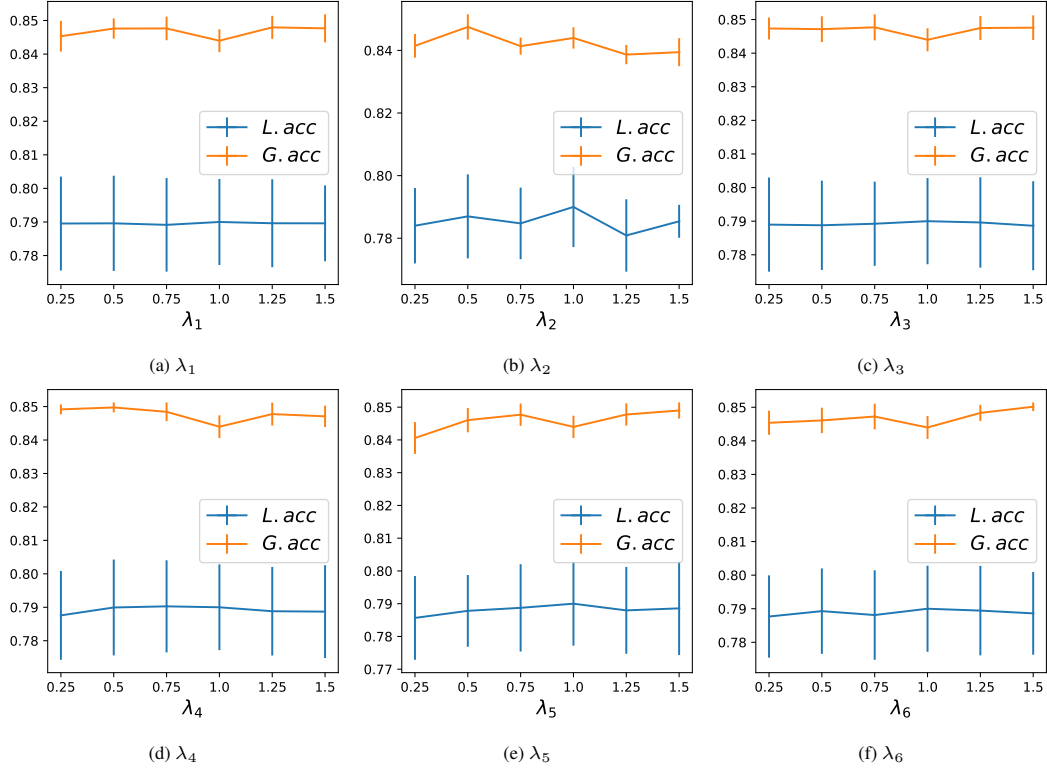


Figure 14: Test performance of FedMD-CG using varying hyperparameters on FMNIST with $\omega = 1.0$.

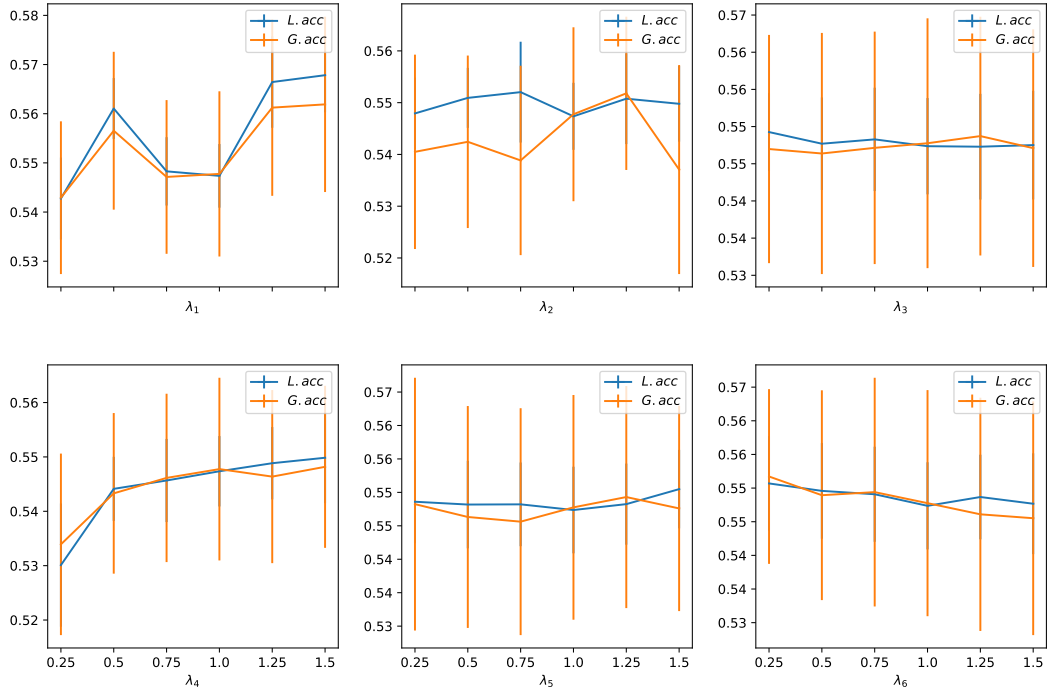


Figure 15: Test performance of FedMD-CG using varying hyperparameters on CIFAR-10 with $\omega = 10.0$.