An Upload-Efficient Scheme for Transferring Knowledge From a Server-Side Pre-trained Generator to Clients in Heterogeneous Federated Learning

Jianqing Zhang[1,*], Yang Liu[2,3,†], Yang Hua[4], Jian Cao[1,5,†]
[1]Shanghai Jiao Tong University [2]Institute for AI Industry Research (AIR), Tsinghua University
[3]Shanghai Artificial Intelligence Laboratory [4]Queen's University Belfast
[5]Shanghai Key Laboratory of Trusted Data Circulation and Governance in Web3
`tsingz@sjtu.edu.cn, liuy03@air.tsinghua.edu.cn, y.hua@qub.ac.uk, cao-jian@sjtu.edu.cn`

We provide more details and results about our work in the appendices. Here are the contents:

- Appendix A: The details and results of using the Stable Diffusion as the pre-trained generator on the server.
- Appendix B: The applicability of the knowledge transfer scheme of our FedKTL in the scenario with only one server and one edge client.
- Appendix C: Additional experimental details, such as download web links of pre-trained generators, hyperparameter settings, *etc*.
- Appendix D: The continued privacy-preserving discussion besides the main body with experimental results.
- Appendix E: Empirical convergence analysis.
- Appendix F: The effects of using different hyperparameter settings for our FedKTL.
- Appendix G: Additional ablation study regarding the ArcFace loss.
- Appendix H: Data distribution visualizations for different scenarios in our experiments.

A. Using the Stable Diffusion Model

A.1. Preliminaries

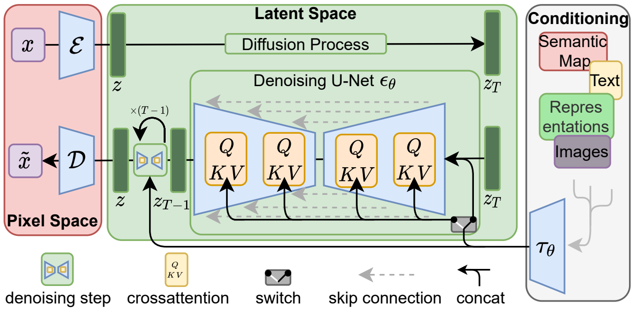


Figure 1. The main components in the Stable Diffusion [10].

Most publicly available pre-trained generators can map from a latent vector (or a matrix, which can be flattened to a vector) to an image, making them compatible with our FedKTL. In Stable Diffusion [10] (v1.5), the latent diffusion model (LDM) combined with VAE's decoder can map a latent vector z_T to an image $\tilde{x}$, which is similar with StyleGANs [4–6, 11] during generation, as shown in Fig. 1. Typically, z_T is randomly generated from a normal distribution. Except for the above similarities, Stable Diffusion includes a conditioning component, which, for instance, can convert a text prompt to a conditional vector and influence the diffusion process to generate images with semantics related to the given prompt. As our FedKTL is agnostic to the semantics of the images produced by the generator, one can select any valid text prompt, such as "*a cat*," and maintain it unchanged throughout the entire FL process.

A.2. Experimental Results

Due to the change of the pre-trained generator, we re-tune some hyperparameters. Specifically, we set $\eta_s = 0.1$, $\lambda = 0.01$, and $\mu = 100$ while maintaining the other hyperparameters consistent with those used for the StyleGAN-XL [11]. As shown in Tab. 1, using Stable Diffusion is also effective. While Stable Diffusion demonstrates excellent image generation performance, it's worth noting that the dimension of the latent vector is 16384, compared to 512 in the StyleGAN-XL. Mapping low-dimensional client prototypes (with a dimension of 10 for the classification task on Cifar10) to such a high dimension space while preserving their correlation is challenging. Perhaps a deeper feature transformer is required. We also show the generated images during the HtFL process in Fig. 2. With more iterations of HtFL, the generated images become clearer and more informative.

