Fine-tuning Global Model via Data-Free Knowledge Distillation for Non-IID Federated Learning

Lin Zhang[1,4] Li Shen[2] Liang Ding[3] Dacheng Tao[2,3] Ling-Yu Duan[1,4]
[1] Peking University, Beijing, China [2] JD Explore Academy, Beijing, China
[3] The University of Sydney, Sydney, Australia [4] Peng Cheng Laboratory, Shenzhen, China
{zhanglin.imre, lingyu}@pku.edu.cn, {mathshenli, dacheng.tao}@gmail.com
ldin3097@sydney.edu.au

1. Supplementary

1.1. Exploration of Long-Tail problem

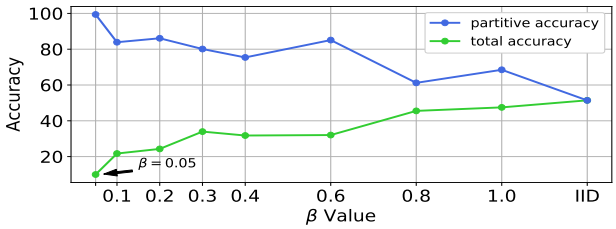


Figure 1. Test accuracy of model trained on class-imbalanced data.

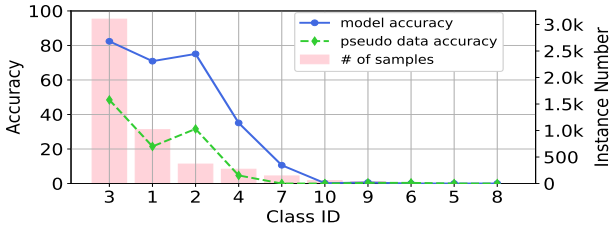


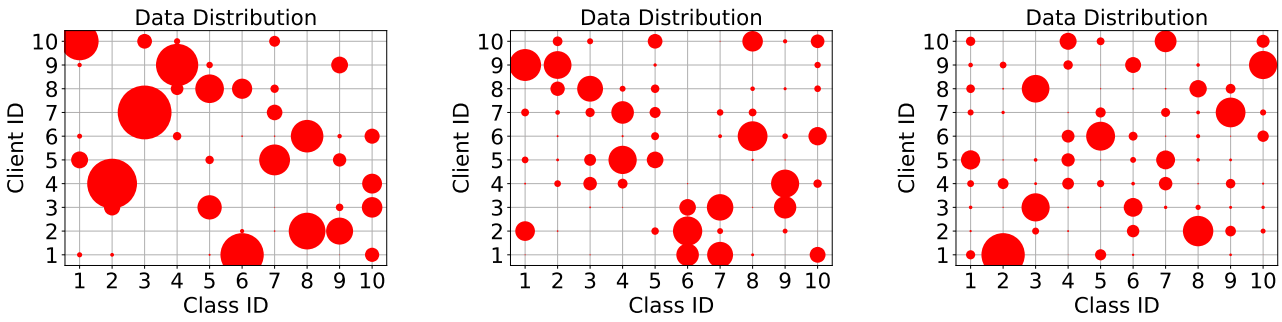
Figure 2. Correlation of model accuracy, pseudo data accuracy and the instance number on each class. Note that the class IDs are ordered via the instance number.

To explore the influence of long-tailed data on model performance, we train models using multiple subsets of CIFAR10, which have different degrees of imbalance. Here the subset is generated by Dirichlet distribution $\mathbf{Dir}(\beta)$, where a smaller β indicates more imbalanced data. The data number of each subset is 5000, and the architecture of the model is ResNet34 [2]. The results are illustrated in Figure 1. Here, the curves in green and blue are the test accuracy on *total test data* and *partitive test data* respectively, where the distribution of partitive test data is the same as the distribution of training data. We can see there is a performance gap between two curves, and the gap becomes larger when the degree of imbalance is increased. This is because the model only learns the majority classes, and these classes also dominate the partitive test data, thus the model achieves high accuracy on partitive test data; whereas for the total test data that contains balanced data for every class, the model can not correctly predict the data of minority classes, thus the model yields lower test accuracy on total test data. The results in Figure 1 verifies that the model tends to learn majority data from imbalanced training data and ignore the minority classes. In the following, we term the model trained using long-tailed data as "the biased model".

