Cooperative Multiple Model Training for Personalized Federated Learning over Heterogeneous Devices

Jian Xu
Tsinghua Shenzhen International Graduate School
Tsinghua University
China
xujian20@mails.tsinghua.edu.cn

Shuo Wan, Yinchuan Li, Zhilin Chen, Yunfeng Shao, Zhitang Chen
Noah's Ark Lab, Huawei Technologies
China

Shao-Lun Huang*
Tsinghua Shenzhen International Graduate School
Tsinghua University
China
shaolun.huang@sz.tsinghua.edu.cn

ABSTRACT

Federated learning (FL) is an increasingly popular paradigm for protecting data privacy in machine learning systems. However, the data heterogeneity and high computation cost/latency are challenging barriers for employing FL in real-world applications with heterogeneous devices. In this paper, we propose a novel personalized FL framework named CompFL allowing cooperative training of models with varied structures to mitigate those issues. First, CompFL initializes a set of expert models in varied sizes and allows each client to choose one or multiple expert models for training according to its capacity. Second, CompFL combines the model decoupling strategy and local-global feature alignment to mitigate the adverse impact of label heterogeneity, where clients only share the representation layers of each model architecture. Third, to encourage mutual enhancement of various models, knowledge distillation in local training is further applied to improve the overall performance. To make our framework workable in real systems, we implement it in both centralized settings with server-coordinated parallel training and decentralized settings by a newly developed device-to-device model training-forwarding scheme. Extensive experiments on benchmark datasets are conducted to verify the potential of our approach.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

Federated Learning, Heterogeneous Devices, Multiple Models

ACM Reference Format:
Jian Xu, Shuo Wan, Yinchuan Li, Zhilin Chen, Yunfeng Shao, Zhitang Chen, and Shao-Lun Huang. 2024. Cooperative Multiple Model Training for Personalized Federated Learning over Heterogeneous Devices. In *FedKDD '24, August 26, 2024, Barcelona, Spain.*. ACM, New York, NY, USA, 8 pages.

*Corresponding author

FedKDD '24, August 26, 2024, Barcelona, Spain.

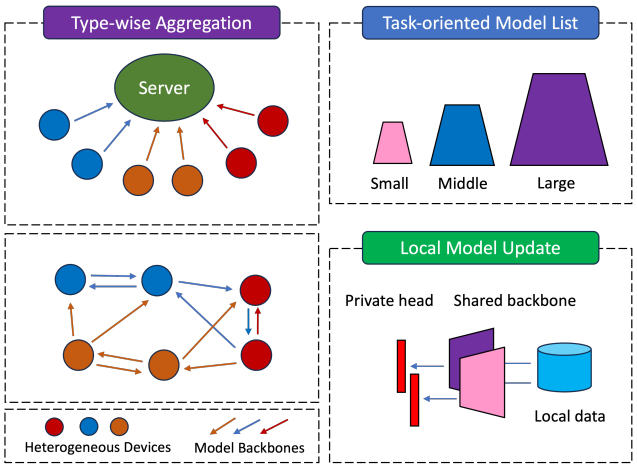


Figure 1: CompFL with multiple model types and heterogeneous devices. Each client can select the proper model architecture from a shared model list based on local computation capability. In server-based FL, server will aggregate each type of models and broadcast back to nodes. In decentralized FL, each type of model can travel over the nodes by random walk.

1 INTRODUCTION

With the rapid advancement of machine learning techniques and the growing demand for privacy protection, federated learning (FL) has been developed to facilitate model training over distributed data sources by exchanging gradients or parameters without explicitly collecting sensitive data [17, 26]. FL has exhibited promising applications in many scenarios, e.g., Internet of Things [6] and healthcare [15]. In a typical FL system, all clients jointly train the same model while a central server is responsible for coordinating the model aggregation. Besides, to mitigate the communication bottleneck in server and tolerate node failures, decentralized FL is also gaining popularity, where the clients directly exchange their model parameters with neighbors in a peer-to-peer manner [14, 20, 21].

Despite the great potential of FL, however, various challenges from system and statistical heterogeneity have been observed, which will substantially hinder the deployment of FL in real applications [7]. System heterogeneity refers to the varied hardware and network capabilities across clients, which mainly affects the learning efficiency and commonly occurs with the emerging mobile computing for wearable devices and smartphones [13]. For example, it is capable for a CPU device to train a simple CNN mode within

acceptable running time while it could be extremely slow to train a ResNet-like model. Different GPU devices also vary significantly in computational capability. To mitigate the system heterogeneity, a wide range of approaches have been investigated, such as asynchronous updating [18], adaptive workload [24], and model pruning [1, 9]. In particular, one promising solution is to allow heterogeneous models across clients depending on their capabilities. However, existing methods usually allow arbitrarily different local models and rely on either a large public dataset or a data generator [10, 16] to conduct knowledge transfer, making it infeasible in many realistic scenarios. The statistical heterogeneity represents the varied distributions of data sets across clients, which raises the problem of non-IID data and significantly affects the model aggregation [27]. To address this issue, a popular direction is personalized FL where each individual client aims to leverage the knowledge from peer clients for a better customized model than that of standalone training [22]. Those two issues could co-exist in real-world systems and further aggravate the training process [12].

