Consensus Optimization at Representation: Improving Personalized Federated Learning via Data-Centric Regularization

Heng Zhu, Arya Mazumdar
University of California, San Diego

Abstract

Federated learning is a large scale machine learning training paradigm where data is distributed across clients, and can be highly heterogeneous from one client to another. To ensure personalization in client models, and at the same time to ensure that the local models have enough commonality (i.e., prevent "client-drift"), it has been recently proposed to cast the federated learning problem as a consensus optimization problem, where local models are trained on local data, but are forced to be similar via a regularization term. In this paper we propose an improved federated learning algorithm, where we ensure consensus optimization at the *representation* part of each local client, and not on whole local models. This algorithm naturally takes into account that today's deep networks are often partitioned into a feature extraction part (representation) and a prediction part. Our algorithm ensures greater flexibility compared to previous works on exact shared representation in highly heterogeneous settings, as it has been seen that the representation part can differ substantially with data distribution. We validate its good performance experimentally in standard datasets.

1 Introduction

Federated learning (FL) has attracted much attention from the machine learning community recently due to rapid development of distributed intelligent devices and the demand of data privacy protection in large scale learning models. A typical FL framework is a machine learning training paradigm that includes a central server to aggregate the local information from participating clients to update a global model. The local data of each client should not be shared with other clients and should ideally be kept private up to certain degree also from the server Konečnỳ et al. (2016); McMahan et al. (2017); Kairouz & McMahan (2021). With M clients, a standard FL algorithm usually tries to solve the following optimization problem:

$$\min_{\omega} \frac{1}{M} \sum_{i=1}^{M} f_i(\omega) \tag{1}$$

where ω is a global model updated at the server, $f_i(\omega)$ is the local objective function at i-th client (the *empirical risk functions* at each of the client evaluated at their respective data samples). At each iteration, a local (stochastic) gradient or the entire local model is sent to the server for global model update.

However, in the context of FL, the data distribution across different clients are usually highly non-identical and heterogeneous. Thus in many practical applications, a single global model is not sufficient to satisfy the requirements of all the clients. To tackle this issue, many personalized FL methods have been proposed to allow each client to maintain a local model. A popular formulation of the problem is to use the concept of consensus optimization Smith et al. (2017); T Dinh et al. (2020);

International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023 (FL@FM-NeurIPS'23). This workshop does not have official proceedings and this paper is non-archival.

Li et al. (2021), that replaces the optimization problem of eq. (1) with the following:

$$\min_{\omega_0,\{\omega_i\}_{i=1}^M} \frac{1}{M}\sum_{i=1}^M f_i(\omega_i) + \frac{\lambda}{2}\|\omega_i - \omega_0\|^2 \tag{2}$$

where ω_0 is the global model maintained at the server, ω_i is an unique local model at i-th client, and λ is a hyper-parameter to balance the local training and forced consensus. The local models are not required to be the exactly same but the regularization forces them to be close to each other, and the parameter λ provides a flexibility to fit local data distribution.

Recent success of centralized multi-task learning is based on the realization that different tasks have shared common representation Bengio et al. (2013); Collins et al. (2021). Inspired by this observation, several studies have tried to exploit shared representation in personalized federated learning to achieve better local performance Arivazhagan et al. (2019); Collins et al. (2021); Pillutla et al. (2022). In this setting, at a high level, the local prediction model at each client is divided into two parts, including a representation part common to all clients. This motivates our first question:

Q1: Can we force the consensus (cf. eq. 2) on the representation level, not the whole model level?

Note that, a regularization at the representation part will include less number of variables, and therefore potentially is less expensive (e.g., in taking gradients) than a constraint on entire model. Indeed, in modern machine learning tasks, the model is usually a deep neural network consisting of a feature extractor and a prediction head. In the personalized FL works mentioned above, the deep neural network model is partitioned into a feature extractor u and prediction head v. They consider the following optimization problem across different clients:

$$\min_{u,\{v_i\}_{i=1}^M} \frac{1}{M}\sum_{i=1}^M f_i(u, v_i) \tag{3}$$

where u is a global feature extractor mapping inputs to a low dimensional space, v_i is the local prediction head at i-th client. The server only maintains the global feature extractor u, not the whole model, and broadcasts it to all the clients at each communication round. This method decouples the representation part and prediction part and obtains better performance on heterogeneous data Collins et al. (2021); Pillutla et al. (2022). However, as we will show in the next section, for different data distributions even the feature extractors can be different across clients. This motivates our second question:

Q2: Can we further allow the feature extractor in one client to be different from others while still learning information on shared representations from other clients?

