FedSPD: A Soft-clustering Approach for Personalized Decentralized Federated Learning

I-Cheng Lin[1] Osman Yağan[1] Carlee Joe-Wong[1]

[1]Department of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

Abstract

Federated learning has recently gained popularity as a framework for distributed clients to collaboratively train a machine learning model using their local data. While traditional federated learning relies on a central server for model aggregation, recent advancements adopt a decentralized framework, enabling direct model exchange between clients and eliminating the single point of failure. However, existing decentralized frameworks often assume all clients train a shared model. Personalizing each client's model can enhance performance, especially with heterogeneous client data distributions. We propose **FedSPD**, an efficient personalized federated learning algorithm for the decentralized setting, and show that it learns accurate models in *low-connectivity* networks. To provide theoretical guarantees on convergence, we introduce a clustering-based framework that enables consensus on models for distinct data clusters while personalizing to unique mixtures of these clusters at different clients. This flexibility, allowing selective model updates based on data distribution, substantially reduces communication costs compared to prior work on personalized federated learning in decentralized settings. Experimental results on real-world datasets show that **FedSPD** outperforms multiple decentralized variants of existing personalized federated learning algorithms in scenarios with *low-connectivity* networks.

1 INTRODUCTION

Federated Learning (FL) is a popular approach for distributed clients to collaboratively learn from their local data. The most popular FL algorithm, **FedAvg** [McMahan et al., 2017], and most of its variants operate within a centralized federated learning (CFL) framework, where a central server coordinates the training process.[1] In CFL, each client in a training round independently trains a model on its local data and then sends the model parameters to a central server for aggregation, after which the aggregated model is broadcast back to the clients to begin a new training round. However, communication delays and bottlenecks often arise when a CFL system includes numerous mobile or IoT (Internet-of-Things) clients, hampering CFL's efficiency. Furthermore, this centralized structure poses risks of attacks and failures due to the single point of failure at the central server [Lalitha et al., 2018].

Decentralized Federated Learning (DFL) addresses these limitations by adopting a fully decentralized architecture where clients share their locally trained model parameters directly with neighboring clients, eliminating the need for a central server [Lalitha et al., 2018]. This approach can also reduce communication and computational costs [Beltrán et al., 2023]. However, most existing DFL methods focus on learning a single global model for all clients, aiming for consensus across clients. Such a global model may under-perform on clients with non-IID (independent and identically distributed) local data, as is commonly the case in federated learning. To address this challenge, we design an efficient **personalized, decentralized federated learning algorithm** that personalizes models to each client's data distribution without relying on a central server and preserves DFL's communication benefits by limiting the required communication between clients. We particularly focus on settings where clients are IoT devices using device-to-device communication protocols. Such settings often feature limited network connectivity, communication resources and computation resources, e.g., sensor-based environmental monitoring or vehicles learning personalized models of human driver preferences [Nakanoya et al., 2021].

[1]Note that clients in the CFL setting still train their models in a distributed manner; the term "centralized" simply refers to the presence of a central server managing the clients' interactions.

Personalization of a shared global model has shown to improve performance in CFL settings [Ruan and Joe-Wong, 2022, Marfoq et al., 2021]. However, extending such personalization methods to DFL poses significant **technical challenges**. DFL algorithms typically strive for consensus by sharing local models among neighboring clients, which represent only a subset of all clients. Ensuring that all clients can benefit from each other's updates despite limited communication is a key challenge [Beltrán et al., 2023]. In contrast, learning personalized models requires intentionally maintaining differences in clients' models, particularly for non-IID data. This makes it difficult to distinguish whether model disparities are due to communication issues or differences in local data distributions. We overcome this challenge by *quantifying similarities between client data* using a clustering-based method, allowing the training of distinct models for different data clusters, which are then personalized to each client's unique data mixture.

Prior works that seek to personalize models in DFL settings, including cluster-based methods, are typically straightforward extensions of personalization methods designed for CFL settings, which do not take into account the distinct communication patterns in DFL and thus perform poorly when the client network has poor connectivity. For example, a naïve clustering method assigns each client to a single cluster based on its data distribution [Ghosh et al., 2020]. However, such "hard" clustering assumes identical distributions within the same cluster, which is rarely the case. Instead, we adopt a **soft clustering** approach, as explored in CFL settings [Ruan and Joe-Wong, 2022, Marfoq et al., 2021], where each client's data is modeled as an unknown mixture of distributions, and a model is trained for each cluster in this mixture. Existing DFL soft clustering approaches require clients to train models for all clusters in every round [Marfoq et al., 2021], imposing **significant training and communication overhead** that scales linearly with the number of clusters. This is particularly problematic in DFL scenarios, where clients often have limited communication and computation capacity [Nguyen et al., 2021]. Therefore, we introduce a training algorithm that (i) learns each client's mixture coefficients, (ii) ensures consensus on models for each cluster, and (iii) unlike prior work, avoids communication resource requirements that scale with the number of clusters. Our **contributions** are as follows:

- We propose **FedSPD**, a novel FL algorithm for clients that utilizes soft clustering to train personalized models in a decentralized manner. **FedSPD** allows clients to reach a consensus on cluster-specific models and adapt their cluster mixture estimates over time, while requiring each client to train only **one** cluster model per training round, significantly reducing communication.
- We **prove the convergence of FedSPD** in Theorem 5.11. This proof adopts a different approach from prior work on soft clustering in DFL, which typically requires clients to train models for every cluster in each round [Marfoq et al., 2021].
- We demonstrate through experiments on real-world datasets that **FedSPD outperforms existing DFL algorithms** (both personalized and non-personalized). In some cases, **FedSPD** even approaches the accuracy of centralized algorithms. Furthermore, we show that **FedSPD** is **particularly effective in low-connectivity networks with computationally constrained clients**.

Following a review of related work in Section 2, we present our DFL model in Section 3 and introduce the **FedSPD** algorithm in Section 4. We then provide a convergence proof in Section 5 and demonstrate the algorithm's superior performance in Section 6, before concluding in Section 7.

2 RELATED WORK

Decentralized Federated Learning has its roots in decentralized optimization [Nedic and Ozdaglar, 2009, Wei and Ozdaglar, 2012, Zhang et al., 2021] and in particular decentralized Stochastic Gradient Descent (SGD) [Lian et al., 2017]. Several methods have been explored for decentralized optimization [Nedic and Ozdaglar, 2009, Wu et al., 2017, Lü et al., 2020], while the convergence analysis of decentralized SGD was first presented by Yuan et al. [2016] and Sirb and Ye [2018] with delayed information, highlighting decentralized SGD's advantages over centralized methods [Lian et al., 2017]. This literature establishes conditions on client connectivity such that all local models will converge to a consensus model [Lian et al., 2017]. The effects of client communication topologies in DFL [Lalitha et al., 2018, Warnat-Herresthal et al., 2021] have also been studied, and gradient tracking techniques based on push-sum algorithms have been proposed to relax the assumptions on client connectivity needed to show consensus [Nedić and Olshevsky, 2014, 2016, Assran et al., 2019].

Personalization in CFL is generally motivated by highly non-IID client data [McMahan et al., 2017, Collins et al., 2021], which can impede convergence and lead to a global model performing poorly at some clients, which may discourage them from participating in the FL process [Huang et al., 2020]. Common techniques include local finetuning [Sim et al., 2019], model interpolation [Mansour et al., 2020], meta-learning [Fallah et al., 2020], adding regularization terms [T Dinh et al., 2020], and multi-task learning [Smith et al., 2017, Yousefi et al., 2019, Li et al., 2021]. Clustered FL in particular includes hard clustering, which partitions clients into clusters based on their data's similarity [Ghosh et al., 2020] and its variations [Xie et al., 2021, Briggs et al., 2020, Duan et al., 2021, Mansour et al., 2020]. In soft clustered FL, one instead assumes that each client's data conforms to a mixture of distributions [Marfoq et al., 2021, Ruan and Joe-Wong, 2022, Wu et al., 2023]. Like these prior works, we use models learned for each cluster as guides for a personalized model; unlike them, we add a final personalization step to ensure good performance. We discuss this comparison in more detail in Section 4.

Some prior works have considered **combining personalization and DFL**. Jeong and Kountouris [2023] proposed a distillation-based algorithm, while Ma et al. [2022] proposed a communication-efficient algorithm with model pruning and neighbor selection. Sadiev et al. [2022] prove lower bounds on personalized DFL algorithms' convergence under specific objectives. Unlike these works, we provide theoretical convergence guarantees under more general learning objectives. Some centralized personalization algorithms also include decentralized versions, such as **FedEM** [Marfoq et al., 2021] and **IFCA** [Ghosh et al., 2020]. We experimentally show (Section 6) that **FedSPD** outperforms both **FedEM** and **IFCA**, particularly in low-connectivity settings. Moreover, we *only require each client to train one cluster model at a time*, which leads to significantly smaller computational and communication overhead than **FedEM**.

Comparison with FedSoft. FedSPD was inspired by **FedSoft** [Ruan and Joe-Wong, 2022]. However, the training methodology is significantly different. **FedSoft** uses a proximal objective and all client data to update its model in each round, while our **FedSPD** maintains separate models for each cluster and has each client update only one of these models, using only data associated with that cluster, in each round. Thus, **FedSPD** avoids bias in gradient updates, which may hamper consensus in decentralized settings. Our theoretical convergence analysis also relaxes the assumptions made by Ruan and Joe-Wong [2022] in analyzing **FedSoft**. We provide a more detailed comparison in Appendix C.

3 PROBLEM FORMULATION

We illustrate our **system model** in Figure 1 and summarize our notation in Table 1. We suppose there are N clients that are connected to each other via a graph with adjacency matrix $\mathbf{A}$ and use $\mathcal{N}_i$ to denote the set of client i's neighbors. Each client $i = 1, 2, \ldots, N$ has a fixed set $\mathcal{D}_i$ of training data. Clients with a shared edge can directly communicate with each other, e.g., to send model parameters.

Each data point $d \in \mathcal{D}_i$ on each client i is randomly sampled from one of S unique probability distributions (clusters) denoted as $P_1, P_2, \ldots P_S$, as illustrated in Figure 1. Consistent with standard clustering methods, we take S as a predetermined hyperparameter [Ruan and Joe-Wong, 2022]. Letting $\mathbf{x}$ denote the parameters of a machine learning model, we define the loss function $\ell(\mathbf{x}; \mathcal{D})$ as measuring the sum of the model losses with parameters $\mathbf{x}$ over all points d in a dataset $\mathcal{D}$. Cross-entropy loss, for example, is a typical loss function for classification problems. The *risk* of cluster s can then be written as: $F_s(\mathbf{x}) = \mathbb{E}_{\mathcal{D}\sim P_s}[\ell(\mathbf{x}; \mathcal{D})]$.

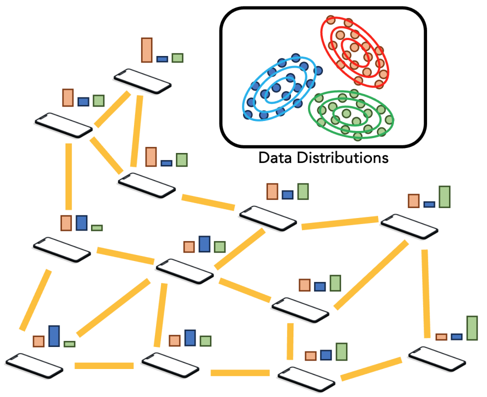


Figure 1: *Illustration of the mixture of data distribution at clients in DFL.*

For each client, the risk on a data point d_{is} belonging to cluster s is defined as: $f_{is}(\mathbf{x}, d_{is}) = \ell(\mathbf{x}, d_{is})$. Our goal is for the clients to collectively find the optimal (i.e., risk-minimizing) model parameters for each cluster, which we also call the *cluster centers* and can be written as: $\mathbf{c}_s^* = argmin_{\mathbf{x}} F_s(\mathbf{x})$, for $s = 1, 2, ..., S$. Given the cluster centers and mixture coefficients u_{is}, which represent the proportions of each cluster s in each client i's data, each client can find a personalized model for its local data mixture (Section 4). By focusing on common cluster centers, personalized learning can be reframed as achieving consensus on these centers, addressing a key challenge in personalized DFL. However, clients cannot directly determine the cluster centers using their local data $\mathcal{D}_i$ since it is a *mixture* of clusters, and they do not know which of their data comes from which cluster. In the next section, we present an algorithm for clients to estimate the cluster centers and use them to derive personalized models.