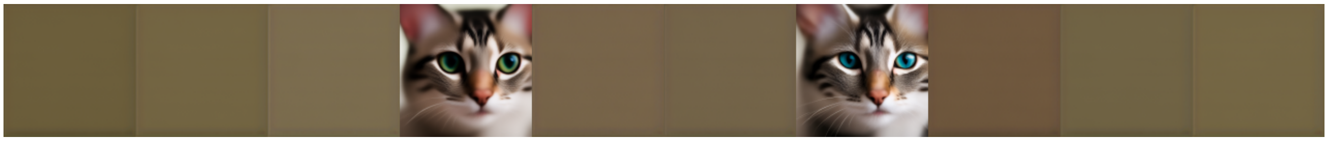
Generator	StyleGAN-XL	Stable Diffusion
Accuracy	87.63	87.71

Table 1. The test accuracy (%) of our FedKTL with different pre-trained generators on Cifar10 in the practical setting using HtFE$_8$.

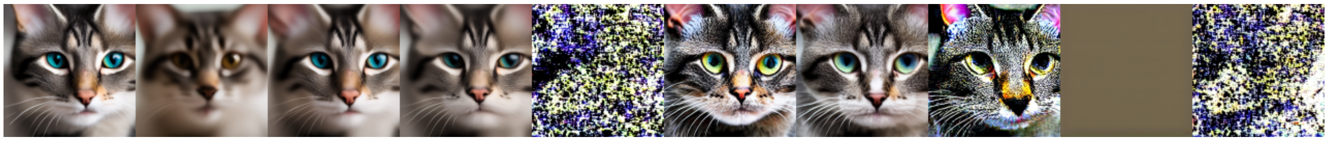
---

[*] Work done during internship at AIR.
[†] Corresponding authors.

(a) Iteration 0th


(b) Iteration 20th

(c) Iteration 40th


(d) Iteration 60th

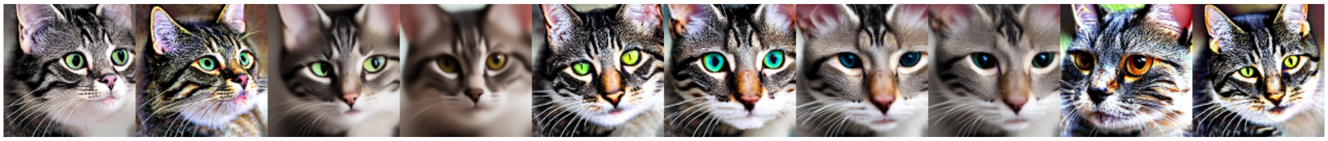

(e) Iteration 80th

Figure 2. The prototypical images generated by Stable Diffusion corresponding to all 10 classes of Cifar10 at different communication iterations during the HtFL process.

B. The Scenario With a Single Edge Client

In the traditional FL scenarios [9, 19], clients mainly fetch extra knowledge from the globally aggregated model parameters. From the view of an individual client, these global model parameters contain fused knowledge from other clients. In our FedKTL, except for the aggregated knowledge from clients, the pre-trained generator consists of common and valuable knowledge that can facilitate client training, particularly in addressing the data scarcity problem on edge devices. Therefore, the knowledge transfer scheme (*i.e.*, the Knowledge-Transfer Loop (KTL)) in our FedKTL offers an additional feature *beyond FL*, expanding its applicability to the scenarios with only one server and one edge client (*e.g.*, the cloud-edge scenarios) and broadening the scope of its application.

We can employ the KTL without modifying FedKTL's workload, as the aggregation step has no effect with only one client. We iteratively execute the knowledge transfer process in each training epoch for this client until the training of the client model converges. Specifically, in each training epoch, the client sends a request (*i.e.*, client prototypes) to the server, the server then sends a response (*i.e.*, image-vector pairs) back to the client, and the response further serves as an additional supervised task to promote client training.

By default, we adopt the StyleGAN-XL as the pre-trained generator on the server. On edge devices, data is usually insufficient [17]. In our considered scenario, the edge client has a few training samples, which is the primary reason this client requires additional common knowledge. Specifically, we only assign $\frac{1}{20}$, $\frac{1}{50}$, and $\frac{1}{200}$ of the Cifar100 dataset to the client, respectively, where the number of samples is the same for all the 100 class. In other words, *the client only has* 23, 9, *and* 2 *training samples per class* in these three settings, respectively. From the view of few-shot learning [15], they are *100-way 23-shot, 100-way 9-shot, and 100-way 2-shot* settings. Then, following the setting in the main body, we split the data into a training set (75%) and a test set (25%).