Motivated by the two questions, in this work we propose a consensus optimization problem at the representation level. The local models are trained with its own data, and with a regularization term to force consensus on representation between local feature extractor and global feature extractor. In this work the global feature extractor is still trained via FedAvg-like method, i.e., from the average of the local feature extractors in all the participating clients. Thus we can write our problem as:

$$\min_{\{u_i\}_{i=0}^M,\{v_i\}_{i=1}^M} \frac{1}{M}\sum_{i=1}^M f_i(u_i, v_i) + \frac{1}{M}\sum_{i=1}^M f_i(u_0, v_{0,i}) + \frac{\lambda}{2}\frac{1}{M}\sum_{i=1}^M H_i(u_i, u_0) \tag{4}$$

where u_i and v_i are the local feature extractor and local prediction head at i-th client, $i = 1, \ldots, M$, respectively, u_0 is a global feature extractor maintained at the server. $v_{0,i}$ is the local prediction head at i-th client so that only the feature extractor is aggregated at the server. $H_i(u_i, u_0)$ is a regularization term to force the representation of the ith client u_i, which is defined on local dataset, and u_0 to be close. In this formulation the local feature extractors are no longer exactly same for each client, which provides more flexibility to fit highly heterogeneous data. The local models are almost trained locally except that their intermediate representations are forced to be close to each other. The local parameters are not covered by the global parameters in the training process, retaining more local knowledge. Fig. 1 displays an overview of our proposed framework. One point that we would like to stress: our regularization of the representation part is data-driven (compare with the regularization term in eq. 2).

For this formulation of the problem, we propose a new federated learning algorithm (detailed in Algorithm 3.1 and outlined in Fig. 1) based on distributed stochastic gradient descent. A limitation of this formulation is that the global feature extractor u_0 is still obtained by the average of the local feature extractors, so that each client needs to send part of the local model to the server, which still remains privacy issues. We aim to train the global feature extractor without knowing the exact model parameters to reduce privacy risk.

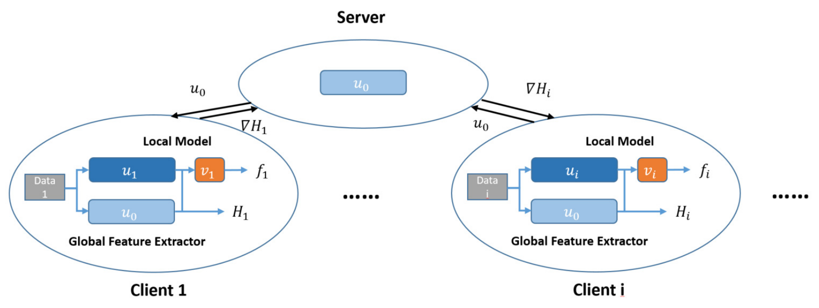


Figure 1: An overview of FedReCo.

2 Motivation and Problem Statement

2.1 Representation similarity across clients in different layers

The FedAvg-like algorithms suffer from this "client drift", which makes the local model far from global model within one communication round. In what follows, we will show the influence of client drift on the representations of different layers in one neural network model. We conduct an experiment on CIFAR10 dataset with a small 5-layer CNN model and ResNet18 He et al. (2016). There are 10 clients, each with 2 classes of data in the CIFAR10 dataset. We train the models via the FedAvg algorithm Li et al. (2020) and after local iterations within one communication round t, we measure the similarity between the representations of local model ω_i^t and global model ω^t before aggregation. We use the centered kernel alignment (CKA) measurement Kornblith et al. (2019); Nguyen et al. (2021); Li et al. (2023) to quantify the similarity of representations.