In this work, we propose a novel **Co**operative **m**ultiple model training framework for **p**ersonalized **FL** with various model sizes to tackle the issues of heterogeneous computing capabilities and tasks as shown in Fig. 1, we name it by `CompFL`. Different from previous works that consider either a single model architecture or independently varied architectures, we consider a mild setup with multiple global model architectures to strike a balance between the homogeneous (all clients share the same model structure) and heterogeneous (each client independently defines its own model) settings. More precisely, each client can choose one specific model architecture from a predefined model list (e.g., small, base and large model variants) that best matches its device capacity in terms of computation. Moreover, to tolerate the label disparity among clients and maintain the model accuracy, we follow the representation-only aggregation strategy in [4] and apply local-global feature alignment-based scheme to enhance the model learning. To the best of our knowledge, we are the first to empirically show the benefit of local-global feature alignment in remedying the shortcoming of only sharing feature extractor schemes in FL. Furthermore, the interaction between different expert models is also explored through local knowledge distillation. The proposed framework is evaluated on benchmark datasets with various settings, which verifies the effectiveness in mitigating the heterogeneity in heterogeneous FL environments and improving personalized model performance. Note that other generic techniques of gradient compression and model pruning are orthogonal to our work and can be integrated to each type of model. The main contributions are three-fold:

- A novel cooperative training framework named `CompFL` is proposed for FL over heterogeneous devices, where each client can choose its own model structure from a predefined model list to best fit its computing capability.
- A holistic training strategy with greedy model selection, local-global feature alignment and cross-model knowledge distillation is employed to optimize the personalized model performance.
- The framework is implemented in both server-based and decentralized settings with extensive experiments conducted on label skew scenarios to verify its effectiveness.

2 PRELIMINARY AND MOTIVATION

2.1 Problem Setup

For a FL setup with m clients, where the goal of each client is to collaboratively train a personalized model that performs better than training based on local data only. The optimization objective is formalized as follows:

$$\min_{W} F(W) := \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_{\xi_i \sim \mathcal{D}_i}\left[\ell_i(\boldsymbol{w}_i; \xi_i)\right] + \mathcal{R}(W) \tag{1}$$

where $W = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_m\}$ denotes the collection of local models, $\mathcal{D}_i = \{x_i^k, y_i^k\}_{k=1}^{n_i}$ denotes the local training data set drawn from an underlying distribution $\mathcal{P}_i$ on $\mathcal{X}_i \times \mathcal{Y}_i$, where $\mathcal{X}_i$ and $\mathcal{Y}_i$ denote the instance space and label space for client i, respectively. $\ell_i(\boldsymbol{w}_i; \xi_i)$ denotes the local loss function and $\mathcal{R}(\cdot)$ is an extra regularization for information sharing among clients.

In this paper, we focus on the classification task with deep neural networks (DNNs), which can be decoupled into the representation layers and the final classifier head as $\boldsymbol{w} = (\boldsymbol{\theta}, \boldsymbol{\phi})$ [1]. Following the previous works [3, 4, 27], we will study the label shift induced data heterogeneity, we further assume that $\mathcal{X}_i = \mathcal{X}_j$ but potentially $\mathcal{Y}_i \neq \mathcal{Y}_j$ for any pair of i and j. Let $\mathcal{Y} = \cup_{i=1}^{m} \mathcal{Y}_i$ denote the set of all possible labels over all clients. Note that the one-hot encoding is employed for treating the categorical variable of label in classification tasks. When the labeling scheme is globally synchronized, the corresponding data category of each class label is consistent across clients (*we call aligned label or unified label*) and thus an output layer with dimension $|\mathcal{Y}|$ could be used as the classifier head in all local models. Otherwise, even two clients share the same one-hot label set, it does not imply that they undertake the same task and thus each client should maintain its own classifier head.

2.2 Motivation

(i) Multiple model architectures for heterogeneous devices. Personalized hardware resources usually make it difficult for all clients to train models with the same structure. As a toy example, we conduct a set of experiments to train a simple CNN and a ResNet-18 on 10 clients using CPU and GPU respectively. The average per-round simulation time and final performance are shown in Table 1, where using CPU to train the ResNet-18 is unacceptably slow. Therefore, using heterogeneous models become a key requirement to enable personalized FL on heterogeneous devices. Previous works usually assume the small model is a sub-network of the large model, which limits the usage of other applicable model architectures and even a sub-network could still be burdensome to train for low-capability nodes. Alternatively, we propose to use multiple model architectures (multiple expert models) as follows.

Assumption 1. *There exists a set of expert models $\mathcal{M} = \{1 : \mathcal{W}_1, 2 : \mathcal{W}_2, .., M : \mathcal{W}_M\}$ with increasing model parameters such that each client can select the suitable expert as its personalized model based on the device capacity.*

Remark 1. *(Rationality) Since popular and commonly used model structures are usually available at the open-source libraries (e.g.,*

[1]The part of representation layers $\boldsymbol{\theta}$ is also known as the feature extractor while the final classifier head is usually a single fully-connected layer.

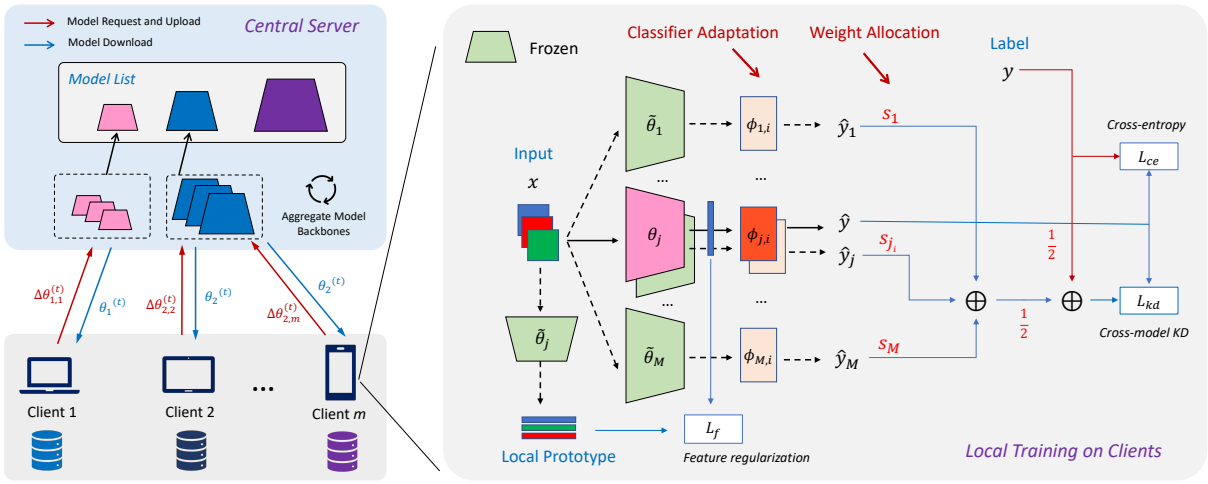


Figure 2: The overall workflow in server-based systems. The local training consists of model adaptation, feature-level alignment and cross-model knowledge distillation, where the back-propagation only exists at the solid line during local model updating.

