

A Detailed Related Work

Label distribution skew poses a challenge in training a model to perform well across all clients. Numerous works have attempted to address label imbalance, which can be mainly categorized into four groups: **1) Incorporate Momentum and Adaptive Methods.** Many recent works try to incorporate optimization methods into FL. For example, FedOpt [6] consists of ServerOpt for the server and ClientOpt for the clients, which are used to update the global and local models, respectively. Both ClientOpt and ServerOpt can be set as any momentum and adaptive optimizers to enhance performance. **2) Reduce the Bias in Local Model Updates.** When dealing with heterogeneous data, local updates can introduce bias into the convergence process. Therefore, some methods focus on reducing this bias. SCAF-FOLD [2] effectively employs control variates, a technique aimed at reducing variance, to correct client-drift within its local updates. **3) Regularize Local Objective Functions.** Some works attempt to penalize local models that deviate significantly from the global model by applying regularization to the local objectives. For instance, FedProx [3] employs the Euclidean distance between local and global models as a regularization function to prevent local models from drifting towards their respective local minima. **4) Consider Alternative Aggregation Methods.** The weighting aggregation determines the ultimate convergence point of the global model. Therefore, recent methods try to design an effective weighting scheme. For example, FedNova [7] optimizes the number of epochs in local updates and introduces a normalized averaging scheme to eliminate inconsistencies in objectives.

However, such optimization methods cannot address the fundamental issue of data distribution heterogeneity, meaning they cannot achieve truly outstanding performance. Moreover, another solution involves generating synthetic data from distributed data sources, as discussed in previous studies [1, 4, 5]. However, these methods cannot generate high-quality data or prevent potential privacy leakage, resulting in subpar performance. Different from the previous approaches, we start by recovering the global data distribution at the local level, thus aiming to fundamentally address the label imbalance in FL.

B More Implementation Details

We implement all methods using PyTorch. Before training, we utilize Stable Diffusion to generate synthetic images for each client based on the prompt “A photo of {class}, real world images, high resolution”. In terms of data volume, for each class, we generate 1.3k synthetic images for the ImageFruit and ImageNet100 datasets, 100 synthetic images for the CUB and Cars datasets, and 3k synthetic images for the EuroSAT dataset. For the adaptive fine-tuning approach, we employ LoRA to fine-tune the Stable Diffusion within each client using their local data and set the α of U-Net to 0.8.

During the global generalization task, the batch size is set to 128, and the round of communication is set to 200. In each communication round, every client updates their weights for 5 epochs using the SGD optimizer. During the local personalization task, we select synthetic data for each client based on the categories of their real data. We fine-tune the global model at each client using their local data for 50 epochs with the SGD optimizer, resulting in a personalized local model for each client.

Algorithm 1: Pytorch-like Pseudocode of Our ReGL.

```

datasets = []
SD = vanilla_Stable_Diffusion()
for client in all_clients:
    # use the real images of each client to fine tune the SD individually
    adaptive_SD = fine_tuning(real_images, SD)
    # use adaptive SD to generate synthetic images
    syn_images = adaptive_SD.generation(prompts, noise)
    # package the real and synthetic images
    all_data = aggregation(real_images, syn_images)
    datasets.append(all_data)

# global generalization task
global_model = network()
for com in Rounds:
    local_weights = []
    # randomly select clients
    for client in selected_clients:
        # SGD update on real and synthetic images
        data = datasets[client]
        weights = local_update(data, global_model)
        local_weights.append(weights)
    # updating the global model by average
    global_model = model_aggregation(local_weights)

# local personalization task
local_models = []
for client in all_clients:
    # each client fine-tune the model for their personalized tasks
    data = datasets[client]
    local_weight = local_update(data, global_model)
    local_models.append(local_weight)

return global_model, local_models

```

C More Ablation Studies and Analyses

Local Epochs Here, we increase the computation load per client in each round by expanding the number of local epochs, which we denote as E_{local} . We conduct numerous experiments on ImageFruit and ImageNet100 datasets, and compare our ReGL with previous algorithms in Tab. 1. It is evident that, when E_{local} is set to 1, the performance of all methods degrades significantly. In this scenario, the number of local updates is too small, resulting in inadequate model training. Nonetheless, our method continues to exhibit competitive performance, achieving an accuracy of 60.7% on the ImageFruit dataset and 60.1% on the ImageNet100 dataset with $\beta = 0.01$. As we increase the value of E_{local} , the performance of all methods generally improves. However, excessively large values of E_{local} can lead to overfitting. Therefore, the optimal choice for our method is 5 epochs per round.