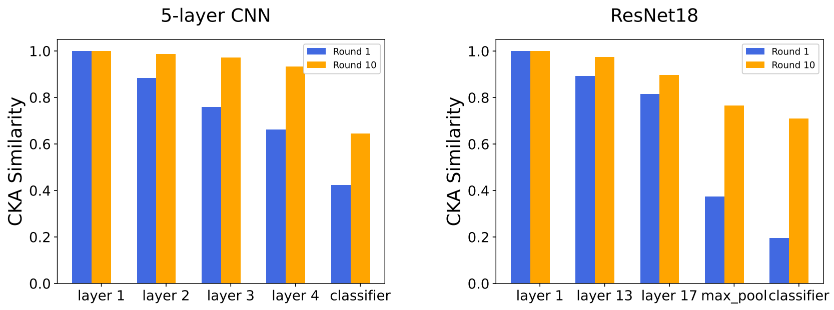


Figure 2: CKA Similarity for different layers.

Fig. 2 display the CKA similarity of representations after 1 round and 10 rounds of training, respectively. After only 1 round local training, the similarity decreases with deeper layers for both models. When training continues, the similarity between local model and global model increases for all the layers. After 10 rounds of training, the first four layers of 5-layer CNN model become close to global model, while the similarity of last (classifier) layer is still low. It shows that the FedAvg can learn a shared representation before the final classifier layer. However, even the previous layers are slightly dissimilar (similarity strictly less than 1). It is more obvious for larger ResNet18 model. Even after many rounds of training, the similarity decreases with deeper layers. The same phenomena has also been observed in Li et al. (2023) for a VGG model.

This observation motivates our work to consider a framework to allow different feature extractors in different clients while still learning the shared representations. The classifier layer or prediction head has the largest difference between local model and global model, thus we wish to train it completely locally. For the feature extractor, we train it locally, but with a regularization term to force it to learn representations from a global model.

2.2 Representation consensus optimization problem

Let us know formally formulate our problem. Consider a federated learning system with M clients, each client i with N samples $\{x_{ij} \in \mathbb{R}^{d_x}, y_{ij} \in \mathbb{R}^{d_y}\}_{j=1}^{N}, i = 1,...,M$. For a designed neural network, the model is partitioned into a feature extractor u and a prediction head v. For i-th client, it maintains its own local model u_i and v_i. And the server maintains a global feature extractor u_0. If we are to mandate that the representations of local data to be the same for local feature extractor and global feature extractor, then the representation consensus optimization problem would be,

$$\min_{u_0, \{u_i, v_i\}_{i=1}^{M}} \frac{1}{M}\sum_{i=1}^{M} f_i(u_i, v_i) + \frac{1}{M}\sum_{i=1}^{M} f_i(u_0, v_{0,i}) \quad \text{s.t. } h_{ij}(u_i) = h_{ij}(u_0), i = 1,2,...,M, j = 1,2,...,N$$

where $f_i(u_i, v_i) = \frac{1}{N}\sum_{j=1}^{N} f_i(u_i, v_i | x_{ij}, y_{ij})$ is the empirical loss function with N samples, and $h_{ij}(u) \triangleq h_{ij}(x_{ij}|u)$ is the mapping function $\mathbb{R}^{d_x} \to \mathbb{R}^{p}$ that maps input x_{ij} to a intermediate representation with dimension p.

However we do not need the representations to be the exactly same for local feature extractor and global feature extractor. Thus we only put a ℓ_2-norm regularization term to constrain the local training:

$$\min_{u_0, \{u_i, v_i\}_{i=1}^{M}} F(u_0, \{u_i, v_i\}_{i=1}^{M}) \triangleq \frac{1}{M}\sum_{i=1}^{M} f_i(u_i, v_i) + \frac{1}{M}\sum_{i=1}^{M} f_i(u_0, v_{0,i}) + \frac{\lambda}{2}\frac{1}{M}\sum_{i=1}^{M} H_i(u_i, u_0) \quad (5)$$

where $H_i(u_i, u_0) = \frac{1}{N}\sum_{j=1}^{N} \|h_{ij}(u_i) - h_{ij}(u_0)\|^2$ is the regularization term to force the representations of all the local data samples to be close for local feature extractors and global feature extractor.