4 PROPOSED FEDSPD ALGORITHM

At each round $t = 1, 2, \ldots, T$, each client i maintains two types of parameters: (i) its estimate of the cluster center $\mathbf{c}_{is}^t$ for each cluster s, and (ii) the cluster to which each data point $d \in \mathcal{D}_i$ is associated, and the corresponding fraction of its data belonging to each cluster s, denoted by u_{is}^t. In each round t, clients update these parameters based on their local data and information received from their neighbors.

Each round of training consists of **four steps**: (1) local training, (2) parameter exchange, (3) parameter (i.e., cluster center) update, and (4) data clustering. Following the last training round, we conduct a **final personalization step**, which involves a local training update to each client's personalized model. Algorithm 1 formalizes this method.

Step 1: Local training (line 12 in Algorithm 1). In round t, each client i has an estimated portion u_{is}^t of its data coming from cluster s, where $\sum_{s=1}^{S} u_{is}^t = 1$. These values are

Name	Notation	Domain	Description
Number of Clients / Clusters	N, S	$N, S \in \mathbb{N}$	Total number of clients / clusters
Learning Rate	η_t	$\eta_t \in \mathbb{R}, 0 < \eta < 1$	Learning rate used in round t
Number of Local Updates	τ	$\tau \in \mathbb{N}$	Number of local updates in each training round
Client Neighbors	$\mathcal{N}_i$	$\mathcal{N}_i \in \mathcal{P}(N)$	Indices (in $\{1, 2, \ldots, N\}$) of client i's neighbors
Final Model Parameters	$\mathbf{x}_i$	$\mathbf{x}_i \in \mathbb{R}^{1\times X}$	Final personalized model parameters of client i
Final Concatenated Model Parameters	$\mathbf{X}$	$\mathbf{X} \in \mathbb{R}^{N\times X}$	Concatenated personalized model parameters
Final Phase Epochs	τ_{final}	$\tau_{final} \in \mathbb{N}$	Number of epochs for the final phase
Local Dataset	$\mathcal{D}_{is}^t$	$\mathcal{D}_{is}^t \subseteq \mathcal{D}_i$, client i's data	Data points at client i associated with cluster s in round t
Cluster Selection	s_i^t	$s_i^t \in \{1, 2, \ldots, S\}$	Index of cluster that client i trains in round t
Portion of Clusters	u_{is}^t	$u_{is}^t \in \mathbb{R}, 0 < u_{is} \leq 1$	Portion of data for client i of cluster s in round t
Concatenated Portions of Clusters	$\mathbf{U}(t)$	$\mathbf{U} \in \mathbb{R}^{N\times S}$	Concatenated portions of data of all clients in round t
Average Cluster Centers	$\bar{\mathbf{c}}_s^t$	$\bar{\mathbf{c}}_s^t \in \mathbb{R}^X$	Average center of cluster s over clients in round t
Concatenated Cluster Centers	$\mathbf{C}_s^t$	$\mathbf{C}_s^t \in \mathbb{R}^{N\times X}$	Concatenated centers of cluster s in round t
Collection of Cluster Centers	$\mathcal{C}(t)$	$\mathcal{C}(t) \in \mathbb{R}^{S\times N\times X}$	$\mathcal{C}(t) = \{\mathbf{C}_1^t, \mathbf{C}_2^t, ..., \mathbf{C}_S^t\}$
Weight Matrix	$\mathbf{W}_s^t$	$\mathbf{W}_s^t \in \mathbb{R}^{N\times N}$	Weight matrix of cluster s in round t
Augmented Adjacency Matrix	$\mathbf{A}$	$\mathbf{A} \in \mathbb{R}^{N\times N}$	Augmented adjacency matrix with diagonal elements equal to 1
Concatenated Gradients	$\mathbf{G}_s^t$	$\mathbf{G}_s^t \in \mathbb{R}^{N\times X}$	Concatenated gradients in round t for cluster s, $\mathbf{G}_s^t := [\nabla F_1, ..., \nabla F_N]$

Table 1: *Mathematical notations used in the paper.*

computed at the end of the previous round (step 4). Client i then selects cluster s to update with probability u_{is}^t, ensuring that clients contribute more to clusters where they have more data. By selecting only one cluster per round, ***FedSPD*** *keeps the training overhead independent of the number of clusters* S, as each client always trains a single cluster's model.

Once a cluster s is selected, the client performs τ SGD updates on its current cluster center estimate $\mathbf{c}_{is}^t$ using learning rate η. Gradients are computed on the risk of the data associated with the selected cluster, $\mathcal{D}_{i,s}^t$, as $\nabla_{\mathbf{c}}\ell(\mathbf{c}; d)$, where d is sampled uniformly at random from $\mathcal{D}_{i,s}^t$. The dataset $\mathcal{D}_{i,s}^t$ is formed in the previous round's clustering step, which assigns each data point $d \in \mathcal{D}_i$ to a cluster.

Step 2: Parameter exchange (line 18 in Algorithm 1). Let s_i^t be the cluster selected by client i in round t, meaning that $\mathbf{c}_{is_i^t}^t$ has been updated. Client i broadcasts s_i^t and $\mathbf{c}_{is_i^t}^t$ to its neighbors $j \in \mathcal{N}_i$. Consequently, each client i receives the communications $\{s_j^t, \mathbf{c}_{js_j^t}^t\}_{j\in\mathcal{N}_i}$ from all its neighbors.

Step 3: Cluster center updates

(line 23 in Algorithm 1). After receiving the updated cluster centers and indices from its neighbors, each client i updates the cluster center of the cluster s it selected to update during this round. The client uses the average of its received cluster centers to update its estimate of $\mathbf{c}_{is}$:

$$\mathbf{c}_{is}^{t+1} = \frac{1}{|j \in \mathcal{N}[i] \cap s_j^t = s|} \sum_{j\in\mathcal{N}[i]\cap s_j^t=s} \mathbf{c}_{js}^t \qquad (1)$$

Here, $\mathcal{N}[i]$ is the closed neighborhood, including client i and its neighboring clients, and $|j \in \mathcal{N}[i] \cap s_j^t = s|$ represents the number of clients j that both updated cluster s and belong to $\mathcal{N}[i]$. If no updates for cluster s are received in round t, i.e., none of the neighbors selected it, the estimated cluster center remains unchanged: $\mathbf{c}_{is}^{t+1} = \mathbf{c}_{is}^t$. This update rule can be expressed in matrix form as $\mathbf{C}_s^{t+1} = \mathbf{W}_s^t\mathbf{C}_s^t$, where $\mathbf{W}_s^t$ is the weight matrix for cluster s at time t, and $\mathbf{C}_s^t = [\mathbf{c}_{1s}^t, \ldots, \mathbf{c}_{Ns}^t]$ contains the concatenated cluster centers.

Step 4: Data clustering (line 29 in Algorithm 1). After updating the cluster centers, each client i associates its data points $d \in \mathcal{D}_i$ with a cluster. It calculates the loss $\ell(\mathbf{c}_{is}^{t+1}, d)$ for each cluster s and assigns data point d to the cluster with the lowest loss. Using these new associations, u_{is}^{t+1}, the fraction of data points linked to cluster s, is computed. This step enables **FedSPD** to adapt the mixture coefficients as cluster center estimates evolve. The process then moves to the next round, $t + 1$, starting again with local training.

Final Step: Personalization (line 37 in Algorithm 1). After T rounds, each client i computes a personalized model as a weighted sum of its cluster centers:

$$\mathbf{x}_i = \sum_{s=1}^{S} u_{i,s}^T \mathbf{c}_{i,s}^T \tag{2}$$

Marfoq et al. [2021] show that Eq. (2) provides the optimal personalized model for client i when the loss function ℓ is convex. However, since most practical loss functions, such as cross-entropy for neural networks, are not convex, this aggregated model may perform poorly in practice. Thus, each client runs a few additional local training iterations, starting from $\mathbf{x}_i$ (Eq. (2)), using its entire local dataset $\mathcal{D}_i$.

Comparison to prior soft clustering algorithms. Marfoq et al. [2021] and Ruan and Joe-Wong [2022] use soft clustering to learn cluster centers and personalized models without this final personalization step, directly learning personalized models in each iteration, with a central server estimating the cluster centers. In DFL, achieving consensus on cluster models is difficult due to the extensive parameter exchanges needed for model propagation, particularly when clients have few neighbors. Marfoq et al. [2021] propose a decentralized algorithm that sets the personalized model as a weighted sum of the cluster centers *at each round's end*, which can be sub-optimal for non-convex loss functions. Such a framework can lead to overfitting in DFL, as clients have low connectivity and thus cannot rely on receiving many other clients' updates in each training round. Adding another final personalization step, as we use in **FedSPD**, may exacerbate this overfitting, as cluster center gradients already incorporate personalized models. In Section 6, we demonstrate that ***FedSPD** outperforms Marfoq et al. [2021]'s **FedEM** algorithm*, which also requires each client to train all models per round, incurring significantly more computation and communication than **FedSPD**.

5 CONVERGENCE ANALYSIS

We prove that **FedSPD** converges in Theorem 5.11. We first outline our technical assumptions and then present our main results. All proof details can be found in Appendix A.

Assumptions. Our analysis relies on the following assumptions on the risk function and gradient estimates, which are common in the literature [Marfoq et al., 2021, Ghosh et al., 2020, Koloskova et al., 2020] and weaker than those of Ruan and Joe-Wong [2022].

Assumption 5.1 *(Strong convexity and smoothness) The risk function F_s for each cluster s is L-smooth and μ-strongly convex. That is, for some $L > 0$ and $\mu \geq 0$:*

$$\begin{aligned} &\|\nabla F_s(\mathbf{x}) - \nabla F_s(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|; \\ &\nabla F_s(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|^2 \leq F_s(\mathbf{y}) - F_s(\mathbf{x}) \end{aligned} \tag{3}$$

Algorithm 1 Our Proposed **FedSPD** Algorithms

```
 1: procedure FEDSPD($\eta$, $\tau$, $S$, $T$, $\mathbf{W}_s^t$)
 2:     for $t = 1, 2, ..., T\tau$ do
 3:         LOCALUPDATE($\mathcal{C}(t)$)
 4:         if $t \bmod \tau = 0$ then
 5:             PARAMETEREXCHANGE($\mathcal{C}(t)$, $\mathbf{A}$)
 6:             PARAMETERUPDATE($\mathcal{C}(t)$, $\mathbf{A}$)
 7:             DATACLUSTERING($\mathcal{C}(t)$, $\mathbf{A}$)
 8:         end if
 9:     end for
10:     FINALPHASE($\mathcal{C}(t)$, $\mathbf{u}(t)$)
11: end procedure
12: procedure LOCALUPDATE($\mathcal{C}(t)$)
13:     for $i = 1, 2, ..., N$ do
14:         Client $i$ selects cluster $s_i^t$ to update
15:         $\mathbf{c}_{s_i^t}^{t+1} = \mathbf{c}_{s_i^t}^t - \eta_t \nabla f_{is}(\mathbf{c}_{s_i}^t)$
16:     end for
17: end procedure
18: procedure PARAMETEREXCHANGE($\mathcal{C}(t)$, $\mathbf{A}$)
19:     for $i = 1, 2, ..., N$ do
20:         For each client $i$, exchange the updated parame-
    ter $\mathbf{c}_{is}$ and the selected cluster $s$ with client $j \in \mathcal{N}_i$
21:     end for
22: end procedure
23: procedure PARAMETERUPDATE($\mathcal{C}(t)$, $\mathbf{A}$)
24:     Construct $\mathbf{W}_s^t$ for each cluster $s$. If client $i$ is not
    selected to update cluster $s$, the row $i$ and column $i$ will
    only have diagonal element equal to 1, else equal to 0 ,
    meaning the model parameter of the cluster that a user
    has not selected to update will remain the same as it
    was in the previous epoch.
25:     for $s = 1, 2, ..., S$ do
26:         $\mathbf{C}_s^{t+1} = \mathbf{W}_s^t \mathbf{C}_s^{t+1}$
27:     end for
28: end procedure
29: procedure DATACLUSTERING($\mathcal{C}(t)$, $\mathbf{A}$)
30:     for $i = 1, 2, ..., N$ do
31:         for $d_k \in \mathcal{D}_i$ do
32:             Label data $d_k$ with the least loss of all the
    model parameters among all clusters.
33:         end for
34:         For $s = 1, ..., S$ update $u_{i,s}^t$ for client $i$
35:     end for
36: end procedure
37: procedure FINALPHASE($\mathcal{C}(t)$, $u^t$)
38:     for $i = 1, 2, ..., N$ do
39:         $\mathbf{X}_i = \sum_{s=1}^{S} u_{i,s}^t \mathbf{C}_s^t(i, :)$
40:     end for
41:     for $t = 1, 2, ..., \tau_{final}$ do
42:         LOCALUPDATE($\mathbf{X}$)      ▷ Run gradient descent
    using all data of the client for the aggregated training.
43:     end for
44: end procedure
```

