Towards Client Driven Federated Learning

Songze Li
Southeast University
songzeli@seu.edu.cn

Chenqing Zhu
Hong Kong University of Science and Technology (Guangzhou)
czhu032@connect.hkust-gz.edu.cn

Abstract

Conventional federated learning (FL) frameworks follow a server-driven model where the server determines session initiation and client participation, which faces challenges in accommodating clients' asynchronous needs for model updates. We introduce Client-Driven Federated Learning (CDFL), a novel FL framework that puts clients at the driving role. In CDFL, each client independently and asynchronously updates its model by uploading the locally trained model to the server and receiving a customized model tailored to its local task. The server maintains a repository of cluster models, iteratively refining them using received client models. Our framework accommodates complex dynamics in clients' data distributions, characterized by time-varying mixtures of cluster distributions, enabling rapid adaptation to new tasks with superior performance. In contrast to traditional clustered FL protocols that send multiple cluster models to a client to perform distribution estimation, we propose a paradigm that offloads the estimation task to the server and only sends a *single* model to a client, and novel strategies to improve estimation accuracy. We provide a theoretical analysis of CDFL's convergence. Extensive experiments across various datasets and system settings highlight CDFL's substantial advantages in model performance and computation efficiency over baselines.

1 Introduction

Federated Learning (FL) (McMahan et al., 2017) is a distributed learning framework that allows for collaborative training of a global model across multiple clients while keeping their raw data local. In nearly all current FL frameworks, the central locus of control invariably resides with the server. That is, the server initiates training sessions for model update, and determines which clients should participate and when. However, this server-driven paradigm may fall short in serving clients' needs, especially in scenarios where clients experience asynchronous changes of their data distributions. Specifically, a client experiencing a concept drift will suffer from sub-optimal performance using the old model, until the server calls for the next model update.

In this paper, we attempt to address the above challenge by proposing a novel **C**lient-**D**riven Federated Learning (CDFL) framework, which *empowers each individual client to assume a more autonomous role in the FL process*. Specifically, we focus on the scenario where each client collects data from a mixture of distributions. As the mixing ratio varies over time, the client may seek help from the server, who acts as a service provider, in updating its local model to match the new distribution. As a real-life example, consider a skincare maintenance application, where users' skin types exhibit complexity — perhaps featuring a combination of oiliness and dryness in different areas of skin, reflecting a mixture of distributions. Additionally, it is common for users' skin conditions to vary with seasons, leading to shifts in distributions. Another example is a retail chain with various branches, each of which sell commodities of different store categories. The commodities offered by these branches may evolve based on changing of customer preferences, creating a dynamic mixture of distributions. In CDFL, in sharp contrast to conventional server-driven paradigm, each client possesses complete autonomy in

deciding when to update its model, and the servers plays a passively assistive role for the clients to adapt to their new distributions.

To tackle clients' varying data distributions, we adopt the clustered FL setting where K base cluster models are maintained at the server (Sattler et al., 2020a,b) to update clients' models. In existing clustered FL works, a crucial consideration is to measure the data distributions of clients. Many works distribute all cluster models to clients, leaving it to clients to determine the distribution based on local empirical loss (Ghosh et al., 2020; Mansour et al., 2020; Ruan and Joe-Wong, 2022). However, such an approach poses several challenges. Firstly, it places a significant communication burden to send all the cluster models; Secondly, it imposes substantial computational demands on clients, requiring them to calculate losses for each cluster and make comparisons. Some other approaches leverage distances between uploaded models to form client groups (Duan et al., 2021a), imposing impractical synchronization requirements on clients for data uploads. In sharp contrast, as illustrated in Figure 1, CDFL *assigns the task of evaluating client data distribution to the server*. Based on the model uploaded by a client, the server analyzes its data distribution, and updates the cluster models. Subsequently, the server generates a personalized model and sends it to the client. This significantly simplifying clients' communication and computation compared with previous clustered FL solutions, reflecting another critical aspect of client-driven FL: ensuring good performance on clients with lightweight operations.

Figure 1: High-level view of CDFL.

In the context of above clustered FL, and building upon the client-driven spirit, we develop an asynchronous CDFL framework that focuses on maximizing clients' performance and minimizing clients' complexity. Specifically, we introduce an effective newcomer cold start mechanism, a feature conspicuously absent in the majority of related works (Duan et al., 2021a; Zeng et al., 2023). Furthermore, our framework exhibits adaptability in addressing client distribution drift, a challenge specifically addressed in only one previous study (Duan et al., 2021b) within the context of clustered FL. *CDFL is the first clustered FL framework that focuses on clients' autonomy, performance, and efficiency*. Compared with existing clustered FL works, the computation overhead of a CDFL client remains minimal: it only conducts the minimal required local model training for any FL protocol; the client's communication overhead is also minimized, with solely uploading and downloading *one single model*, at any time completely determined by the client.

We provide convergence analysis that theoretically validates the convergence of client models on cluster and local tasks. Extensive experiments over different datasets and network settings attest to the substantial improvements of CDFL on cluster and client accuracies. Additionally, it significantly alleviates both communication and computational costs at clients over baselines.

2 Related Work

Clustered Federated Learning (clustered FL). Hard clustering algorithms assume clients in the same group have identical data distribution (Briggs et al., 2020; Ghosh et al., 2020; Mansour et al., 2020); while soft clustering methods assume the data of each client follows a mixture of multiple distributions (Ruan and Joe-Wong, 2022; Li et al., 2021). In most cases, expectation-maximization (EM) methods are used to compute clients' distribution (Long et al., 2023; Ma et al., 2022; Ghosh et al., 2022), and global updates leverage methods based on FedAvg (Briggs et al., 2020). Some works add proximal terms on clients' objectives for personalization (Tang et al., 2021).

Asynchronous Federated Learning (asynchronous FL). Asynchronous FL operates on resource-constrained devices (Xu et al., 2021). In typical asynchronous setups, the central server conducts global aggregation immediately upon receiving a local model (Xie et al., 2019; Wang et al., 2022; Chen et al., 2020), or a set of local models (Nguyen et al., 2022; Wu et al., 2020). These asynchronous clients may be grouped into tiers for updating based on factors like staleness or model similarities (Park et al., 2021; Wang and Wang, 2022), referred to as semi-asynchronous. However, this clustering
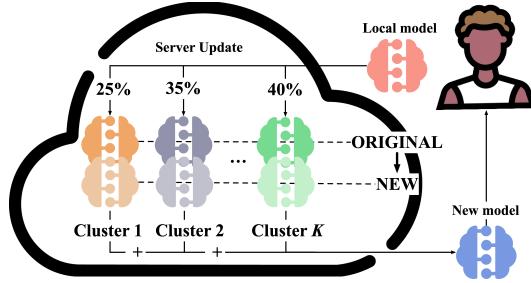
typically contributes to a single global model, and sometimes, the server still selects the clients (Zhang et al., 2021). Existing clustered FL frameworks primarily operate within a synchronous setting. In the context of asynchronous FL, clients are sometimes grouped only to control staleness. Our framework is the first, to the best of our knowledge, to integrate clustered FL within an asynchronous setting.

Personalized Online FL. Mainstream online FL approaches primarily address the communication and computation burden on clients, with the goal of ensuring efficient FL training sessions, irrespective of whether the system operates synchronously or asynchronously (Jiang et al., 2023; Zheng et al., 2023; Gauthier et al., 2023; Damaskinos et al., 2022). Other research efforts in this domain focus on assisting clients in adapting to new task streams (Gogineni et al., 2022; Ganguly et al., 2023; M Ghari and Shen, 2022). Chen et al. (2020) is also an asynchronous FL framework which accomplishes personalization by aggregation of local and global models. Yu et al. (2021) proposes a personalized FL system based on human activity recognition (HAR), albeit with limited applicability to other data types. Additionally, Deng et al. (2020) introduces an adaptive personalized method that combines local and global models, but overlooks potential asynchrony issues. In light of these observations, our proposed framework addresses personalized online FL challenges within an asynchronous setting by leveraging clustered FL algorithms.

User-centric FL Frameworks. Few works have studied FL from a comprehensive user's perspective. Mestoukirdi et al. (2021, 2023) claim to be user-centric, but are indeed personalized FL frameworks dealing with communication burdens. In Khan et al. (2023), the authors point out that existing FL works take away clients' autonomy to make decisions themselves, and propose a token-based incentive mechanism that rewards personalized training. However, this work fails to consider the asynchrony among clients, making it insufficient to provide full autonomy to clients. Note that the shift in clients' distribution is distinct from Federated Continual Learning (FCL) Yoon et al. (2021), which primarily aims to minimize catastrophic forgetting. Our focus lies solely in enabling clients to seamlessly adapt their models to new data during distribution shifts.