Table 1: Comparison of test accuracy and per-round simulation time on CIFAR-10. 10 clients with IID data distribution.

Device	Model	Accuracy (%)	Time/Round (s) (relative)
CPU	CNN	60.25	18.64 (1.0)
	ResNet-18	N/A	**974.55 (52.28)**
GPU	CNN	60.56	13.74 (0.74)
	ResNet-18	78.12	22.47 (1.21)

torchvision), it is possible to construct a model list that could offer suitable models for nodes with diverse capacities. Moreover, in realistic scenarios, devices with similar capacity could be clustered into a group that adopts the same model architecture so that only limited types of model are needed.

(ii) Cooperative model training. We observe that ***the accuracy improvement over communication rounds is significant at the initial stage but becomes marginal as training going on (see Fig. 3)***. When the high-capacity model reaches convergence, it would present little benefits if high-capability nodes always train the big models. Therefore, those nodes could turn to assist the training of smaller models to make them better by feeding more diverse training data, where the larger models could further offer soft-supervision for knowledge distillation.

3 OUR PROPOSED FRAMEWORK

3.1 Knowledge Distillation-Aided Training

Feature-based Distillation. Our preliminary results have shown that inconsistent classifier heads across clients would cause degraded feature representation learning. To remedy this, we incorporate a simple-yet-effective local-global feature regularization to encourage the locally learned features stay close to the global feature anchors (averaged feature vector).

$$L_f = \frac{1}{|n_i|}\sum_{k=1}^{|n_i|}\|z_k - \tilde{z}_{y_k}\|^2, \tag{2}$$

where $z_k = f(x_k)$ is the feature vector of the k-th data point extracted by current local model, and $\tilde{z}_{y_k}$ is the feature anchor of class y_k from previous global model. Since the global data information is not accessible, we use the local data to estimate $\tilde{z}_{y_k}$. As a result, the diversity/drift of locally learned feature distributions could be reduced, which is beneficial for the generalization of aggregated feature extractor.

Logits-based Ensemble Distillation. So far, the objective only involves the specific local model, however, the knowledge of other expert models is also beneficial for local model training. To this end, we fit a local head for each other experts and apply the ensemble knowledge distillation method for utilizing the rich knowledge from other model architectures. Since each data point has its ground-truth $\boldsymbol{y}_k$ and the predictions of expert models might be unreliable, we propose to use the average of true label and weighted prediction as the final soft-label for knowledge distillation:

$$L_{kd} = \frac{1}{|n_i|}\sum_{k=1}^{|n_i|} KL(\tilde{\boldsymbol{y}}_k \| 0.5\boldsymbol{y}_k + 0.5\sum_{j=1}^{M} s_j \hat{\boldsymbol{y}}_{j,k}), \tag{3}$$

where $\tilde{\boldsymbol{y}}_k$ is the prediction of local model, s_j and $\hat{\boldsymbol{y}}_{j,k}$ are weight and prediction of j-th expert model, respectively. Finally, the local objective function is defined as follows:

$$L_i(\boldsymbol{w}_i) := L_{ce} + \lambda L_f + \mu L_{kd}, \tag{4}$$

where L_{ce} denotes the basic cross-entropy loss, λ and μ are two coefficients for balancing different terms.

3.2 Local Training Procedure

Here we present the main workflow of CompFL as in Fig. 2, which implements the optimization steps in a cooperative fashion. For simplicity, we use $j_i \in [M]$ to denote the type of model at client i. At round t, the local execution steps are listed as follows:

- **Step 1: Model Adaptation** For local model training, we first replace the local representation layers by the global aggregate

Table 2: Model accuracy (%) comparison of different methods under label heterogeneous scenarios.