Assumption 5.2 *(Bounded risk function) The risk function F_s for each cluster s is lower-bounded by some $F_{inf} > 0$, i.e., $F_s(\mathbf{x}) \geq F_{inf}$.*

Assumption 5.3 *(Unbiased gradient estimation) The gradient is unbiased, i.e., $\mathbb{E}[\nabla f_{is}(\mathbf{x})] = \nabla F_s(\mathbf{x})$.*

Assumption 5.4 *(Bounded gradient) We have $\mathbb{E}\|\nabla f_{is}(\mathbf{x})\|^2 \leq \sigma^2$ for some $\sigma^2 > 0$.*

Assumption 5.5 *(Bounded variance of gradient estimation) The variance of the estimated gradient is bounded:*

$$\mathbb{E}\|\nabla f_{is}(\mathbf{x}) - \nabla F_s(\mathbf{x})\|^2 \leq v^2, \text{for some } v^2 > 0. \quad (4)$$

Assumption 5.6 *(Bounded cluster error) Following Ruan and Joe-Wong [2022], Ghosh et al. [2020], during all training steps t, all estimated cluster centers have bounded distance to the optimal centers. That is, for some $\delta > 0$:*

$$\|\mathbf{c}_{is}^t - \mathbf{c}_s^*\| \leq (0.5 - \alpha_0)\sqrt{\frac{\mu}{L}}\delta, \forall s \in 1, 2, ..., S \quad (5)$$

where $0 < \alpha_0 \leq 0.5$.

Note that this assumption will always hold for some value of δ; however, a larger δ, and thus larger cluster error, will also lead to slower convergence.

We finally follow Koloskova et al. [2020] in assuming that clients communicate sufficiently for consensus:

Assumption 5.7 *(Expected consensus rate) Define $\mathbf{C_s}$ as the concatenated model parameter matrix of cluster s. Then for some constant $p \in (0, 1]$ and integer $\beta \geq 1$, for all non-negative integers $l \leq \frac{T}{\beta}$ we have:*

$$\mathbb{E}\left\|\mathbf{C}_s \prod_{t=l\beta}^{(l+1)\beta-1} \mathbf{W}_s^t - \overline{\mathbf{C}_s}\right\|_F^2 \leq (1-p)\|\mathbf{C}_s - \overline{\mathbf{C}_s}\|_F^2 \quad (6)$$

where $\bar{\mathbf{C}}_s := \underbrace{[\bar{\mathbf{c}}_s, ..., \bar{\mathbf{c}}_s]}_{\text{total } N \text{ terms.}}$ is the matrix with every column equal to the average of the model parameters.

For simplicity, we further assume that all clients have the same amount of data (i.e., $\mathcal{D}_i$ has the same number of data points for all clients i) and that the number of local updates $\tau = 1$ in the remainder of this section. These can be easily relaxed if needed.

Results. Without loss of generality, we present our results for a specific cluster i, where $i = 1, \ldots, S$. Since the convergence proof is identical for each of the S clusters, we omit the cluster index for clarity. Let n be the number of clients chosen to update the selected cluster. If the total data across clients is roughly uniform for each cluster, then $n \approx \frac{N}{S}$. We begin by bounding the distance of the average cluster center to its optimal center:

Theorem 5.8 *(Descent lemma) The distance $\mathbb{E}\left\|\overline{\mathbf{c}}^{(t+1)} - \mathbf{c}^\star\right\|^2$ between the average cluster center and its optimum $\mathbf{c}^\star$ satisfies the bound (7) with proper choice of learning rate η_t:*

$$\begin{aligned}
&\leq \frac{\eta_t(L+\mu)}{n}\sum_{i=1}^{n_1}\left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}_i^{(t)}\right\|^2 + \frac{18L^2\epsilon_N^2\eta_t^2}{n^2} + v^2\eta_t^2 \\
&+ \left(1 - \eta_t\mu + \frac{\eta_t\mu\epsilon_N}{n}\right)\left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star\right\|^2 + \frac{2\epsilon_N(S-1)v^2\eta_t^2}{n^2} \\
&+ \left(\frac{4\eta_t^2(n-\epsilon_N)^2L}{n^2} + 2\eta_t\frac{1-\epsilon_N}{n}\right)\left(f\left(\overline{\mathbf{c}}^{(t)}\right) - f\left(\mathbf{c}^\star\right)\right)
\end{aligned} \quad (7)$$

Here ϵ_N is the bound of the expected number of clients using the wrong data in Lemma A.2.

We then derive an expression for the cluster centers estimated by individual clients.

Theorem 5.9 *(Update rule) Clients' estimated centers of the cluster after time t can be written as:*

$$\mathbf{C}^t = \mathbf{C}^{l\beta}\prod_{m=l\beta}^{t-1}\mathbf{W}^m - \sum_{m=l\beta}^{t-1}\left(\eta_t\mathbf{G}^m\prod_{r=t-1}^{m}\mathbf{W}^r\right) \quad (8)$$

Here $l \in \mathbb{N}$ and β is the constant in Assumption 5.7. Given this expression, we can relate the clients' cluster center estimates to their average, showing that they eventually reach a near-consensus:

Theorem 5.10 *(Consensus distance) Define $\mathbf{E}_t = \frac{1}{N}\sum_{i=1}^N \mathbb{E}\|\mathbf{c}_i^{(t)} - \overline{\mathbf{c}}^{(t)}\|^2$, the expected squared distance of the model parameters of client i to the average model parameter. It is upper-bounded by*

$$\left(1 - \frac{p}{2}\right)\mathbf{E}_{m\beta} + \sum_{j=m\beta}^{t-1}\left(\frac{p\mathbf{E}_j}{16\beta} + \frac{18\beta n\sigma^2 + nv^2p}{Np}\eta_j^2 + \left(\frac{36Ln\beta\eta_t^2}{pN}\right)\left(f(\overline{\mathbf{c}^j}) - f(\mathbf{c}^\star)\right)\right)$$

Here p is the constant defined in Assumption 5.7.

Theorem 5.11 *(Cluster convergence rate) For given target accuracy ϵ, there exists a constant learning rate for which ϵ accuracy can be reached after T iterations:*

$$\begin{aligned}
&\left[1 + \left(\frac{n-\epsilon_N}{n}\right)\eta L\right]\sum_{t=0}^{T}\frac{r_t}{R_T}\left(\mathbb{E}f(\overline{\mathbf{c}}^t) - f(\mathbf{c}^\star)\right) \\
&\qquad + \mu\mathbb{E}\|\overline{\mathbf{c}}^{(T+1)} - \mathbf{c}^\star\|^2 \leq \epsilon
\end{aligned} \quad (9)$$

Here r_t is a sequence of positive weights defined in Lemma A.3 in Appendix A.4 and $R_T = \sum_{t=1}^{T} r_t$. Rearranging, we find that the number of required iterations T to reach an error ϵ is of the order:

$$\tilde{\mathcal{O}}\Bigg(\frac{\sqrt{L+\mu}}{\sqrt{\epsilon}\mu}\Big(\sigma + \frac{\sqrt{n}}{\sqrt{N}}v\Big) + \frac{L\beta n^{\frac{3}{2}}}{\mu p\sqrt{N}(n-\epsilon_N)} \ln\Big(\frac{1}{\epsilon}\Big) + v^2\frac{n^2 + L^2\epsilon_N^2 + \epsilon_N(S-1)}{\mu n^2 \epsilon}\Bigg).$$

The convergence rate, asymptotically requiring $O(1/\sqrt{\epsilon})$ training rounds to reach an error ϵ, aligns with previous works on DFL without personalization [Koloskova et al., 2020], leading us to conjecture that **FedSPD** will converge well. We note that the network connectivity appears in this bound through the constant $p \in (0, 1]$ (Assumption 5.7), where higher connectivity indicates a larger p. However, the second term in the convergence rate that involves p is not the dominant term. Thus, as long as the network is connected, we expect that the effect of network connectivity on convergence will be relatively minor. Our simulation results in Section 6.2 support this observation.

6 SIMULATION RESULTS

In this section, we evaluate the performance of our proposed **FedSPD** algorithm and compare it with existing methods. We also analyze how different network connectivity and topology influence **FedSPD**'s performance.

Datasets and models. Unless specified, we use $N = 100$ clients for all experiments on hand-written character recognition (MNIST and EMNIST datasets [Cohen et al., 2017]) and $N = 25$ clients for all experiments on image classification (CIFAR-10 and CIFAR-100 datasets [Krizhevsky et al., 2009]). We use a CNN (convolutional neural network) model for each client, following the settings of Ruan and Joe-Wong [2022] with data from a mixture of $S = 2$ distributions, $\mathcal{D}_A$ and $\mathcal{D}_B$. Additional results using MobileNet-v2 are also included in Appendix B.2.7. Each client randomly draws 10% to 90% of its data from $\mathcal{D}_A$and the remainder from $\mathcal{D}_B$ with unbalanced class [Marfoq et al., 2021], image rotation [Ruan and Joe-Wong, 2022], or both. We thus create a portion of clients with significantly unbalanced data and guarantees the unique distribution of each client. We follow Ruan and Joe-Wong [2022] and Marfoq et al. [2021] for other parameter settings. Details are described in the Appendix B. The test accuracy is evaluated on each client's local test dataset, which is unseen during training.

Client communications. Unless specified, the client graph is a connected Erdős–Rényi (ER) random graph [Erdos et al., 1960] with an average degree from 5 to 12; more details are in Appendix B. To avoid the label switching problem [Stephens, 2000], we calculate the cosine similarities of the model parameters received from other clients to ensure the consensus of the cluster.

Baselines. We compare **FedSPD** with: (i) centralized and decentralized **FedAvg** [McMahan et al., 2017]; (ii) centralized and decentralized **FedEM** [Marfoq et al., 2021], a prior soft clustering method; (iii) centralized and decentralized versions of **FedSoft** [Ruan and Joe-Wong, 2022], which also uses soft clustering; (iv) centralized and decentralized **IFCA** [Ghosh et al., 2020] using hard clustering; (v) centralized and decentralized **pFedMe** [T Dinh et al., 2020], another state-of-the-art FL personalization approach without clustering; and (vi) **local** training on local dataset only.

Additional results will be included in the appendix due to page limit. These include:

- We conduct ablation studies to analyze the impact of different factors on the performance of **FedSPD**, specifically evaluating:
  - The influence of the number of local training epochs in Section B.2.1;
  - The contribution of the final training phase in Section B.2.2;
  - The impact of the number of clusters in Section B.2.3; and
  - More details of the effect of network connectivity in Section B.2.4. We also show how the dynamic network topology influences the performance of **FedSPD**.
- In Section B.2.5, we evaluate the performance of **FedSPD** under a more challenging setting where the total amount of data is imbalanced across clients in addition to the data heterogeneity across clusters.
- To explore the potential for enhancing privacy guarantees in DFL, we incorporate Differential Privacy into **FedSPD** and present the results in Section B.2.6.
- For most experiments on the CIFAR-10/CIFAR-100 datasets, we adopt the same CNN model used by Ruan and Joe-Wong [2022] to ensure a fair comparison. To further assess the scalability of **FedSPD**, we evaluate its performance using a more complex architecture, MobileNet-V2, in Section B.2.7.