3 Problem Definition

Consider an FL system with one central server and many distributed clients. The server maintains K cluster models, corresponding to K distributions $P_1, \ldots, P_K$, and has a proxy dataset D_k for each P_k. Note that the existence of small-size proxy datasets is rather a mild and common assumption in FL literature (Wang et al., 2024; Li and Wang, 2019; Lin et al., 2020). The value of K is determined a priori, according to the type of service (e.g., genders or ethnicities in the skincare service), or is deducted from a small amount of data collected in advance. Given a loss function $l(w; x, y)$, each cluster $k \in [K] \triangleq \{1, \ldots, K\}$ aims to find an optimal model w_k that minimizes the objective

$$F_k(w_k) = \mathbb{E}_{(x,y) \sim P_k}[l(w_k; x, y)]. \tag{1}$$

The training takes T global epochs. For each epoch $t \in [T]$, some client m collects local data following a mixture of distribution $P_m^t = \sum_{k=1}^{K} \alpha_{mk}^t P_k$, with $\alpha_{mk}^t \in [0, 1]$ and $\sum_{k=1}^{K} \alpha_{mk}^t = 1$. Here α_{mk}^t is the importance weight of cluster k to client m at epoch t. The importance weights may vary over time, i.e., $\alpha_{mk}^t \neq \alpha_{mk}^{t'}$ for $t \neq t'$, and are unknown to the client. Each time when client m's data distribution shifts, it may choose to fit the local model v_m^t to the new distribution, and uploads it to the server. The server enhances v_m^t to construct a personalized model u_m^t for client m:

$$u_m^t = g(v_m^t, \{w_k^{t-1}\}_{k=1}^{K}, \{D_k\}_{k=1}^{K}), \tag{2}$$

following some rule g, and returns u_m^t to client m. In the meantime, server also updates the cluster models using some function $\mathbf{h}$, such that

$$(w_1^t, \ldots, w_K^t) = \mathbf{h}(v_m^t, \{w_k^{t-1}\}_{k=1}^{K}, \{D_k\}_{k=1}^{K}). \tag{3}$$

Specifically, the local model v_m^t is obtained by optimizing the local objective

$$J_m^t(v_m^t; u_m^\tau) = \frac{1}{n_m^t} \mathbb{E}_{(x^i, y^i) \sim P_m^t} \Big[\sum_{i=1}^{n_m^t} l(v_m^t; x^i, y^i) \Big] + \frac{\rho}{2} \left\| v_m^t - u_m^\tau \right\|^2. \tag{4}$$

Here n_m^t is the number of data samples at client m in epoch t; ρ is some scaling parameter; $\tau < t$ is the last epoch when client m uploads its model v_m^τ to the server.
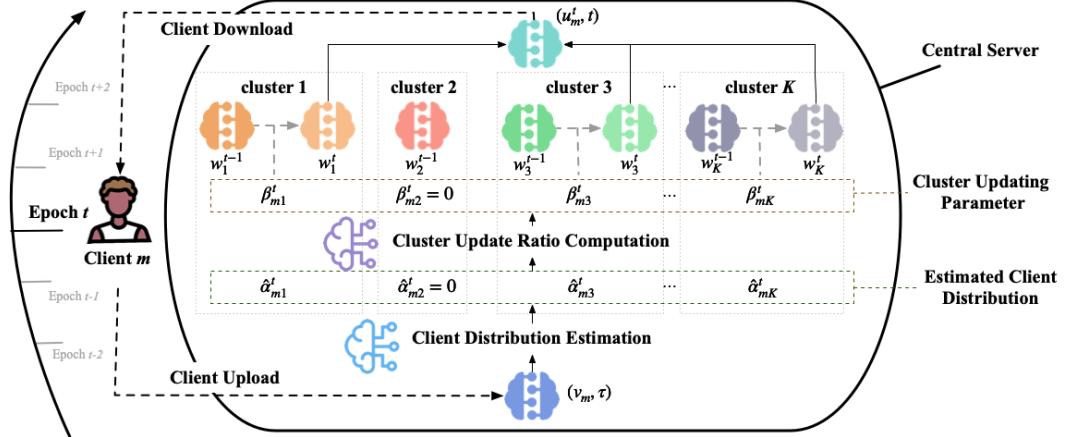
Figure 2: CDFL workflow. Client m uploads model and epoch index of last model update (v_m, τ) to the server. Server performs distribution estimation, derives cluster updating parameters to update the cluster models, and finally constructs an aggregated model u_m^t to send back to the client. Note here as the client's distribution is estimated not to contain P_2, cluster 2's model is not updated and not used in computing u_m^t.

4 Client-driven Federated Learning

We describe the operations of the client and the server respectively in the proposed CDFL framework. The entire workflow of CDFL is depicted in Figure 2 and detailed in Algorithm 1.

4.1 Client Update

Initialization. CDFL is designed to be fully open and dynamic, which does not presuppose a fixed total number of clients. A client can seamlessly joint the system, simply via retrieving an initialization tuple from the server, comprising an initial model and the index of current epoch, denoted as (u_m^t, t).

Training and Uploading. Whenever a client m feels necessary to perform a model update, perhaps due to the occurrence of distribution shift and the resulting performance degradation, it performs local training on current dataset to obtain a local model v_m, and uploads (v_m, τ) to the server, where τ is the index of the latest epoch in which client m updates its model with the server. Having received v_m, the server generates an enhanced model u_m^t for current epoch t, and returns (u_m^t, t) to client m. Finally, client m updates $\tau = t$, and starts to use u_m^t for local tasks.

4.2 Server Update

Throughout the entire process of CDFL, the server passively waits for clients' update requests. Upon receipt of an uploaded model, the server first increments the epoch counter to obtain the current epoch t, then the server initiates a two-step evaluation process. Firstly, it checks if the client's model is too stale. Specifically, as client m uploads (v_m, τ), if $t - \tau > \tau_0$ for some preset staleness threshold τ_0, the server refrains from updating the cluster models, i.e., $w_k^t = w_k^{t-1}$ for all k, and returns the client a personalized model as an aggregation of current cluster models. Otherwise, the server proceeds to estimate client m's data distribution. Subsequently, it updates each cluster model, and constructs a new personalized model for the client, as an aggregation of the updated cluster models.

Distribution Estimation. Upon client m uploading v_m at epoch t (referred to as v_m^t for clarity), the estimation of its distribution hinges on several components, including v_m^t, the latest cluster models $w_1^{t-1}, \ldots, w_K^{t-1}$, and their proxy datasets $D_1, \ldots, D_K$. Two key metrics are evaluated to estimate the importance weights in client m's distribution. The first is the empirical loss of v_m^t on D_k, denoted by $F(v_m^t; D_k)$, for all $k \in [K]$. Specifically, when $F(v_m^t; D_k) < F(v_m^t; D_{k'})$, v_m^t fits P_k better than $P_{k'}$, which indicates that cluster k should have a higher importance weight than cluster k' in client m's distribution P_m^t. The second metric is a measure on the similarity between v_m^t and the cluster models. This similarity can be materialized either through the loss difference between the cluster model and

Algorithm 1: CDFL

Input: Server pre-trained model w_k^0, server proxy dataset $D_k \sim P_k$ ($k \in [K]$), staleness threshold $\tau_0 < T$, cluster update threshold $\beta_0 \in (0, 1)$

Output: Local model parameter v_m, cluster model parameters $w_1, \ldots, w_K$

Initialization: Server sends $(u^0, 0)$ to each client, $u^0 = \frac{1}{K} \sum_{k=1}^{K} w_k^0$. Global epoch $t \leftarrow 0$. Run `Client()` thread and `Server()` thread asynchronously in parallel.

Thread `Server()`:
> **while** *no client uploads* **do**
>> Wait for client update. **if** *client m uploads* (v_m, τ) **then**
>>> $t \leftarrow t + 1$; $u_m^t \leftarrow$ `ServerUpdate` (v_m, τ, t); send (u_m^t, t) to client m.

Thread `Client()`:
> **foreach** *client m in parallel* **do**
>> Receive pair (u_m, t) from server. Set local model $v_m \leftarrow u_m$, local timestamp $t_m \leftarrow t$.
>> **while** *active* **do**
>>> **if** *choose to upload* **then**
>>>> Define $J_m(v_m; u_m)$ as in (4).
>>>> **foreach** *local iteration h* **do**
>>>>> $v_{m,h} \leftarrow v_{m,h-1} - \gamma \nabla J_m(v_{m,h-1}; u_m)$
>>>>> /* learning rate γ */
>>>> Upload (v_m, t_m) and wait for server response.

Function `ServerUpdate`(v_m, τ, t):
> **if** $t - \tau > \tau_0$ **then**
>> /* Client deprecated. Do not update cluster models.   */
>> **foreach** $k \in [K]$ **do** $w_k^t \leftarrow w_k^{t-1}$
>> **return** $u_m^t = \sum_{k=1}^{K} \hat{\alpha}_{mk}^\tau w_k^t$.
> $\hat{\alpha}_{m1}^t, \ldots, \hat{\alpha}_{mK}^t \leftarrow$ `DistributionEstimate` $(v_m, w_1^{t-1}, \ldots, w_K^{t-1}, D_1, \ldots, D_K, t)$
> $\beta_{m1}^t, \ldots, \beta_{mK}^t \leftarrow$ `UpdateRatioCompute` $(\hat{\alpha}_{m1}^t, \ldots, \hat{\alpha}_{mK}^t, \beta_0, \tau, t)$
> **foreach** $k \in [K]$ **do** $w_k^t \leftarrow (1 - \beta_{mk}^t) w_k^{t-1} + \beta_{mk}^t v_m$
> **return** $u_m^t = \sum_{k=1}^{K} \hat{\alpha}_{mk}^t w_k^t$.