3 FedReCo Algorithm

3.1 Algorithm Description

Since $H_i(u_i, u_0)$ is based on the local data samples, we can apply stochastic gradient descent (SGD) to solve the problem 5. We first update global model $u_0, v_{0,i}$, and then update the local model u_i and v_i with the global feature extractor u_0. When updating u_i, v_i, we can exploit the same batch of data samples to calculate the stochastic gradients of $f_i(u_i, v_i)$ and $H_i(u_i, u_0)$ simultaneously, just passing the same batch of samples twice to model $\{u_i, v_i\}$ and feature extractor u_0, respectively. Similarly, when updating u_0, We can exploit the same batch of data samples to calculate the stochastic gradients of $f_i(u_0, v_{0,i})$ and $H_i(u_i, u_0)$ simultaneously. To reduce the communication burden, we can perform multiple local SGD steps before transmitting the stochastic gradient. And due to the decoupling of feature extractor and prediction head, we can apply different learning rates and numbers of local steps to the two parts, respectively. In the following we use symbol $\tilde{\nabla}$ to represent stochastic gradient.

Specifically, our proposed FedReCo (Representation Consensus) algorithm is as follows: At each communication round t, the server broadcasts the u_0^t to all the clients. When updating u_0 and $v_{0,i}$, the i-th client first updates the prediction head $v_{0,i}$ via K_v SGD local steps with learning rate η_v, then fixs $v_{0,i}$ and updates $u_{0,i}$ from u_0^t via K_u local steps with learning rate η_u. When updating u_i, v_i, each client first updates the local prediction head v_i via K_v SGD local steps with learning rate η_v. Then the client fixes v_i and updates local feature extractor u_i via K_u local steps with learning rate η_u. After local training, the client sends $u_{0,i}$ to server. The server aggregates the $u_{0,i}$ and averages them to update u_0. The details of FedReCo algorithm are described in Algorithm 3.1.

For local training in FedReCo, each client needs to pass the same batch of samples to two models and calculate the stochastic gradients. Although the global feature extractor enlarges the demand of local computation and memory, the local computation power is not usually the bottleneck in the whole system. For the communication stage, each client needs to send $u_{0,i}$ to the server, which only includes a part of the model parameters, less than the whole model.

Algorithm 1 FedReCo Algorithm

```
Input: Step size $\eta_u$,$\eta_v$,$\eta_0$, penalty parameter $\lambda$
Initialize: Initialize $u_0^0$ for server, initialize $u_i$ and $v_i$ for $i$-th client
 1: for $t=0,1,...,T-1$ do
 2:     Server:
 3:     Broadcast $u_0^t$ to all the clients
 4:     Receive $u_{0,i}^{t+1}$ from all the clients
 5:     Update $u_0$: $u_0^{t+1}=\frac{1}{M}\sum_{i=1}^{M}u_{0,i}^{t+1}$
 6:     client $i$:
 7:     Receive $u_0^t$ from server, let $u_i^{t,0}=u_i^t$
 8:     for $k=0,1,...,K_v-1$ do
 9:         Randomly select one batch of samples, pass the samples to the model $\{u_0^t,v_{0,i}^{t,k}\}$ and calculate
            stochastic gradient $\tilde{\nabla}_{v_{0,i}}f_i(v_{0,i}^{t,k},u_0^t)$
10:         Update $v_{0,i}^{t,k+1}=v_{0,i}^{t,k}-\eta_v\tilde{\nabla}_{v_{0,i}}f_i(v_{0,i}^{t,k},u_0^t)$
11:     end for
12:     Let $v_{0,i}^{t+1}=v_{0,i}^{t,K_v}$
13:     Let $u_{0,i}^{t,0}=u_0^t$
14:     for $k=0,1,...,K_u-1$ do
15:         Randomly select one batch of samples, pass the samples to model $\{u_{0,i}^{t,k},v_{0,i}^{t+1}\}$ and calculate
            stochastic gradients $\tilde{\nabla}_{u_{0,i}}f_i(v_{0,i}^{t+1},u_{0,i}^{t,k})$, pass the same batch of samples to feature extractor
            $u_{0,i}^{t,k}$ and calculate stochastic gradient $\tilde{\nabla}_{u_{0,i}}H_i(u_i^t,u_{0,i}^{t,k})$
16:         Update $u_{0,i}^{t,k+1}=u_{0,i}^{t,k}-\eta_u\left(\tilde{\nabla}_{u_{0,i}}f_i(v_{0,i}^{t+1},u_{0,i}^{t,k})+\frac{\lambda}{2}\tilde{\nabla}_{u_{0,i}}H_i(u_{0,i}^{t,k},u_i^t)\right)$
17:     end for
18:     Let $u_{0,i}^{t+1}=u_{0,i}^{t,K_u}$
19:     Send $u_{0,i}^{t+1}$ to the server
20: end for
21: for $k=0,1,...,K_v-1$ do
22:     Randomly select one batch of samples, pass the samples to the model $\{u_i^t,v_i^{t,k}\}$ and calculate
        stochastic gradient $\tilde{\nabla}_{v_i}f_i(v_i^{t,k},u_i^t)$
23:     Update $v_i^{t,k+1}=v_i^{t,k}-\eta_v\tilde{\nabla}_{v_i}f_i(v_i^{t,k},u_i^t)$
24: end for
25: Let $v_i^{t+1}=v_i^{t,K_v}$
26: for $k=0,1,...,K_u-1$ do
27:     Randomly select one batch of samples, pass the samples to model $\{u_i^{t,k},v_i^{t+1}\}$ and calculate
        stochastic gradients $\tilde{\nabla}_{u_i}f_i(v_i^{t+1},u_i^{t,k})$, pass the same batch of t samples to feature extractor
        $u_0^t$ and calculate stochastic gradient $\tilde{\nabla}_{u_i}H_i(u_i^{t,k},u_0^t)$
28:     Update $u_i^{t,k+1}=u_i^{t,k}-\eta_u\left(\tilde{\nabla}_{u_i}f_i(v_i^{t+1},u_i^{t,k})+\frac{\lambda}{2}\tilde{\nabla}_{u_i}H_i(u_i^{t,k},u_0^t)\right)$
29: end for
30: Let $u_i^{t+1}=u_i^{t,K_u}$
```