6.1 COMPARISON WITH BASELINES

We first compare our method with other decentralized personalized methods. Our results on EMNIST, CIFAR-10, and CIFAR-100 are shown in Table 2 and Table 3. **FedSPD** outperforms other DFL methods in most cases, approaching the accuracy of CFL. The centralized methods still outperform decentralized methods, as expected from prior literature [Sun et al., 2023]. However, decentralized methods offer advantages such as lower communication traffic and increased

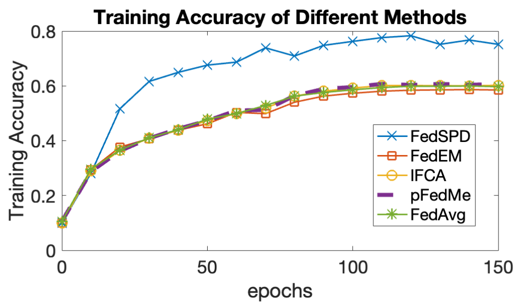


Figure 2: *Training accuracy of different DFL methods versus number of epochs on CIFAR-10 ($N = 25$).* ***FedSPD*** *converges faster in terms of training accuracy compared to all other DFL methods.*

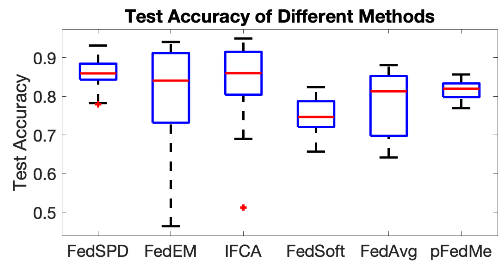


Figure 3: *Box-plot for accuracy across clients on EMNIST dataset.* ***FedSPD*** *has much lower variance in test accuracy across clients.*

robustness, as they do not rely on a single point of failure like a centralized server.

Figure 2 shows the training accuracy versus number of epochs on the CIFAR-10 dataset. **FedSPD** *converges faster* than all other DFL algorithms in terms of training accuracy. This shows that each of the clusters in **FedSPD** does converge as desired. Note that compared to **FedEM**, another soft clustering method, our **FedSPD** needs half the communication cost, since **FedEM** clients exchange the information of all $S = 2$ clusters.

To guarantee the **fairness across clients**, we show the box plot of the final test accuracy across different clients on EMNIST in Figure 3. **FedSPD** has much *less variance in accuracy* across different clients compared to all baselines except **pFedMe**, validating that its improvement in average accuracy does not come from high accuracy in a few clients.

6.2 EFFECTS OF NETWORK CONNECTIVITY

In this section, we investigate how the performance varies with the connectivity of the client network. Figure 4 shows the test accuracy of different DFL methods under varying client connectivity on the CIFAR-100 dataset using the ER Random Graph, averaged over three experimental runs. **FedSPD** consistently shows the highest test accuracies, though

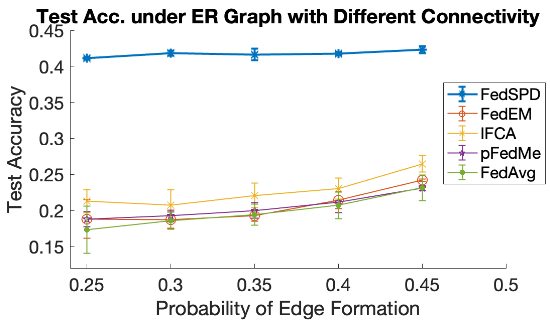


Figure 4: *Test accuracy of different methods under different connectivity levels of an ER Random Graph on the CIFAR-100 dataset ($N = 15$).* ***FedSPD*** *shows consistently high test accuracies compare to other DFL methods.*

other methods' performance begins to increase as the graph becomes more connected (a higher probability of link formation).

Tables 4 and 5 show the test accuracy of **FedSPD** in different types of networks and connectivity levels. We use three different network topologies: the ER Random Graph; the Barabasi-Albert (BA) Model [Albert and Barabási, 2002] with preferential attachment representing the network following the power law; and the Random Geometrical Graph (RGG) [Penrose, 2003], which is often used in wireless communication and IoT scenarios that have high clustering effects. We observe that the final test accuracy does not vary significantly across different network topologies and levels of connectivity in MNIST. In EMNIST, the test accuracy slightly increases when the average degree increases. The test accuracy is more stable in RGG under different connectivity, which we conjecture is due to RGG's highly clustered nature. Thus, as long as the network is connected, **FedSPD** performs well in both high and low connectivity scenarios and across various types of networks. As we expect from Theorem 5.11, **FedSPD** converges regardless of the network topology.

6.3 COMMUNICATION OVERHEAD

FedSPD requires transmitting 50% less data compared to **FedEM** (in the case $S = 2$) since only a single model is transmitted by each client. As the number of clusters S increases, our communication volume advantage grows. Compared to the decentralized versions of **FedAvg** and **FedSoft**, **FedSPD** requires each client to send the same volume of data (equivalent to one model's parameters) in each round. However, since **FedSPD** only requires clients to send their local models to their neighbors training models for the same cluster, the number of clients to which each client communicates in **FedSPD** is smaller than in algorithms like **FedAvg** and **FedSoft**, which requires each client to send its local model to *all* of its neighbors. **FedSPD**, **FedAvg** and **FedSoft**

	DFL	CFL				
Dataset	**FedSPD**	FedEM	IFCA	FedAvg	FedSoft	pFedMe
EMNIST	83.07	88.83	89.42	88.81	84.97	90.95
CIFAR-10	68.72	79.64	79.52	79.36	76.62	79.43
CIFAR-100	40.38	44.25	43.91	43.11	39.76	8.74[2]

Table 2: ***FedSPD*** *has comparable test accuracy to CFL algorithms. Accuracy in percentage (%)*

	DFL						
Dataset	**FedSPD**	FedEM	IFCA	FedAvg	FedSoft	pFedMe	Local
EMNIST	83.07	80.47	**83.88**	78.61	74.30	81.16	56.91
CIFAR-10	**68.72**	50.45	52.39	49.21	42.38	49.48	41.82
CIFAR-100	**40.38**	18.59	17.18	17.20	13.17	18.27	13.31

Table 3: ***FedSPD*** *achieves higher test accuracy than other DFL algorithms in most cases. Accuracy in percentage (%)*

thus have comparable communication overhead if clients use multicast communication, but if they use point-to-point communication, **FedSPD** will require less communication than **FedAvg** and **FedSoft** with full participation, due to having fewer recipients per client.

6.4 DISCUSSION

As shown in Table 2 and Table 3, local learning performs the worst among all algorithms, validating that all other methods benefit from exchanging information between clients to learn a better model. Among the DFL algorithms, **FedAvg**, the only one without personalization, typically performs the worst, indicating that personalization is beneficial in non-iid data distributions, as we would intuitively expect. However, an exception is observed with the **FedSoft** algorithm. In the CIFAR-10 and CIFAR-100 datasets, **FedSoft** performs poorly, nearing the accuracy of local training. We conjecture that this is due to the way **FedSoft** aggregates models, making it difficult to learn the correct cluster centers in a low-connectivity network, leading to suboptimal performance. Our **FedSPD** designs a new model update method to avoid such an issue. More detailed discussion comparing **FedSPD** and **FedSoft** can be found in Section C.

Avg. Degree	6	8	10	12	14
ER	92.86	92.93	93.37	93.31	93.26
BA	93.06	92.58	92.56	92.87	93.17
RGG	92.86	92.61	92.84	93.49	92.97

Table 4: ***FedSPD*** *shows consistently high test accuracies on MNIST data for* $N = 50$ *clients across different client network topologies.*

Avg. Degree	8	12	16	20
ER	79.79	82.26	84.28	84.49
BA	79.45	82.13	84.58	84.73
RGG	82.26	83.49	84.06	84.08

Table 5: ***FedSPD*** *shows consistently high test accuracies on EMNIST data for* $N = 50$ *clients across different client network topologies.*

7 CONCLUSION

We propose **FedSPD**, a soft clustering approach that enables federated training of personalized models in a decentralized setting. **FedSPD** models each FL client's data as a mixture of cluster distributions and aims to learn a distinct model for each cluster. In the final phase, all models are aggregated and further personalized for each client. Importantly, **FedSPD** requires each client to train only one cluster model per training round, ensuring scalability with the number of clusters, and works well when communication resource is limited. We theoretically demonstrate that **FedSPD** can achieve consensus within each cluster. Our experiments on real-world datasets show that **FedSPD** outperforms previous algorithms for personalized, decentralized FL and performs well even in *low-connectivity* networks. For future extensions, this work can serve as a foundation for various applications, such as environmental monitoring in IoT, object identification in mixed reality, or autonomous driving, all of which benefit from the low latency of direct communication and collaborative learning across adjacent devices with similar data.

Acknowledgements

This work was supported in part by the National Science Foundation (NSF) under Grants CNS-2312761 and CNS-1751075, and by the Office of Naval Research (ONR) under Grant N00014-23-1-2275.

[2]The centralized pFedMe on CIFAR-100 does not converge in the various settings of hyperparameters we tried.

References

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74 (1):47, 2002.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*, 2023.

Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

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

Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. Fedgroup: Efficient federated learning via decomposed similarity-based clustering. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 228–237. IEEE, 2021.

Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

Tiansheng Huang, Weiwei Lin, Wentai Wu, Ligang He, Keqin Li, and Albert Y Zomaya. An efficiency-boosting client selection scheme for federated learning with fairness guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1552–1564, 2020.

Eunjeong Jeong and Marios Kountouris. Personalized decentralized federated learning with knowledge distillation, 2023.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Anusha Lalitha, Shubhanshu Shekhar, Tara Javidi, and Farinaz Koushanfar. Fully decentralized federated learning. In *Third workshop on bayesian deep learning (NeurIPS)*, volume 2, 2018.

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

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.

Qingguo Lü, Xiaofeng Liao, Huaqing Li, and Tingwen Huang. A computation-efficient decentralized algorithm for composite constrained optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 6:774–789, 2020.

Zhenguo Ma, Yang Xu, Hongli Xu, Jianchun Liu, and Yinxing Xue. Like attracts like: Personalized federated learning in decentralized edge computing. *IEEE Transactions on Mobile Computing*, pages 1–17, 2022. doi: 10.1109/TMC.2022.3230712.

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 Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Manabu Nakanoya, Junha Im, Hang Qiu, Sachin Katti, Marco Pavone, and Sandeep Chinchali. Personalized federated learning of driver prediction models for autonomous driving. *arXiv preprint arXiv:2112.00956*, 2021.

Angelia Nedić and Alex Olshevsky. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2014.

Angelia Nedić and Alex Olshevsky. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE Transactions on Automatic Control*, 61(12):3936–3947, 2016.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.

Mathew Penrose. *Random geometric graphs*, volume 5. OUP Oxford, 2003.

Yichen Ruan and Carlee Joe-Wong. Fedsoft: Soft clustered federated learning with proximal local updating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8124–8131, 2022.

Abdurakhmon Sadiev, Ekaterina Borodich, Aleksandr Beznosikov, Darina Dvinskikh, Saveliy Chezhegov, Rachael Tappenden, Martin Takáč, and Alexander Gasnikov. Decentralized personalized federated learning: Lower bounds and optimal algorithm for all personalization modes. *EURO Journal on Computational Optimization*, 10:100041, 2022.

Khe Chai Sim, Françoise Beaufays, Arnaud Benard, Dhruv Guliani, Andreas Kabel, Nikhil Khare, Tamar Lucassen, Petr Zadrazil, Harry Zhang, Leif Johnson, et al. Personalization of end-to-end speech recognition on mobile devices for named entities. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 23–30. IEEE, 2019.

Benjamin Sirb and Xiaojing Ye. Decentralized consensus algorithm with delayed and stochastic gradients. *SIAM Journal on Optimization*, 28(2):1232–1254, 2018.

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

Matthew Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, 2000.

Yan Sun, Li Shen, and Dacheng Tao. Which mode is better for federated learning? centralized or decentralized. *arXiv preprint arXiv:2310.03461*, 2023.