---

the client's model on proxy data, i.e., $|F(w_k^{t-1}; D_k) - F(v_m^t; D_k)|$, or the l_2 distance between the models, i.e., $\left\| v_m^t - w_k^{t-1} \right\|$. Leveraging these metrics, we propose `DistributionEstimation` in Algorithm 2 to estimate the importance weights of P_m^t as $\hat{\alpha}_{m1}^t, \ldots, \hat{\alpha}_{mK}^t$.

Cluster Updating. The server updates the model of each cluster k as follows,

$$w_k^t = (1 - \beta_{mk}^t) w_k^{t-1} + \beta_{mk}^t v_m^t, \tag{5}$$

where β_{mk}^t is the updating ratio contributed by client m to cluster k at epoch t. The value of β_{mk}^t depends on 1) the correlation between v_m^t and w_k^{t-1} as measured by the weight $\hat{\alpha}_{mk}^t$ evaluated for distribution estimation; and 2) the staleness of v_m^t indicated by the epoch index τ in which client m's model is lastly updated. Detailed procedures for computing the updating ratio are elucidated in `UpdateRatioCompute` in Algorithm 2.

4.3 Convergence Analysis

To assist the convergence analysis of CDFL, we make the following assumptions that are standard in analyses of asynchronous and clustered FL algorithms (see, e.g., Xie et al. (2019); Ghosh et al. (2020); Ruan and Joe-Wong (2022)).

Assumption 1. F_k is L_k-smooth and μ_k-strongly convex and for some $L_k, \mu_k > 0$, for all $k \in [K]$.

Assumption 2. Each client executes at least H_{min} and at most H_{max} local model updates before uploadin to server.

Assumption 3. In (4), we simplify v_m^t as v denoting local model of client m at current epoch t, and u_m^τ as u denoting the latest model received from server. Let $f_m^t(v) \triangleq$

$\frac{1}{n_m^t}\mathbb{E}_{(x^i,y^i)\sim P_m^t}[\sum_{i=1}^{n_m^t} l(v;x^i,y^i)]$, then $J_m^t(v;u) = f_m^t(v) + \frac{\rho}{2}\|v-u\|^2$. We assume $\forall m$, and $\forall t \in T$, we have $\|\nabla f_m^t(v)\|^2 \leq V_1$ and $\|\nabla J_m^t(v;u)\|^2 \leq V_2$, for some constants V_1 and V_2.

Assumption 4. The distance between optimal models of different clusters is bounded as $a_0\Delta \leq \|w_k^* - w_{k'}^*\| \leq \Delta$, for some constant Δ, and $0 \leq a_0 \leq 1$, for all $k \neq k'$.

Assumption 5. The l_2 norm of cluster k's model w_k, $\forall k \in [K]$, is bounded as $\|w_k\| \leq a_k\Delta$, for some $a_k > 0$.

Theorem 1. *For a client with model v and a cluster k, we consider consecutive S_k epochs, such that in each epoch the data of the client contains an non-zero component of P_k, and the client and cluster k updates v and w_k respectively as in Algorithm 1, then with the above assumptions, choosing $\rho \geq \frac{2V_1 + \frac{1}{2}\|v-u\|^2 + \sqrt{4\|v-u\|^2(1+V_1)\epsilon}}{2\|v-u\|^2}$ for all possible v and u, and a small constant $\epsilon > 0$, we have*

$$\mathbb{E}[\|\nabla F_k(v)\|^2] \leq \frac{\mathbb{E}[F_k(w_k^0) - F_k(w_k^{S_k})]}{\beta_0\gamma\epsilon S_k H_{min}} + \frac{\left(\frac{L_k}{2} + \rho H_{max} + \frac{\rho H_{max}^2}{2}\right)\gamma H_{max}V_2}{\epsilon H_{min}}$$

$$+ \frac{\sqrt{V_1}\left(2\sum_{i=1}^K a_i + (2K+1)a_k + K\right)\Delta}{\gamma\epsilon H_{min}} + \frac{\left(\frac{L_k}{2} + \rho\right)\left(2\sum_{i=1}^K a_i + (2K+1)a_k + K\right)^2\Delta^2}{\gamma\epsilon H_{min}},$$

where w_k^0 and $w_k^{S_k}$ denotes cluster k's initial model and the model after S_k updates respectively.

Proof. See proof in Appendix B. □

This theorem demonstrates the convergence of a client's model on the loss function of a single cluster k. The upper bound on the gradient norm increases with H_{max} and Δ, and decreases with H_{min}. Intuitively, a larger H_{min} indicates a more sufficient local training, and a smaller H_{max} reduces the effect of overfitting; Δ represents the distance between different clusters, and larger distance leads to slower convergence. Since a client's local loss is a convex combination of the cluster losses, and the above theorem holds true for each cluster, it implies a good performance on the client's local task.

Algorithm 2: DistributionEstimation & UpdateRatioCompute

Function DistributionEstimation($v_m, w_1, \ldots, w_K, D_1, \ldots, D_K, t$):

    foreach $k \in [K]$ **do**

        $l_k \leftarrow F(v_m; D_k)$; $d_{1k} \leftarrow |F(w_k; D_k) - l_k|$; $d_{2k} \leftarrow \|v_m - w_k\|$

        $l_k \leftarrow l_k - l_{bar}$; $d_{1k} \leftarrow d_{1k} - d_{1bar}$; $d_{2k} \leftarrow d_{2k} - d_{2bar}$

        /* l_{bar}, d_{1bar}, d_{2bar} are hyper-parameters to control the scale.   */

        $\hat{\alpha}_{mk}^t \leftarrow \frac{1}{K-1} \cdot \left(c_1 \cdot \frac{\sum_{i\neq k} l_i}{\sum_i l_i} + c_2 \cdot \frac{\sum_{i\neq k} d_{1i}}{\sum_i d_{1i}} + (1-c_1-c_2) \cdot \frac{\sum_{i\neq k} d_{2i}}{\sum_i d_{2i}}\right)$

        /* $c_1, c_2 \in [0,1]$ are hyper-parameters. $c_1 + c_2 \in [0,1]$.   */

    $\hat{\alpha}_{m1}^t, \ldots, \hat{\alpha}_{mK}^t \leftarrow \text{softmax}(\hat{\alpha}_{m1}^t \cdot A, ..., \hat{\alpha}_{mK}^t \cdot A)$

    /* $A > 0$ is the hyper-parameter to magnify the difference of

        estimation results of clusters. Estimated weights $\hat{\alpha}_{mk}^t \in [0,1]$ for

        $k \in [K]$, and $\sum_{k=1}^K \hat{\alpha}_{mk}^t = 1$.   */

Function UpdateRaTioCompute($\hat{\alpha}_{m1}^t, \ldots, \hat{\alpha}_{mK}^t, \beta_0, \tau, t$):

    foreach $k \in [K]$ **do**

        $\beta_{11}, \ldots, \beta_{1K} \leftarrow \hat{\alpha}_{m1}^t, \ldots, \hat{\alpha}_{mK}^t$; $\beta_{1max} \leftarrow \max(\beta_{1k})$.

        if $\beta_{1k} < \beta_{1bar}$ **then** $\beta_{1k} \leftarrow 0$; **else then** $\beta_{1k} \leftarrow \beta_{1k}/\beta_{1max}$; $e_k \leftarrow t$

        /* β_{1bar} is a preset threshold. Do not update cluster k when

            $\beta_{1k} < \beta_{1bar}$. e_k is the last epoch when cluster k is updated.   */

        if $e_k - \tau < b$ **then** $\beta_{2k} \leftarrow 1$; **else then** $\beta_{2k} \leftarrow 1/(a(e_k - \tau) + 1)$

        /* a, b are hyper-parameters to control staleness.   */

        $\beta_{mk}^t \leftarrow \beta_0 \cdot \beta_{1k}\beta_{2k}$ /* $\beta_{mk}^t \in [0,\beta_0]$; β_0 governs the maximal local model

           modification to the cluster model.   */

    return $\beta_{m1}^t, ..., \beta_{mK}^t$

---

5 Experiments

5.1 Setup

We create FL clustered datasets via three public datasets: FashionMNIST (Xiao et al., 2017), CIFAR-100 (Krizhevsky et al., 2009), MiniImageNet-100 (Vinyals et al., 2016). In order to simulate different distributions, we augment the datasets using rotation, and create the Rotated FashionMNIST, Rotated CIFAR-100 and Rotated MiniImagenet-100 datasets. **Rotated FashionMNIST**: each cluster has 60,000 training images and 10,000 testing images with 10 classes; **Rotated CIFAR-100**: each cluster has 50,000 training images and 10,000 testing images with 100 classes; **Rotated MiniImagenet-100**: each cluster has 48,000 training images and 12,000 testing images with 100 classes. Each dataset is applied by $i * \frac{360}{K} (i = 0, ..., K - 1)$ degrees of rotation to the images, resulting in K clusters. We conduct experiments on various values of $K = 2, 3, 4, 6$.