Dataset	Method	Unified Labels			Independent Labels		
		$N = 3$	$N = 5$	$N = 10$	$N = 3$	$N = 5$	$N = 10$
CIFAR-10	Standalone	78.28 (± 0.82)	60.25 (± 0.61)	40.17 (± 0.22)	78.17 (± 0.45)	61.03 (± 0.43)	39.78 (± 0.25)
	FedAvg	60.73 (± 0.56)	64.85 (± 0.31)	68.27 (± 0.20)	40.88 (± 0.63)	28.73 (± 1.63)	17.68 (± 0.57)
	FedAvg-FT	84.52 (± 0.51)	74.42 (± 0.65)	66.27 (± 0.87)	82.17 (± 0.64)	68.57 (± 0.53)	50.38 (± 0.47)
	LG-FedAvg	79.61 (± 0.54)	64.31 (± 0.48)	43.86 (± 0.53)	79.71 (± 0.53)	64.24 (± 0.43)	43.66 (± 0.74)
	FedPer	82.71 (± 0.35)	72.58 (± 0.17)	61.17 (± 0.16)	81.62 (± 0.27)	71.78 (± 0.16)	59.08 (± 0.11)
	FedRep	83.48 (± 0.21)	73.62 (± 0.14)	61.62 (± 0.31)	83.87 (± 0.20)	73.72 (± 0.43)	61.19 (± 0.33)
	FedBABU	85.52 (± 0.26)	75.13 (± 0.33)	67.68 (± 0.43)	82.38 (± 0.26)	68.95 (± 0.37)	51.58 (± 0.55)
	Ditto	85.37 (± 0.23)	76.37 (± 0.21)	67.53 (± 0.09)	84.03 (± 0.24)	71.12 (± 0.11)	53.56 (± 0.14)
	FedRoD	84.32 (± 0.32)	76.12 (± 0.21)	68.38 (± 0.12)	82.38 (± 2.13)	68.58 (± 0.93)	52.55 (± 1.05)
	CompFL	**86.22** (± 0.33)	**79.98** (± 0.25)	**74.29** (± 0.24)	**84.17** (± 0.23)	**77.25** (± 0.25)	**68.85** (± 0.24)
CIFAR-100	Standalone	76.37 (± 0.20)	60.32 (± 0.19)	38.94 (± 0.21)	76.31 (± 0.22)	61.27 (± 0.17)	37.94 (± 0.32)
	FedAvg	39.68 (± 0.33)	41.75 (± 0.43)	43.28 (± 0.54)	50.92 (± 0.35)	30.41 (± 0.34)	17.86 (± 0.38)
	FedAvg-FT	77.28 (± 0.54)	66.57 (± 0.48)	48.77 (± 0.53)	75.21 (± 0.53)	60.81 (± 0.43)	40.10 (± 0.74)
	LG-FedAvg	75.62 (± 0.54)	60.49 (± 0.48)	39.79 (± 0.53)	76.19 (± 0.53)	60.78 (± 0.43)	39.31 (± 0.74)
	FedPer	76.25 (± 0.31)	64.66 (± 0.25)	45.95 (± 0.41)	73.13 (± 0.29)	61.82 (± 0.24)	44.54 (± 0.32)
	FedRep	75.43 (± 0.28)	63.53 (± 0.27)	46.21 (± 0.34)	75.78 (± 0.27)	63.64 (± 0.22)	46.45 (± 0.30)
	FedBABU	77.75 (± 0.35)	66.21 (± 0.43)	48.42 (± 0.31)	74.76 (± 0.41)	60.87 (± 0.37)	40.96 (± 0.62)
	Ditto	77.92 (± 0.43)	65.21 (± 0.44)	48.65 (± 0.46)	73.64 (± 0.44)	59.16 (± 0.41)	33.93 (± 0.72)
	FedRoD	76.14 (± 0.39)	65.35 (± 0.39)	51.23 (± 0.55)	75.15 (± 0.47)	59.97 (± 0.56)	38.52 (± 0.61)
	CompFL	**78.26** (± 0.25)	**68.65** (± 0.33)	**55.95** (± 0.23)	**77.72** (± 0.62)	**65.78** (± 0.22)	**49.78** (± 0.21)

and remain the private classifier:

$$\boldsymbol{w}_i^{(t)} := \{\tilde{\boldsymbol{\theta}}_{j_i}^{(t)}, \boldsymbol{\phi}_i^{(t)}\}. \tag{5}$$

We can then re-initialize the head $\boldsymbol{\phi}_i$ by computing the class prototypes on private data and (optionally) fine-tune it by gradient descent for one epoch:

$$\boldsymbol{\phi}_i^{(t)} \leftarrow \boldsymbol{\phi}_i^{(t)} - \eta_g \nabla_{\boldsymbol{\phi}} \ell(\boldsymbol{\theta}_i^{(t)}, \boldsymbol{\phi}_i^{(t)}; \xi_i), \tag{6}$$

where ξ_i denotes the mini-batch of data, η_g is the learning rate for updating classifier head. For other expert models, if a client wants to utilize them for knowledge distillation, it also needs to fit a local classifier head. We use the class-wise prototypes as the initialization and train the low-dimensional classifier for one epoch efficiently.

- **Step 2: Model Evaluation** As mentioned before, the local ensemble knowledge distillation need to obtain a weighted average of soft-labels for guiding local training. Intuitively, the higher accuracy a model can achieve, the higher weight it should be assigned. Therefore, we simply allocate the weights based on the normalized local validation accuracy. More precisely, we use the following normalization method to generate the weights:

$$s_j = \frac{\exp\left(\tilde{a}_j/\tau\right)}{\sum_{j=1}^{M} \exp\left(\tilde{a}_j/\tau\right)}, \tag{7}$$

where $\tilde{a}_j$ is the 0-1 normalized value and a positive constant $0 < \tau < 1$ is further introduced to sharpen the weight allocation as we empirically found that it is more beneficial to assign higher weights to the high performance models.

- **Step 3: Model Selection** In the traditional FL, a client will only train the local model by default. However, the high-capability nodes actually could help train the feature extractors of low-capacity models as well, which is more computation-communication efficient. Thus, we propose to use the ε-greedy model selection strategy for a high-capability node that has a chance to sample model $\hat{j}_i$ uniformly with a probability of ε, instead of always choosing the local model j_i.

$$\hat{j}_i \leftarrow \begin{cases} j_i & \text{with prob. } 1-\varepsilon \\ \mathcal{U}\{1, ..., j_i | j_i \le M\} & \text{with prob. } \varepsilon \end{cases} \tag{8}$$

- **Step 4: Model Updating** We consider the basic cross-entropy loss, feature alignment loss and knowledge distillation loss together. Then, we apply the stochastic gradient decent (SGD) to train the local model for multiple epochs:

$$\boldsymbol{w}_i^{(t)} \leftarrow \boldsymbol{w}_i^{(t)} - \eta \nabla_{\boldsymbol{w}} L_i(\boldsymbol{w}_i^{(t)}) \tag{9}$$

Note that we train the whole model in an end-to-end manner instead of alternating training of representation layers and classifier head as in FedRep, as we already conducted local model adaptation and the end-to-end manner training could avoid additional data iteration with is time-consuming.