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

Stefanie Warnat-Herresthal, Hartmut Schultze, Krishnaprasad Lingadahalli Shastry, Sathyanarayanan Manamohan, Saikat Mukherjee, Vishesh Garg, Ravi Sarveswara, Kristian Händler, Peter Pickkers, N Ahmad Aziz, et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature*, 594(7862): 265–270, 2021.

Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 5445–5450. IEEE, 2012.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS 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.

Tianyu Wu, Kun Yuan, Qing Ling, Wotao Yin, and Ali H Sayed. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 4(2):293–307, 2017.

Yue Wu, Shuaicheng Zhang, Wenchao Yu, Yanchi Liu, Quanquan Gu, Dawei Zhou, Haifeng Chen, and Wei Cheng. Personalized federated learning under mixture of distributions. In *International Conference on Machine Learning*, pages 37860–37879. PMLR, 2023.

Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning. *arXiv preprint arXiv:2108.08647*, 2021.

Fariba Yousefi, Michael T Smith, and Mauricio Alvarez. Multi-task learning for aggregated data using gaussian processes. *Advances in Neural Information Processing Systems*, 32, 2019.

Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Jiaojiao Zhang, Qing Ling, and Anthony Man-Cho So. A newton tracking algorithm with exact linear convergence for decentralized consensus optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 7:346–358, 2021.

Supplementary Material

I-Cheng Lin[1] **Osman Yağan**[1] **Carlee Joe-Wong**[1]

[1]Department of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

A PROOF OF THE THEOREMS

A.1 PROOF OF THEOREM 5.8

Without loss of generality, we select a single cluster, cluster 1 for analysis; the same analysis applies to the other $S-1$ clusters. For readability, we eliminate the subscription indicating the cluster number 1. Consider each client running single step of SGD, we use n to indicate the number of clients selected to update this cluster and n_1 and n_0 to indicate the number of clients using the correct data and incorrect data, respectively (i.e. the data is drawn from this selected cluster is consider a correct data.), so that $n_1 + n_0 = n$. S indicates the set of the client selected to update this cluster, $S*$ indicates the set of clients using the correct data, and $\overline{S*}$ indicates the set of clients using the incorrect data.

Lemma A.1 *(Doubly-stochastic weight matrix preserves the average) At the communication step, if the model of each client in the network is updated according to a doubly-stochastic weight matrix* $\mathbf{W}^t$ *then the average after the communication step remains the same. Formally, we have:*

$$\mathbf{C}^{t+1}\frac{\mathbf{1}\mathbf{1}^T}{N} = \mathbf{C}^t\mathbf{W}^t\frac{\mathbf{1}\mathbf{1}^T}{N} = \mathbf{C}^t\frac{\mathbf{1}\mathbf{1}^T}{N} \tag{10}$$

From Lemma A.1, we can write the left-hand side of Theorem 5.8 as:

$$\begin{aligned}
\left\|\overline{\mathbf{c}}^{(t+1)} - \mathbf{c}^\star\right\|^2 &= \left\|\overline{\mathbf{c}}^{(t)} - \frac{\eta_t}{n}\sum_{i=1}^{n}\nabla F_i\left(\mathbf{c}_i^{(t)}, D_i^{(t)}\right) - \mathbf{c}^\star\right\|^2 \\
&= \left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star - \frac{\eta_t}{n}\sum_{i\in S\cap S*}\nabla F_i\left(\mathbf{c}_i^{(t)}\right) - \frac{\eta_t}{n}\sum_{i\in S\cap\overline{S*}}\nabla F_i\left(\mathbf{c}_i^{(t)}\right)\right\|^2 \\
&= \left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star - \frac{\eta_t}{n}\sum_{i\in S\cap S*}\nabla F_i\left(\mathbf{c}_i^{(t)}\right)\right\|^2 + \left\|\frac{\eta_t}{n}\sum_{i\in S\cap\overline{S*}}\nabla F_i\left(\mathbf{c}_i^{(t)}\right)\right\|^2 \\
&\quad - \frac{2\eta_t}{n}\left\langle\overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star - \frac{\eta_t}{n}\sum_{i\in S\cap S*}\nabla F_i\left(\mathbf{c}_i^{(t)}\right), \sum_{i\in S\cap\overline{S*}}\nabla F_i\left(\mathbf{c}_i^{(t)}\right)\right\rangle
\end{aligned} \tag{11}$$

We let the first and the second term on the right-hand side as $\|T_1\|^2$ and $\|T_2\|^2$ respectively. Thus the above equation can be written as:

$$\left\|\overline{\mathbf{c}}^{(t+1)} - \mathbf{c}^\star\right\|^2 = \|T_1\|^2 + \|T_2\|^2 + 2\langle T_1, T_2\rangle \leq (1+\alpha)\|T_1\|^2 + (1+\alpha^{-1})\|T_2\|^2 \tag{12}$$

for all $\alpha > 0$.

The T_1 part is the typical decentralized SGD items. Inspired by [Koloskova et al., 2020], we write T_1 as:

$$\left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star - \frac{\eta_t}{n}\sum_{i\in S\cap S*}\nabla F_i\left(\mathbf{c}_i^{(t)}\right)\right\|^2 \leq \left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star\right\|^2 + \eta_t^2\frac{n_1^2}{n^2}\underbrace{\left\|\frac{1}{n_1}\sum_{i=1}^{n_1}\nabla f_i\left(\mathbf{c}_i^{(t)}\right)\right\|^2}_{T_{11}} + 2\eta_t\frac{n_1}{n}\underbrace{\left\langle \overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star, \frac{-1}{n_1}\sum_{i=1}^{n_1}\nabla f_i\left(\mathbf{c}_i^{(t)}\right)\right\rangle}_{T_{12}} + \eta_t^2 v^2 \tag{13}$$

We can bound T_{11} and T_{12} separately as:

$$\begin{aligned} T_{11} &= \left\|\frac{1}{n_1}\sum_{i=1}^{n_1}\left(\nabla f_i\left(\mathbf{c}_i^{(t)}\right) - \nabla f_i\left(\overline{\mathbf{c}}^{(t)}\right) + \nabla f_i\left(\overline{\mathbf{c}}^{(t)}\right) - \nabla f_i\left(\mathbf{c}^\star\right)\right)\right\|^2 \\ &\leq \frac{2}{n_1}\sum_{i=1}^{n_1}\left\|\nabla f_i\left(\mathbf{c}_i^{(t)}\right) - \nabla f_i\left(\overline{\mathbf{c}}^{(t)}\right)\right\|^2 + 2\left\|\frac{1}{n}\sum_{i=1}^{n_1}\nabla f_i\left(\overline{\mathbf{c}}^{(t)}\right) - \frac{1}{n}\sum_{i=1}^{n_1}\nabla f_i\left(\mathbf{c}^\star\right)\right\|^2 \\ &= \frac{2L^2}{n_1}\sum_{i=1}^{n_1}\left\|\mathbf{c}_i^{(t)} - \overline{\mathbf{c}}^{(t)}\right\|^2 + 4L\left(f\left(\overline{\mathbf{c}}^{(t)}\right) - f(\mathbf{c}^\star)\right) \end{aligned} \tag{14}$$

$$\begin{aligned} -T_{12} &= -\frac{1}{n_1}\sum_{i=1}^{n_1}\left[\left\langle\overline{\mathbf{c}}^{(t)} - \mathbf{c}_i^{(t)}, \nabla f_i\left(\mathbf{c}_i^{(t)}\right)\right\rangle + \left\langle\mathbf{c}_i^{(t)} - \mathbf{c}^\star, \nabla f_i\left(\mathbf{c}_i^{(t)}\right)\right\rangle\right] \\ &\leq -\frac{1}{n_1}\sum_{i=1}^{n_1}\left[f_i\left(\overline{\mathbf{c}}^{(t)}\right) - f_i\left(\mathbf{c}_i^{(t)}\right) - \frac{L}{2}\left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}_i^{(t)}\right\|^2 + f_i\left(\mathbf{c}_i^{(t)}\right) - f_i\left(\mathbf{c}^\star\right) + \frac{\mu}{2}\left\|\mathbf{c}_i^{(t)} - \mathbf{c}^\star\right\|^2\right] \\ &\leq -\left(f\left(\overline{\mathbf{c}}^{(t)}\right) - f\left(\mathbf{c}^\star\right)\right) + \frac{L+\mu}{2n_1}\sum_{i=1}^{n_1}\left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}_i^{(t)}\right\|^2 - \frac{\mu}{4}\left\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star\right\|^2 \end{aligned} \tag{15}$$

Now we deal with T_2. From [Ruan and Joe-Wong, 2022] and [Ghosh et al., 2020] we have the following Lemma:

Lemma A.2 *(Mis-classified probability) For a data point belongs to cluster j, the probability of error classification $\mathbb{P}(\epsilon^{j,j'})$ to cluster $j' \neq j$ can be bound as:*

$$\mathbb{P}(\epsilon^{j,j'}) \leq \frac{c_1}{\alpha_0^2\delta^4} \tag{16}$$

And by union bound, the error probability is bounded as:

$$\mathbb{P}(\overline{\epsilon}) \leq \frac{c_1 S}{\alpha_0^2\delta^4} \tag{17}$$

The expected number of clients using wrong cluster of data is bounded as:

$$\mathbb{E}[S \cap \overline{S*}] \leq \frac{c_1 N}{\alpha_0^2\delta^4} = \epsilon_N \tag{18}$$

for some constant c_1. We define this bound as ϵ_N

Inspired by [Ghosh et al., 2020], define T_{2k} as the clients selecting the mis-classified data points that should be belongs to cluster k where $k \neq 1$(The correct cluster). That is:

$$T_{2k} = \sum_{i\in S\cap\overline{S*}\cap S_k*}\nabla F_i(\mathbf{c}_i) \tag{19}$$

For each T_{2k}, we use n_k to indicate the number of clients using mis-classified data that should be belongs to cluster k. We have:

$$T_{2k} = \sum_{i=1}^{n_k} \nabla F_i^k(\mathbf{c}_i) + \sum_{i=1}^{n_k} \nabla F_i(\mathbf{c}_i) - \nabla F_i^k(\mathbf{c}_i) \tag{20}$$

Taking the expectation and by Markov's inequality:

$$\begin{aligned} \|T_{2k}\| &= \left\| \sum_{i=1}^{n_k} \nabla F_i^k(\mathbf{c}_i) + \sum_{i=1}^{n_k} \nabla F_i(\mathbf{c}_i) - \nabla F_i^k(\mathbf{c}_i) \right\| \\ &\le 3 n_k L + \frac{\sqrt{n_k} v}{\theta_1} \end{aligned} \tag{21}$$

For any $\theta_1 \in (0,1)$ with probability equal or greater than $1-\theta_1$. The above used Lemma A.2, Assumption 5.5 and Assumption 5.6 and the Markov inequality.

Using the union bound we see that $T_2 = \sum_k T_{2k}$ is bounded as the following with probability greater or equal to $1-(S-1)\theta_1 - \theta_2$:

$$\begin{aligned} \|T_2\|^2 = \| \sum_{k=2}^{S} T_{2k} \|^2 &\le (S-1) \sum_{k=2}^{S} \|T_{2k}\|^2 \\ &\le \frac{18 L^2 \epsilon_N^2}{\theta_2^2} + \frac{2\epsilon_N (S-1) v^2}{\theta_1^2 \theta_2} \end{aligned} \tag{22}$$

When $\sum_{k=2}^{S} n_k \le \frac{\epsilon_N}{\theta_2}$ with probability at least $1-\theta_2$.

