DISENTANGLED KNOWLEDGE TRANSFER: A NEW PERSPECTIVE FOR PERSONALIZED FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

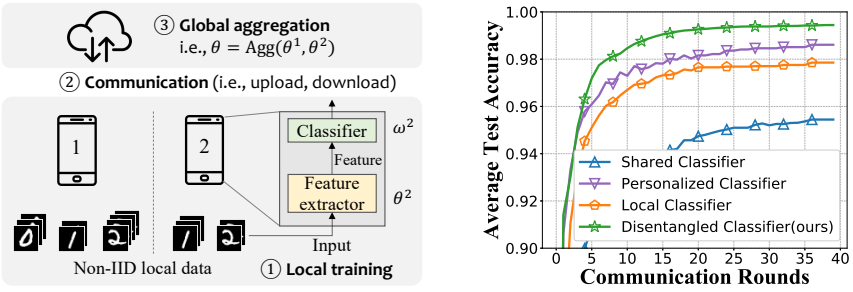
ABSTRACT

Personalized federated learning (pFL) is to collaboratively train non-identical machine learning models for different clients to adapt to their heterogeneously distributed datasets. State-of-the-art pFL approaches pay much attention on exploiting clients' inter-similarities to facilitate the collaborative learning process, meanwhile, can barely escape from the irrelevant knowledge pooling that is inevitable during the aggregation phase (e.g., inconsistent classes among clients), and thus hindering the optimization convergence and degrading the personalization performance. To tackle such conflicts between facilitating collaboration and promoting personalization, we propose a novel pFL framework, dubbed pFedC, to disentangle the global aggregated knowledge into several compositional branches and only aggregate relevant branches for supporting conflicts-aware collaboration among contradictory clients. Specifically, by reconstructing each local model into a shared feature extractor and multiple disentangled task-specific classifiers, the training on each client transforms into a mutually reinforced and relatively independent multi-task learning process, which provides a new perspective for pFL. Besides, we conduct a personalized aggregation mechanism on disentangled classifiers via quantifying the combination weights for each client to capture clients' common prior, as well as mitigate potential conflicts from the divergent knowledge caused by the heterogeneous data. Extensive empirical experiments are conducted over various models and datasets to verify the effectiveness and superior performance of the proposed algorithm.

1 INTRODUCTION

Recently, Federated Learning (FL) has gained growing attention for its capability of collaboratively training machine learning models among distributed clients without accumulating their private data in a central repository McMahan et al. (2017b); Yang et al. (2020). FL has successfully contributed to a wide range of smart applications, such as the next-word prediction Hard et al. (2018), voice recognition Sattler et al. (2020), and health care applications Xu et al. (2021). However, there exists a fundamental challenge that prevents its further application to users with data that having statistical defects, i.e., the data could be non-independent and identically distributed (non-IID). In such circumstance, the knowledge learnt from local training may have conflicts in the aggregation process, and thus sharing a global model for all clients may significantly slow down the training convergence process and seriously degrade the inference performance Jiang et al. (2019).

To deal with such data heterogeneity issue, personalized federated learning (pFL) has emerged that allows FL clients to train personalized models rather than an identical sharing model with others Hanzely et al. (2020); Tan et al. (2021); Huang et al. (2021). The principle is to explore the inter-user similarities from the joint training process meanwhile producing unique models that can adapt to their diverse data distributions Li et al. (2021a); T Dinh et al. (2020); Fallah et al. (2020); Collins et al. (2021); Liang et al. (2020); Li et al. (2021b); Zhang et al. (2021b); Huang et al. (2021); Zhang et al. (2021a). Among the statue quo pFL methods, two of the best popular solutions are similarity-based multiple-model scheme Zhang et al. (2021b); Huang et al. (2021); Zhang et al. (2021a) and parameter decomposition way Collins et al. (2021); Liang et al. (2020); Li et al. (2021b). The former one is to produce a personalized model for each client by dissecting the similarity among different



(a) Federated training (b) Different classifier

Figure 1: Motivation example: we consider 2 client to collaboratively learn their personalized models for a 3-class classification task. In a general federated training framework (a), the local model can be presented as a concatenation of a feature extractor (i.e., θ) and a classifier (i.e., ω). (b) comparison on different classifier aggregation mechanisms, *blue*: shared classifier, i.e., FedAvg McMahan et al. (2017b); *orange*: local classifier, i.e., FedRep Collins et al. (2021); *purple*: personalized classifier, i.e., FedFomo Zhang et al. (2021b); *green*: our proposed disentangled classifier, in which the original classifier is reconstructed into multiple class-specific branches, so as to avoid the knowledge of class 2 being transferred to client 2.

clients, while the latter one focuses on decoupling the whole model into two parts, top layers (e.g., feature extractor) and bottom layers (e.g., classifier), and only aggregating specific parameters (e.g., feature extractor) in the server.

Existing pFL methods have demonstrated improvements on FL performance via facilitating the collaboration among clients with similar data distributions, while very few literature have paid attention to the contradictory knowledge from users with irrelevant data (i.e., users have different classes or tasks), which may potentially degrade the aggregation efficiency if their cooperation is reinforced during the pFL process. For example, considering the aggregation phase on two clients with heterogeneous data distributions (i.e., Figure 1), the knowledge transferred from client 1 to 2 can be simply divided into two types: relevant knowledge (i.e., information on class 0) and irrelevant knowledge (i.e., information on class 2, 3). Traditional client-level aggregation mechanisms in pFL take knowledge from all other clients as a whole, and apply a unified combination weight for both the relevant and irrelevant parts. Worse still, the intertwined irrelevant knowledge transfer obtained from other clients would be aggravated among users with higher data similarity. Namely, **facilitating personalized collaboration cannot avoid irrelevant knowledge pooling.** As shown in Figure 1, by using different classifier aggregation mechanisms, the average model accuracy on customized classifiers outperforms that of a shared classifier, while the disentangled classifier has the best performance. This inspires us to investigate a more efficient approach to disentangle the coupled relationship between relevant and irrelevant knowledge transfer during the pFL aggregation process.

Motivated by the multi-task learning (MTL) paradigm, commonalities and differences across multiple related tasks can be exploited simultaneously and further reused to improve the learning performance for task-specific models Hashimoto et al. (2016); Ruder (2017); Sener & Koltun (2018). We seek to develop a novel pFL training framework that can circumvent the aggregation conflicts of the irrelevant knowledge derived from the distributed non-IID data during pFL training. To this end, we formulate the learning phase in pFL to an MTL-based conflicts-aware knowledge transfer training algorithm, dubbed pFedC, whose main idea is to disentangle the global aggregated knowledge into several compositional branches that can be trained in a mutually reinforced or relatively independent way according to their knowledge conflicts. Specifically, we disentangle the original model in a brand-new perspective and reconstruct the parameters into two types, i.e., the global representation parameters shared by all clients and multiple task-specific parameters owned by each client. The shared representation is to capture the generic knowledge across all clients, while the task-specific parameters are to optimize the personalization.

Different from previous pFL methods, e.g., FedFomo Zhang et al. (2021b) and FedRep Collins et al. (2021), pFedC disentangles the classifier at the semantic dimension to further improve the FL personalization. We demonstrate that pFedC can let clients selectively acquire relevant knowledge from others while excluding irrelevant knowledge transfer to avoid potential conflicts. The pro-

posed pFedC can even deal with the extreme case of user heterogeneity, e.g., each client needs to solve different classification tasks. Extensive experiments are conducted and we show that pFedC can outperform the state-of-the-art baselines over widely used models and datasets, i.e., EMNIST, FashionMNIST, CIFAR10 and CIFAR100.

The contributions of the paper are summarized as follows:

- To the best of our knowledge, we are the first to consider the conflicts between collaboration and personalization among different clients during the pFL training, and explicitly reveal the benefits of classifier disentanglement in pFL with non-IID data distributions.
- A flexible classifier disengagement mechanism based on semantic knowledge from different classes is proposed to avoid conflicts in knowledge transfer for each client.
- We further design a conflict-aware training framework, pFedC, that can efficiently exploit the inter-similarities among all clients while producing accurate personalized models for each client.
- We conduct extensive experiments on four typical image classification tasks and two non-IID data settings. The empirical evaluation shows the superior performance of pFedC over the state-of-the-art approaches.

2 Related Work

Personalized Federated Learning. Existing works on pFL can be categorized into data-based and model-based methods. The former ones focus on reducing the statistical heterogeneity among clients' datasets to boost the model convergence, i.e., data-sharing Zhao et al. (2018), data-augmentation Jeong et al. (2018); Duan et al. (2020). Model-based methods emphasize on producing customized model structures or parameters for different clients, which can also be divided into two types: single-model and multiple-model methods. Single-model based methods extended from the conventional FL algorithms like FedAvg McMahan et al. (2017a) combine the optimization of the local models and global model, including local fine-tuning Wang et al. (2019); Schneider & Vlachos (2019); Arivazhagan et al. (2019), regularization T Dinh et al. (2020); Hanzely & Richtárik (2020); Hanzely et al. (2020), model mixture Mansour et al. (2020); Deng et al. (2020), meta-learning Fallah et al. (2020); Jiang et al. (2019) and parameter decomposition Bui et al. (2019); Collins et al. (2021); Arivazhagan et al. (2019); Oh et al. (2022). Considering the diversity and inherent relationship of local data, a multi-model-based approach is applied to train multiple global models for heterogeneous clients Huang et al. (2021); Smith et al. (2017a); Ghosh et al. (2020); Mansour et al. (2020); Huang et al. (2021); Zhang et al. (2021b). The main idea is to facilitate the collaboration among different clients with similar data distributions.