As demonstrated in Tab. 2, our KTL yields more improvement when the client has limited data, attributed to the introduction of additional pre-existing knowledge from the server-side pre-trained generator. However, according to the results for *100-way 23-shot*, if the data on the edge client is not particularly scarce, simply introducing a pre-

Settings	100-way 23-shot	100-way 9-shot	100-way 2-shot
Client Data	12.53±0.39	7.55±0.41	4.44±1.66
Our KTL	13.02±0.43	8.88±0.62	8.76±2.25
Improvement	0.49	1.33	4.32
Improvement Ratio	3.91%	17.61%	97.29%

Table 2. The test accuracy (%) with Cifar100's subsets on a single client using a small model *i.e.*, the 4-layer CNN.

trained generator on the server without collaboration with other clients (such as using HtFL) can hardly bring improvement. These findings regarding KTL highlight its ability to transfer knowledge from a pre-trained model to an edge device with very limited data.

C. Additional Experimental Details

Datasets, pre-trained generators, and environment. We use four datasets with their respective download links: Cifar10[1], Cifar100[2], Flowers102[3], and Tiny-ImageNet[4]. We can fetch the public pre-trained generators with their respective download links: StyleGAN-XL[5], StyleGAN3 (pre-trained on AFHQv2)[6], StyleGAN3 (pre-trained on Bench)[7], StyleGAN3 (pre-trained on FFHQ-U)[8], StyleGAN3 (pre-trained on WikiArt)[9], and Stable Diffusion (v1.5)[10]. All our experiments are conducted on a machine with 64 Intel(R) Xeon(R) Platinum 8362 CPUs, 256G memory, eight NVIDIA 3090 GPUs, and Ubuntu 20.04.4 LTS.

Hyperparameter settings. Besides the hyperparameter setting provided in the main body, we follow each baseline method's original paper for their respective hyperparameter settings. LG-FedAvg [8] has no additional hyperparameters. For FedGen [20], we set the noise dimension to 32, its generator learning rate to 0.1, its hidden dimension to be equal to the feature dimension, *i.e.*, K, and its server learning epochs to 100. For FedGH [18], we set the server learning rate to be the same as the client learning rate, *i.e.*, 0.01. For FML [12], we set its knowledge distillation hyperparameters $\alpha = 0.5$ and $\beta = 0.5$. For FedKD [16], we set its auxiliary model learning rate to be the same as the client learning rate, *i.e.*, 0.01, $T_{start} = 0.95$, and $T_{end} = 0.95$. For FedDistill [3], we set $\gamma = 1$. For FedProto [13], we set $\lambda = 0.1$. For our FedKTL, we set $K = C$, $\mu = 50$, $\lambda = 1$, $\eta_S = 0.01$ (server learning rate), $B_S = 100$ (server batch size), and $E_S = 100$ (the number of server training epochs), by using grid search in the following ranges on Tiny-ImageNet:

- μ: $\{1, 10, 20, 50, 80, 100, 200\}$
- λ: $\{0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100\}$
- η_S: $\{0.0001, 0.001, 0.01, 0.1, 1\}$
- B_S: $\{1, 10, 50, 100, 200, 500\}$
- E_S: $\{1, 10, 50, 100, 200, 500, 1000\}$

Besides, we use Adam [7] for F training following FedGen, set $s = 64$ and $m = 0.5$ following ArcFace loss [1], and use the radial basis function (RBF) kernel for the kernel function κ in L^{MMD}. We use these settings for all the tasks.

Auxiliary model in FML and FedKD. According to FedKD and FML, the auxiliary model needs to be designed as small as possible to reduce the communication overhead for model parameter transmitting, so we choose the smallest model to be the auxiliary model for FedKD and FML in any model heterogeneity scenarios.

D. Privacy-Preserving Discussion (Continued)

Here we further discuss the privacy-preserving capability of our FedKTL when a client has the potential to recover data from other clients. When a client receives additional ***global knowledge*** (with data belonging to the labels never seen before), ***the client is still unable to discern which image-vector pair belongs to which client (or group of clients), and thus cannot disclose the local data of other individual clients***. As a result, transmitting class-level prototypes is a common practice in FL (*e.g.*, FedProto [13]). Secondly, in *§3.3.5*, we have provided three reasons supporting the privacy-preserving capabilities of our FedKTL based on its design philosophy. Moreover, our FedKTL is ***compatible with privacy-preserving techniques***, such as adding noise, resulting in only a slight decrease in accuracy (see Table 3).