Combining the above three terms and Lemma A.2, we have:

$$\begin{aligned} \mathbb{E} \left\| \overline{\mathbf{c}}^{(t+1)} - \mathbf{c}^\star \right\|^2 \le (1 - \eta_t \mu + \eta_t \mu \frac{\epsilon_N}{n}) \left\| \overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star \right\|^2 + \frac{18 L^2 \epsilon_N^2 \eta_t^2}{n^2} + \frac{2\epsilon_N (S-1) v^2 \eta_t^2}{n^2} + \eta_t^2 v^2 \\ + \frac{\eta_t (L+\mu)}{n} \sum_{i=1}^{n_1} \left\| \overline{\mathbf{c}}^{(t)} - \mathbf{c}_i^{(t)} \right\|^2 + \left(\frac{4\eta_t^2 (n-\epsilon_N)^2 L}{n^2} + 2\eta_t - \frac{2\eta_t \epsilon_N}{n} \right) \left(f\left(\overline{\mathbf{c}}^{(t)}\right) - f\left(\mathbf{c}^\star\right) \right) \end{aligned} \tag{23}$$

A.2 PROOF OF THEOREM 5.9

For time $t-1$, after the local updating round, the cluster parameters can be expressed as:

$$\mathbf{C}^{t-1'} = \mathbf{C}^{t-1} - \eta_t \mathbf{G}^{t-1} \tag{24}$$

After the communication round, the parameters can be expressed as:

$$\mathbf{C}^{t} = \mathbf{C}^{t-1'} \mathbf{W}^{t-1} = \mathbf{C}^{t-1} \mathbf{W}^{t-1} - \eta_t \mathbf{G}^{t-1} \mathbf{W}^{t-1} \tag{25}$$

Thus, recursively expanding the parameters at time t back to $l\beta$, we can get the final form:

$$\mathbf{C}^{t} = \mathbf{C}^{l\beta} \prod_{m=l\beta}^{t-1} \mathbf{W}^m - \sum_{m=l\beta}^{t-1} \left(\eta_t \mathbf{G}^m \prod_{r=t-1}^{m} \mathbf{W}^r \right) \tag{26}$$

A.3 PROOF OF THEOREM 5.10

Following the same flow of Lemma 9 in [Koloskova et al., 2020], applying Theorem 5.10, we have for all $\alpha > 0$:

$$\begin{aligned}
\mathbb{E}\|\mathbf{C}^t - \overline{\mathbf{C}}^t\|_F^2 = N\mathbf{E}_t &\le \mathbb{E}\left\|\mathbf{C}^{(m\beta)} \prod_{i=t-1}^{m\beta} \mathbf{W}^{(i)} - \bar{\mathbf{C}}^{(m\beta)} + \sum_{j=m\beta}^{t-1} \eta_j \nabla \mathbf{F}\left(\mathbf{C}^{(j)}\right) \prod_{i=t-1}^{j} \mathbf{W}^{(i)}\right\|_F^2 \\
&\le \mathbb{E}\left\|\mathbf{C}^{(m\beta)} \prod_{i=t-1}^{m\beta} \mathbf{W}^{(i)} - \bar{\mathbf{C}}^{(m\beta)} + \sum_{j=m\beta}^{t-1} \eta_j \left(\nabla \mathbf{F}\left(\mathbf{C}^{(j)}\right) - \nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right) \prod_{i=t-1}^{j} \mathbf{W}^{(i)}\right\|_F^2 \\
&+ \left\|\sum_{j=m\beta}^{t-1} \eta_j \left(\nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right) \prod_{i=t-1}^{j} \mathbf{W}^{(i)}\right\|_F^2 \\
&\le (1+\alpha)\mathbb{E}\left\|\mathbf{C}^{(m\beta)} \prod_{i=t-1}^{m\beta} \mathbf{W}^{(i)} - \bar{\mathbf{C}}^{(m\beta)}\right\|_F^2 \\
&+ (1+\alpha^{-1})\mathbb{E}\left\|\sum_{j=m\beta}^{t-1} \eta_j \left(\nabla \mathbf{F}\left(\mathbf{C}^{(j)}\right) - \nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right) \prod_{i=t-1}^{j} \mathbf{W}^{(i)}\right\|_F^2 \\
&+ \left\|\sum_{j=m\beta}^{t-1} \eta_j \left(\nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right) \prod_{i=t-1}^{j} \mathbf{W}^{(i)}\right\|_F^2
\end{aligned} \tag{27}$$

Using Assumption 5.7, the above can be further simplified:

$$\begin{aligned}
\mathbb{E}\|\mathbf{C}^t - \overline{\mathbf{C}}^t\|_F^2 &\le (1+\alpha)(1-p)\mathbb{E}\left\|\mathbf{C}^{(m\beta)} - \overline{\mathbf{C}}^{(m\beta)}\right\|_F^2 \\
&+ (1+\alpha^{-1})2\beta \sum_{j=m\beta}^{t-1} \eta_j^2 \mathbb{E}\left\|\left(\nabla \mathbf{F}\left(\mathbf{C}^{(j)}\right) - \nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right)\right\|_F^2 \\
&+ \sum_{j=m\beta}^{t-1} \eta_j^2 \mathbb{E}\left\|\left(\nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right)\right\|_F^2 \\
&\le (1+\alpha)(1-p)\mathbb{E}\left\|\mathbf{C}^{(m\beta)} - \overline{\mathbf{C}}^{(m\beta)}\right\|_F^2 \\
&+ (1+\alpha^{-1})2\beta \sum_{j=m\beta}^{t-1} \eta_j^2 \mathbb{E}\left\|\left(\nabla \mathbf{F}\left(\mathbf{C}^{(j)}\right) - \nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right)\right\|_F^2 \\
&+ \sum_{j=m\beta}^{t-1} \eta_j^2 n v^2
\end{aligned} \tag{28}$$

The expectation of the second term on the right-hand side can be bounded as:

$$\begin{aligned}
&\mathbb{E}\left\|\left(\nabla \mathbf{F}\left(\mathbf{C}^{(j)}\right) - \nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right)\right\|_F^2 \\
&= \mathbb{E}\left\|\left(\nabla \mathbf{F}\left(\mathbf{C}^{(j)}\right) - \nabla \mathbf{F}\left(\overline{\mathbf{C}}\right) + \nabla \mathbf{F}\left(\overline{\mathbf{C}}\right) - \nabla \mathbf{F}\left(\mathbf{C}^{\star}\right) + \nabla \mathbf{f}\left(\mathbf{C}^{\star}\right)\right)\right\|_F^2 \\
&\le 3\frac{n}{N} L^2 \|\mathbf{C}^{(j)} - \overline{\mathbf{C}}^{(j)}\|_F^2 + 3n\sigma^2 + 6nL(f(\overline{\mathbf{c}^j}) - f(\mathbf{c}^{\star})) \\
&\le 3\frac{n}{N} L^2 \|\mathbf{C}^{(j)} - \overline{\mathbf{C}}^{(j)}\|_F^2 + 3n\sigma^2 + 6nL(f(\overline{\mathbf{c}^j}) - f(\mathbf{c}^{\star}))
\end{aligned} \tag{29}$$

Putting the above equations together and setting a proper α to make the first term become $1 - \frac{p}{2}$, similar to [Koloskova et al., 2020] with stepsize $\eta_j \leq \frac{p\sqrt{N}}{12\sqrt{2n\beta}L}$, we can get the desired bound:

$$\begin{aligned}\mathbf{E}_t \leq (1 - \frac{p}{2})\mathbf{E}_{m\beta} + \frac{p}{16\beta}\sum_{j=m\beta}^{t-1}\mathbf{E}_j + \frac{36Ln\beta}{pN}\sum_{j=m\beta}^{t-1}\eta_j^2(f(\overline{\mathbf{c}^j}) - f(\mathbf{c}^\star)) \\ + \left(\frac{18\beta n}{Np}\sigma^2 + \frac{n}{N}v^2\right)\sum_{j=m\beta}^{t-1}\eta_j^2\end{aligned} \tag{30}$$

A.4 PROOF OF THEOREM 5.11

We adapted the following Lemma A.3 from [Koloskova et al., 2020]:

Lemma A.3 *(Simplify the Recursive Equations) For a bound of the cluster distance to the optimal $d_t = \mathbb{E}\|\overline{\mathbf{c}}^{(t)} - \mathbf{c}^\star\|^2$ in the following form:*

$$d_{t+1} \leq (1 - a\eta_t)\, d_t - b\eta_t e_t + c\eta_t^2 + \eta_t B\mathbf{E}_t, \tag{31}$$

and for any non-negative sequences $\{\mathbf{E}_t\}_{t\geq 0}, \{e_t\}_{t\geq 0}, \{\eta_t\}_{t\geq 0}$ that satisfy the following form:

$$\mathbf{E}_t \leq \left(1 - \frac{p}{2}\right)\mathbf{E}_{m\beta} + \frac{p}{16\beta}\sum_{j=m\beta}^{t-1}\mathbf{E}_j + D\sum_{j=m\beta}^{t-1}\eta_j^2 e_j + A\sum_{j=m\beta}^{t-1}\eta_j^2, \tag{32}$$

then if the learning rate $\{\eta_t^2\}_{t\geq 0}$ and $\{r_t\}_{t\geq 0}$ are respectively a $\frac{8\beta}{p}$-slow decreasing sequence and $\frac{16\beta}{p}$-slow increasing non-negative sequence, then for some constant $E > 0$ with learning rate $\eta_t \leq \frac{1}{16}\sqrt{\frac{pb}{DB\beta}}$ the following holds:

$$E\sum_{t=0}^{T} r_t\mathbf{E}_t \leq \frac{b}{2}\sum_{t=0}^{T} r_t e_t + 64BA\frac{\beta}{p}\sum_{t=0}^{T} r_t\eta_t^2 \tag{33}$$

By combining the above equations we have:

$$\frac{1}{2R_T}\sum_{t=0}^{T} br_t e_t \leq \frac{1}{R_T}\sum_{t=0}^{T}\left(\frac{(1 - a\eta_t)\, r_t}{\eta_t}d_t - \frac{r_t}{\eta_t}d_{t+1}\right) + \frac{c}{R_T}\sum_{t=0}^{T} r_t\eta_t + \frac{64BA}{R_T}\sum_{t=0}^{T} r_t\eta_t^2 \tag{34}$$

Where $R_T = \sum_{t=0}^{T} r_t$

Following the previous Lemma, we adapt Lemma 13 from [Koloskova et al., 2020] as the following Lemma A.4

Lemma A.4 *(Main Recursion) The main recursion can be bounded as the following with a constant step-size $\eta_t = \eta < \frac{1}{h}$:*

$$\frac{1}{2R_T}\sum_{t=0}^{T} be_t r_t + ad_{T+1} \leq \tilde{\mathcal{O}}\left(d_0 h\exp\left[-\frac{a(T+1)}{h}\right] + \frac{c}{aT} + \frac{BA}{a^2T^2}\right) \tag{35}$$

For the following two cases, tuning η we have: If $\frac{1}{h} \geq \frac{\ln\left(\max\left\{2, a^2 d_0 T^2/c\right\}\right)}{aT}$ η is chosen to be equal to this value and that:

$$\tilde{\mathcal{O}}\left(ad_0 T\exp\left[-\ln\left(\max\left\{2, a^2 d_0 T^2/c\right\}\right)\right]\right) + \tilde{\mathcal{O}}\left(\frac{c}{aT}\right) + \tilde{\mathcal{O}}\left(\frac{BA}{a^2T^2}\right) = \tilde{\mathcal{O}}\left(\frac{c}{aT}\right) + \tilde{\mathcal{O}}\left(\frac{BA}{a^2T^2}\right) \tag{36}$$

If else choose $\eta = \frac{1}{h}$ and that:

$$\tilde{\mathcal{O}}\left(d_0 h\exp\left[-\frac{a(T+1)}{h}\right] + \frac{c}{h} + \frac{BA}{h^2}\right) \leq \tilde{\mathcal{O}}\left(d_0 h\exp\left[-\frac{a(T+1)}{h}\right] + \frac{c}{aT} + \frac{BA}{a^2T^2}\right) \tag{37}$$

Using the above Lemma A.3, Lemma A.4 and Theorem 5.8 and Theorem 5.10, we can get the final bound.

B SIMULATION DETAILS AND ADDITIONAL SIMULATIONS

B.1 EXPERIMENT DETAILS

The following shows the detailed settings of our experiments. We largely follow Marfoq et al. [2021], Ruan and Joe-Wong [2022] in our experiment settings.

B.1.1 MNIST/EMNIST Data

Half of the dataset was selected to undergo a 90-degree rotation. Each client received the same amount of data, but the ratio of rotated to non-rotated data was set uniformly at random in the range from 10% and 90%. The number of clients was fixed at $N = 100$ for comparison with the baselines. A CNN (convolutional neural network) model was employed, consisting of two convolutional layers with kernel size and padding set to 5 and 2, respectively. Each convolutional layer was followed by a max-pooling layer with a kernel size of 2. After the convolutional layers, fully connected layers were used, with a dropout layer of size 50. The ReLU activation function was applied to each convolution layer and fully-connected layer. All clients utilized SGD as the optimizer. The number of local epochs was set to 5, with the initial step having double the local epochs to accelerate the initial learning, leading to a faster reduction in global loss. The initial learning rate was 5e-2, with a decay factor of 0.80. Training was carried out over 150 global epochs. Regarding network topology, unless otherwise specified, ER Random Graph with connecting probability $p = 0.06$ and total number of clients $N = 100$ were used in the experiments. The results in the table are averaged over five individual experiments.