Global Model Aggregation. The server will generate a common feature extractor for each type of architecture:

$$\tilde{\boldsymbol{\theta}}_j^{(t+1)} = \sum_{i=1}^{m_j} \alpha_i \boldsymbol{\theta}_{j,i}^{(t)}, \quad \text{s.t.} \sum_{i=1}^{m_j} \alpha_i = 1 \tag{10}$$

where m_j is the number of clients connecting to server and returning j-th architecture in t-th round, $\alpha_i \geq 0$ is the weight of client i and we choose $\alpha_i = n_i / \sum_{c=1}^{m_j} n_c$, which is proportional to the local data size. After the global aggregation, the server will broadcast the model set to the selected clients to start the next round and each individual client could choose to download either partial or full set of the expert models based on the personalized need. By default, we assume that each client will only download the expert models that have less memory and computation requirements than its own personalized choice as mentioned in the model selection part.

4 EVALUATION

In this part, we provide some main results in server-based setup, while more results as well as extension to decentralized setup can refer to the Appendix D-E.

4.1 Experimental Setup

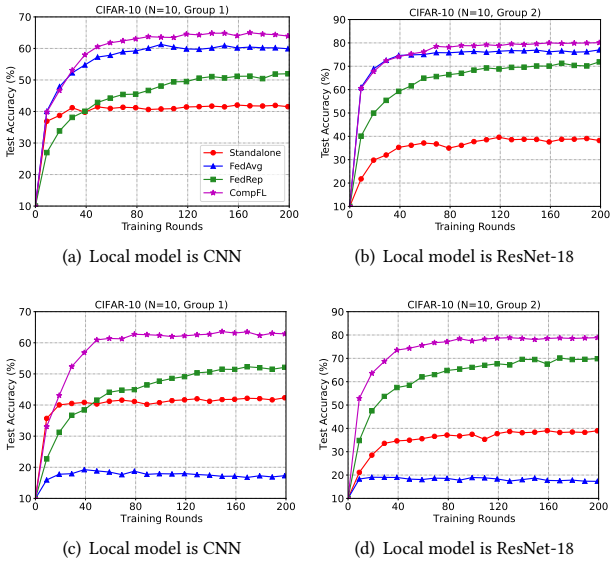
Datasets and Models. We focus on the image classification tasks on popular benchmark datasets Fashion-MNIST [25] and CIFAR-10/CIFAR-100 [8]. In this work, we consider M=2 expert models. For Fashion-MNIST, we build two convolutional neural networks (CNNs) with different numbers of convolution-pooling/fully-connected layers. For CIFAR-10/CIFAR-100, a CNN model [17] and a ResNet-18 [5] are selected as the expert models.

Data Partitioning. Like previous study [23], we sample a subset of classes $C_i \subseteq \mathcal{Y}$ with $|C_i| = N$ for client i and each client will have access to data of only N classes. And the number of samples belonging to each class in C_i is balanced. We use the coarse labels with 20 categories for CIFAR-100.

4.2 Evaluation Results and Analyses

Performance of Personalized Models. We choose N from {3, 5, 10} on three datasets, respectively. We keep ***local data size as*** 500 ***on each client*** and evaluate all methods under the same training conditions. The results of average test accuracy across clients are reported in Table 2, where our CompFL is able to achieve the highest average test accuracy for most local tasks. The improvement by our method increases as the local task becomes harder. For example, our method improve the accuracy by more than 5% when N=10 and more than 2% when N=5 on the CIFAR-10. The performance gaps between FedPer/FedRep and FedBABU on various cases indicate that though learning a shared feature extractor is promising, diverse local classifiers could make the feature learning sub-optimal. By contrast, our framework leverages the explicit local-global feature alignment to facilitate the representation learning. We also separately provide the curves of average model accuracy over rounds for each group to verify how our framework improves the overall performance. The results in Fig. 3 show that our method can improve the performance of both expert models when only the feature extractor layers are shared.

Ablation Studies. Since there are three key components in CompFL, including feature self-alignment (Align), greedy model selection strategy (Greedy) and cross-model knowledge distillation (KD), it is necessary to conduct ablation studies to verify their individual efficacy. We choose the CIFAR-10/100 datasets under N=10 cases with unified labels. For performance comparison, we report both



(a) Local model is CNN (b) Local model is ResNet-18

(c) Local model is CNN (d) Local model is ResNet-18

Figure 3: Group-wise average test accuracy over training rounds on CIFAR-10. (a)-(b): unified labels; (c)-(d):independent labels.

Table 3: Ablation studies of the N=10 case with unified labels. Baseline is equivalent to FedPer with 61.17%/45.95% Acc..

Align	Greedy	KD	Final Acc. & # Rnds to Target			
			CIFAR-10		CIFAR-100	
✓			68.65	—	52.28	100
	✓		65.69	—	46.57	—
		✓	68.21	—	49.63	—
✓	✓		70.68	89	52.94	104
✓		✓	71.88	81	53.24	83
✓	✓	✓	**74.29**	**63**	**55.95**	**61**

the average final accuracy and the number of rounds to achieve a target accuracy (70%/50%). From the results presented in Table 3, it can be found that any arbitrary combination of the designed components can always outperforms the baseline of FedPer with a non-trivial improvement, which demonstrates that leveraging knowledge from global model and other expert models is effective in improving personalized performance.

5 CONCLUSION

In this paper, we study the FL with both label skew and heterogeneous devices. We propose a new framework that allows each client to cooperatively train multiple models with varied architectures and choose the suitable one for inference. It shows potential in striking a good balance between high personalized performance and training efficiency. As future work, it would be interesting to investigate training with multi-modal data and extend CompFL to other complicated computer vision tasks.