---

[1] https://pytorch.org/vision/main/generated/torchvision.datasets.CIFAR10.html

[2] https://pytorch.org/vision/stable/generated/torchvision.datasets.CIFAR100.html

[3] https://pytorch.org/vision/stable/generated/torchvision.datasets.Flowers102.html

[4] http://cs231n.stanford.edu/tiny-imagenet-200.zip

[5] https://s3.eu-central-1.amazonaws.com/avg-projects/stylegan_xl/models/imagenet64.pkl

[6] https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-t-afhqv2-512x512.pkl

[7] https://g-75671f.f5dc97.75bc.dn.glob.us/benches/network-snapshot-011000.pkl

[8] https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-r-ffhqu-256x256.pkl

[9] https://lambdalabs.com/blog/stylegan-3

[10] https://huggingface.co/runwayml/stable-diffusion-v1-5/tree/main

	—	NC	NG	NC + NG
FedKTL	53.16	52.73	51.16	50.51

Table 3. The test accuracy (%) on Flowers102 in the practical setting using HtFE_8 with *noisy uploaded client prototypes* (NC) and *noisy generated image-vector pairs* (NG). Following FedPCL [14]'s privacy-preserving settings, we add Gaussian noise to the images and vectors before transmitting with a controllable parameters scale (s) and perturbation coefficient (p). We follow FedPCL to set $s = 0.05$ and $p = 0.2$ for vectors (or prototypes) and set $s = 0.2$ and $p = 0.2$ for images.

E. Convergence Analysis

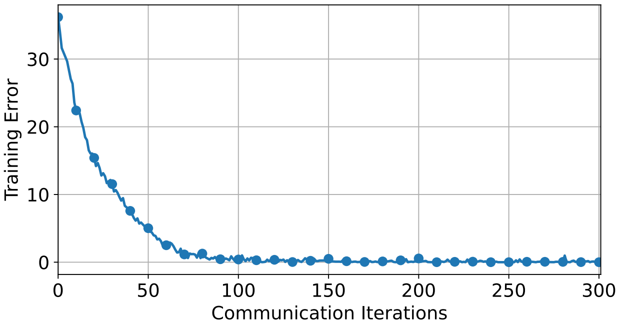


Figure 3. The training error curve for our FedKTL on Flowers102 using HtFE_8 in the practical setting.

We show the training error curve of our FedKTL in Fig. 3, where we calculate the training error on clients' training sets in the same way as calculating test accuracy in the main body. According to Fig. 3, our FedKTL optimizes quickly in the initial 80 iterations and gradually converges in the subsequent iterations. Besides, our FedKTL maintains stable performance after converging at around the 120th iteration.

F. Hyperparameter Experiments

To study the effect of hyperparameters in our FedKTL, we vary the value of each hyperparameter when keeping other parameters fixed, which are tuned on Tiny-ImageNet. Increasing the ETF dimension K transmits more client knowledge to the server and improves accuracy, but this approach increases communication cost (see Tab. 4). To save communication resources, we set $K = C$ in practice. According to Tab. 5, setting a value larger than 50 for μ can achieve an accuracy larger than 53%, which means that the importance of L_i^M should be emphasized. However, overly large values of μ can also lead to a decrease in accuracy. In contrast, in Tab. 6 we find that the optimal value for the server's λ is typically less than 10 on Flowers102, as too large values of λ tend to weaken the domain alignment. Regarding the server hyperparameters η_S and E_S, our FedKTL can achieve better performance when using larger values for these parameters, as shown in Tab. 7 and Tab. 8. On the contrary, a smaller B_S usually improves the performance of our FedKTL (see Tab. 9). In addition to the findings mentioned above, we also discover that the best combination of hyperparameters for Tiny-ImageNet is not necessarily the best for the Flowers102 dataset. While the default hyperparameter setting performs excellently, it is important to note that for a new dataset, one may need to re-tune the hyperparameters to achieve the best performance.