B.1.2 CIFAR-10 & CIFAR-100 Data

The dataset was divided into even and odd labels by its number of label marked in the dataset, and half of the data was randomly selected to undergo a 90-degree rotation. This process potentially created four different data distributions (rotated even, un-rotated even, rotated odd, un-rotated odd). Each client received an equal amount of data, but the proportion of odd-labeled and even-labeled data was randomly assigned, ranging uniformly at random from 10% to 90%. The number of clients was set to $N = 25$ for comparison with the baselines. A CNN model with four convolutional layers was used. The first two layers had a kernel size and padding of 5 and 2, respectively, while the last two layers had a kernel size and padding of 3 and 1, respectively. Each convolutional layer was followed by batch normalization. After the second and fourth convolutional layers, max-pooling with a kernel size of 2 and a dropout layer were applied. Following the convolutional layers, two fully connected layers with dropout and batch normalization were used, containing 1024 and 512 hidden neurons, respectively. The activation function was ReLU on each layer. All clients used SGD as the optimizer. The number of local epochs was set to 5, with the initial step doubling the local epochs. The initial learning rate was set to 5e-2, with a decay factor of 0.85. Training was conducted for 150 global epochs. Regarding network topology, unless otherwise specified, ER Random Graph with connecting probability $p = 0.20$ and total number of clients $N = 25$ were used in the experiments. The results in the table are averaged over two independent runs, except for FedSoft, which was only run once due to its extensive runtime. The results in the Figure are averaged over three independent runs with slightly lower number of clients $N = 15$.

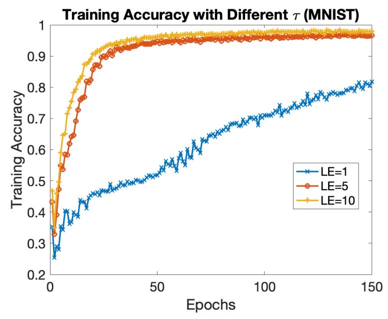
For the code for FedSPD, please refer to the Link.

B.2 ADDITIONAL SIMULATION RESULTS

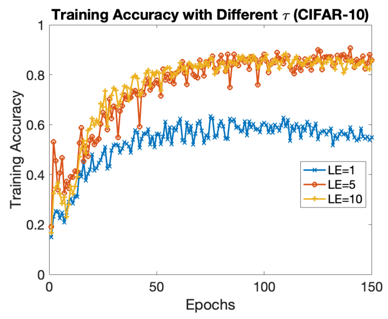
B.2.1 Varying the Number of Local Epochs

We conducted experiments for 150 epochs on the MNIST, CIFAR-10, and CIFAR-100 datasets. As shown in Figure 5, increasing the number of local epochs in **FedSPD** leads to faster convergence. For $\tau = 1$, the training did not converge even after 150 epochs on MNIST, and for CIFAR-10 and CIFAR-100, it seemed to converge to a lower training accuracy. We observed that as the dataset and model complexity increased, increasing the number of local epochs tended to improve performance.

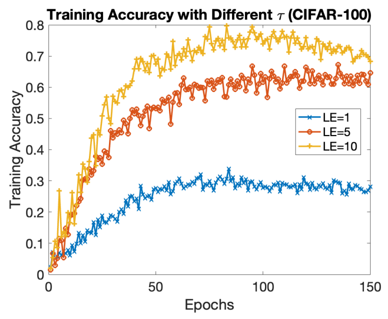
Table 6 presents the final **FedSPD** testing accuracies for different numbers of local epochs across the datasets. On MNIST, the testing accuracies were 93.27% and 93.47%, respectively, showing only a slight difference, likely because the MNIST dataset is relatively simple, so the learning hyperparameters do not make much of a difference in model performance. For CIFAR-10, the testing accuracies for $\tau = 5$ and $\tau = 10$ were 70.61% and 66.52%, respectively, where a larger number of local epochs actually reduced the final performance. However, for CIFAR-100, $\tau = 10$ resulted in the best performance. This suggests that for more complex datasets, a higher number of local epochs can be beneficial, as indicated by the training accuracy curves. Nevertheless, it is important to note that setting τ too high may lead to overfitting to the local data, as was the case with $\tau = 10$ on the CIFAR-10 dataset. These findings are consistent with known results in general federated learning, where a higher number of local epochs can effectively increase the number of gradient steps taken, accelerating convergence as long as the local models do not diverge too much due to a large number of local steps.



(a) *Training accuracy on MNIST.*



(b) *Training accuracy on CIFAR-10.*



(c) *Training accuracy on CIFAR-100.*

Figure 5: ***FedSPD*** *training accuracy with different numbers of local steps τ. When the data become more complicated, increasing local epochs may be a better choice.*

Local Epochs	1	5	10
MNIST	74.20	93.27	93.47
CIFAR-10	41.34	70.61	66.52
CIFAR100	19.86	43.35	44.99

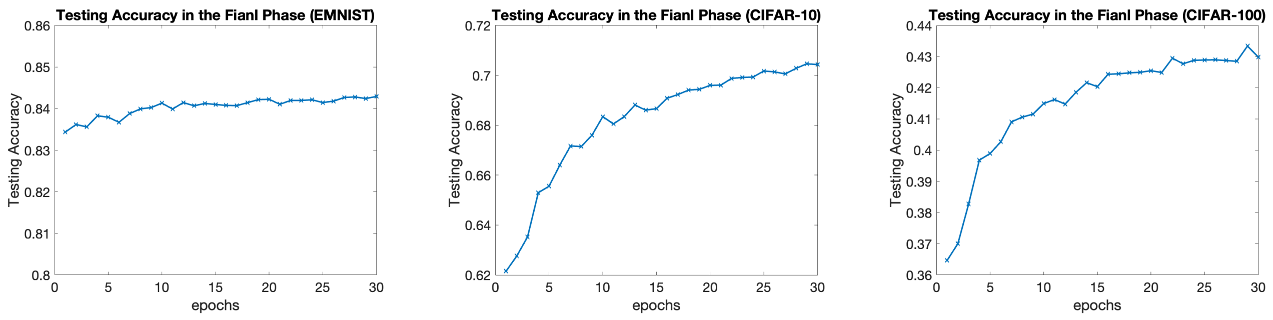
Table 6: *Final* ***FedSPD*** *testing accuracies for different number of local epochs.*

B.2.2 Influence of the Final Phase

Our **FedSPD** algorithm uses a final phase that follows the typical federated learning training process. The optimal number of epochs for this final phase varies depending on the dataset and learning model. Due to the simplicity of EMNIST and its model, the testing accuracy is already sufficiently high after aggregation. In our EMNIST setup, using 10 epochs in the final phase increases performance by 0.5%, and beyond 10 epochs, the testing accuracy stabilizes. For CIFAR-10 and CIFAR-100, the testing accuracy improves by 7% and 6%, respectively, after 15 epochs. Around 30 epochs are sufficient to achieve optimal performance for both datasets. It is important to note that choosing the correct number of epochs and learning rate for this final phase is crucial. Too many epochs, or a learning rate that is too high (or with insufficient decay), may lead to overfitting to the local data. Since this final phase is trained locally without any communication overhead, it presents a key advantage of our **FedSPD** algorithm in communication-constrained settings. Additionally, note that for EMNIST, CIFAR-10, and CIFAR-100, our **FedSPD** already achieves higher accuracies compared to other methods, even without this final phase. Other algorithms like **FedEM** perform aggregation during the regular training phase, so adding extra local rounds in a final phase of training may lead to overfitting.

B.2.3 Influence of the Number of Clusters

The testing accuracy with different hyperparameters S (number of clusters) for the CIFAR-10 and CIFAR-100 datasets is shown in Figure 7. In the experimental settings, we potentially created four different distributions by using varying labels and image rotations. In our **FedSPD** algorithm, setting S too high does not necessarily improve performance. This may be because most practical loss functions, such as the cross-entropy used in neural networks, are non-convex, meaning that the aggregated model may not perform optimally in practice. Aggregating more models in the final phase can exacerbate



(a) *Testing accuracy on EMNIST.* (b) *Testing accuracy on CIFAR-10.* (c) *Testing accuracy on CIFAR-100.*

Figure 6: *Testing accuracy of the final phase.*

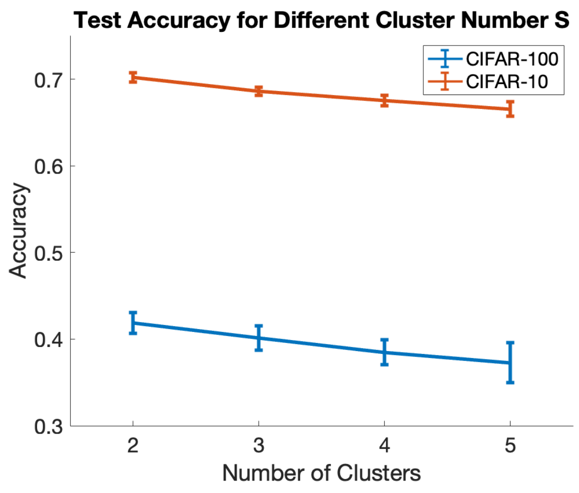


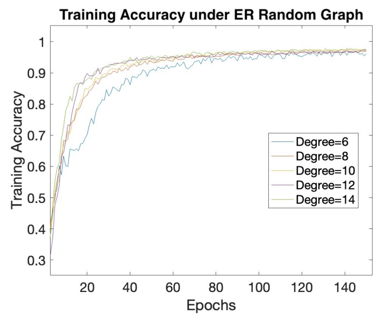
Figure 7: **FedSPD** test accuracy for different numbers of clusters S.

this issue. However, in our **FedSPD** algorithm, setting $S = 2$ already gives excellent performance in terms of the final test accuracy.

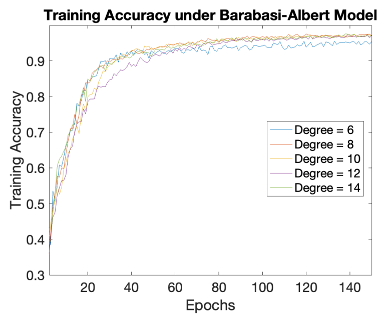
B.2.4 Extra Details for Experiments with Different Graph Connectivity

FedSPD's training accuracy versus epochs for MNIST across different topologies is shown in Figure 8. We observe that networks with lower connectivity typically converge more slowly than those with higher connectivity, in each topology. Additionally, RGG exhibits more oscillations compared to other topologies, likely due to its high clustering effect [Penrose, 2003]. However, all topologies eventually reach the same level of training accuracy, regardless of the network structure, indicating that, as predicted by Theorem 5.11, **FedSPD** converges as long as the network is connected.

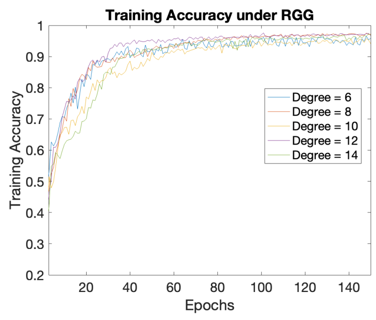
In addition to static network topology settings, we evaluate the performance of our **FedSPD** algorithm under dynamic network conditions. In the following experiments, we use CIFAR-100 with 25 clients, initializing the network with an Erdős–Rényi (ER) random graph. To simulate a dynamic network, at each epoch, every existing edge has a probability p of being removed, while each non-existent edge has a probability p_{add} of being added. The value of p_{add} is adjusted at each epoch to maintain a roughly constant average connectivity across the network. A larger value of p corresponds to a more dynamic network topology over time. The results are summarized in Table 7. From the results, we observe that network dynamics have little effect on performance—our **FedSPD** consistently maintains its effectiveness across different edge removal probabilities p.