Existing pFL researches focus on designing efficient methods to facilitate collaboration among clients by measuring inter-similarities, yet none of them have considered the conflict problem caused by irrelevant knowledge aggregated from other clients, especially for those who have higher similarities among their local data distributions. To this end, we need to explore a novel pFL method to efficiently and selectively capture the relevant knowledge and circumvent the irrelevant knowledge transfer from other clients.

Multi-task Learning. Multi-task learning (MTL) refers to learning a single model that can solve multiple different tasks Hashimoto et al. (2016); Ruder (2017); Sener & Koltun (2018). By sharing parameters between all tasks, MTL can learn more efficiently with an overall model with a smaller size compared with traditional separate model learning methods Zhang & Yang (2021); Vandenhende et al. (2021); Chen et al. (2018); Liu et al. (2019); Kendall et al. (2018). Multi-class learning can be seen as a special case of MTL, in which the classification on each class can be considered as one task Li et al. (2020); Huang et al. (2020). Smith et al. Smith et al. (2017a) are the first to apply multi-task learning to the FL settings. Unfortunately, the proposed MOCHA algorithm can only train linear models. FedEM Marfoq et al. (2021) learns personalized models by implicitly clustering all clients into M groups. However, all above federated MTL approaches treat each client as a single task, neglecting the inherent MTL attribute (i.e., multiple classes) of each client.

Summary. Aiming to eliminate the conflict between collaboration and personalization during pFL training, we are motivated to reformulate the original pFL task on each client into an MTL problem

to improve the learning efficiency meanwhile mitigating the aggregation conflicts from heterogeneous clients. By such means, the global aggregated knowledge can be disentangled into multiple complementary branches, so as to help optimize both the local training and global aggregation.

3 PROBLEM FORMULATION

3.1 NOTATIONS AND PRELIMINARY

In pFL, the goal is to collaboratively train personalized models among multiple clients to adapt to the distributed heterogeneous data. Considering N clients with non-IID datasets, let $\mathcal{D}_i = \{x_j^i, y_j^i\}_{i=1}^{D_i}$ $(1 \leq i \leq N)$ be the dataset on the i-th client, where $x_j \in \mathcal{X}$ is the j-th input data sample, $y_j \in \mathcal{Y}$ is the corresponding label. $\mathcal{X} \in \mathbb{R}^d$ and $\mathcal{Y} = \{1, 2, \ldots, C\} \in \mathbb{R}$ represent the input and output space, respectively. We define $\Pi_i = [\pi_i^1, \pi_i^2, \ldots, \pi_i^C]$ as the label distribution on i-th client, in which each element π_i^c is a binary indicator that represents whether label c exits in i-th client (i.e., $\pi_i^c = 1$) or not (i.e., $\pi_i^c = 0$). Let $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_N\}$ denotes the whole label distributions on all clients. The size of the datasets on the i-th client is denoted by D_i. The size of all clients' datasets is $D = \sum_{i=1}^{N} D_i$. The model on i-th client is denoted by q_i: $\mathcal{X} \to \mathcal{Y}$, which maps input to predicted labels, and is parameterized by θ_i. The objective of pFL can be formulated as

$$\min_{\Theta} \sum_{i=1}^{N} \frac{D_i}{D} \mathcal{L}_i(\theta_i), \tag{1}$$

where $\mathcal{L}_i(\theta_i) = \frac{1}{D_i} \sum_{j=1}^{D_i} \ell(\theta_i; x_j^i, y_j^i)$ and $\Theta = \{\theta_i, \ldots, \theta_N\}$ is the set of personalized parameters for all clients. $\mathcal{L}_i$ is loss function of i-th client associated with dataset $\mathcal{D}_i$. The difference between the predicted value and the true label of data samples is measured by ℓ, e.g., cross-entropy loss.

3.2 TASKWISE MODEL DISENTANGLEMENT

We reformulate the original pFL problem into an MTL process to avoid potential aggregation conflicts. Inspired by recent works on disentangled representations Luo et al. (2022); Sarhan et al. (2020), we first re-construct the model parameters into two different types of parameters: *shared representation (ϕ)* which denote the global and common knowledge for all tasks, and *disentangled task-specific parameters (ω)* that capture the local conflicts-aware knowledge for each client. The latter one can be further disentangled into T parts: $\omega \triangleq \{\omega^1, \ldots, \omega^T\}$. Note that the number of virtual tasks T can be set flexibly, i.e., $T \leq C$. When $T < C$, it means a C-class classification task is grouped into T virtual sub-tasks, in which each virtual task becomes a new classification task. For simplicity, in the formulation part, we disentangle the task-specific parameters according to the number of classes, which is reasonable in practical applications. In other words, we disentangle the original task into C (i.e., $T = C$) binary classification tasks, in which the output space for task c covert to $\mathcal{Y}^c = \{0, 1\}$. Then the whole local model at i-th client can be represented as

$$\theta_i = \left\{ \phi_i \circ \omega_i | \omega_i \triangleq \{\omega_i^1, \ldots, \omega_i^c, \ldots, \omega_i^C\} \right\}. \tag{2}$$

Given the MTL-based version of original model, we then reformulate the whole optimization problem. Referring to existing hypothesis and loss functions definitions in existing MTL literatureSener & Koltun (2018), we apply the typical minimization formulation at each client i as follows:

$$\mathcal{L}_i(\theta_i) = \sum_{c=1}^{C} \lambda_c \mathcal{L}_c(\phi_i, \omega_i^c), \tag{3}$$

where $\mathcal{L}_c(\phi_i, \omega_i^c) = \frac{1}{D_i^c} \sum_{j=1}^{D_i^c} \ell(\phi_i, \omega_i^c; x_j^i, \hat{y}_j^c)$. λ_c is the weight for task c, which can be static or dynamically computed during the training process. D_i^c denotes the number of data samples belonging to the class c. Since we reformulate the local objective in an MTL version, the j-th data sample on i-th client $\{x_j^i, y_j^i\}$ can be re-represented as $\{x_j^i, \hat{y}_j^{i,1}, \ldots, \hat{y}_j^{i,C}\}$. We omit the symbol i in $\hat{y}_j^{i,c}$ in the later content for simplicity.

For each client i, the local training becomes a multi-task learning process. To achieve conflicts-aware pFL, we should design an efficient MTL-based training and aggregation mechanism that can facilitate collaboration among clients with similar data distribution meanwhile reducing the negative/irrelevant knowledge transfer from other clients.

Algorithm 1 pFedC Algorithm

Require: $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N\}$, η, E, $\{\Pi_i\}_{i=1}^N$, T.
Ensure: Trained personalized models $\{\theta_1, \theta_2, \ldots, \theta_N\}$.
Initialize the clients' model parameters
procedure SERVER EXECUTES
  for each communication round $t \in \{1, \ldots, T\}$ **do**
    Receive $\{\theta_i\}_{i=1}^N$ from all the clients
    Calculate ϕ via Eq. (6)
    for each client i **in parallel do**
      Calculate ω_i via Eq. (7)
    Distributed ϕ and $\{\omega_i\}_{i=1}^N$ accordingly
    for each client i **in parallel do**
      $\theta_i \leftarrow \phi \circ \omega_i$
      $\theta_i \leftarrow$ CLIENTUPDATE(θ_i)
  return Trained personalized models $\{\theta_1, \ldots, \theta_N\}$
procedure CLIENTUPDATE(θ_i)
  Client i receives θ_i from the server.
  for each local epoch $e \in \{1, \ldots, E\}$ **do**
    for mini-batch $\xi_t \subseteq \mathcal{D}_i$ **do**
      Update model parameters θ_i via Eq. (5)
  return θ_i

4 pFedC ALGORITHM

To conduct efficient pFL via the MTL training, we update the shared representation and task-specific parameters at the server side. We apply a conflict-aware personalized federated learning framework to select only the relevant knowledge from other clients during the aggregation phase.

4.1 CONFLICTS-AWARE PERSONALIZED FEDERATED LEARNING

In this section, we elaborate the conflicts-aware model training and aggregation mechanism in pFedC step by step. Since the shared representation should capture the generic and common knowledge from all data samples, we let each client to maintain a global version of shared representation at the server side, and download it to local side to form a new local model before each communication round, i.e., $\phi \leftarrow \sum_{i=1}^N \frac{D_i}{D}\phi_i$. As for task-specific parameters, we introduce a mask vector m_i at the server side to guide the aggregation phase of i-th client:

$$m_i = \left[m_i^1, m_i^2, \ldots, m_i^C\right] = \begin{bmatrix} m_{i,1}^1 & m_{i,1}^2 & \cdots & m_{i,1}^C \\ m_{i,2}^1 & m_{i,2}^2 & \cdots & m_{i,2}^C \\ \vdots & \vdots & \ddots & \vdots \\ m_{i,N}^1 & m_{i,N}^2 & \cdots & m_{i,N}^C \end{bmatrix}, \tag{4}$$

where m_i^c represents the aggregation weight vector of c-th task in i-th client, while each element $m_{i,n}^c$ denotes the aggregation decision of task c on n-th client for i-th client, i.e., $m_{i,n}^c = 1$ if and only if client i and n both have class c; otherwise $m_{i,n}^c = 0$. Therefore, the value of $m_{i,n}^c$ can be easily calculated by $m_{i,n}^c = \pi_i^c \cdot \pi_n^c$ $(1 \leq i, n \leq N)$.