To further explore the influence of the biased model on pseudo data generation, we evaluate the accuracy of a biased model trained by a class-imbalanced CIFAR10 subset, and the quality of pseudo data generated via the biased model. The data quality is displayed in terms of the percentage of pseudo data that are correctly classified by a well-trained classifier, which is trained on all data of CIFAR10 and achieves 81.38% test accuracy. The results are illustrated in Figure 2. We can see that the model tends to learn majority classes and yields extremely low even zero accuracies for minority classes 7,10 and 9. Moreover, the quality of pseudo data is highly related to original data distribution. For the minority classes, the test accuracy of the pseudo data is less than 10%, i.e., the quality of the pseudo data is even worse than random noise. This indicates that the pseudo data generated via biased model could be invalid to conduct knowledge transfer, which motivates as to customize the sample probability of label during data generation to facilitate effective knowledge transfer.

1.2. Visualization of Data Heterogeneity

In Figure 3, we figure out the data distributions of clients that generated by Dirichlet distribution $\mathbf{Dir}(\beta)$ with different β as well as IID data distributions. For each β value, we display the data distributions of 10 clients. In Figure 3, the data distributions of clients are significantly dif-



(a) $\beta = 0.1$ (b) $\beta = 0.3$ (c) $\beta = 0.6$

(d) $\beta = 1.0$ (e) $\beta = 10.0$ (f) IID

Figure 3. Visualization of the instance number per class allocated to each clients (indicated by dot size), for different β values of Dirichlet distribution $\mathbf{Dir}(\beta)$.

Table 1. The architectures of generators used in Section 4.1 $\sim$ Section 4.3.

(a) Generator for FedFTG and FedDF

$z \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ $m =$Map$(y) \in \mathbb{R}^M, y \in [1, ..., M]$
FC$(z) \rightarrow 4096$ FC$(m) \rightarrow 4096$
Concat $\rightarrow 8192$
Reshape, BN $\rightarrow 128 \times 8 \times 8$
Conv2D, BN, LeakyReLU $\rightarrow 128 \times 8 \times 8$
Upsampling $\rightarrow 128 \times 16 \times 16$
Conv2D, BN, LeakyReLU $\rightarrow 64 \times 16 \times 16$
Upsampling $\rightarrow 64 \times 32 \times 32$
Conv2D, Tanh $\rightarrow 3 \times 32 \times 32$

(b) Generator for FedGen

$z \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ $m =$Map$(y) \in \mathbb{R}^M, y \in [1, ..., M]$
FC$(z) \rightarrow 4096$ FC$(m) \rightarrow 4096$
Concat, BN $\rightarrow 8192$
FC, BN, LeakyReLU $\rightarrow 8192$
FC $\rightarrow 512$

ferent when β is small, and the client even has no data for some classes. When β grows, the data is distributed more evenly in each client, and the discrepancy of data distributions among clients becomes smaller.

1.3. Detailed Hyperparameters

Here we introduce the setting of hyperparameters for baselines during experiments. For FedProx, the proximal regularization parameter μ is $1e-4$. α in FedDyn is $1e-2$. We set the local update round in SCAFFOLD following [1], which is 50 according to our experiment setting. Following [3], we set $\tau = 0.5$, tune μ from $\{0.1, 1, 5\}$ and report the best result. For FedGen and FedDF, the learning rate for the generator is the same as FedFTG, i.e., it is initialized as 0.01 and is decayed quadratically with weight 0.998. As Resnet18 only has one fully-connected layer, l in FedGen is $L-1$, where L is the total layer number.

1.4. Detailed Architecture of Generator

Table 1 lists the architectures of generators for FedFTG, FedDF and FedGen used in Section 4.1 $\sim$ Section 4.3. Here, d is the dimension of noise data z, and it is 100 and

Table 2. The architectures of generators used in Section 4.4.