G. Additional Ablation Study

By default, in the main body, we adopt the ArcFace loss [1] as L_i^A. Specifically, we have

$$L_i^A = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_i} - \log \frac{e^{s(\cos{(\theta_y + m)})}}{e^{s(\cos{(\theta_y + m)})} + \sum_{c=1,c\neq y}^{C} e^{s\cos\theta_c}}, \tag{1}$$

where θ_y is the angle between $g_i(\boldsymbol{x})$ and $\boldsymbol{v}_y$, s and m are the re-scale and additive hyperparameters [1], respectively. Here, we adopt a more classical practice for L_i^A. Specifically, we replace the ArcFace loss with the contrastive loss [1, 2]. In other words, we set $s = 1$ and $m = 0$ in L_i^A to achieve this replacement, so we have

$$L_i^A = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_i} - \log \frac{e^{\cos\theta_y}}{e^{\cos\theta_y} + \sum_{c=1,c\neq y}^{C} e^{\cos\theta_c}}. \tag{2}$$

We denote this variant of FedKTL as $*L_i^A$.

We conduct experiments on four datasets using Eq. (2) and show the test accuracy in Tab. 10. We observe that the impact of replacing L_i^M varies across different datasets. However, it is consistent that removing the ArcFace loss leads to a decrease in accuracy.

H. Visualizations of Data Distributions

We illustrate the data distributions (including training and test data) in the experiments here.

	$K = C = 102$	$K = 500$	$K = 1000$
Accuracy	53.16	54.42	53.90
Upload	0.07M	0.35M	0.69M

Table 4. The test accuracy (%) and upload communication cost of our FedKTL on Flowers102 in the practical setting using HtFE_8 with different K. "M" is shorter for a million.

	$\mu = 1$	$\mu = 10$	$\mu = 20$	$\mu = 50$	$\mu = 100$	$\mu = 200$
Accuracy	48.09	51.01	52.83	53.16	53.99	53.43

Table 5. The test accuracy (%) of our FedKTL on Flowers102 in the practical setting using HtFE_8 with different μ.

	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$	$\lambda = 100$
Accuracy	53.28	53.30	53.16	53.06	48.45

Table 6. The test accuracy (%) of our FedKTL on Flowers102 in the practical setting using HtFE_8 with different λ.

	$\eta_S = 0.0001$	$\eta_S = 0.001$	$\eta_S = 0.01$	$\eta_S = 0.1$
Accuracy	49.84	51.51	53.16	53.62

Table 7. The test accuracy (%) of our FedKTL on Flowers102 in the practical setting using HtFE_8 with different η_S.

	$E_S = 10$	$E_S = 50$	$E_S = 100$	$E_S = 200$	$E_S = 500$
Accuracy	52.00	52.94	53.16	53.83	54.35

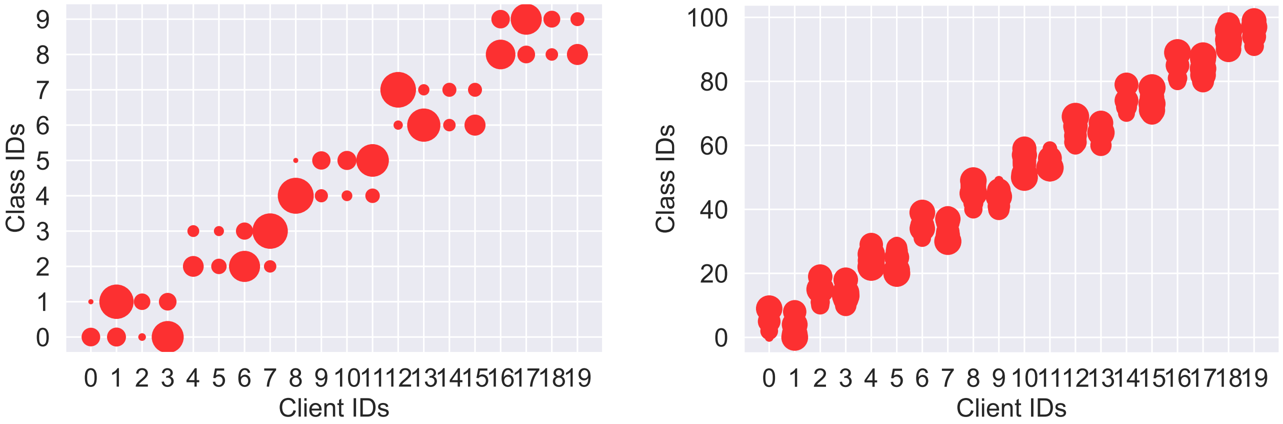
Table 8. The test accuracy (%) of our FedKTL on Flowers102 in the practical setting using HtFE_8 with different E_S.