In aggregation phase, each client i calculates the personalized task-specific parameters ω_i by an element-wise multiplication of m_i and all local $\{\omega_n\}_{n=1}^N$. Then, the server can update the personalized model consisting of the shared representation and the task-specific parameters for each client.

4.2 WORKFLOW

As discussed earlier, pFedC enables each client to perform an MTL locally, and the server aggregates the local model parameters in a conflict-aware manner. To do so, pFedC alternates between client updates and a server update during each communication round. The overall learning scheme is summarized in Algorithm 1.

Client Update. In each communication round, the clients download the shared representation ϕ and personalized task-specific parameters ω_i from the server. In the client update phase, each of them

makes several local gradient-based updates (e.g., E epochs) to optimize the objective in Eq. (3) by adapting the MGDA-UB algorithm Sener & Koltun (2018) from MTL. Namely, for each iteration, client i updates its model as follows,

$$\theta_i \leftarrow \theta_i - \eta \nabla_{\theta_i} \mathcal{L}_i(\theta_i), \tag{5}$$

where $\theta_i = \{\phi \circ \omega_i\}$, and η denotes the learning rate. $\nabla_{\theta_i} \mathcal{L}_i(\theta_i)$ is the gradient of local objective function $\mathcal{L}_i(\theta_i)$ with respect to θ_i.

Server Update. After training the local models at the client-side, corresponding models will be sent to the server for further aggregation. Specifically, the server first aggregates the shared representation by taking a weighted averaged of them:

$$\phi = \sum_{i=1}^{N} \frac{D_i}{D} \phi_i. \tag{6}$$

The personalized task-specific parameters for each task c of client i can be aggregated by using mask vector m_i, i.e.,

$$\omega_i^c = \begin{cases} \frac{1}{\sum_{n=1}^{N} m_{i,n}^c} \sum_{n=1}^{N} \omega_n^c \cdot m_{i,n}^c, & \text{if } \pi_i^c = 1, \\ \omega_i^c, & \text{otherwise.} \end{cases} \tag{7}$$

Updated shared representation and personalized task-specific parameters are then distributed to the local clients for next round of training.

4.3 Analysis on pFedC

Convergence Analysis. We prove that pFedC can assist clients converge to their respective local optimums under the following assumptions: 1) $\mathcal{L}_1, \cdots, \mathcal{L}_N$ are all μ-strongly-convex, 2) $\mathcal{L}_1, \cdots, \mathcal{L}_N$ are all L-smooth, 3) the variance of stochastic gradients in each user is bounded by σ_i^2 and 4) the expected squared norm of stochastic gradients is uniformly bounded by G^2.

Theorem 1. *Let all above assumptions hold and μ, L, σ_i, G are defined therein. Choose $\kappa = \frac{L}{\mu}$,$\gamma = \max\{8\kappa, E\}$ and the learning rate $\eta_t = \frac{2}{\mu(\gamma+t)}$. Suppose task c exists in client i (i.e., $\pi_i^c = 1$), then client i in pFedC satisfies*

$$\mathbb{E}[\mathcal{L}_i(\theta_i)] - \mathcal{L}_i^* \leq \frac{\kappa}{\gamma + T - 1} \left(\frac{2B}{\mu} + \frac{\mu\gamma}{2} \mathbb{E}\|\theta_{i,0} - \theta_i^*\|^2 \right) \tag{8}$$

where

$$B = \sum_{n=1}^{N} \frac{{p_n}^2 {\pi_n^c}^2}{{\epsilon^c}^2} \sigma_n^2 + 6L\Gamma + 8(E-1)^2 G^2, \Gamma = \mathcal{L}_i^* - \sum_{n=1}^{N} \frac{p_n \pi_n^c}{\epsilon^c} \mathcal{L}_n^* \geq 0 \tag{9}$$

The full convergence analysis proof is elaborated in Appendix 7.3. From Theorem 1, we show that pFedC converges to the respective local optimums at a rate of $\mathcal{O}(\frac{1}{T})$ for strongly convex and smooth functions.

Privacy Preserving. In pFedC, the server requires to access $\{\Pi_i\}_{i=1}^N$ from local clients, which may lead to privacy concerns on user data profiles. However, there are already some literature Tan et al. (2022); Zhu et al. (2021) show that transmitting local category-level information is considered reasonable and will not cause privacy leakage in image classification tasks, where different clients have different subsets of the whole classes. Thus, it is feasible to upload data distribution information to boost personalization performance in a federated framework. Moreover, pFedC can be integrated with various privacy-preserving techniques, such as Homomorphic Encryption (HE) Phong et al. (2018), differential privacy Wei et al. (2020) and even trusted execution environment Mondal et al. (2021) to enhance the reliability of the training framework. Taking Homomorphic Encryption as an example, we design a privacy-preserved version of pFedC. As space is limited, we omit it in the main content and explain the details in Appendix 7.4.

Communication Efficiency. Compared with conventional pFL methods that only transmit local model parameters or partial parameters, e.g., top layers in FedRep Collins et al. (2021), our proposed pFedC requires the exchange of multi-branches model parameters, which may lead to additional communication overhead. However, we clarify that the additional communication cost on pFedC

Table 1: Task Accuracy over 20 clients on the MNIST, EMNIST and CIFAR10, CIFAR100 datasets(Pathological non-IID setting, T denotes the number of disentangled tasks).

	MNIST (%)	EMNIST (%)	CIFAR10 (%)	CIFAR100 (%)
(# Tasks, # Local Epochs)	(10, 5)	(10, 5)	(10, 10)	(20, 10)
FedAvg McMahan et al. (2017b)	91.94±0.76	85.32±0.38	53.70±0.98	46.29±0.26
Per-FedAvg Fallah et al. (2020)	92.95±0.28	81.81±1.14	60.33±1.83	54.49±1.55
pFedMe T Dinh et al. (2020)	93.30±0.44	86.56±0.50	57.20±0.86	51.70±0.38
FedBN Li et al. (2021b)	97.73±0.34	93.53±0.42	61.43±0.37	61.35±0.09
FedRep Collins et al. (2021)	97.85±0.15	93.94±0.12	60.93±0.26	63.20±0.57
FedFomo Zhang et al. (2021b)	98.61±0.29	92.31±0.17	58.65±0.34	58.50±0.12
pFedC (Ours)	**99.44**±0.41	**97.94**±1.35	**90.37**±0.29	**93.49**±0.66

can be negligible, which only has about 0.011% increments. We will conduct detailed explanations on communication efficiency of pFedC in the experimental part.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate the pFedC framework over four datasets, EMNIST, FashionMNIST, CIFAR10 and CIFAR100. We consider two non-IID scenarios: 1) pathological non-IID setting — each client is randomly assigned two classes with the same amount of data on each class McMahan et al. (2017b); 2) real-world non-IID setting — each client contains all classes, while the data on each class is not uniformly distributed, i.e., using a Dirichlet distribution **Dir**(α) Hsu et al. (2019); Lin et al. (2020), where α indicates the non-IID level, and a smaller value of α indicates higher data heterogeneity. All data are divided into 75% training set, and 25% test set. The test set and the training set have the same data distribution for all clients.

Baselines. We compared the performance of pFedC with the state-of-the-art methods. In addition to **FedAvg** McMahan et al. (2017b), we also include **Per-Fedavg** Fallah et al. (2020), a pFL algorithm based on meta-learning; **pFedMe** T Dinh et al. (2020), a pFL algorithm with regularization term added in the objective function; **FedBN** Li et al. (2021b), keeps each client's BN layer updating locally, while other layers are aggregated according to the FedAvg algorithm; **FedRep** Collins et al. (2021), a pFL algorithm that keeps each client's classifier updating locally, while the other parts are aggregated at the server; **FedFomo** Zhang et al. (2021b), a pFL algorithm that uses distance to calculate the aggregation weights based on the model and loss differences.

Training Details. For MNIST and EMNIST dataset, we use a CNN architecture based on LeNet LeCun et al. (1998). We treat all layers except the last as the shared representation function and put ten fully-connected layers as task-specific functions. For CIFAR10 and CIFAR100 dataset, we use an architecture based on ResNet-18 He et al. (2016) and make similar modifications to it as LeNet. The detailed architectures used for experiments is visualized in Appendix 7.2. It's worth noting that we disentangle the model of CIFAR100 based on the number of superclasses (i.e., 20). In all baselines, we use the original LeNet and RestNet-18 architectures. All the models have the same structure between different clients under the same setting. We evaluate the performance of pFedC on 20 clients with 100% participation, and 100 clients with 10% participation, respectively. The average model accuracy of all clients is obtained after 1000 communication rounds training for all baselines and 40 communication rounds training for pFedC, respectively.

Implementation. We simulate all clients and the server on a workstation with an RTX 2080Ti GPU, a 3.6-GHZ Intel Core i9-9900KF CPU and 64GB of RAM.