REFERENCES

[1] Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. 2022. FedRolex: Model-Heterogeneous Federated Learning with Rolling Sub-Model Extraction. In *NeurIPS*.

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

[3] Hong-You Chen and Wei-Lun Chao. 2022. On Bridging Generic and Personalized Federated Learning for Image Classification. In *ICLR*.

[4] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting Shared Representations for Personalized Federated Learning. In *ICML*.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Sun Jian. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

[6] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M. Hadi Amini. 2022. A Survey on Federated Learning for Resource-Constrained IoT Devices. *IEEE Internet Things J.* 9, 1 (2022), 1–24.

[7] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, and et al. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210.

[8] Alex Krizhevsky and G. Hinton. 2009. Learning Multiple Layers of Features from Tiny Images. *University of Toronto* (2009).

[9] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. 2021. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *ACM MobiCom*. 420–437.

[10] Daliang Li and Junpu Wang. 2019. FedMD: Heterogenous Federated Learning via Model Distillation. *CoRR* abs/1910.03581 (2019). arXiv:1910.03581

[11] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and Robust Federated Learning Through Personalization. In *ICML*.

[12] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems*.

[13] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* 22, 3 (2020), 2031–2063.

[14] Wei Liu, Li Chen, and Wenyi Zhang. 2022. Decentralized Federated Learning: Balancing Communication and Computing Costs. *IEEE Trans. Signal Inf. Process. over Networks* 8 (2022), 131–143.

[15] Zelei Liu, Yuanyuan Chen, Yansong Zhao, Han Yu, Yang Liu, Renyi Bao, Jinpeng Jiang, Zaiqing Nie, Qian Xu, and Qiang Yang. 2022. Contribution-Aware Federated Learning for Smart Healthcare. In *AAAI*.

[16] Disha Makhija, Xing Han, Nhat Ho, and Joydeep Ghosh. 2022. Architecture Agnostic Federated Learning for Neural Networks. In *ICML*.

[17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[18] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. 2022. Federated Learning with Buffered Asynchronous Aggregation. In *AISTATS*.

[19] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. 2022. FedBABU: Toward Enhanced Representation for Federated Image Classification. In *ICLR*.

[20] Noa Onoszko, Gustav Karlsson, Olof Mogren, and Edvin Listo Zec. 2021. Decentralized federated learning of deep neural networks on non-iid data. *CoRR* abs/2107.08517 (2021).

[21] Tao Sun, Dongsheng Li, and Bao Wang. 2021. Decentralized Federated Averaging. *CoRR* abs/2104.11375 (2021). arXiv:2104.11375

[22] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–17. https://doi.org/10.1109/TNNLS.2022.3160699

[23] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, and Jing Jiang. 2022. FedProto: Federated Prototype Learning over Heterogeneous Devices. In *AAAI*.

[24] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE J. Sel. Areas Commun.* 37, 6 (2019), 1205–1221.

[25] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR* abs/1708.07747 (2017).

[26] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* 10, 2 (Jan. 2019).

[27] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated Learning with Non-IID Data. *arXiv:1806.00582* (2018).

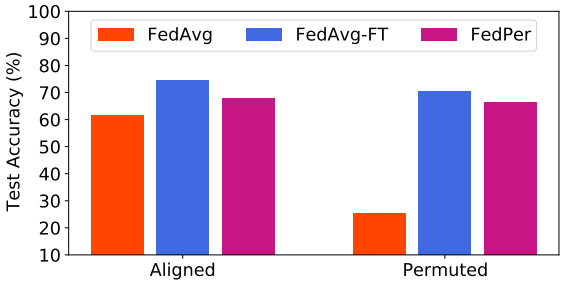


Figure 4: Preliminary results on CIFAR-10. FedAvg with fine-tuning outperforms FedPer that only keeps private head even in the presence of permuted labels among clients.

A MODEL DECOUPLING IS SUB-OPTIMAL

Generally, the deep neural network for classification task can be decoupled into the representation layers and the final classifier head as $\boldsymbol{w} = (\boldsymbol{\theta}, \boldsymbol{\phi})$, where the former is also known as feature extractor. Previous work has shown the potential of decoupling the feature extractor (the common role across similar local tasks) and the classifier head (the personalized part for each specific task) during model aggregation. As a preliminary experiment, we consider a FL setup with 20 clients and each of them has 500 samples from 5 classes of CIFAR-10. We construct a simple CNN model with a 10-class classifier head and evaluate both ***globally aligned*** and ***locally randomly permuted*** labels scenarios. Fig. 4 presents the results of FedAvg, FedAvg with local classifier fine-tuning, and FedPer that only aggregates the feature extractor. It can be observed that the vanilla FedAvg is vulnerable to label shift and local fine-tuning could lead to promising performance. In contrast, FedPer is more robust to label heterogeneity but results in slightly lower performance than FedAvg-FT. We provide the following analyses to explain such phenomenons. First, the last classifier head only accounts for a limited percentage of parameters and is more robust to local over-fitting. Therefore, it could be learned locally given sufficient local data when the feature learning is accomplished, which explains the satisfactory performance of FedAvg-FT. Second, when the local data is limited, locally learned classifier parameters are still inaccurate and lead to non-negligible diversity across clients, which will have an adverse effect on the feature learning as the representations and classifiers are highly coupled during the training process. As a result, the locally learned representations may have limited transferability to other clients. Therefore, it is crucial to learn more consistent feature representations across clients so as to improve the generalization ability of the shared feature extractor.

B EXPERIMENTAL DETAILS

Compared Methods. We adopt the following baselines from FL to aggregate each model architecture: (i) FedAvg [17]; (ii) FedPer [2] and FedRep [4] that keep personal classifiers, and FedBABU [19] that freezes the initialized classifier head during feature learning and re-trains the head during local adaptation; (iii) Ditto [11] that conducts bi-level optimization; (iv) FedRoD [3] that combines the local and global heads. For our method, after careful tuning we fix λ=1.0, ε=0.3, τ=0.1 and μ=1.0 for all experiments.

