META KNOWLEDGE CONDENSATION FOR FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

1 ADDITIONAL EXPERIMENTAL RESULTS

1.1 DATASETS

The additional experiments are also conducted on four benchmarks: MNIST (LeCun et al., 2010), SVHN (Netzer et al., 2011), CIFAR10 (Krizhevsky & Hinton, 2009), and CIFAR100 (Krizhevsky & Hinton, 2009).

1.2 COMPARATIVE STUDIES

Compared with Prior Works when setting α to 0.1 and 0.25: We set the α value in Dirichlet distribution $D(\alpha)$ (Zhu et al., 2021) to 0.1 and 0.25, and run all methods under limited communication budgets (10 rounds) on four datasets. We report the results in Table 1. As shown in Table 1, when communication budgets are limited (10 rounds) and the α value is set to 0.25 and 0.10, our method can still learn a model outperforming competing works by a remarkable margin.

Impact of the smooth parameter τ in Eq. 6: We conduct a study on four datasets to explore the impact of smooth parameter τ in Eq. 6. We set the τ value to 1.0, 5.0, and 10.0, respectively. As shown in Table 2, we achieve the highest performance in most cases when we set τ to 5.0. Therefore, we set τ in Eq. 6 to 5.0 for all our experiments.

2 DISCUSSIONS

Why does utilizing meta-knowledge decrease the required communication rounds? Our method utilizes extracted meta knowledge as normal training data to train a global model on the server. The meta knowledge is extracted from original data via a bi-level optimization, which encodes the "gradient of gradient" with respect to the model. The optimization methods based on the second order gradient generally have a higher convergence speed than the methods using the first order gradient (Battiti, 1992; Xie et al., 2022). Therefore, utilizing meta-knowledge endows our algorithm with a fast convergence speed and decreases the communication round number.

Why increasing the meta knowledge size can not necessarily improve final performance? In the meta knowledge extraction process, the calculated meta-knowledge in each batch represents the average of the model update direction. The average gradient is stable when the batch number increases in a certain range (from ×1 to ×10). As a result, increasing the meta-knowledge sizes does not necessarily increase the performance. Intuitively, the meta knowledge is highly dense and compressed, encoding the knowledge from original data (Zhou et al., 2022). In principle, using the meta knowledge approximates employing the original data. As the amount of information in the original data is constant, the training performance will not necessarily increase as the meta knowledge size increases.

We conduct an experiment on MNIST to show the information change between meta knowledge with different sizes. Concretely, we set the meta-knowledge size (S) as 10, 20, 30, 40, 50, 60, 70, and 80, respectively. As earth mover's distance (EMD) has been utilized to compute a structural distance between two data sets to determine their similarity (Zhang et al., 2020), we use it to evaluate differences according to meta-knowledge with different sizes. The results are listed in Table 3. It can be seen that the EMDs with respect to meta-knowledge sizes are stable, indicating the amount of

Table 1: Results with 10 rounds.

Setting	FedAvg	FedProx	FedDistill	FedEnsem	FedGen	FedMK
			MNIST			
α=0.10	61.95%	61.41%	58.46%	67.89%	64.83%	**77.37**%
α=0.25	69.52%	68.43%	71.78%	72.23%	73.41%	**90.07**%
			SVHN			
α=0.10	20.10%	18.39%	25.44%	24.60%	24.38%	**57.24**%
α=0.25	23.56%	25.01%	22.70%	23.21%	28.79%	**65.66**%
			CIFAR10			
α=0.10	23.71%	21.88%	24.93%	24.80%	20.16%	**38.45**%
α=0.25	21.85%	22.17%	20.84%	23.98%	22.94%	**40.75**%
			CIFAR100			
α=0.10	10.19%	9.41%	12.41%	10.64%	10.79%	**18.62**%
α=0.25	11.73%	10.43%	8.73%	12.42%	8.22%	**22.14**%

Table 2: Impact of the smooth parameter τ.

	τ=1.0	τ=5.0	τ=10.0
		MNIST	
α=0.50	91.70%	**92.95**%	91.79%
α=0.75	91.90%	**92.86**%	92.23%
α=1.0	91.53%	**93.63**%	91.91%
		SVHN	
α=0.50	72.24%	**74.11**%	71.60%
α=0.75	71.47%	**74.90**%	74.03%
α=1.0	71.74%	**74.84**%	72.19%
		CIFAR10	
α=0.50	**47.72**%	47.33%	46.62%
α=0.75	48.17%	49.04%	**49.27**%
α=1.0	47.82%	**50.32**%	48.54%
		CIFAR100	
α=0.50	26.06%	**26.74**%	26.45%
α=0.75	26.93%	**27.43**%	26.98%
α=1.0	25.43%	**28.20**%	26.15%

information in the meta-knowledge does not change significantly with respect to the meta knowledge size.

The possibility of restoring original data from meta-knowledge. We conduct an experiment on MNIST to explore the possibility of restoring data from extracted meta-knowledge. The results are shown in Figure 1. The original images are shown in the top row, and the extracted meta-knowledge is shown in the middle row. We feed the extracted meta-knowledge and a trained model to Deep Leakage (Zhu & Han, 2020), which is one of the state-of-the-art methods for restoring data from leaked knowledge. The data restored by Deep Leakage is shown in the bottom row. It can be seen that it is hard to construct correspondence between entries in restored data and original data.