5.2 Experimental Results

Performance Comparison. We compare the performance of pFedC with all baselines under two different non-IID cases: pathological non-IID setting and real-world non-IID setting. For all experiments, we use SGD optimizer with a batch size of 32 for MNIST and EMNIST, 16 for CIFAR10 and CIFAR100. The learning rate is 0.01 for CIFAR10 and CIFAR100, and 0.005 for MNIST and EMNIST. The performance of all baselines is evaluated by the mean testing accuracy, which is the

Table 2: Task Accuracy over 20 clients on the MNIST, EMNIST and CIFAR10, CIFAR100 datasets (Real-world non-IID setting, T denotes the number of disentangled tasks). For MNIST and EMNIST, the local epochs is set as 5; For CIFAR10 and CIFAR100, the local epochs is set as 10.

Average Test Accuracy								
Dataset	α	FedAvg	Per-FedAvg	pFedMe	FedBN	FedRep	FedFomo	pFedC (Ours)
MNIST $T=10$	0.05	54.59±0.12	65.64±0.59	67.27±0.28	68.77±0.42	74.82±0.29	70.66±0.37	**80.05**±0.28
	0.1	57.72±0.32	88.21±0.57	92.84±0.38	83.57±0.21	71.88±0.37	92.25±0.74	**94.99**±0.46
	1	84.56±0.18	90.08±1.12	94.03±0.95	93.24±0.82	89.44±0.44	80.60±1.11	**96.87**±0.28
EMNIST $T=10$	0.05	60.39±0.40	67.92±0.94	**80.18**±0.19	64.90±1.19	64.20±1.07	65.68±0.86	76.34±0.96
	0.1	74.68±0.51	69.02±0.11	82.77±0.82	65.34±0.94	65.65±1.66	65.29±0.93	**89.18**±0.82
	1	77.89±0.16	69.59±0.91	84.37±0.14	75.02±0.33	72.23±0.42	66.02±0.50	**91.01**±0.54
CIFAR10 $T=10$	0.05	34.88±0.08	48.08±0.56	52.19±0.23	53.27±0.73	53.61±0.57	53.17±0.74	**70.65**±0.34
	0.1	48.12±0.11	64.37±0.38	69.20±0.29	70.13±0.31	70.34±0.49	70.13±0.51	**77.85**±0.24
	1	61.37±0.18	67.24±1.07	73.12±0.56	74.98±0.92	74.20±0.16	70.70±0.08	**83.04**±0.34
CIFAR100 $T=20$	0.05	36.99±1.04	58.63±0.08	62.91±0.14	64.72±0.51	64.06±0.88	63.41±0.11	**81.79**±1.60
	0.1	54.16±0.09	62.44±0.91	66.24±0.98	68.36±0.96	68.18±1.89	67.31±0.83	**85.91**±0.26
	1	58.56±0.17	75.66±0.37	79.38±0.25	81.75±0.13	82.25±0.70	79.50±1.01	**87.69**±0.37

Table 3: Performance comparison (%) on CIFAR-100 with 100 clients ($T = 20$).

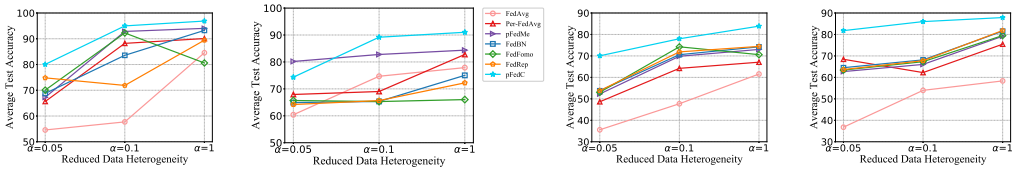
Method / Data Setting	FedAvg	Per-FedAVg	pFedMe	FedBN	FedRep	FedFomo	pFedC(Ours)
classs = 2	45.19	43.72	53.60	56.16	56.47	44.28	92.15
$\alpha = 0.05$	40.89	42.56	44.35	46.42	43.51	29.63	84.03
$\alpha = 0.1$	54.03	54.40	58.23	60.62	60.36	47.08	89.25
$\alpha = 1$	60.10	77.28	80.12	82.01	81.19	78.07	90.98

average of the testing accuracy on all clients and the $\pm$ indicates the error range of the mean testing accuracy after 3 repeated experiments. For pFedC, we evaluate the performance by calculating the mean testing accuracy on all tasks. During the training process, MGDA-UB Sener & Koltun (2018) is used to solve the local optimization problem on each client. The experimental results are listed in Table 1 and Table 2, respectively.

Table 1 demonstrates the mean testing accuracy of all methods on four datatsets with the pathological non-IID data setting. It can be observed that pFedC constantly outperforms other competitors by large margins under a variety of settings. The reason mainly lies in the MTL-based local training and personalized aggregation of the pFL models, which encourages the clients to optimize the objective in a conflict-aware manner. For real-world non-IID setting, Table 2 shows the results of FedAvg, Per-FedAvg, pFedMe, FedBN, FedRep, FedFomo and our pFedC on four datasets under three different level of data heterogeneity (i.e., $\alpha = 0.05, 0.1, 1$). From Table 2, we can notice that pFedC provides superior performance than the baselines over different datatsets and data distributions in most cases. Besides, we emphasize that even if all clients own all types of classes (e.g., $\alpha = 1$), the performance of pFedC is still better than that of FedAvg. This is enabled by the disentangled architecture and task-specific aggregation process.

Effect of Data Heterogeneity. To further verify the efficiency of our proposed pFedC, we investigate the effect of data heterogeneity on training performance. The final mean test accuracies of each algorithm under different value of $\alpha = 0.05, 0.1, 1$ are recorded to plot the trends between performance and data heterogeneity. As shown in Figure 2, pFedC can provide stable gain over other methods when the data distributions are highly heterogeneous (i.e., with smaller α). In most cases, pFedC shows an obvious advantage in training performance, while the advantage of Per-FedAvg, FedRep, FedFomo vanishes as data heterogeneity alleviates. These results show the robustness of pFedC under different levels of data heterogeneity.

Scalability. To verify the scalability of pFedC, we further conduct experiments on large-scale FL systems, where the number of clients reaches 100. During the training process of pFedC, partial clients (sampling rate: 0.1) will be selected and related partial elements in the mask matrix will be used for aggregation in each communication round. In Table 3, the experimental results demonstrate the efficiency and scalability of pFedC. Besides, we notice that disentangling the original bottom layer (i.e., classifier) into T parts in pFedC only expands the width in the last FC layer while the depth has not changed at all. Taking ResNet-18 on CIFAR-100 for example, one FC layer (50×1) is disentangled and reconstructed into twenty FC layers (50×1) for pFedC (in our experimental setting). The parameter size on the original model and our proposed model are 8,869,427 and



(a) MNIST Dataset. (b) EMNIST Dataset. (c) CIFAR10 Dataset. (d) CIFAR100 Dataset.

Figure 2: Performance v.s. data heterogeneity

Table 4: Effect of different disentanglement strategies on CIFAR100 (Real-world non-IID setting, T denotes the number of disentangled tasks). The local epochs is set as 10.

Data Setting	# Number of Tasks			
	$T=1$	$T=10$	$T=20$	$T=100$
$\alpha=0.05$	37.99	97.30	74.94	62.48
$\alpha=0.1$	54.16	95.38	76.14	62.50
$\alpha=1$	58.56	90.69	76.36	62.57

8,870,396, respectively, which has only 0.011% increments. Compared with the original model, the additional cost on memory, computation and communication are negligible.

Comparison with Federated MTL methods. To further demonstrate the efficiency of MTL process in pFedC, we compare it with MOCHA Smith et al. (2017b) and FedEM Marfoq et al. (2021). Since MOCHA only works for linear models, we conduct experiments on two different models: linear models and CNN models (i.e., ReNet-18). We tuned the parameter λ of MOCHA on a holdout validation set via grid search in $\{10^1, 10^0, 10^1, 10^2, 10^3\}$, and we found that the optimal value of λ is 10^0. Unless otherwise stated, the number of components considered by FedEM is $M = 3$. We show that pFedC can obtain the best performance when compared with other federated MTL methods. The detailed results are shown in Table 6.

Effect of Different Disentanglement Strategies. For large-scale classification datasets, disentangling the original model at a class level may be infeasible due to the huge number of classes in the dataset. In this case, how to disentangle the model becomes important and should be carefully investigated. Therefore, We conduct analysis on the effect of disentanglement strategies on CIFAR100. Three different disentanglement strategies are proposed in the experiments: 1) $T = 10$, splitting the 100 classes into 10 groups and considering each 10 consecutive classes as a task von Oswald et al. (2019); 2) $T = 20$, splitting the dataset according to the superclasses (i.e., 20 superclasses) Yoon et al. (2019); 3) $T = 100$, splitting the dataset according to the original classes (i.e., 100 classes). $T = 1$ represents no model disentanglement in the training process, which is equivalent to FedAvg.