We also perform a practical digit recognition task on three distinct datasets: the hand-written dataset composing of MNIST(LeCun et al., 2010) and USPS(Hull, 1994); the Street View House Numbers (SVHN(Netzer et al., 2011)); and the lisence plate numbers collected from CCPD(Li et al., 2020). The three datasets correspond to three clusters (or three different data distributions) in our setting. Consider an autonomous driving application where the client is a smart car that needs to recognize digits with its model on board, as the car traveling to different areas, the proportion of the digits from different types of images (digits from car plates or from house numbers) also changes, which is modeled by changing of mixing ratio of the three datasets in our experiment. More details about the employed training datasets and models are provided in Appendix A.

Baselines. 1) `FedSoft-Async`. An asynchronous adaptation of the soft-clustering baseline Ruan and Joe-Wong (2022). Clients receive all cluster models from the server, and performs distribution estimation locally through identifying the model with the smallest loss on each data point. The estimated importance weights $\hat{\alpha}_{m1}, \ldots, \hat{\alpha}_{mK}$ are sent to the server alongside the local model for global updates. The cluster models' updates are performed similarly as in `CDFL` using the received importance weights. As there are no existing works addressing both asynchrony and soft-clustering concurrently in FL, `FedSoft-Async` serves as the most suitable baseline method; and 2) `Local`. The clients only perform local optimizations and never interact with the server.

In the initialization phase, clients perform computations using the averaged cluster model. Each client possesses a dataset ranging from 500 to 2000 data points, with 40% to 90% originating from a primary distribution and the remainder from other clusters. After initialization, clients autonomously and randomly decide when to update their models. Each upload-download cycle prompts a client to receive new data, necessitating further model updates. In the experiments presented in Table 1, the number of clients is 20 times the number of clusters. The value of staleness threshold τ_0 is set to the number of clients. In the FashionMNIST experiments, on average, each client undergoes 25 model updates; and 20 updates in the CIFAR-100 and MiniImagenet-100 experiments. In Digit Recognition experiment, we set the client number as 100 and on average, each client undergoes 40 model updates. For each cluster in above experiments, we randomly choose 3000 samples from the training set to pre-train cluster models and 1000 samples from the test dataset as the proxy dataset on server. We set parameters in Algorithm 2 as $\beta_0 = 0.025$, $a = 10$, and $b = 5$, and the regularization parameter in (4) as $\rho = 0.1$. Values for other hyper-parameters are given in Appendix A.

All experiments are conducted on a single machine with two Intel Xeon 6226R CPUs, 384GB of memory, and four NVIDIA 3090 GPUs. Each experiment is repeated 5 times, and the average values and the standard deviations are reported.

5.2 Results

Accuracy. Average accuracy results across clients and clusters are shown in Table 1. Notably, both `CDFL` and `FedSoft-Async` exhibit significant enhancements in client performance compared to only local training, underscoring the importance of client collaboration. Across most experiments, `CDFL` outperforms `FedSoft-Async` for both clients and clusters, particularly for larger K.

We also plot the client and cluster accuracies over time in Figure 3. In FashionMNIST experiments, both `CDFL` and `FedSoft-Async` require a few sessions (each client has updated its model at least once in each session) for the downloaded model to surpass the performance of their locally trained counterparts. This may attribute to the relatively poor performance of the cluster models in the initial
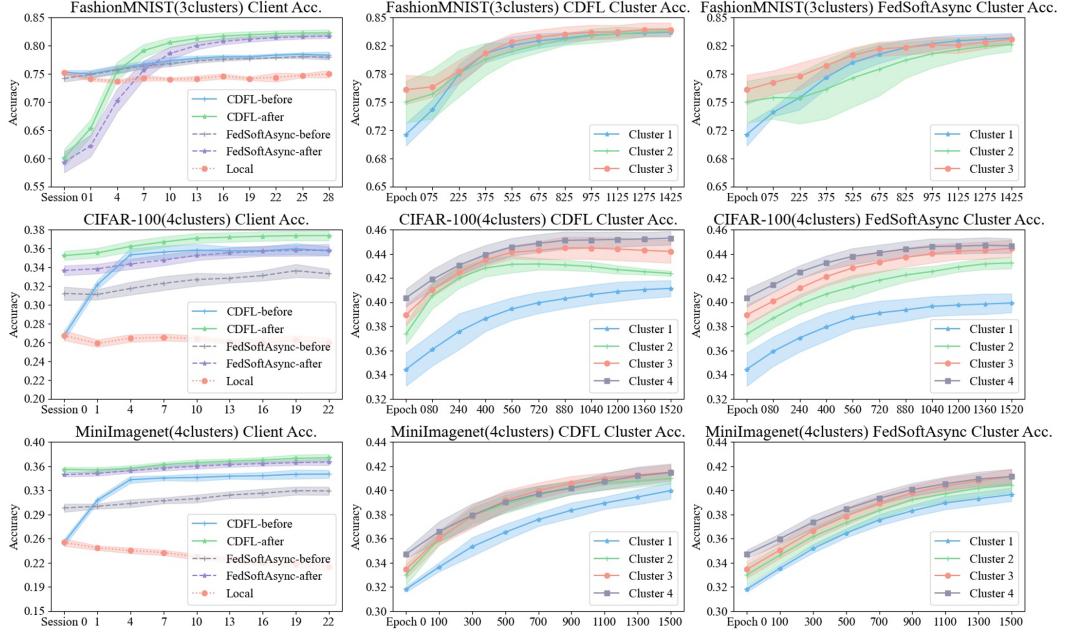
Figure 3: Average accuracy of clients and clusters over time. For client accuracy, in each session all the clients have updated their models for at least once; for cluster accuracy, exact one client updates its model with the sever in each epoch.

epochs. As the accuracies of cluster models improve, the benefit of model updating starts to prevail. Nevertheless, this phenomenon is not observed in CIFAR-100 and MiniImagenet experiments, and model updating with the server helps to boost clients' performance from very early epochs. Additional experiment results on different numbers of clusters can be found in Appendix C. The results for Digit Recognition Task are shown in Table 1 and in Figure 4. We observe that for both `CDFL` and `FedSoft-Async`, client accuracy before and after model updates have increased. This improvement suggests that clients are receiving better models from the server, enhancing the performance of their local models. While the difference on client accuracy between the proposed algorithm and the baseline is small, `CDFL` has much higher cluster accuracy than the baseline method.

Figure 4: Average accuracy of clients and clusters on Digit Recognition task over time.

Distribution Estimation. To assess the proposed distribution estimation method in Algorithm 2, we empirically compare the estimation outcomes of `CDFL` and those of `FedSoft-Async`. To quantify this assessment, we employ the KL-divergence metric $KL(P||Q)$, which measures the information loss when approximating the true distribution P with the estimated distribution Q. Lower KL divergence value implies more accurate estimation. As shown in Figure 5(b), over all datasets and cluster numbers, `CDFL` constantly outperforms `FedSoft-Async`.

Communication and Computation Overhead. We compare the communication and computation overheads between `FedSoft-Async` and `CDFL` in Figure 6. Specifically, we focus on download communication overhead, as both methods upload one local model to the server. We normalize both the communication and computation overhead of `CDFL` to 1, and measure the relative values of `FedSoft-Async`. Due to the fact that clients in `CDFL` solely download a single aggregated model, and do not need additional computations beyond local model training, the communication and computation overhead is significantly reduced compared to `FedSoft-Async`.

(a) M-I(6clusters) Estimation of Distribution

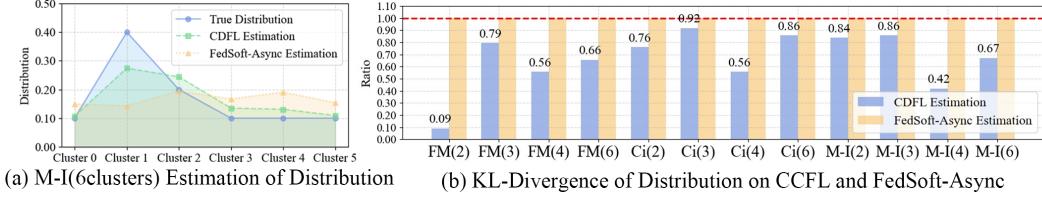(b) KL-Divergence of Distribution on CCFL and FedSoft-Async

Figure 5: (a) Estimated distributions in the MiniImagenet (6 clusters) experiment; (b) KL-divergence between the true distribution and the distribution estimated by CDFL and FedSoft-Async. FM(k) denotes FashionMNIST (k clusters), Ci as CIFAR-100, M-I as MiniImagenet-100.