(a) Generator for FedFTG and FedDF

$z \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ $m =$Map$(y) \in \mathbb{R}^M, y \in [1, ..., M]$
FC$(z) \rightarrow s^2$ FC$(m) \rightarrow s^2$
Concat $\rightarrow 2s^2$
Reshape, BN $\rightarrow 512 \times (s\backslash 16) \times (s\backslash 16)$
Conv2D, BN, LeakyReLU $\rightarrow 256 \times (s\backslash 8) \times (s\backslash 8)$
Conv2D, BN, LeakyReLU $\rightarrow 128 \times (s\backslash 4) \times (s\backslash 4)$
Conv2D, BN, LeakyReLU $\rightarrow 64 \times (s\backslash 2) \times (s\backslash 2)$
Conv2D, BN, LeakyReLU $\rightarrow 64 \times s \times s$
Conv2D, Tanh $\rightarrow 3 \times s \times s$

(b) Generator for FedGen

$z \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ $m =$Map$(y) \in \mathbb{R}^M, y \in [1, ..., M]$
FC$(z) \rightarrow s^2$ FC$(m) \rightarrow s^2$
Concat, BN $\rightarrow 2s^2$
FC, BN, LeakyReLU $\rightarrow s^2$
FC, BN, LeakyReLU $\rightarrow s^2$
FC $\rightarrow 512$

Table 3. Evaluation of different FL methods on CIFAR10 and CIFAR100 ($\beta = 0.6$), in terms of the number of communication rounds to reach target test accuracy (acc). Note that we highlight the **best** and ***second best*** results in bold.

	CIFAR10		CIFAR100	
	$acc = 75\%$	$acc = 80\%$	$acc = 40\%$	$acc = 50\%$
FedAvg	104.33±6.67	270.67±13.33	81.67±2.33	563.67±163.33
FedProx	109.67±8.33	263.0±27.0	81.67±11.33	476.00±199.00
MOON	102.67±1.33	252.33±32.67	83.67±3.33	354.00±21.00
FedDyn	**72.67**±7.33	**133.33**±28.67	***56.00***±6.00	213.67±6.33
SCAFFOLD	77.00±3.00	161.00±8.00	61.67±7.33	***186.33***±10.67
FedGen	114.00±8.00	284.33±30.67	82.00±5.00	571.33±78.67
FedDF	97.67±8.33	246.33±24.67	90.00±6.00	445.00±42.00
FedFTG	***73.67***±4.33	***143.33***±5.67	**55.00**±3.00	**152.33**±10.67

256 for CIFAR10 and CIFAR100, respectively. M is the class number of datasets, and it is 10 and 100 for CIFAR10 and CIFAR100 respectively. The inplace of LeakReLU is 0.2 here. Note that in Table 1(b) the output of generator is 512-dimensional, as the input of the last FC layer in ResNet18 is 512-dimensional. If using the other classifiers, the dimension of the generator's output should be adjusted accordingly.

Table 2 lists the architectures of generators used in Section 4.4. Here $d = 256$ for all the datasets MIO-TCD, CompCar and Tiny-ImageNet. s is the image size, and $s = 112, 112, 64$ for MIO-TCD, CompCar and Tiny-ImageNet respectively. Note that for the experiments of VGG11 in Table 3 in the main paper, we also adopt these two generators for FedDF, FedGen and FedFTG.

1.5. Supplementary Experiment Results

Table 3 illustrates the communication rounds of different methods to reach the target test accuracy (75% and 80% for CIFAR10, 40% and 50% for CIFAR100) when $\beta = 0.6$, which is a supplement to Table 2 in the main paper. Same as Table 2, FedFTG achieves the second best and the best convergence for CIFAR10 and CIFAR100 respectively. Besides, it greatly reduces the round numbers required by its FL optimizer SCAFFOLD.

Table 4. Test Accuracy (%) of different methods on CIFAR10 using VGG11 and ResNet34 networks ($\beta = 0.3$).

	VGG11	ResNet34
FedAvg	82.05±0.59	80.48±0.89
FedProx	82.10±0.53	81.02±0.53
FedDyn	85.38±0.44	81.13±1.11
MOON	83.69±0.76	81.15±0.46
SCAFFOLD	86.78±0.37	83.31±0.71
FedGen	84.38±0.56	80.72±0.44
FedDF	84.71±0.78	81.20±0.46
FedFTG	**87.46**±0.49	**85.00**±0.45

Table 4 displays the test accuracy when adopting VGG11 [4] and ResNet34 [2] as the classifier. In this table, FedFTG yields the best performance in all scenarios, which validates the effectiveness of FedFTG on various architectures of deep neural network.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2020. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 1, 3

[3] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021. 2

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3