As shown in Table 4, the training performance on CIFAR100 with different data heterogeneity varies greatly with the number of tasks (i.e., T). Specifically, comparing with FedAvg (i.e., $T = 1$), pFedC achieves a significant improvement on model accuracy, e.g., a 32.13% - 59.31% improvement over FedAvg when $T = 10$, and 4.01% - 24.49% improvement over FedAvg when $T = 100$. These results verify the efficiency of our proposed model disentanglement-based pFL training framework. However, larger values of T cannot always benefit the convergence of the personalized models. The reason is that a large number of tasks makes it extremely difficult to learn the relationship between these different tasks. Moreover, larger values of T may bring disadvantages to the whole training process due to the huge increase in model size and local training time. Thus, a proper setting of T is necessary and there are many things to take into consideration, e.g., the computation overhead, communication overhead, and model performance.

6 Conclusion

In this paper, we have investigated the potential conflicts caused by the irrelevant knowledge transfer from heterogeneous clients in pFL, and integrated a novel model disentanglement method (pFedC) into the pFL training framework, which can improve the training efficiency and model performance during the learning process over non-IID data. It is shown that pFedC can capture the relevant knowledge and circumvent the irrelevant knowledge transfer from other clients. Extensive empirical experiments have been conducted over various models and datasets to verify the effectiveness and superior performance of pFedC framework.

REFERENCES

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

Duc Bui, Kshitiz Malik, Jack Goetz, Honglei Liu, Seungwhan Moon, Anuj Kumar, and Kang G Shin. Federated user representation learning. *arXiv preprint arXiv:1909.12535*, 2019.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pp. 794–803, 2018.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2089–2099, 2021.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(1):59–71, 2020.

Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. In *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.

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.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Gale Yan Huang, Jiahao Chen, Haochen Liu, Weiping Fu, Wenbiao Ding, Jiliang Tang, Songfan Yang, Guoliang Li, and Zitao Liu. Neural multi-task learning for teacher question detection in online classrooms. In *International Conference on Artificial Intelligence in Education*, pp. 269–281. Springer, 2020.

Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, volume 35, pp. 7865–7873, 2021.

Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Na Li, Huizhen Hao, Zhiwei Jiang, Feng Jiang, Ronghua Guo, Qing Gu, and Xiumian Hu. A multi-task multi-class learning method for automatic identification of heavy minerals from river sand. *Computers & Geosciences*, 135:104403, 2020.

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

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*, 2021b.

Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1871–1880, 2019.

Zhengquan Luo, Yunlong Wang, Zilei Wang, Zhenan Sun, and Tieniu Tan. Disentangled federated learning for tackling attributes skew via invariant aggregation and diversity transferring. In *International Conference on Machine Learning*, pp. 14527–14541. PMLR, 2022.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems*, 34:15434–15447, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2017a.

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

Arup Mondal, Yash More, Ruthu Hulikal Rooparaghunath, and Debayan Gupta. Poster: Flatee: Federated learning across trusted execution environments. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 707–709. IEEE, 2021.

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

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Mhd Hasan Sarhan, Nassir Navab, Abouzar Eslami, and Shadi Albarqouni. Fairness by learning orthogonal disentangled representations. In *European Conference on Computer Vision*, pp. 746–761. Springer, 2020.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3400–3413, 2020.

Johannes Schneider and Michail Vlachos. Personalization of deep learning. *arXiv preprint arXiv:1909.02803*, 2019.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *NeurIPS*, 2018.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2017a.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *NeurIPS*, pp. 4424–4434, 2017b.

Canh T Dinh, Nguyen Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 33, 2020.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *arXiv preprint arXiv:2103.00710*, 2021.

Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *AAAI Conference on Artificial Intelligence*, volume 1, 2022.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19, 2021.

Miao Yang, Akitanoshou Wong, Hongbin Zhu, Haifeng Wang, and Hua Qian. Federated learning with class imbalance reduction. *arXiv preprint arXiv:2011.11266*, 2020.

Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. *arXiv preprint arXiv:1902.09432*, 2019.

Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, and Feijie Wu. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems*, 34, 2021a.

Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. Personalized federated learning with first order model optimization. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*, 2021b.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 12878–12889. PMLR, 2021.

7 APPENDIX

7.1 COMPARISON WITH OTHER PFL METHODS

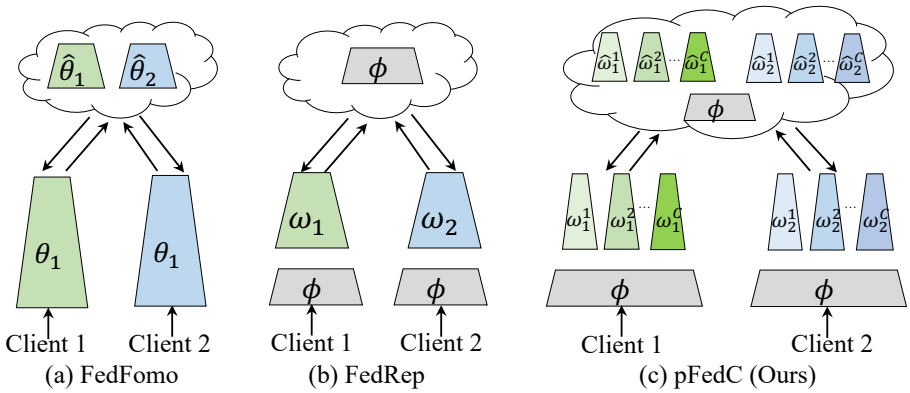


Figure 3: Illustration of different pFL methods. The previous approaches either maintain a personalized model for each client in a model-wise mechanism, e.g., FedFomo Zhang et al. (2021b) (a) or simply decouple the whole model into two parts: feature extractor and classifier in a layer-wise mechanism, e.g., FedRep Collins et al. (2021) (b). Instead, we achieve conflicts-aware model personalization via task-wise disentanglement (c).

As shown in Figure 3, different from previous pFL methods, e.g., FedFomo Zhang et al. (2021b) and FedRep Collins et al. (2021), pFedC disentangles the classifier at the semantic dimension to further improve the FL personalization. We demonstrate that pFedC can let clients selectively acquire relevant knowledge from others while excluding irrelevant knowledge transfer to avoid potential conflicts. The proposed pFedC can even deal with the extreme case of user heterogeneity, e.g., each client needs to solve different classification tasks.

7.2 ADDITIONAL EXPERIMENTAL DETAILS

In this section, we provide more concrete experimental settings and more numerical results to examine the advantages of our proposed pFedC, i.e., the convergence rate.

7.2.1 NON-IID DATA SETTING

Pathological non-IID setting: each client is randomly assigned two classes with the same amount of data on each class.

Real-world non-IID setting: each client contains all classes, while the data on each class is not uniformly distributed, i.e., using a Dirichlet distribution **Dir**(α), where α indicates the non-IID level, and a smaller value of α indicates higher data heterogeneity.

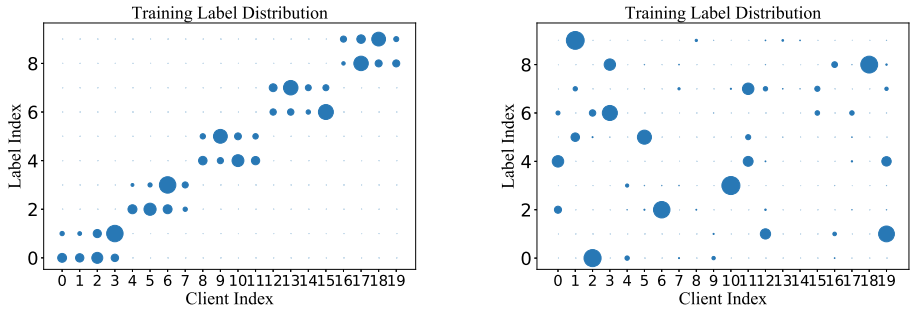
We visualize the effects of adopting different α on the data heterogeneity for the MNIST dataset in Figure 4.

7.2.2 DETAILED ARCHITECTURES USED FOR PFEDC

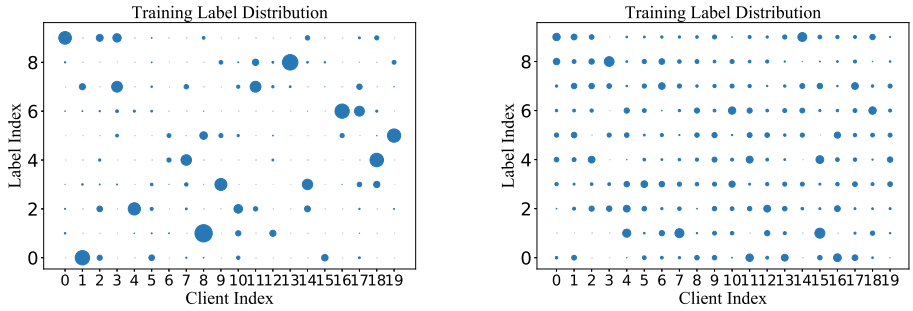
For MNIST and EMNIST dataset, we use a CNN architecture based on LeNet. We treat all layers except the last as the shared representation function and put ten fully-connected layers as task-specific functions. For CIFAR10 and CIFAR100 dataset, we use an architecture based on ResNet-18. Similarity, all layers except the last one are treated as shared representation for all tasks, and the fully connected layer are used as task-specific functions. The architectures used for different datasets are visualized in Figure. 5.