Number of Clients To analyze the effect of the number of clients on performance, we train these methods with different numbers of clients M on ImageFruit and ImageNet100 datasets. Specifically, we set $M = \{5, 50, 100\}$ for the ImageFruit and $M = \{10, 50, 100\}$ for the ImageNet100, with a skew degree of $\beta = 0.01$. As demonstrated in Tab. 2, when M increases, the performance of all previous methods experiences a significant decline. This decline is especially pronounced when $M = 100$, where the accuracy of previous methods drops to approximately 10% on ImageFruit and 15% on ImageNet100. We conjecture that as the number of clients increases, there are more skewed local models, leading to a poorer aggregated model. However, our method remains robust as M increases, with almost no

E_{local}	Methods	ImageFruit		ImageNet100	
		$\beta = 0.01$	$\beta = 0.5$	$\beta = 0.01$	$\beta = 0.5$
1	FedAvg	23.7	39.2	28.8	33.6
	FedNova	27.9	40.6	32.9	37.7
	MOON	28.5	41.8	31.8	38.5
	Ours	60.7	65.2	60.1	63.2
5	FedAvg	30.1	52.1	37.0	43.9
	FedNova	31.9	54.0	43.7	51.3
	MOON	33.7	55.9	45.6	53.8
	Ours	77.3	<u>79.2</u>	<u>77.2</u>	78.6
10	FedAvg	31.9	52.7	36.5	42.8
	FedNova	32.3	53.5	43.9	52.0
	MOON	33.1	54.6	47.0	53.1
	Ours	76.8	80.2	78.2	<u>77.9</u>
20	FedAvg	30.6	50.6	34.1	43.1
	FedNova	31.8	52.3	43.6	53.5
	MOON	30.9	51.6	46.2	54.1
	Ours	<u>77.0</u>	78.6	76.0	77.2

Table 1: Performance comparison of different local epochs.

Methods	ImageFruit			ImageNet100		
	$M = 5$	$M = 50$	$M = 100$	$M = 10$	$M = 50$	$M = 100$
FedAvg	30.1	14.5	9.1	37.0	20.5	13.3
FedProx	30.7	14.9	9.6	38.8	20.8	14.1
FedNova	31.9	18.8	11.7	43.7	22.6	14.9
FedOpt	32.8	19.6	11.9	44.1	23.0	16.8
MOON	33.7	20.2	12.7	45.6	25.9	17.1
Ours	77.3	77.0	76.2	77.2	76.8	75.0

Table 2: Performance comparison of different number of clients.

performance degradation. Based on these results, we can conclude that our method is highly suitable for scenarios involving a large number of clients. In cases where the number of clients is particularly high, for instance, exceeding 100, our method outperforms the previous approaches by about 60%.

Analysis of Class-level Accuracy Label distribution skew can lead to class inconsistency across clients, so we compare the accuracy of FedAvg and our method for each class before and after a certain local update on the ImageFruit dataset using the $\beta = 0.01$ setting. *For the sake of display, we generate synthetic images for each class in our method to ensure that the total number of real and synthetic images is 1.3k, which is slightly different from the settings in our main experiment.* The performance is shown in Fig. 1. Here, all local models on the test set have the same test accuracy before local updates, because these local models are equivalent to the global model. First, we can observe that in FedAvg, the local model is restricted to learning samples solely from the majority classes, leading to a sharp decline in accuracy for the remaining classes. This indicates that label distribution skew can lead



Figure 1: Class-level accuracy of FedAvg and our method on the skewed ImageFruit dataset. The blue histogram and the orange histogram represent the number of real and synthetic images for each class, respectively. The green line and the red line indicate the accuracy of each class before and after a certain local update, respectively.

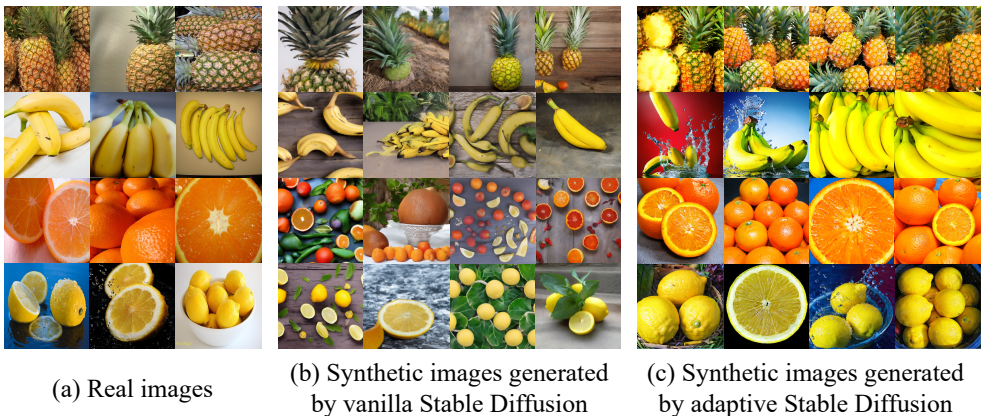


Figure 2: Visualization of real and synthetic images in ImageFruit dataset. While there is a significant difference between the real images and synthetic images generated by vanilla Stable Diffusion, adaptive approach can generate synthetic images in the style of real images.

to a biased model, severely impacting global model performance. While in our method, we eliminate label skew by generating synthetic images for each class, which enables a more effective local model update. The test accuracy of each class improves after local updates, *resulting in a more robust federated learning system under label distribution skew.*

References

- [1] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. In *2019 IEEE international parallel and distributed processing symposium (IPDPS)*, pages 866–877. IEEE, 2019.
- [2] 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*, pages 5132–5143. PMLR, 2020.
- [3] 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, 2020.
- [4] Vaikkunth Mugunthan, Vignesh Gokul, Lalana Kagal, and Shlomo Dubnov. Bias-free fedgan: A federated approach to generate bias-free datasets. *arXiv preprint arXiv:2103.09876*, 2021.
- [5] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228*, 2020.
- [6] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.
- [7] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.