The difference between FedGen and FedMK: There are significant differences between FedGen (Zhu et al., 2021) and our FedMK, which are listed as follows:

- [Local model training by original data v.s. Meta Knowledge extraction]: FedGen utilizes original data on local clients to train local models, while our method conducts meta knowledge extraction to synthesize meta knowledge, which is used for global model training on a server. In FedGen, the trained local models might diverge due to the data distribution variations among clients.

- [Global model aggregation v.s. Global model training]: FedGen constructs a global model by aggregating uploaded local models; while our method learns a global model based on meta knowledge uploaded from clients. In our method, the global model learning utilizes knowledge from all active clients, therefore mitigating the bias issue compared to FedGen.

- [The role of conditional generator]: The conditional generator in FedGen is trained on the server and transmitted to clients. On clients, it is used as a constraint in the local model training. On the contrary, the conditional generator in our method is trained and utilized on the server, participating in the global model training. Compared to FedGen, our method has a less communication cost without performance deterioration. In conclusion, compared to FedGen, our method performs more effectively and efficiently under both practical and pathological non-iid settings.

Table 3: EMDs with respect to meta-knowledge sizes.

Meta-Knowledge size(S)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
Difference w.r.t. S=10	0	10	20	30	40	50	60	70
EMD($meta_S, meta_{10}$)	0.000	0.023	0.054	0.046	0.053	0.045	0.045	0.043

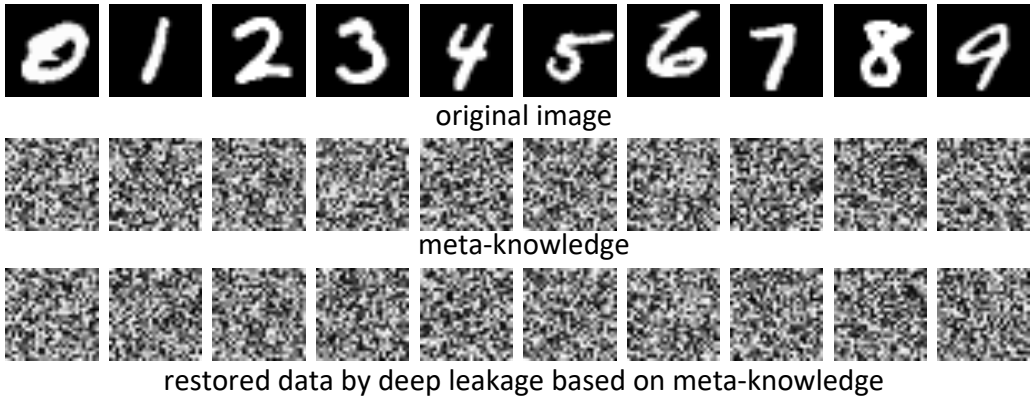

Figure 1: The visualization of original images (the top row), extracted meta-knowledge (the middle row), and restored data by deep leakage based on extracted meta-knowledge (the bottom row).

Algorithm 1: FedMK

Input: Original data $\mathcal{D}$; global parameters $\mathbf{W}_G$; generator parameter $\mathbf{w}^{\mathcal{G}}$; the communication budget .
Output: Optimal $\mathbf{W}_G^*$

while *not over the communication budget* **do**
  the server selects active clients C uniformly at random, broadcasts $\mathbf{W}_G$ to the selected clients C.
  ▷Federated Meta Knowledge Extraction on selected clients C:
  for *all user* $c \in C$ *in parallel* **do**
    $\mathbf{w}^c \leftarrow \mathbf{W}_G$;
    for *t = 1, ..., #Round* **do**
      conduct the conditional initialization: $\hat{\mathcal{D}}_{ini}^c \leftarrow \hat{\mathcal{D}}_{t-1}^{c'}, c' \sim randint[1, C], c' \neq c$;
      calculate dynamic weights by Eq. 6;
      generate $\hat{\mathcal{D}}^c$ by Eq. 3;
    end
    send the $\hat{\mathcal{D}}^c$ to the server.
  end
  ▷Global Model Training on the server:
  update generator parameter $\mathbf{w}^{\mathcal{G}}$ by Eq. 9;
  generate $\hat{\mathcal{D}}^{pseu}$ by the updated generator $\mathcal{G}$;
  update global parameter $\mathbf{W}_G$ by Eq. 10.
end
return $\mathbf{W}_G$ as $\mathbf{W}_G^*$;

3 ALGORITHM

The algorithm of FedMK is illustrated in Alg. 1.

REFERENCES

Roberto Battiti. First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 1992.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Corinna Cortes, and Chris Burges. Mnist handwritten digit database, 2010.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS-Workshop*, 2011.

Zeke Xie, Xinrui Wang, Huishuai Zhang, Issei Sato, and Masashi Sugiyama. Adaptive inertia: Disentangling the effects of adaptive learning rate and momentum. In *ICML*, 2022.

Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *CVPR*, 2020.

Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022.

Ligeng Zhu and Song Han. Deep leakage from gradients. In *NeurIPS*, 2020.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *ICML*, 2021.