7.2.3 THE TRAINING PERFORMANCE ON DIFFERENT NON-IID SETTINGS

We compare pFedC with state-of-the-art baselines on four datasets: MNIST, EMNIST, CIFAR10 and CIFAR100. There are two non-IID data settings: 1) Pathological non-IID setting, where the



(a) classes per client = 2 (b) $\alpha = 0.05$



(c) $\alpha = 0.1$ (d) $\alpha = 1$

Figure 4: Visualization of statistical heterogeneity among users on MNIST dataset, where the x-axis indicates client index, the y-axis indicates label index, and the size of scattered points indicates the number of training samples for a label available to that client. (a) Pathological non-IID setting; (b) - (d) Real-world non-IID setting.

data on each client only contains the specific number of labels (maybe only two labels); 2) Real-world non-IID setting, where the number of labels for each client is randomly chosen.

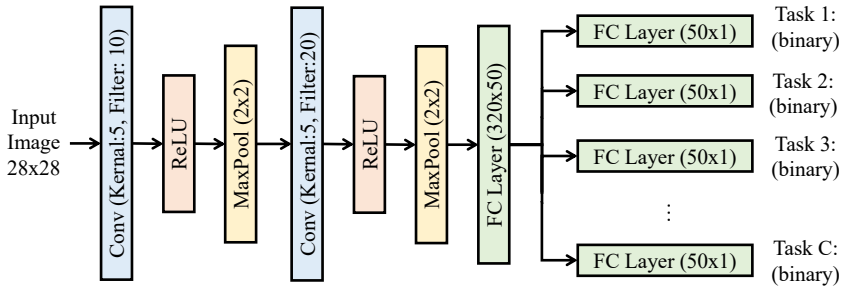
Figure 6, 7, 8, 9 shows the empirical convergence results of pFedC along with other baselines. Specifically, we focus on the changes of average test accuracy of these algorithms in each communication round. It is obvious that pFedC converges to higher average test accuracy on all four datasets than baselines. This phenomenon validate the effectiveness of the model decomposition based training and aggregation on pFL. Moreover, pFedC can achieve faster convergence speed than baselines in most cases, which further verifies the advantages of model decomposition and conflict-aware personalized aggregation on training performance.

Table 5: Required Rounds (Baselines|pFedC) on CIFAR-100 with different T.

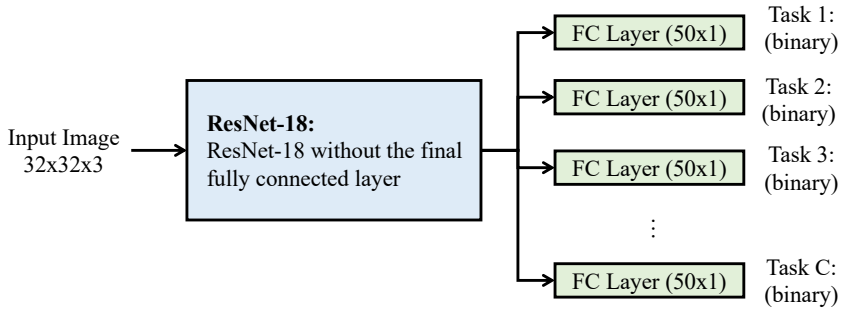
Setting	classes per client =2	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 1$
$T = 1$	244 |244	250 |250	200 |200	200 |200
$T = 10$	244 |55	250 |58	200 |41	200 |49
$T = 20$	244 |32	250 |28	200 |25	200 |22
$T = 100$	244 |22	250 |18	200 |19	200 |17

7.2.4 The Required Communication Rounds on Different Decomposition Strategies

Finally, we conduct experiments to analyze the convergence rate. From table 5, the required communication rounds to reach the convergence in pFedC is far less than baseline methods.



(a) Architecture used for MNIST and EMNIST.



(b) Architecture used for CIFAR10 and CIFAR100

Figure 5: Architecture used for MNIST, EMNIST, CIFAR10 and CIFAR100 experiments, respectively.

Table 6: Performance comparison over 20 clients on CIFAR10, dataset(T denotes the number of disentangled tasks).

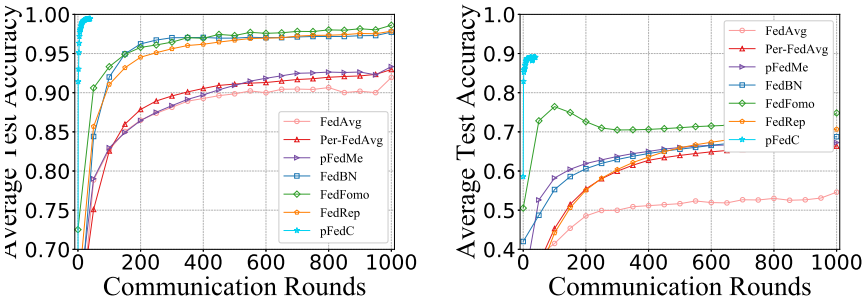
	Linear Model	ResNet-18		
(# Data heterogeneity)	classes per client=2	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 1$
MOCHA Smith et al. (2017a)	64.75±0.65	-	-	-
FedEM Marfoq et al. (2021)	65.40±0.01	67.12±0.57	75.79±0.66	81.66±0.32
FedMA Wang et al. (2020)	-	67.81±0.77	69.54±0.70	81.21±0.81
pFedMA (i.e., FedMA+FT)	-	68.83±0.51	75.15±0.53	82.09±0.65
pFedC (Ours)	77.23±0.52	70.65±0.34	77.85±0.24	83.04±0.34

7.2.5 Comparison with more pFL Methods

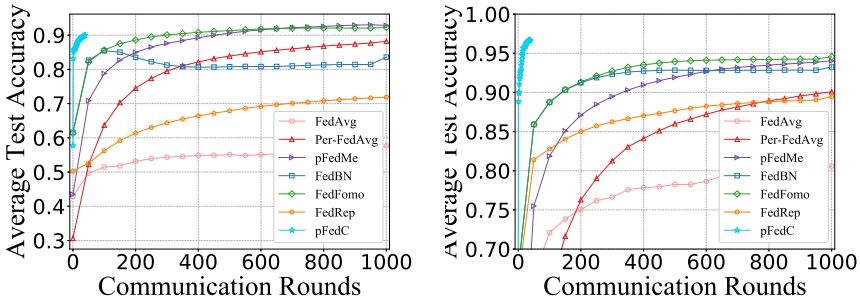
To further demonstrate the efficiency of MTL process in pFedC, we compare it with MOCHA Smith et al. (2017b) and FedEM Marfoq et al. (2021). Since MOCHA only works for linear models, we conduct experiments on two different models: linear models and CNN models (i.e., ReNet-18). We tune the parameter λ of MOCHA on a holdout validation set via grid search in $\{10^1, 10^0, 10^1, 10^2, 10^3\}$, and find that the optimal value of λ is 10^0. Unless otherwise stated, the number of components considered by FedEM is $M = 3$. From Table 6, we show that pFedC can obtain the best performance when compared with other federated MTL methods.

7.2.6 Effect of Local Epochs

Figure 10 demonstrates the performance of pFedC under different settings of local epochs (the local epochs E in each communication round are set to 3, 5, 10 and 20 respectively). It is observed that larger values of E are not always optimal for obtaining the best performance. For example, the model can achieve the highest averaged accuracy(e.g., 96.86%) on MNIST with $\alpha = 1$ when pFedC performs 5 epochs during one communication round, while 10 epochs are better for CIFAR10 ($\alpha = 1$).



(a) classes per client = 2 (b) $\alpha = 0.05$



(c) $\alpha = 0.1$ (d) $\alpha = 1$

Figure 6: Performance of pFedC comparede with baselines on MNIST datatset. (a) Pathological non-IID setting; (b) - (d) Real-world non-IID setting.

7.3 Convergence Analysis of pFedC

The personalized performance convergence analysis for each client is the same, so we only focus on one client $i, 1 \leq i \leq N$. Consider that all clients participate in the aggregation phase in pFedC.

7.3.1 Additional Notation

Let $\theta_{i,t}$ be the model parameters of i-th client at t-th step. Let E be the number of local updates in each round. Let $\mathcal{I}_E$ be the set of synchronization steps, i.e., $\mathcal{I}_E = \{nE | n = 1, 2, \cdots\}$. If $t+1 \in \mathcal{I}_E$, i.e., the step to communication. Then, the model update with all participants can be described as:

$$\mathrm{v}_{i,t+1} = \theta_{i,t} - \eta_t \nabla \mathcal{L}_i(\theta_{i,t}, \xi_{i,t}) \tag{10}$$

$$\theta_{i,t+1} = \begin{cases} \mathrm{v}_{i,t+1}, & \text{if} \quad t+1 \notin \mathcal{I}_E \\ \mathrm{Agg}(\mathrm{v}_{i,t+1} | i = 1, \cdots, N), & \text{if} \quad t+1 \in \mathcal{I}_E \end{cases} \tag{11}$$