Algorithm 1 CompFL

Input: Learning rate η, communication rounds T, number of clients m, local epochs E

Initialize: Initial models $\{\boldsymbol{w}_j^0 \in R^{|\mathcal{W}_j|}\}_{j=1}^M$

Main Loop:

for $t = 0, 1, ..., T-1$ **do**
  Server Executes:
  Select active client set S_t
  Clients request the chosen model $\hat{j}_i$
  Send $\{\boldsymbol{\theta}_{\hat{j}_i}^t\}_{i \in S_t}$ to selected clients
  for *Client* $i \in S_t$ *in parallel* **do**
    $\{\boldsymbol{\theta}_{\hat{j}_i,i}^{(t+1)}, \boldsymbol{\phi}_{\hat{j}_i,i}^{(t+1)}\} \leftarrow$ **LocalTraining**$(i, \theta_{\hat{j}_i}^t)$
    Send $\Delta\boldsymbol{\theta}_{\hat{j}_i,i}^{(t+1)} := \boldsymbol{\theta}_{\hat{j}_i,i}^{(t+1)} - \boldsymbol{\theta}_{\hat{j}_i}^{(t)}$ back
  end for
  Server aggregates $\{\tilde{\boldsymbol{\theta}}_j^{(t+1)}\}_{j=1}^M$:
    $\tilde{\boldsymbol{\theta}}_j^{(t+1)} = \boldsymbol{\theta}_j^{(t)} + \sum_{k=1}^{m_j} \alpha_k \Delta\boldsymbol{\theta}_{j,k}^{(t+1)}$
end for

LocalTraining$(i, \tilde{\theta}_{\hat{j}_i}^t)$:

Retain local model: $\boldsymbol{w}_i^{(t)} \leftarrow \{\tilde{\boldsymbol{\theta}}_{\hat{j}_i}^{(t)}, \boldsymbol{\phi}_i^{(t)}\}$
for *local epoch* $e = 0, 1, ..., E-1$ **do**
  for *mini-batch* $\xi_i \in \mathcal{D}_i$ **do**
    Update model: $\boldsymbol{w}_i^{(t)} \leftarrow \boldsymbol{w}_i^{(t)} - \eta \nabla_{\boldsymbol{w}} L_i(\boldsymbol{w}_i^{(t)})$
  end for
end for
return $\{\boldsymbol{\theta}_{\hat{j}_i,i}^{(t+1)}, \boldsymbol{\phi}_{\hat{j}_i,i}^{(t+1)}\} := \boldsymbol{w}_i{}^{(t)}$

Training Settings. We launch $m = 20$ clients and equally divide them into 2 groups with two levels of capability corresponding to two different model selections. SGD is employed as the local optimizer with batch size $B = 50$, momentum coefficient 0.5 and weight decay 5e-4. We run 200 communication rounds with local epochs $E = 5$.

C PRACTICAL CONSIDERATION

During the uplink communication, no extra information except the parameters of one specific feature extractor will be transmitted for each client, compatible with the general settings. For efficiency, we let i-th client conduct ensemble knowledge distillation only from expert models that have less parameters than its own model j_i after some warm-up rounds. Depending on the device capability, a client could choose to only use the selected local model or the ensemble of locally adapted expert models for inference. In this work, we consider the former option for cost efficiency and leave the local mixture-of-experts direction as the future work. The pseudo-code is listed in Algorithm 1.

D FURTHER DISCUSSIONS

Effect of Local Data Size. To investigate how local data size affects the overall model performance, we fix N=5 and vary the local data size over {100, 300, 500, 1000, 1500, 2000}, recording the average accuracy in Fig. 5. The results clearly indicate that clients with

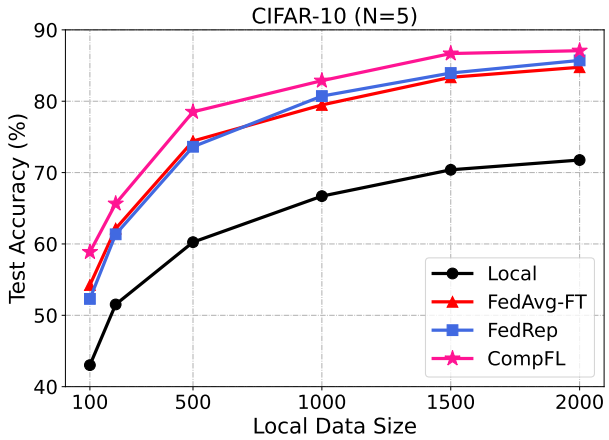


Figure 5: Accuracy on CIFAR-10 with varying local data size.

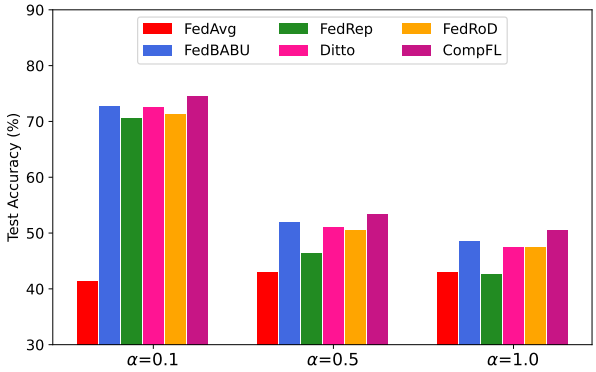


Figure 6: Model accuracy (%) comparison of different methods on CIFAR-100 under Dirichlet sampling based label shifts.

different data sizes can consistently benefit from participating FL and our method always achieves higher performance gain than FedAvg-FT and FedRep.