	$B_S = 10$	$B_S = 50$	$B_S = 100$	$B_S = 200$	$B_S = 500$
Accuracy	54.97	54.76	53.16	53.93	53.26

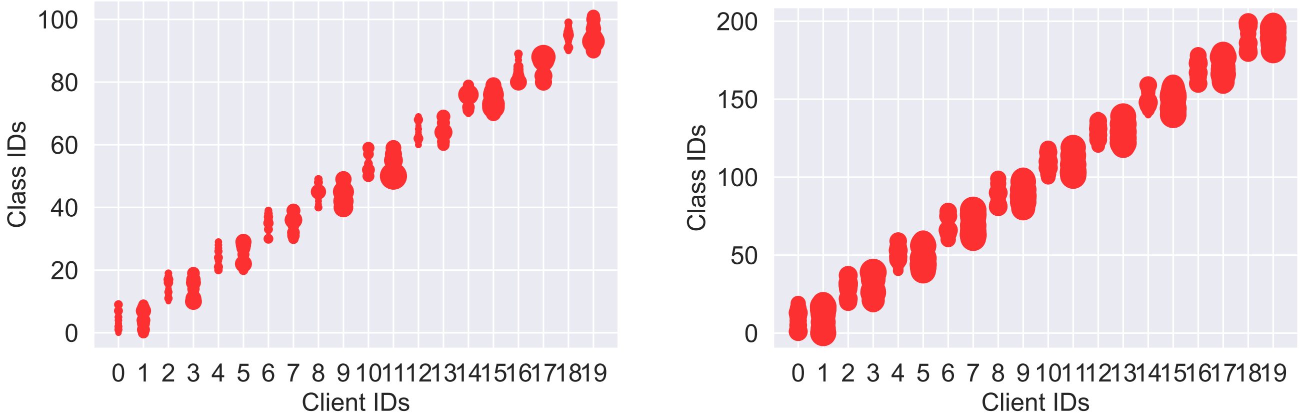
Table 9. The test accuracy (%) of our FedKTL on Flowers102 in the practical setting using HtFE_8 with different B_S.

	Cifar10	Cifar100	Flowers102	Tiny-ImageNet
FedKTL	87.63	46.94	53.16	28.17
$*L_i^A$	85.28	41.63	51.30	27.52
Δ	-2.35	-5.31	-1.86	-0.65

Table 10. The test accuracy (%) of our FedKTL's variant $*L_i^A$ on four datasets in the practical setting using HtFE_8.