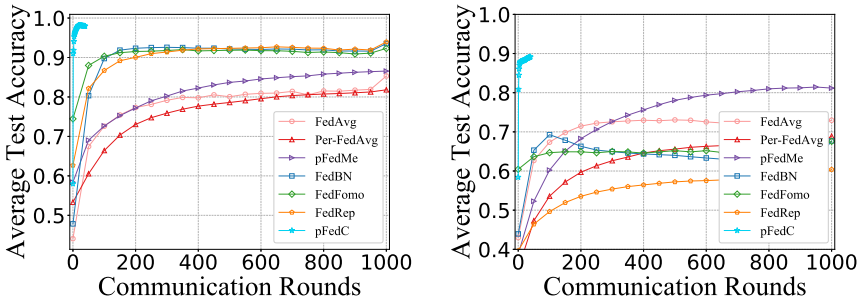
where

$$\mathrm{Agg}(\mathrm{v}_{i,t+1} | i = 1, \cdots, N) = \sum_{i=1}^{N} p_i \phi_{i,t+1} \circ \widetilde{\omega}_{i,t+1}, \tag{12}$$

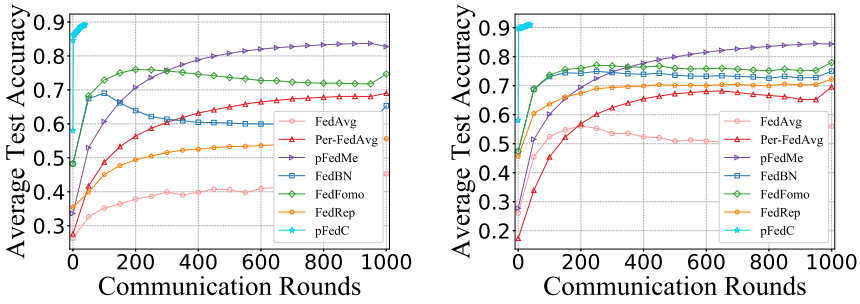
$$\widetilde{\omega}_{i,t+1} = \{\omega_{i,t+1}^c | c = 1, \cdots, C\}, \quad \omega_{i,t+1}^c = \begin{cases} \omega_{i,t+1}^c, & \text{if} \quad \pi_i^c = 0 \\ \sum_{n=1}^{N} \frac{p_n \pi_n^c}{\epsilon^c} \omega_{n,t+1}^c, & \text{if} \quad \pi_i^c = 1 \end{cases} \tag{13}$$

and $p_n = \frac{D_n}{D}$, $\epsilon^c = \sum_{n=1}^{N} m_n^c$, $\mathrm{v}_{i,t+1} = \{\phi_{i,t+1} \circ \omega_{i,t+1}\}$. Here, an additional variable $\mathrm{v}_{i,t+1}$ is introduced to represent the immediate result of one step SGD update from $\theta_{i,t+1}$. We regard $\theta_{i,t+1}$ as the parameter obtained after communication steps.

In following analysis, we define two virtual sequences $\bar{\mathrm{v}}_{i,t} = \mathrm{Agg}(\mathrm{v}_{i,t} | i = 1, \cdots, N)$ and $\bar{\theta}_{i,t} = \mathrm{Agg}(\theta_{i,t} | i = 1, \cdots, N)$. As the whole original model is divided into two different roles in our proposed pFedC: global feature extractor (i.e., ϕ) and task-specific classifier (i.e., ω). According to Eq. 11, the update policy for ϕ is same as FedAvg algorithm, so we omit the analysis on ϕ and only consider the task-specific parts. Moreover, for each task c, π_i^c has two different values, i.e., 0 or



(a) classes per client = 2 (b) $\alpha = 0.05$



(c) $\alpha = 0.1$ (d) $\alpha = 1$

Figure 7: Performance of pFedC comparede with baselines on EMNIST datatset. (a) Pathological non-IID setting; (b) - (d) Real-world non-IID setting.

1. For convenience, we only consider the case with $\pi_i^c = 1$. Therefore, in the rest of the analysis, $\mathrm{v}, \bar{\mathrm{v}}, \theta$ and $\bar{\theta}$ denote the parameter related to ω^c unless other specified. It is noticed that $\bar{\mathrm{v}}_{i,t+1}$ comes from an single step of SGD from $\bar{\theta}_{i,t}$. We also define $\bar{g}_{i,t}^c(\omega_i^c) = \sum_{n=1}^N \frac{p_n \pi_n^c}{\epsilon^c} \nabla_{\omega_n^c} \mathcal{L}_n(\theta_{n,t})$ and $g_{i,t}^c(\omega_i^c) = \sum_{n=1}^N \frac{p_n \pi_n^c}{\epsilon^c} \nabla_{\omega_n^c} \mathcal{L}_n(\theta_{n,t}, \xi_{n,t})$. For simplicity, we omit the w_i^c in the above expression of $\bar{g}_{i,t}^c$ and $g_{i,t}^c$. Thus, we have $\bar{\mathrm{v}}_{i,t+1} = \bar{\theta}_{i,t} - \eta_t g_{i,t}$ and $\mathbb{E} g_{i,t} = \bar{g}_{i,t}$.

7.3.2 Key Lemmas

To clearly show our proof, it is necessary to define some lemmas before the main theorem. The proof of these lemmas can be easily found in Li et al. (2019) and we only focus on the main theorem.

Lemma 1. *(Results of one step SGD). Assume Assumption 1 and 2 hold, we have*

$$\mathbb{E}\|\bar{\mathrm{v}}_{i,t+1} - \theta_i^*\| \leq (1 - \eta_t \mu)\mathbb{E}\|\bar{\theta}_{i,t} - \theta_i^*\|^2 + \eta_t^2 \mathbb{E}\|g_{i,t} - \bar{g}_{i,t}\|^2 + 6L\eta_t^2 \Gamma + 2\mathbb{E} \sum_{n=1}^N \frac{p_n \pi_n^c}{\epsilon^c} \|\bar{\theta}_{i,t} - \theta_{n,t}\|^2 \tag{14}$$

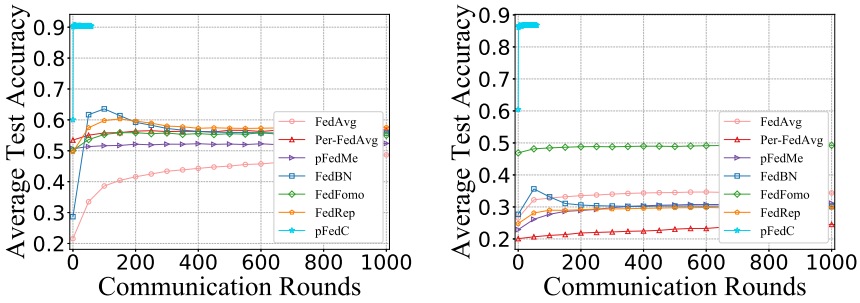
where $\Gamma = \mathcal{L}_i^* - \sum_{n=1}^N \frac{p_n \pi_n^c}{\epsilon^c} \mathcal{L}_n^* \geq 0$.

Lemma 2. *(Bounding the variance). Assume Assumption 3 holds. It follows that*

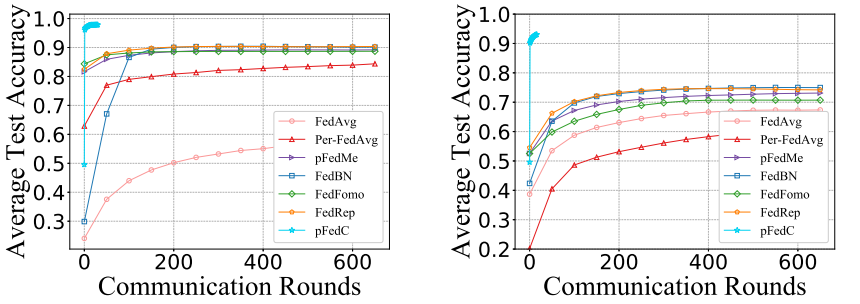
$$\mathbb{E}\|g_{i,t} - \bar{g}_{i,t}\|^2 \leq \sum_{n=1}^N \frac{{p_n}^2 {\pi_n^c}^2}{{\alpha_n^c}^2} \sigma_n^2 \tag{15}$$

Lemma 3. *(Bounding the divergence of $\theta_{n,t}$). Assume Assumption 4 holds, that η_t is non-increasing and $\eta_t \leq 2\eta_{t+E}$ for all $t \geq 0$. It follows that*

$$\mathbb{E}\left[\sum_{n=1}^N \frac{p_n \pi_n^c}{\epsilon^c} \|\bar{\theta}_{i,t} - \theta_{n,t}\|^2\right] \leq 4\eta_t^2 (E-1)^2 G^2. \tag{16}$$



(a) classes per client = 2 (b) $\alpha = 0.05$



(c) $\alpha = 0.1$ (d) $\alpha = 1$

Figure 8: Performance of pFedC comparede with baselines on CIFAR10 datatset. (a) Pathological non-IID setting; (b) - (d) Real-world non-IID setting.