4 Experiments

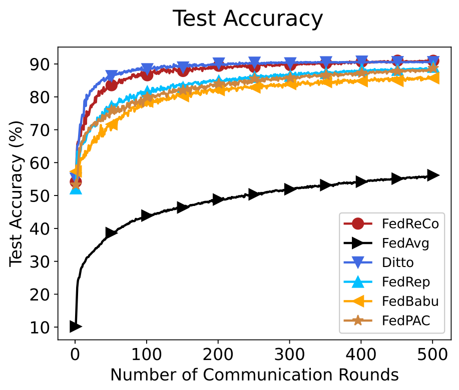
4.1 Performance on Benchmark Datasets

We perform the experiments on FashionMNIST/FMNIST and CIFAR10 datasets with a 5-layer CNN model, with two convolution layers and three fully connected layers. The first four layers are considered as the feature extractor and one last classifier layer as the prediction head trained totally locally. The compared methods include: FedAvg McMahan et al. (2017), FedAvg-FineTuning (FT) Collins et al. (2022), Ditto Li et al. (2021), FedRep Collins et al. (2021), FedBabu Oh et al. (2022), FedPAC Xu et al. (2023), FedCR Zhang et al. (2023). There are 50 clients in the network, each with 4 classes of data for FMNIST dataset and 2 classes of data for CIFAR10 dataset, to form a hetegenerous data distribution. The results are obtained after 500 rounds of communication, each with local SGD updates for 2 epochs of local samples, 1 epoch on training local prediction head, 1 epoch on training local feature extractor. More details of settings and hyper-parameters are provided in Appendix A.

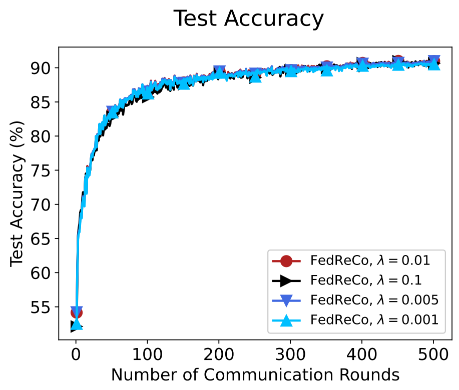
For the relatively simple dataset FMNIST, FedAvg can already get an acceptable accuracy, and other algorithms obtain similar final accuracy. Note that in this case the FedAvg+fine-tuning is competitive to other methods, getting the highest accuracy. For the more complex CIFAR10 dataset and more heterogeneous setting, fine-tuning is still competitive to some personalization methods, with FedReCo outperforming all compared methods, showing the higher flexibility to more heterogeneous setting. Fig. 3(a) displays the test accuracy of different algorithms on CIFAR10 dataset. It is seen that Ditto and FedReCo converge faster than other compared algorithms in this setting.