Table 1: Average client and cluster accuracy of FashionMNIST, CIFAR100, and MiniImagenet-100 after final epoch. "Cli Brf" and "Cli Aft" denote the accuracy of client's model before and after uploading. Average accuracy and standard deviation over 5 trials are reported.

| Dataset (Cluster No.) | CDFL ACC. | | | FedSoft-Async ACC. | | | Local |
	Cli Bfr	Cli Aft	Cluster	Cli Bfr	Cli Aft	Cluster	Client ACC.
FashionMNIST (2)	.799±.011	.834±.003	.838±.007	.798±.012	.836±.003	.833±.008	.781±.019
FashionMNIST (3)	.783±.015	.823±.003	.835±.003	.780±.014	.818±.003	.823±.006	.746±.053
FashionMNIST (4)	.765±.020	.800±.006	.829±.005	.759±.021	.786±.007	.800±.017	.694±.072
FashionMNIST (6)	.768±.020	.788±.018	.818±.010	.760±.024	.761±.023	.769±.051	.697±.074
CIFAR-100 (2)	.370±.022	.392±.007	.417±.015	.369±.025	.397±.004	.416±.011	.280±.029
CIFAR-100 (3)	.284±.033	.303±.029	.362±.032	.274±.031	.295±.008	.348±.024	.208±.033
CIFAR-100 (4)	.360±.029	.376±.012	.432±.019	.337±.037	.361±.015	.430±.022	.261±.034
CIFAR-100 (6)	.293±.033	.304±.008	.376±.019	.267±.043	.392±.017	.371±.035	.214±.038
MiniImagenet (2)	.342±.020	.369±.003	.385±.006	.340±.026	.374±.003	.388±.002	.225±.030
MiniImagenet (3)	.291±.029	.314±.018	.359±.008	.276±.033	.308±.011	.353±.004	.180±.028
MiniImagenet (4)	.351±.026	.376±.014	.411±.007	.328±.035	.371±.009	.407±.008	.219±.029
MiniImagenet (6)	.309±.029	.334±.007	.386±.011	.280±.038	.323±.011	.380±.013	.192±.029
DigitRecognition	.835±.077	.883±.056	.890±.130	.846±.070	.890±.052	.870±.102	.820±.083

5.3 Ablation Study

Size of Proxy Dataset. We evaluate the performance of CDFL for various sizes of public datasets on FashionMNIST(4 clusters) and CIFAR100(4clusters) in Figure 7, ranging from 500 to 4000 samples. In those experiments, although the performance difference for using smaller sizes of proxy datasets has relatively lower performances, but the difference is negligible. We demonstrate that even a proxy dataset size of 500 samples (**approximately 0.8% of the training data**) can yield relatively favorable performance outcomes. Therefore, we consider the inclusion of a small but available proxy dataset to be a reasonable and beneficial practice in machine learning research. Other ablation studies on different values of ρ, a, b, and different number of clients can be found in Appendix C.1.
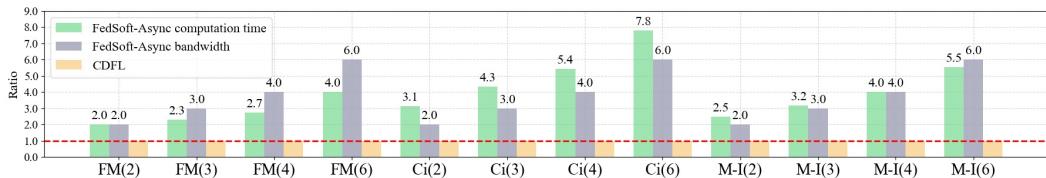

Figure 6: Relative communication and computation overheads of FedSoft-Async with CDFL.
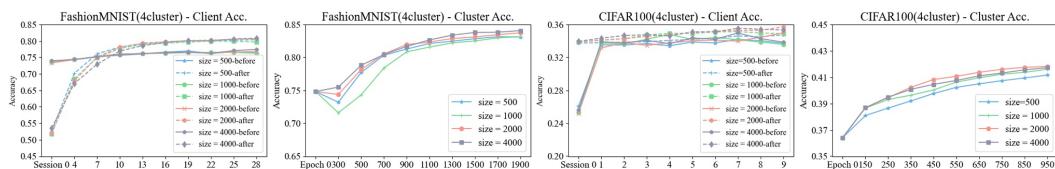

Figure 7: Average client and cluster accuracy for different sizes of proxy datasets at the server.

9

6   Conclusion

We introduce Client-Driven Federated Learning (CDFL), a novel FL paradigm that emphasizes on clients' autonomy, performance, and efficiency. In CDFL, a client autonomously decides when to update its model with the server, to accommodate potential distribution shifts. CDFL proposes novel distribution estimation strategies, for the server who hosts multiple cluster models to accurately estimate the distribution of the client, facilitating a rapid adaptation of the client's model to the new tasks. We theoretically and experimentally demonstrate the superiority of CDFL over baselines.

References

Briggs, C., Fan, Z., Andras, P., 2020. Federated learning with hierarchical clustering of local updates to improve training on non-iid data, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–9.

Chen, Y., Ning, Y., Slawski, M., Rangwala, H., 2020. Asynchronous online federated learning for edge devices with non-iid data, in: 2020 IEEE International Conference on Big Data (Big Data), IEEE. pp. 15–24.

Damaskinos, G., Guerraoui, R., Kermarrec, A.M., Nitu, V., Patra, R., Taiani, F., 2022. Fleet: Online federated learning via staleness awareness and performance prediction. ACM Transactions on Intelligent Systems and Technology (TIST) 13, 1–30.

Deng, Y., Kamani, M.M., Mahdavi, M., 2020. Adaptive personalized federated learning. arXiv preprint arXiv:2003.13461 .

Duan, M., Liu, D., Ji, X., Liu, R., Liang, L., Chen, X., Tan, Y., 2021a. 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), IEEE. pp. 228–237.

Duan, M., Liu, D., Ji, X., Wu, Y., Liang, L., Chen, X., Tan, Y., Ren, A., 2021b. Flexible clustered federated learning for client-level data distribution shift. IEEE Transactions on Parallel and Distributed Systems 33, 2661–2674.

Ganguly, B., Hosseinalipour, S., Kim, K.T., Brinton, C.G., Aggarwal, V., Love, D.J., Chiang, M., 2023. Multi-edge server-assisted dynamic federated learning with an optimized floating aggregation point. IEEE/ACM Transactions on Networking .

Gauthier, F., Gogineni, V.C., Werner, S., Huang, Y.F., Kuh, A., 2023. Asynchronous online federated learning with reduced communication requirements. IEEE Internet of Things Journal .

Ghosh, A., Chung, J., Yin, D., Ramchandran, K., 2020. An efficient framework for clustered federated learning. Advances in Neural Information Processing Systems 33, 19586–19597.

Ghosh, A., Mazumdar, A., et al., 2022. An improved algorithm for clustered federated learning. arXiv preprint arXiv:2210.11538 .

Gogineni, V.C., Werner, S., Huang, Y.F., Kuh, A., 2022. Communication-efficient online federated learning strategies for kernel regression. IEEE Internet of Things Journal 10, 4531–4544.

Hull, J.J., 1994. Usps handwritten digits database. `https://cs.nyu.edu/~roweis/data/usps_all.mat`.

Jiang, Z., Xu, Y., Xu, H., Wang, Z., Liu, J., Chen, Q., Qiao, C., 2023. Computation and communication efficient federated learning with adaptive model pruning. IEEE Transactions on Mobile Computing .

Khan, A.F., Wang, X., Le, Q., Khan, A.A., Ali, H., Ding, J., Butt, A., Anwar, A., 2023. Pi-fl: Personalized and incentivized federated learning. arXiv preprint arXiv:2304.07514 .

Krizhevsky, A., Hinton, G., et al., 2009. Learning multiple layers of features from tiny images .

LeCun, Y., Cortes, C., Burges, C.J., 2010. Mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

Li, C., Li, G., Varshney, P.K., 2021. Federated learning with soft clustering. IEEE Internet of Things Journal 9, 7773–7782.

Li, D., Wang, J., 2019. Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581 .

Li, Y., Yu, Q., Jin, H., Lu, X., 2020. Ccpd: Chinese city parking dataset. `https://github.com/detectRecog/CCPD`.

Lin, T., Kong, L., Stich, S.U., Jaggi, M., 2020. Ensemble distillation for robust model fusion in federated learning. Advances in Neural Information Processing Systems 33, 2351–2363.

Long, G., Xie, M., Shen, T., Zhou, T., Wang, X., Jiang, J., 2023. Multi-center federated learning: clients clustering for better personalization. World Wide Web 26, 481–500.

M Ghari, P., Shen, Y., 2022. Personalized online federated learning with multiple kernels. Advances in Neural Information Processing Systems 35, 33316–33329.