7.3.3 Full Proof of Theorem 1

Proof. It is clear that no matter whether $t+1 \in \mathcal{I}_E$ or $t+1 \notin \mathcal{I}_E$, we always have the following equation: $\bar{\theta}_{i,t+1} = \bar{\mathrm{v}}_{i,t+1}$. Let $\Delta_t = \mathbb{E}\|\bar{\theta}_{i,t} - \theta_i^*\|^2$. From Lemma 1, Lemma 2 and Lemma 3, it follows that

$$\Delta_{t+1} \leq (1-\eta_t\mu)\Delta + \eta_t^2 B \tag{17}$$

where

$$B = \sum_{n=1}^{N} \frac{{p_n}^2 {\pi_n^c}^2}{{\epsilon^c}^2}\sigma_n^2 + 6L\Gamma + 8(E-1)^2G^2 \tag{18}$$

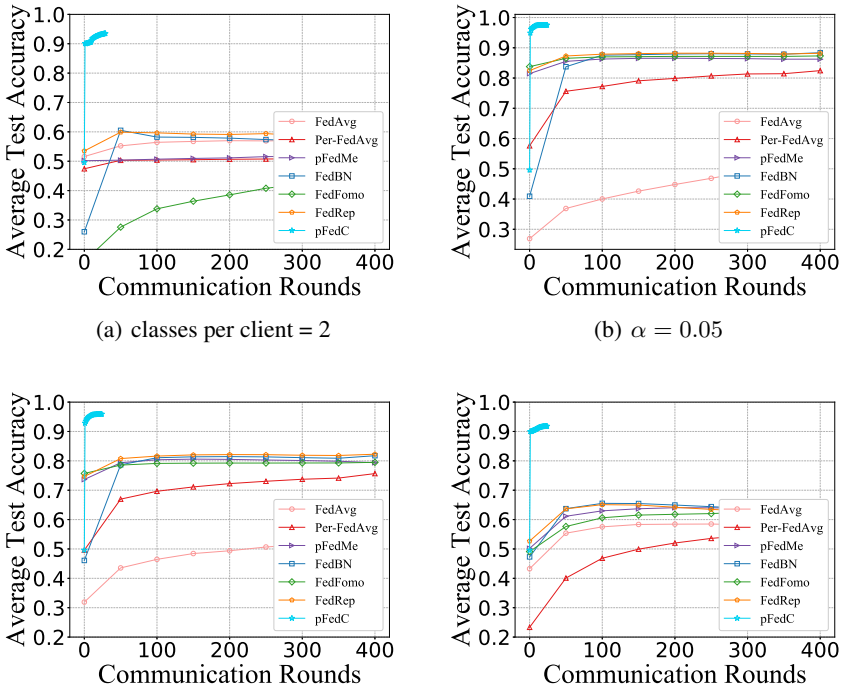
For a diminishing stepsize, $\eta_t = \frac{\beta}{t+\gamma}$ for some $\beta > \frac{1}{\mu}$ and $\gamma > 0$ such that $\eta_1 \leq \min\{\frac{1}{\mu}, \frac{1}{4L}\}$ and $\eta_t \leq 2\eta_{t+E}$. We prove that $\Delta_t \leq \frac{v}{\gamma+t}$ where $v = \max\{\frac{\beta^2 B}{\beta\mu - 1}, (\gamma+1)\Delta_1\}$.

First, the definition of v ensures that it holds for $t = 1$. Assume the conclusion holds for some t, it follows that

$$\begin{aligned}
\Delta_{t+1} &\leq (1-\eta_t\mu)\Delta_t + \eta_t^2 B \\
&\leq \left(1 - \frac{\beta\mu}{t+\gamma}\right)\frac{v}{t+\gamma} + \frac{\beta^2 B}{(t+\gamma)^2} \\
&= \frac{t+\gamma-1}{(t+\gamma)^2}v + \left[\frac{\beta^2 B}{(t+\gamma)^2} - \frac{\beta\mu - 1}{(t+\gamma)^2}v\right] \\
&\leq \frac{v}{t+\gamma+1}
\end{aligned} \tag{19}$$

Then, according to the L-smoothness of $\mathcal{L}_i(\cdot)$, we have

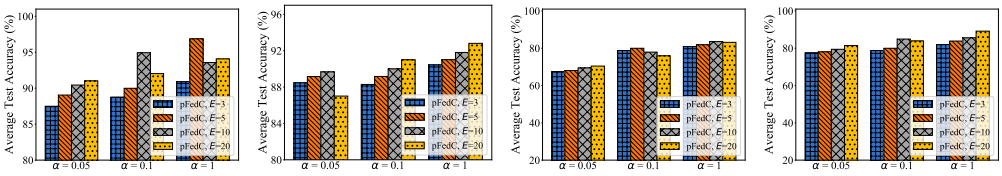
$$\mathbb{E}[\mathcal{L}_i(\bar{\theta}_{i,t})] - \mathcal{L}_i^* \leq \frac{L}{2}\Delta_t \leq \frac{L}{2}\frac{v}{\gamma+t}. \tag{20}$$



(a) classes per client = 2 (b) $\alpha = 0.05$

(c) $\alpha = 0.1$ (d) $\alpha = 1$

Figure 9: Performance of pFedC compared with baselines on CIFAR10 datatset. (a) Pathological non-IID setting; (b) - (d) Real-world non-IID setting.



(a) MNIST Dataset. (b) EMNIST Dataset. (c) CIFAR10 Dataset. (d) CIFAR100 Dataset.

Figure 10: Training Performance under different setting of local epochs, i.e., E.

Specifically, if we choose $\beta = \frac{2}{\mu}$, $\gamma = \max\{8\frac{L}{\mu} - 1, E\}$ and denote $\kappa = \frac{L}{\mu}$, then $\eta_t = \frac{2}{\mu}\frac{1}{\eta+t}$. One can verify that the choice of $\eta_t \leq 2\eta_{t+E}$ for $t \geq 1$. Then we have

$$v = \max\{\frac{\beta^2 B}{\beta\mu - 1}, (\gamma+1)\Delta_1\} \leq \frac{\beta^2 B}{\beta\mu - 1} + (\gamma+1)\Delta_1 \leq \frac{4B}{\mu^2} + (\gamma+1)\Delta_1, \tag{21}$$

and

$$\mathbb{E}[\mathcal{L}_i(\bar{\theta}_{i,t})] - \mathcal{L}_i^* \leq \frac{L}{2}\frac{v}{\gamma+t} \leq \frac{\kappa}{\gamma+t}\left(\frac{2B}{\mu} + \frac{\mu(\gamma+1)}{2}\Delta_1\right) \tag{22}$$

□

7.4 Analysis on Privacy Preserving

The proposed pFedC requires the exchange of local label distribution between the server and the clients. This property may bring privacy concerns to the user profile or other sensitive information. In this section, we integrate the homomorphic encryption technique with pFedC to further protect the local data distribution. Specifically, we propose an evolutionary version of pFedC, dubbed Privacy-pFedC, to achieve a privacy-preserving training framework. Instead of directly uploading shared feature extractor ϕ, multiple task-specific parameters ω_i and local label distributions Π_i to the server, the results of $\omega_i \cdot \Pi_i^*$ will be encrypted and sent to the server, where $\Pi_i^* = [1, \pi_i^1, \cdots, \pi_i^C]$. The

server will aggregate the local updates to obtain the encrypted personalized parameters and then distribute the updated results back to the clients. Then, the clients can decrypt the updated results to obtain personalized models and then continue with the next round of training. To this end, the privacy of the training framework can be guaranteed. Attackers from the server cannot reconstruct the raw data of clients from aggregated results or encrypted results without access to the private key on each client. The detailed workflow of Privacy-pFedC is illustrated in Algorithm 2.

Algorithm 2 Privacy-pFedC Algorithm

Require: $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N\}$, η, E, T, $\{\Pi_1^*, \Pi_1^*, \cdots, \Pi_N^*\}$, Encryption function $Enc(\cdot)$, Decryption function $Dec(\cdot)$.

Ensure: Trained personalized models $\{\theta_1, \theta_2, \ldots, \theta_N\}$.

Generate public key pk and secret keys sk
procedure SERVER EXECUTES
  for each communication round $t \in \{1, \ldots, T\}$ **do**
    Receive ϕ_i and $\{\widetilde{\omega}_i\}_{i=1}^N$ from the selected clients
    $\phi \leftarrow \sum_{i=1}^N \frac{D_i}{D} \phi_i$
    for each task c **do**
      $\widetilde{\omega}^c \leftarrow \sum_{n=1}^N \widetilde{\omega}_n^c$
    Distribute ϕ and $\{\widetilde{\omega}^c\}_{c=1}^C$ to selected clients
    for each client i **in parallel do**
      $\theta_i \leftarrow$ CLIENTUPDATE$(\phi, \{\widetilde{\omega}^c\}_{c=1}^C)$
      $\widetilde{\omega}_i \leftarrow Enc(\omega_i \cdot \Pi_i^*, pk)$
  return Trained personalized models $\{\theta_1, \ldots, \theta_N\}$
procedure CLIENTUPDATE$(\phi, \{\widetilde{\omega}^c\}_{c=1}^C)$
  $\{\omega^c\}_{c=1}^C \leftarrow Dec(\{\widetilde{\omega}^c\}_{c=1}^C, sk)$
  for each task c **do**
    if $\pi_i^c == 1$ **then**
      $\omega_i^c \leftarrow \omega^c$
    else
      $\omega_i^c \leftarrow \omega_i^c$
  $\theta_i \leftarrow \phi \circ \omega_i$
  for each local epoch $e \in \{1, \ldots, E\}$ **do**
    for mini-batch $\xi_t \subseteq \mathcal{D}_i$ **do**
      Update model parameters θ_i via: $\theta_i \leftarrow \theta_i - \eta \nabla_{\theta_i} \mathcal{L}_i(\theta_i)$
  return θ_i