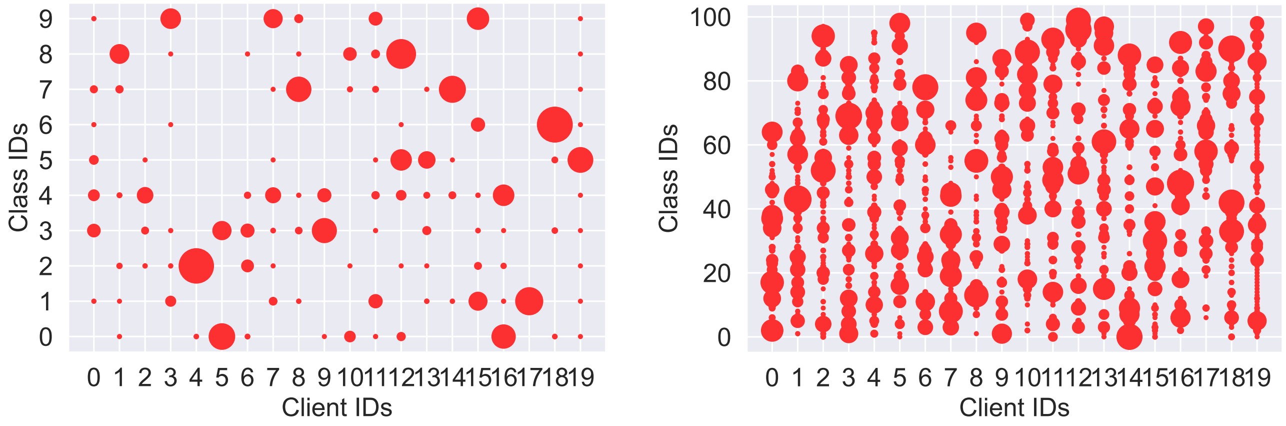


(a) Cifar10 (b) Cifar100



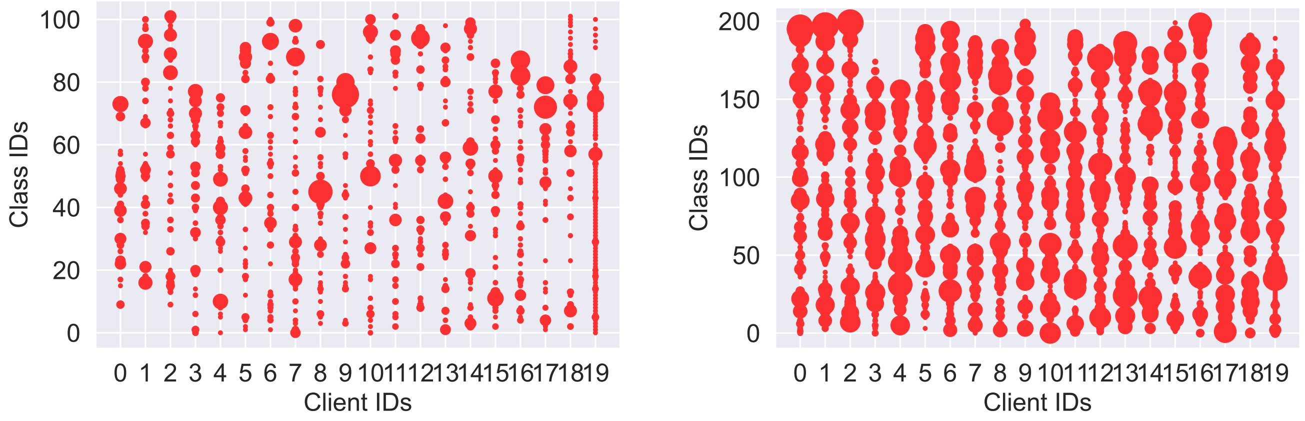
(c) Flowers102 (d) Tiny-ImageNet

Figure 4. The data distribution of each client on Cifar10, Cifar100, Flowers102, and Tiny-ImageNet, respectively, in the pathological settings. The size of a circle represents the number of samples.