Ma, J., Long, G., Zhou, T., Jiang, J., Zhang, C., 2022. On the convergence of clustered federated learning. arXiv preprint arXiv:2202.06187 .

Mansour, Y., Mohri, M., Ro, J., Suresh, A.T., 2020. Three approaches for personalization with applications to federated learning. arXiv preprint arXiv:2002.10619 .

McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A., 2017. Communication-efficient learning of deep networks from decentralized data, in: Artificial intelligence and statistics, PMLR. pp. 1273–1282.

Mestoukirdi, M., Zecchin, M., Gesbert, D., Li, Q., 2023. User-centric federated learning: Trading off wireless resources for personalization. arXiv preprint arXiv:2304.12930 .

Mestoukirdi, M., Zecchin, M., Gesbert, D., Li, Q., Gresset, N., 2021. User-centric federated learning, in: 2021 IEEE Globecom Workshops (GC Wkshps), IEEE. pp. 1–6.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y., 2011. Street view house numbers (svhn) dataset. `http://ufldl.stanford.edu/housenumbers/`.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., Huba, D., 2022. Federated learning with buffered asynchronous aggregation, in: International Conference on Artificial Intelligence and Statistics, PMLR. pp. 3581–3607.

Park, J., Han, D.J., Choi, M., Moon, J., 2021. Sageflow: Robust federated learning against both stragglers and adversaries. Advances in neural information processing systems 34, 840–851.

Ruan, Y., Joe-Wong, C., 2022. Fedsoft: Soft clustered federated learning with proximal local updating, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 8124–8131.

Sattler, F., Müller, K.R., Samek, W., 2020a. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. IEEE transactions on neural networks and learning systems 32, 3710–3722.

Sattler, F., Müller, K.R., Wiegand, T., Samek, W., 2020b. On the byzantine robustness of clustered federated learning, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 8861–8865.

Tang, X., Guo, S., Guo, J., 2021. Personalized federated learning with clustered generalization .

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al., 2016. Matching networks for one shot learning. Advances in neural information processing systems 29.

Wang, H., Xiang, T., Guo, S., He, J., Liu, H., Zhang, T., 2024. Transtroj: Transferable backdoor attacks to pre-trained models via embedding indistinguishability. arXiv preprint arXiv:2401.15883 .

Wang, Q., Yang, Q., He, S., Shi, Z., Chen, J., 2022. Asyncfeded: Asynchronous federated learning with euclidean distance based adaptive weight aggregation. arXiv preprint arXiv:2205.13797 .

Wang, X., Wang, Y., 2022. Asynchronous hierarchical federated learning. arXiv preprint arXiv:2206.00054 .

Wu, W., He, L., Lin, W., Mao, R., Maple, C., Jarvis, S., 2020. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. IEEE Transactions on Computers 70, 655–668.

Xiao, H., Rasul, K., Vollgraf, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. `arXiv:cs.LG/1708.07747`.

Xie, C., Koyejo, S., Gupta, I., 2019. Asynchronous federated optimization. arXiv preprint arXiv:1903.03934 .

Xu, C., Qu, Y., Xiang, Y., Gao, L., 2021. Asynchronous federated learning on heterogeneous devices: A survey. arXiv preprint arXiv:2109.04269 .

Yoon, J., Jeong, W., Lee, G., Yang, E., Hwang, S.J., 2021. Federated continual learning with weighted inter-client transfer, in: International Conference on Machine Learning, PMLR. pp. 12073–12086.

Yu, H., Chen, Z., Zhang, X., Chen, X., Zhuang, F., Xiong, H., Cheng, X., 2021. Fedhar: Semi-supervised online learning for personalized federated human activity recognition. IEEE transactions on mobile computing 22, 3318–3332.

Zeng, D., Hu, X., Liu, S., Yu, Y., Wang, Q., Xu, Z., 2023. Stochastic clustered federated learning. arXiv preprint arXiv:2303.00897 .

Zhang, Y., Duan, M., Liu, D., Li, L., Ren, A., Chen, X., Tan, Y., Wang, C., 2021. Csafl: A clustered semi-asynchronous federated learning framework, in: 2021 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–10.

Zheng, J., Li, K., Mhaisen, N., Ni, W., Tovar, E., Guizani, M., 2023. Federated learning for online resource allocation in mobile edge computing: A deep reinforcement learning approach, in: 2023 IEEE Wireless Communications and Networking Conference (WCNC), IEEE. pp. 1–6.

A   Data Structures, Model Structures and Hyper-parameters

For the Digit Recognition experiment, the details are as below: **MNIST&USPS**: contains 76,000 training images and 14,000 testing images. **SVHN**: contains 73,257 training images and 26,032 testing images. Numbers from **CCPD**: contains 80,000 training images and 20,000 testing images. We normalize the all the images to the size of 32*32*3 and employ the CNN model to do the training. All the other settings are similar to those in FashionMNIST experiment. Examples from training data are shown in Figure 8.

We use CNN models to train FashionMNIST and Digit Recognition task and ResNet18 to train Cifar-100 and MiniImagenet-100. The model structures are as follows.

Images from Rotated FashionMNIST

Images from Digit Recognition Task

Figure 8: Images from conducted experiments.

- This CNN model consists of two convolutional layers with 5x5 kernels and ReLU activation, each followed by max-pooling layers with 2x2 kernels. This is followed by two fully-connected layers with 512 and 10 neurons, respectively. For the FashionMNIST task, the model takes 28x28 grayscale images as input; for the digit recognition task, the model takes 32*32*3 umages as input. The model produces class probabilities for 10 classes. It employs ReLU activation throughout.

- ResNet18: ResNet-18 is characterized by its residual blocks, and it's a smaller variant of the original ResNet architecture. The model consists of two types of residual blocks: BasicBlock, containing two convolutional layers with 3x3 kernels and ReLU activations, along with batch normalization. It also includes shortcut connections to handle different input and output dimensions when the stride is not equal to 1; BottleNeck, including three convolutional layers with 1x1, 3x3, and 1x1 kernels, along with ReLU activations and batch normalization and uses shortcut connections for dimension matching. The ResNet-18 architecture is structured around an initial convolutional layer with 64 output channels and a 3x3 kernel with padding set to 1, followed by four stages of residual blocks. Each stage contains a specific number of residual blocks, with the configuration typically set as [2, 2, 2, 2].Within each residual block, the block type can be either BasicBlock or BottleNeck. The number of output channels in the convolutional layers is typically set according to the block type: 64 for BasicBlock and 256 for BottleNeck. Stride values are often set to 1 for BasicBlock and 2 for BottleNeck to achieve downsampling. The expansion factor is 1 for BasicBlock and 4 for BottleNeck. The model uses adaptive average pooling to produce a fixed-sized output, typically (1, 1) spatial dimensions, and a fully-connected layer for classification, with the number of output classes of 100 for CIFAR-100 and activation functions serving as essential hyperparameters.

For all experiments, batch size is chosen as 128. For CNN model, Adam optimizer is chosen with weight decay of 0.005 and learning rate of 0.01. For ResNet models, SGD opitimizer is chosen with weight decay of 5e-4, momentum of 0.9 and learning rate as 0.1. Cross entropy loss is computed for optimization through the experiments. Distribution estimation and server updating parameters are listed as follows in Table 2.

In the experiment, each client is initially assigned a main cluster index before training. During the training's outset, every client is randomly provided with 500-2000 data samples, with 40%-90% sourced from the designated main cluster and the remainder from other cluster distributions. Clients conduct local training and autonomously decide when to upload their models. If a client initiates an upload session, the accuracy of the uploaded model is first evaluated on the test sets (consisting of the same distribution as their training sets), termed as "Client Before Accuracy." Subsequently, upon downloading the new personalized model, the accuracy of the received model is assessed as "Client After Accuracy." After each update of every single client, he would receive new data sampled as the
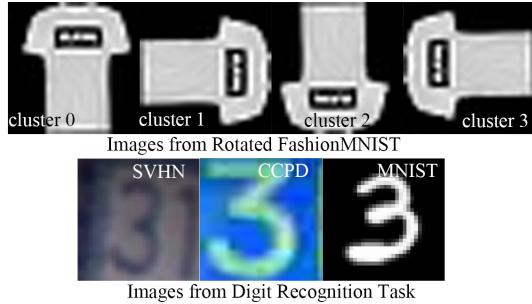
same way mentioned above. Since the distribution has changed, the client needs to do training on new data again and decides the next upload.

Table 2: Distribution estimation and server updating parameters. *min* means to use the minimum of the computed l_k (or d_{1k}, d_{2k}) as bar. *ave* means to use the average of the computed β_{1k} as bar. If multiple values are resented in A, it means to do *softmax* multiple times with respective given value.