Other Label Distribution Shift. The main results focus on the typical pathological setting where the data-heterogeneity is determined by the distinct types of available labels. Here, we evaluate another typical setting with label distribution shift determined by the Dirichlet sampling with a concentration parameter α. The following Fig. 6 reports the numerical results of several baselines and our methods under independent labeling setting on coarsely-labeled CIFAR-100, where our framework always outperforms the baselines with non-marginal gains when α varies over {0.1, 0.5, 1.0}, demonstrating its effectiveness in a wide range of label shifts.

Leverage Pre-trained Expert Models. So far, we always train the expert models from scratch, however, plentiful pre-trained models are widely applied in real-world applications and also promising to accelerate the decentralized federated learning. For the ***Proof of Concept***, we simply adopt the CNN / ResNet-18 pre-trained on CIFAR-100 as the backbone followed by a re-initialized private head. We evaluate such system designs on the CIFAR-10, where each client is assigned with 5 distinct classes. The resulted average accuracy and consumed communication rounds for achieving the target accuracy are reported in Table 4, which show that leveraging the pre-trained model parameters could lead to better local models with

Table 4: Performance comparison on CIFAR-10 for methods with and without model pre-training.

Method	Performance Metric	
	Accuracy (%)	# Rounds for 75 %
Standalone	60.25	–
+ pre-training	77.04	–
FedAvg-FT	74.42	–
+ pre-training	80.19	4
CompFL	**79.98**	44
+ pre-training	**83.21**	**3**

Table 5: The test accuracy (%) with varying number of clients (m) on CIFAR-10. 50 clients are active in each training round.

Method	System Size		
	$m = 50$	$m = 100$	$m = 150$
FedAvg	59.71	62.38	68.38
FedAvg-FT	73.81	76.81	80.04
FedBABU	71.48	74.75	76.78
FedRep	63.19	63.85	65.68
Ditto	72.63	75.88	77.83
FedRoD	72.81	76.76	79.51
CompFL (ours)	**76.67**	**79.59**	**82.54**

less transmitting bits, demonstrating the extraordinary advantage and extensibility of our proposed framework.

Scale the System. Finally, to further verify the adaptability and scalability of our framework, we set the number of clients over $\{50, 100, 150\}$ and local data size to 200 for simulating larger systems with data scarcity in clients. Each client will be allocated with 5 classes of CIFAR-10 and the number of expert models is set to 3 for the 150 clients setting, including the simple CNN, ResNet-9 and ResNet-18. Client sampling is employed for communication reduction, where only 20 clients are active in each round. We compare our CompFL with baselines in Table 5. As expected, our method achieved the best performance, demonstrating its adaptability and robustness in a wide range of environments.

E EXTENSION TO DECENTRALIZED SYSTEMS

Extending the multi-model training scheme from the server-based systems to the decentralized systems with only device-to-device communication is non-trivial as each client may need to send and receive multiple models from each neighboring peer, which could make the communication protocol too complicated to design or increase the overhead significantly. In this work, we propose two simple yet effective frameworks by (1) sequential training over clients for each model architecture, where each model is trained by random walk over clients as illustrated in Figure 1, and different architectures can be trained simultaneously (**device-by-device, D/D**); (2) distributed parallel training over clients for each model architecture, where the model training is conducted as standard decentralized optimization and different architectures are sorted according the sizes in a descending manner and sequentially trained

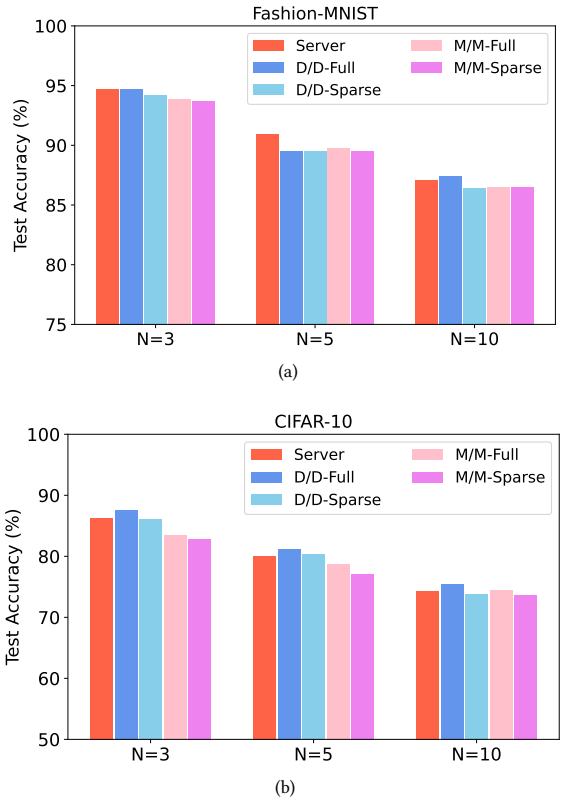


Figure 7: Performance comparison of server-based and two decentralized variants. Decentralized implementations can generally have similar accuracy as server-based implementation and sparsely connected network will result in slightly lower accuracy.

such that the smaller model can benefit from the larger model during training (**model-by-model, M/M**). By this way, the proposed training scheme in Section 3 can be easily adopted to support decentralized multi-model training. In the D/D settings, each client can decide whether to receive each expert model to conduct local training or directly forward to the next client. In the M/M settings, each expert model is trained in order and each client could choose to save the historically trained models to benefit future training.

We provide empirical evaluation results to verify the effectiveness in those two system settings. We apply the Erdos-Renyi graph model to simulate fully-connected network with edge creation probability p=1.0 and sparsely connected network with p=0.6. The experiment results in Figure 7 show that decentralized implementations can result in similar performance under various settings, which demonstrates the potential of our scheme in broader systems.