(a) *Training accuracy of ER Graph.*



(b) *Training accuracy of BA Model.*



(c) *TrainingaAccuracy of RGG.*

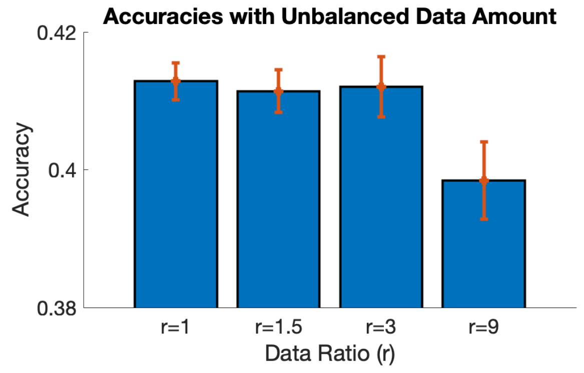
Figure 8: ***FedSPD** converges slightly faster on networks of higher average degree, with noisier convergence on highly clustered RGG graphs, on MNIST Data.*

p	0.3	0.2	0.1	0 (Static)
Test Accuracy	37.56	37.22	37.42	37.14

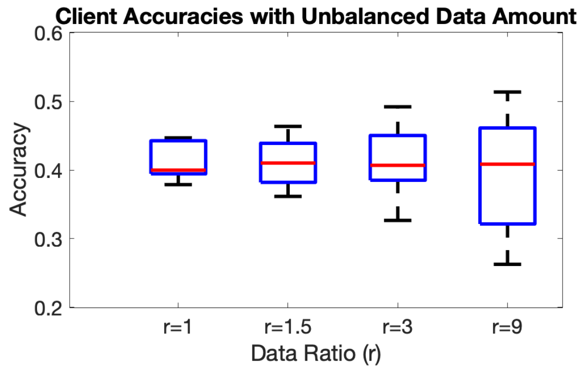
Table 7: *Performance of Dynamic Network Topology.*

B.2.5 Impact of Data Quantity Imbalance Across Clients

In addition to considering data imbalance across clusters, we further evaluate the performance of our method under a more challenging setting where both inter-cluster imbalance and total data imbalance across clients are present. Specifically, we conduct experiments using the CIFAR-100 dataset with the same configuration as described in Figure 4. To simulate varying amounts of data per client, we categorize clients into three groups: low, average, and high data holders. Let r denote the ratio of data volume between clients with the highest and lowest data quantities.



(a) *Average Test accuracy with unbalanced data amount.*



(b) *Box plot of test accuracy across clients.*

Figure 9: *Test accuracy with unbalanced data amount.*

The results of this experiment are presented in Figure 9. We observe that the average accuracy remains stable as the imbalance ratio r increases. Notably, even under the most skewed setting ($r = 9$), the clients with the lowest test accuracy achieve approximately 30% accuracy—substantially higher than the performance of local training under uniform data allocation, which yields only around 14% accuracy. This demonstrates that clients with limited data can significantly benefit from collaborative training and knowledge sharing with other clients.

B.2.6 Experiments Incorporating Differential Privacy (DP)

We follow Wei et al. [2020] conduct the experiments on MNIST dataset with 50 clients. The parameters of DP is selected as follow: Clipping Threshold $C = 1$, $\delta = 0.01$ thus c is chosen to be $\sqrt{2\ln\frac{1.25}{0.01}}$. We select ϵ to be 10, 50 and 100 to do the experiment. Table 8 shows the results of different settings with two different accuracies. One is the test accuracy of our **FedSPD** right after the model aggregation. The other one is the accuracy after 10 local epochs of our final phase. The reason to include 2 different accuracies is because the final phase is local training, need not to do the DP. If we only show the accuracy after the final phase, the influence of DP might not be clear.

Metrics	No DP	DP ($\epsilon = 100$)	DP ($\epsilon = 50$)	DP ($\epsilon = 10$)
Test Accuracy (Post Aggregation)	92.51	92.75	92.46	92.13
Test Accuracy (After Final Phase)	93.89	93.99	93.87	93.70

Table 8: *Results with DP on MNIST dataset.*

From the results, we see that our **FedSPD** algorithms combine perfectly with DP. The accuracies keep at a high level with different settings. The $\epsilon = 100$ case even have a slightly higher accuracy compare to the case without DP. This may be potentially due to a moderate additive noise actually preventing over-fitting in certain level. Another observation is that, actually the final phase of our algorithms do enhance the model and reduce the gap of the test accuracies across different settings. This is another evidence showing that the final phase of our **FedSPD** do further personalize the local model well.

B.2.7 Experiments Using MobileNet-v2

We conducted experiments on the CIFAR-100 dataset and the CIFAR-10 dataset and its mixtures with MNIST and FashionMNIST, using MobileNet-v2 as the machine learning model. The mixed datasets were created by sampling 25,000 data points from CIFAR-10 and 25,000 from MNIST/FashionMNIST. Each client randomly drew between 10% and 90% of its data from one of the sampled datasets, with the remainder sourced from the other. The network topology was modeled as an Erdős–Rényi (ER) random graph with a connection probability of $p = 0.20$ and a total of 20 clients. The results are presented in Table 9.

For CIFAR-100, our method outperforms other methods. However in the data mixture settings, the results indicate that as the model size and complexity increase, FedAvg outperforms all other algorithms. As models and datasets become more complex, personalization methods may occasionally exhibit reduced performance due to many reasons. This may be attributed to the model's expressiveness, which allows it to effectively capture variations across different clients. In this case, the global model is sufficiently robust and more effective than personalizing to local data distributions. Similar results were also observed in the decentralized FedEM Marfoq et al. [2021], where performance degradation occurred under specific conditions in their experiments. In specific, their decentralized FedEM performs worse than FedAvg on FEMNIST and CIFAR-10.

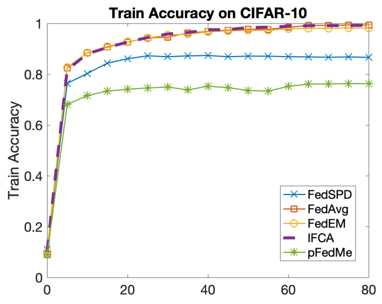
Our proposed method, **FedSPD**, experiences greater challenges in such scenario, as it splits the local dataset into two clusters for separate training and postpone the aggregation. However, when the distributions of the two clusters differ significantly, such as in the mixture of CIFAR-10 with MNIST or FashionMNIST, **FedSPD** achieves faster convergence, as demonstrated in Figures 10b and 10c. This highlights **FedSPD**'s ability to accurately distinguish data sampled from different distributions, showing that **FedSPD** can be trained efficiently when the communication/computing resources are limited.

	FedSPD	FedEM	IFCA	pFedMe	FedAvg	Local
CIFAR-10	72.14	78.02	78.56	61.00	79.01	57.72
CIFAR-10 + MNIST	86.20	88.88	89.08	74.64	89.27	77.76
CIFAR-10 + FashionMNIST	80.04	85.01	85.47	69.37	85.58	72.74
CIFAR-100	46.13	42.29	44.48	32.77	45.70	18.52

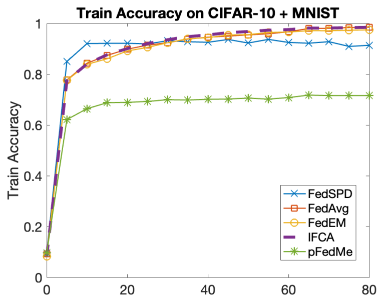
Table 9: *Results using MobileNet-v2.*

C DETAILED COMPARISON BETWEEN FEDSPD AND FEDSOFT

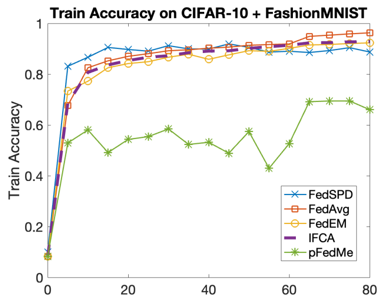
The training processes of **FedSPD** and **FedSoft** differ fundamentally. In **FedSoft**, local training utilizes a proximal objective function with regularization terms to account for the distance to each cluster center. Conversely, **FedSPD** trains using data associated with the selected cluster, where model parameters are updated based on an objective function that does not include the regularization term. This modification is critical in decentralized federated learning, where ensuring both convergence



(a) *Training Accuracy of Single CIFAR-10 Dataset.*



(b) *Training Accuracy of Mixture of CIFAR-10 + MNIST.*



(c) *Training Accuracy of Mixture of CIFAR-10 + FashionMNIST.*

Figure 10: *Experiments on Various Datasets Using MobileNet-v2.*

to an optimal solution and consensus on cluster centers is essential. Unlike centralized systems, which aggregate cluster centers at a single server, decentralized systems lack this central coordination, making consensus challenging. **FedSPD** was inspired by **FedSoft**. Attempts were made to bound the consensus distance of cluster centers in **FedSoft**. However, the results suggest that in decentralized settings of **FedSoft**, the consensus may not be reached. Experimental results in Section 6 demonstrate **FedSoft**'s limitations in the decentralized scenarios. Specifically, **FedSoft**'s cluster centers fail to reach the optimal values due to its update rules:

- Uniform Data Utilization: **FedSoft** updates each cluster center using all available data.
- Probabilistic Contribution: **FedSoft** uses probabilities proportional to the estimated data distribution among clusters to guide contributions to the selected cluster.

These update rules lead to gradients during local updates being biased towards a mixture of optimal cluster centers from all clusters, rather than the correct optimal center for the selected cluster. As datasets grow more complex and the optimal cluster centers diverge significantly (e.g., CIFAR-10 or CIFAR-100), this bias becomes more pronounced, causing degraded performance. This is evident in Section 6, where **FedSoft**'s performance deteriorates as the datasets shift from EMNIST to CIFAR-10 and CIFAR-100 in both centralized and decentralized scenarios.

In decentralized settings, the sparse updates exacerbate the difficulty for clients to estimate optimal cluster centers accurately. This is especially problematic for complex datasets like CIFAR-10 and CIFAR-100, where **FedSoft** performs significantly worse. This limitation highlights the need for a different approach, such as the one introduced by **FedSPD**.

In sum, in **FedSPD** during each round of the first step, clients train separate models using data associated with their selected clusters. This approach ensures consensus and optimality of the cluster centers. Unlike centralized scenarios, where clients share a unified cluster center, each client in decentralized systems maintains different cluster centers. This necessitates rigorous proof of consensus across clients, as described in our theoretical analysis. Thus, the proof techniques of **FedSPD** and **FedSoft** are completely different.

Once consensus on cluster centers is achieved, the second phase of **FedSPD** aggregates models by computing a weighted average of the cluster centers, aligning with **FedSoft**'s objective. To address the suboptimality of non-convex models, **FedSPD** incorporates an additional final phase of local training. This phase enables further exploration of the model parameters, mitigating suboptimality and enhancing overall performance.

Differences in Theoretical Analysis.

- Relax of Assumption 2 in the **FedSoft**. Since **FedSoft** requires the Assumption 2 in their paper which state the β similarity among all subproblems of different clusters. In other words, the optimals of different cluster centers need to be close enough to guarantee the bounded distance between the learned cluster center and the optimal cluster center. This is because they use all data to update the cluster center. If using the data from other cluster to update the selected cluster, this assumption is required to guarantee the gradient update is not going to far away from the optimal of the selected cluster. In contrast, the different update rule of our algorithm **FedSPD** on cluster center further guarantee the optimality of our algorithms and have a tighter bound of convergence without the requirement of this additional assumption.

- The proof of **FedSoft** is based on the centralized FL (CFL). Thus, consensus is automatically met with the centralized aggregation. However, they do not proof the effectiveness under the decentralized settings. In contrast, we proof the consensus of the cluster centers in **FedSPD** under decentralized FL. This is one of the main challenges of the theoretical analysis. Proving the consensus and convergence is much more difficult in decentralized FL (DFL) than the CFL, since all clients keep the different estimation of the cluster centers. Our attempts try to bound the consensus distance of **FedSoft** in DFL was failed since the way that FedSoft update its cluster center may not yield the same optimal for all clients. It may differ based on different neighboring clients.