Table 1: Test Accuracy (%) on benchmark datasets; F: FMNIST, C: CIFAR10

	FedAvg	FedAvg-FT	Ditto	FedRep	FedBABU	FedPAC	FedCR	FedReCo
F	85.38	**93.85**	92.92	93.04	92.85	92.62	92.71	93.42
C	56.17	89.05	90.55	89.12	85.69	88.71	89.21	**91.07**



(a) Different Algorithms



(b) Differential Value of λ

Figure 3: (a) Test accuracy on CIFAR10 dataset. (b) Test accuracy with different λ on CIFAR10 dataset.

We furthur explore the impact of λ in FedReCo. Fig. 3(b) shows that with different values of λ, the performance of FedReCo is not affected a lot. They nearly achieve the same performance.

5 Conclusions

We have proposed a federated learning algorithm, FedReCo, that enforces the representation part of local models to be similar in a data-driven manner. FedReCo takes a step to study how layer sensitivity in neural networks can be fully exploited in federated learning, which hopefully will result in further interesting works. In fact, the framework of FedReCo can be easily extended to the partition of neural network at any layer, not limited to last classifier layer, and even to partitioning at different layers for different clients.

References

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, pp. 2089–2099. PMLR, 2021.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Fedavg with fine tuning: Local updates lead to representation learning. *Advances in Neural Information Processing Systems*, 35:10572–10586, 2022.

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*, pp. 770–778, 2016.

Peter Kairouz and H Brendan McMahan. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1):1–210, 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL `http://dx.doi.org/10.1561/2200000083`.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Bo Li, Mikkel N Schmidt, Tommy S Alstrøm, and Sebastian U Stich. On the effectiveness of partial variance reduction in federated learning with heterogeneous data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3964–3973, 2023.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pp. 6357–6368. PMLR, 2021.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of FedAvg on non-IID data. In *International Conference on Learning Representations*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2021.

Jaehoon Oh, SangMook Kim, and Se-Young Yun. Fedbabu: Toward enhanced representation for federated image classification. In *International Conference on Learning Representations*, 2022.

Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pp. 17716–17758. PMLR, 2022.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

Jian Xu, Xinyi Tong, and Shao-Lun Huang. Personalized federated learning with feature alignment and classifier collaboration. In *The Eleventh International Conference on Learning Representations*, 2023.

Hao Zhang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. Fedcr: Personalized federated learning based on across-client common representation with conditional mutual information regularization. In *International Conference on Machine Learning*, 2023.

A Experiment Setup

A.1 Datasets

We use two benchmark datasets in the experiments, FashionMNIST (FMNIST) and CIFAR10, both consisting of 10 classes of data. The samples are divided into a training set with 70% data, and a testing set with 30% data. The data samples are distributed to 50 clients. To make the data distribution heterogeneous, we assign different classes of data to the clients. For FMNIST, each client has the data from 4 classes, and for CIFAR10, each client only has the data from 2 classes.

Table 2: Detailed information about datasets

Dataset	All Samples	Training Set	Test Set	Samples per Client	Classes per Client
FMNIST	70000	49000	21000	1400	4
CIFAR10	60000	42000	18000	1200	2

A.2 Model and Hyper-parameters

The model for both datasets is a 5-layer CNN model, consisting of two convolutional layers, each followed by a 2×2 max pooling layers, and two fully connected layers with 1024 neurons, and finally a softmax layer as classifier. The first four layers are considered as the feature extractor for FedRep, FedBabu, FedPAC, FedCR, FedReCo.

For the hyper-parameters in different algorithms, we set the same learning rates as 0.01, and batch size as 48. In the local training, the standard gradient clipping is used with a maximum norm 10. For FedReCo, we use the same learning rate for feature extractor and prediction head as 0.01, and 0.001 for the global feature extractor. The λ in the regularization term is 0.01. For Ditto, the λ is set as 0.01. For FedPAC, the λ is set to 1. For FedCR, the β is set as 0.001.