Dataset (cluster No.)	Distribution Estimation						Server Updating
	c_1	c_2	A	l_{bar}	d_{1bar}	d_{2bar}	β_{1bar}
FashionMNIST (2)	0.5	0.4	3	8	0	0	ave
FashionMNIST (3)	0.5	0.25	3	8	0	0	ave
FashionMNIST (4)	0.5	0.25	7	8	0	0	ave
FashionMNIST (6)	0.7	0.2	15	8	0	0	ave
CIFAR-100 (2)	0.5	0.2	1	40	0	0	ave
CIFAR-100 (3)	0.5	0.2	10	min	min	min	ave
CIFAR-100 (4)	0.5	0.2	10	min	min	min	ave
CIFAR-100 (6)	0.5	0.2	10	min	min	min	ave
MiniImagenet (2)	0.5	0.2	1	70	0	0	ave
MiniImagenet (3)	0.5	0.4	7	min	min	min	ave
MiniImagenet (4)	0.6	0.3	15	min	min	min	ave
MiniImagenet (6)	0.6	0.3	10, 10	min	min	min	ave
DigitRecognition	0.9	0	5	min	min	min	ave

B Proof of Theorem 1

Without the loss of generality, we assume client m uploads (v_m^t, τ) to the server at epoch t. We assume the client is not stale ($t - \tau < \tau_0$), and cluster k's model w_k^{t-1} would be updated with parameter $\beta_{mk}^t > 0$. We assume v_m^t is the result of applying $H_{min} \leq H \leq H_{max}$ local updates to u_m^τ. We define $\mathbb{J}_k(v; u) = F_k(v) + \frac{\rho}{2} \|v - u\|^2$. For convenience we denote $v_{m,h}^t$ as v_h ($h \in [H]$), u_m^τ as u_τ, w_k as w, w_k^* as w^*.

Conditioning on v_{h-1}, for $\forall h \in [H]$ we have

$$\mathbb{E}[F_k(v_h) - F_k(w^*)] \leq \mathbb{E}[\mathbb{J}_k(v_h; u_\tau) - F_k(w^*)] \leq \mathbb{E}[\mathbb{J}_k(v_h; u_\tau)] - F_k(w^*) \tag{6}$$

With the updating function

$$v_h = v_{h-1} - \gamma \nabla J_m(v_{h-1}; u) \tag{7}$$

and Taloy's Expansion $f(w - y) = f(x) - \langle \nabla f(x), y \rangle + \frac{1}{2} \nabla^2 f(x) y^2$ and using L_k-smoothness,

$$\begin{aligned}
\mathbb{J}_k&(v_h; u_\tau) \\
&= \mathbb{J}_k \left(v_{h-1} - \gamma \nabla J_m(v_{h-1}; u_\tau); u_\tau \right) \\
&= \mathbb{J}_k(v_{h-1}; u_\tau) - \langle \nabla \mathbb{J}_k(v_{h-1}; u_\tau), \gamma \nabla J_m(v_{h-1}; u_\tau) \rangle \\
&\quad + \frac{1}{2} \nabla^2 \mathbb{J}_k(v_{h-1}; u_\tau) \gamma^2 \|\nabla J_m(v_{h-1}; u_\tau)\|^2 \\
&\leq \mathbb{J}_k(v_{h-1}; u_\tau) - \langle \nabla \mathbb{J}_k(v_{h-1}; u_\tau), \gamma \nabla J_m(v_{h-1}; u_\tau) \rangle + \frac{1}{2} L_k \gamma^2 \|\nabla J_m(v_{h-1}; u_\tau)\|^2
\end{aligned} \tag{8}$$

Since $\|v_{h-1} - u_\tau\|^2 \leq H_{max}^2 \gamma^2 V_2$ and $\|J_m(v_{h-1}; u_\tau)\|^2 \leq V_2$, with (8),

$$\begin{aligned}
\mathbb{E}[F_k&(v_h) - F_k(w^*)] \\
&\leq \mathbb{J}_k(v_{h-1}; u_\tau) - F_k(w^*) - \gamma \mathbb{E}[\langle \nabla \mathbb{J}_k(v_{h-1}; u_\tau), \nabla J_m(v_{h-1}; u_\tau) \rangle] \\
&\quad + \frac{1}{2} L_k \gamma^2 \mathbb{E}[\|\nabla J_m(v_{h-1}; u_\tau)\|^2] \\
&= F_k(v_{h-1}) - F_k(w^*) + \frac{\rho}{2} \|v_{h-1} - u_\tau\|^2 - \gamma \mathbb{E}[\langle \nabla \mathbb{J}_k(v_{h-1}; u_\tau), \nabla J_m(v_{h-1}; u_\tau) \rangle] \\
&\quad + \frac{1}{2} L_k \gamma^2 \mathbb{E}[\|\nabla J_m(v_{h-1}; u_\tau)\|^2] \\
&\leq F_k(v_{h-1}) - F_k(w^*) - \gamma \mathbb{E}[\langle \nabla \mathbb{J}_k(v_{h-1}; u_\tau), \nabla J_m(v_{h-1}; u_\tau) \rangle] + \frac{L_k \gamma^2}{2} V_2 + \frac{\rho H_{max}^2 \gamma^2}{2} V_2
\end{aligned} \tag{9}$$

Take a small constant $\epsilon > 0$, and with inequality of arithmetic and geometric mean, if we choose some $\rho \geq \frac{2V_1 + \frac{1}{2} \|v_{h-1} - u_\tau\|^2 + \sqrt{4\|v_{h-1} - u_\tau\|^2 (1 + V_1)\epsilon}}{2\|v_{h-1} - u_\tau\|^2}$ for all possible v_{h-1}, u_τ, we have

15

$$\langle \nabla \mathbb{J}_k(v_{h-1}; u_\tau), \nabla J_m(v_{h-1}; u_\tau) \rangle - \epsilon \|\nabla F_k(v_{h-1})\|^2$$

$$= \langle \nabla F_k(v_{h-1}) + \rho(v_{h-1} - u_\tau), \nabla f_m(v_{h-1}) + \rho(v_{h-1} - u_\tau) \rangle - \epsilon \|\nabla F_k(v_{h-1})\|^2$$

$$= \langle \nabla F_k(v_{h-1}), \nabla f_m(v_{h-1}) \rangle + \rho \langle \nabla F_k(v_{h-1}) + \nabla f_m(v_{h-1}), v_{h-1} - u_\tau \rangle$$

$$\qquad + \rho^2 \|v_{h-1} - u_\tau\|^2 - \epsilon \|\nabla F_k(v_{h-1})\|^2$$

$$\geq -\frac{1}{2} \|\nabla F_k(v_{h-1})\|^2 - \frac{1}{2} \|\nabla f_m(v_{h-1})\|^2 - \frac{\rho}{2} \|\nabla F_k(v_{h-1}) + f_m(v_{h-1})\|^2$$

$$\qquad - \frac{\rho}{2} \|v_{h-1} - u_\tau\|^2 + \rho^2 \|v_{h-1} - u_\tau\|^2 - \epsilon \|\nabla F_k(v_{h-1})\|^2$$

$$\geq -\frac{1}{2} \|\nabla F_k(v_{h-1})\|^2 - \frac{1}{2} \|\nabla f_m(v_{h-1})\|^2 - \rho \|\nabla F_k(v_{h-1})\|^2 - \rho \|\nabla f_m(v_{h-1})\|^2 \qquad (10)$$

$$\qquad - \frac{\rho}{2} \|v_{h-1} - u_\tau\|^2 + \rho^2 \|v_{h-1} - u_\tau\|^2 - \epsilon \|\nabla F_k(v_{h-1})\|^2$$

$$= \|v_{h-1} - u_\tau\|^2 \rho^2 - \left(\|\nabla F_k(v_{h-1})\|^2 + \|\nabla f_m(v_{h-1})\|^2 + \frac{1}{2} \|v_{h-1} - u_\tau\|^2 \right) \rho$$

$$\qquad - \left(\frac{1}{2} \|\nabla F_k(v_{h-1})\|^2 + \frac{1}{2} \|\nabla f_m(v_{h-1})\|^2 + \epsilon \|\nabla F_k(v_{h-1})\|^2 \right)$$

$$\geq \|v_{h-1} - u_\tau\|^2 \rho^2 - \left(2V_1 + \frac{1}{2} \|v_{h-1} - u_\tau\|^2 \right) \rho - (1 + V_1)\epsilon \geq 0$$

Thus, we have

$$\gamma \langle \nabla \mathbb{J}_k(v_{h-1}; u_\tau), \nabla J_m(v_{h-1}; u_\tau) \rangle \geq \gamma \epsilon \|\nabla F_k(v_{h-1})\|^2 \qquad (11)$$

Taking (11) into (9), we have

$$\mathbb{E}[F_k(v_h) - F_k(w^*)] \leq F_k(v_{h-1}) - F_k(w^*) - \gamma \epsilon \|\nabla F_k(v_{h-1})\|^2 + \frac{L_k \gamma^2}{2} V_2 + \frac{\rho H_{max}^2 \gamma^2}{2} V_2 \quad (12)$$

By iterating (12) for $h = 0, ..., H - 1$, we have

$$\mathbb{E}[F_k(v_h) - F_k(v_0)] \leq -\gamma \epsilon \sum_{h=0}^{H-1} \|\nabla F_k(v_h)\|^2 + \frac{H_{max} L_k \gamma^2}{2} V_2 + \frac{\rho H_{max}^3 \gamma^2}{2} V_2 \qquad (13)$$