(a) Cifar10

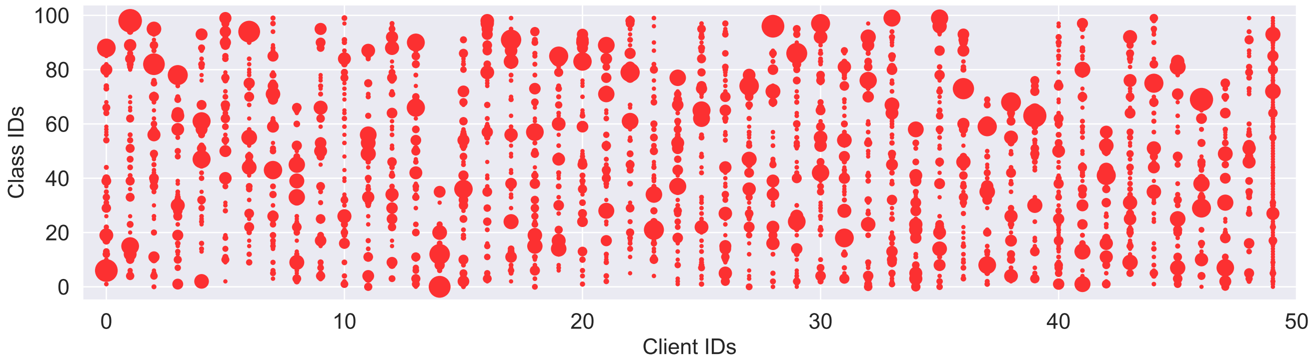
(b) Cifar100



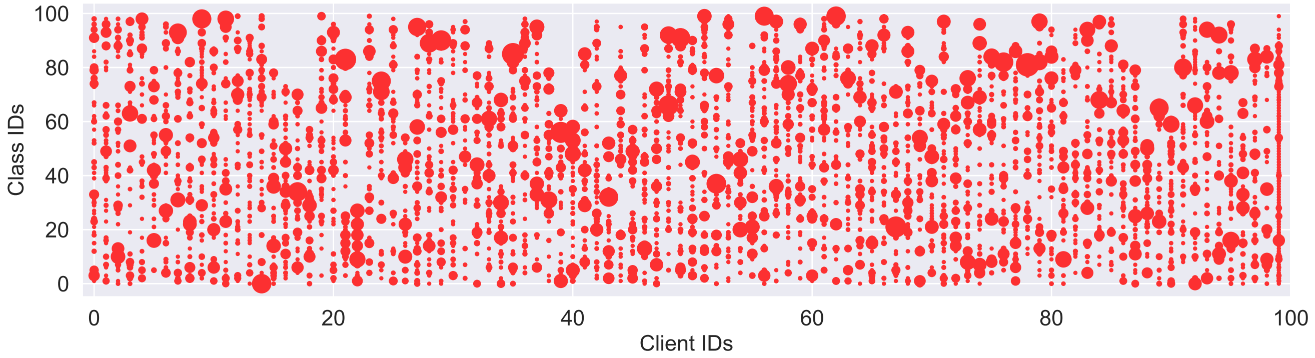
(c) Flowers102

(d) Tiny-ImageNet

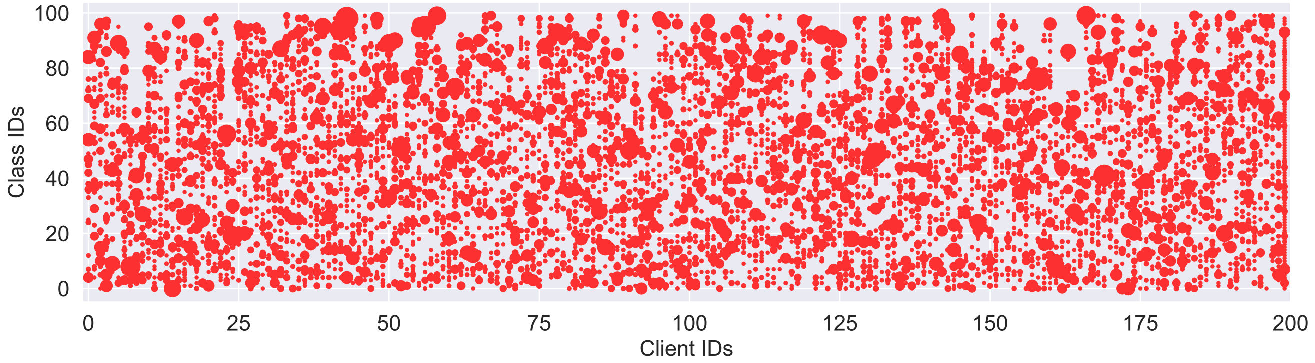
Figure 5. The data distribution of each client on Cifar10, Cifar100, Flowers102, and Tiny-ImageNet, respectively, in practical settings ($\beta = 0.1$). The size of a circle represents the number of samples.



(a) 50 clients



(b) 100 clients



(c) 200 clients

Figure 6. The data distribution of each client on Cifar100 in the practical setting ($\beta = 0.1$) with 50, 100, and 200 clients, respectively. The size of a circle represents the number of samples.

References

[1] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4

[2] Munawar Hayat, Salman Khan, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Gaussian affinity for max-margin class imbalanced learning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 6469–6479, 2019. 4

[3] 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. 3

[4] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[5] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1

[7] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 3

[8] 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. 3

[9] 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 (AISTATS)*, 2017. 2

[10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[11] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022. 1

[12] Tao Shen, Jie Zhang, Xinkang Jia, Fengda Zhang, Gang Huang, Pan Zhou, Kun Kuang, Fei Wu, and Chao Wu. Federated mutual learning. *arXiv preprint arXiv:2006.16765*, 2020. 3

[13] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated Prototype Learning across Heterogeneous Clients. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022. 3

[14] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 4

[15] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020. 2

[16] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1): 2032, 2022. 3

[17] Qiong Wu, Xu Chen, Zhi Zhou, and Junshan Zhang. Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring. *IEEE Transactions on Mobile Computing*, 21(8):2818–2832, 2020. 2

[18] Liping Yi, Gang Wang, Xiaoguang Liu, Zhuan Shi, and Han Yu. Fedgh: Heterogeneous federated learning with generalized global header. In *Proceedings of the 31st ACM International Conference on Multimedia*, 2023. 3

[19] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. FedALA: Adaptive Local Aggregation for Personalized Federated Learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023. 2

[20] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-Free Knowledge Distillation for Heterogeneous Federated Learning. In *International Conference on Machine Learning (ICML)*, 2021. 3