Since v_0 is initiated from u_τ, we can rewrite above equation as

$$\mathbb{E}[F_k(v_h) - F_k(u_\tau)] \leq -\gamma \epsilon \sum_{h=0}^{H-1} \|\nabla F_k(v_h)\|^2 + \frac{H_{max} L \gamma^2}{2} V_2 + \frac{\rho H_{max}^3 \gamma^2}{2} V_2 \qquad (14)$$

We know that cluster k is not updated in every iteration t. If we relabel the iterations when cluster k is updated as $0, 1, 2, ..., s - 1, s = t$, then we have $w_k^s = (1 - \beta_{mk}^s)w_k^{s-1} + \beta_{mk}^s v_m^s$. For simplicity, we denote w_k^s as w_s, β_{mk}^s as β_s, then

$$\begin{aligned}
&\mathbb{E}[F_k(w_s) - F_k(w_{s-1})] \\
&\leq \mathbb{E}[\mathbb{J}_k(w_s; w_{s-1}) - F_k(w_{s-1})] \\
&= \mathbb{E}[\mathbb{J}_k((1 - \beta_s)w_{s-1} + \beta_s v_m^s; w_{s-1}) - F_k(w_{s-1})] \\
&= \mathbb{E}[\mathbb{J}_k((1 - \beta_s)w_{s-1} + \beta_s v_h; w_{s-1}) - F_k(w_{s-1})] \\
&\leq \mathbb{E}[(1 - \beta_s)\mathbb{J}_k(w_{s-1}; w_{s-1}) + \beta_s \mathbb{J}_k(v_h; w_{s-1}) - F_k(w_{s-1})] \\
&= \mathbb{E}[\beta_s (F_k(v_h) - F(w_{s-1})) + \frac{\beta_s \rho}{2} \|v_h - w_{s-1}\|^2] \\
&\leq \beta_s \mathbb{E}[F_k(v_h) - F_k(w_{s-1})] + \beta_s \rho \|v_h - u_\tau\|^2 + \beta_s \rho \|u_\tau - w_{s-1}\|^2
\end{aligned} \qquad (15)$$

We have

$$
\begin{aligned}
&\|u_\tau - w_{s-1}\| \\
&= \left\|u_m^\tau - w_k^{s-1}\right\| \\
&= \left\|\sum_{i=1}^K \hat{\alpha}_{mi}^s w_i^\tau - w_k^{s-1}\right\| \\
&= \left\|\sum_{i=1}^K \hat{\alpha}_{mi}^s w_i^\tau - w_k^\tau + w_k^\tau - w_k^{s-1}\right\| \\
&= \left\|\sum_{i=1}^K \hat{\alpha}_{mi}^s \left(w_i^\tau - w_k^\tau\right) + \left(w_k^\tau - w_k^{s-1}\right)\right\| \\
&\leq \sum_{i=1}^K \hat{\alpha}_{mi}^s \left\|w_i^\tau - w_k^\tau\right\| + \left\|w_k^\tau - w_k^{s-1}\right\| \\
&\leq \sum_{i=1}^K \left\|w_i^\tau - w_k^\tau\right\| + \left\|w_k^\tau - w_k^{s-1}\right\|
\end{aligned}
\tag{16}
$$

We can notice that

$$
\begin{aligned}
&\|w_i^\tau - w_k^\tau\| \\
&= \|w_i^\tau - w_i^* + w_i^* - w_k^* + w_k^* - w_k^\tau\| \\
&\leq \|w_i^\tau - w_i^*\| + \|w_i^* - w_k^*\| + \|w_k^* - w_k^\tau\|
\end{aligned}
\tag{17}
$$

Since $\|w_k\| \leq a_k \Delta$, then $\|w_k - w_k^*\| \leq 2a_k \Delta$, thus

$$
\|w_i^\tau - w_k^\tau\| \leq 2a_i \Delta + \Delta + 2a_k \Delta = (2a_i + 2a_k + 1)\Delta
\tag{18}
$$

And,

$$
\begin{aligned}
\left\|w_k^\tau - w_k^{s-1}\right\| &= \left\|w_k^\tau - w_k^* + w_k^* - w_k^{s-1}\right\| \\
&\leq \left\|w_k^\tau - w_k^*\right\| + \left\|w^* - w_k^{s-1}\right\| \leq 2a_k \Delta + 2a_k \Delta = 4a_k \Delta
\end{aligned}
\tag{19}
$$

Thus,

$$
\|u_\tau - w_{s-1}\| \leq \left(2\sum_{i=1}^K a_i + (2K+1)a_k + K\right)\Delta
\tag{20}
$$

And with $\|v_h - u_\tau\|^2 \leq H_{max}^2 \gamma^2 V_2$, from (15) we have

$$
\begin{aligned}
&\mathbb{E}[F_k(w_s) - F_k(w_{s-1})] \\
&\leq \beta_s \mathbb{E}[F_k(v_h) - F_k(w_{s-1})] + \beta_s \rho H_{max}^2 \gamma^2 V_2 + \beta_s \rho \left(2\sum_{i=1}^K a_i + (2K+1)a_k + K\right)^2 \Delta^2 \\
&\leq \beta_s \mathbb{E}[F_k(v_h) - F_k(u_\tau) + F_k(u_\tau) - F_k(w_{s-1})] + \beta_s \rho H_{max}^2 \gamma^2 V_2 \\
&\quad + \beta_s \rho \left(2\sum_{i=1}^K a_i + (2K+1)a_k + K\right)^2 \Delta^2
\end{aligned}
\tag{21}
$$

Using L_k-smoothness, we have

17

$$\mathbb{E}[F_k(u_\tau) - F_k(w_{s-1})]$$

$$\leq \langle \nabla F_k(w_{s-1}), u_\tau - w_{s-1} \rangle + \frac{L_k}{2} \|u_\tau - w_{s-1}\|^2$$

$$\leq \|F_k(w_{s-1})\| \|u_\tau - w_{s-1}\| + \frac{L_k}{2} \|u_\tau - w_{s-1}\|^2 \tag{22}$$

$$\leq \sqrt{V_1} \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right) \Delta + \frac{L_k}{2} \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right)^2 \Delta^2$$

With the inequalities from (14), (22), we can rewrite (21) into

$$\mathbb{E}[F_k(w_s) - F_k(w_{s-1})]$$

$$\leq -\beta_s \gamma \epsilon \sum_{h=0}^{H-1} \|\nabla F_k(v_h)\|^2 + \frac{H_{max} L_k \gamma^2}{2} \beta_s V_2 + \frac{\rho H_{max}^3 \gamma^2}{2} \beta_s V_2$$

$$+ \beta_s \sqrt{V_1} \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right) \Delta + \frac{\beta_s L}{2} \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right)^2 \Delta^2$$

$$+ \beta_s \rho H_{max}^2 \gamma^2 V_2 + \beta_s \rho \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right)^2 \Delta^2$$

$$= -\beta_s \gamma \epsilon \sum_{h=0}^{H-1} \|\nabla F_k(v_h)\|^2 + \left(\frac{L_k}{2} + \rho H_{max} + \frac{\rho H_{max}^2}{2} \right) \gamma^2 H_{max} \beta_s V_2$$

$$+ \beta_s \sqrt{V_1} \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right) \Delta$$

$$+ \beta_s \left(\frac{L}{2} + \rho \right) \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right)^2 \Delta^2 \tag{23}$$

Denoting H_s as the number of the local iterations applied on client m before be uploads at epoch s, by rearranging terms in (23), we have

$$\sum_{h=0}^{H_s-1} \|\nabla F_k(v_h)\|^2$$

$$\leq \frac{\mathbb{E}[F_k(w_{s-1}) - F_k(w_s)]}{\beta_s \gamma \epsilon} + \frac{\left(\frac{L_k}{2} + \rho H_{max} + \frac{\rho H_{max}^2}{2} \right) \gamma H_{max} V_2}{\epsilon}$$

$$+ \frac{\sqrt{V_1} \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right) \Delta}{\gamma \epsilon} \tag{24}$$

$$+ \frac{\left(\frac{L}{2} + \rho \right) \left(2\sum_{i=1}^{K} a_i + (2K+1)a_k + K \right)^2 \Delta^2}{\gamma \epsilon}$$

Denote τ_s as if client m uploads at epoch s, the last time of client's communication (client last uploads at epoch τ_s before s), by taking total expectation, after S_k global epoch on cluster k, where S_k is the total number of validate updates on cluster k, we have

$$\mathbb{E}[\|\nabla F_k(v)\|^2]$$

$$= \mathbb{E}_{\tau_s, s\in\{0,...,T-1\}, h\in\{0,...,H'_t-1\}}[\|\nabla F(v_{\tau_s,h})\|^2]$$

$$= \frac{1}{\sum_{s=1}^{S_k} H_s} \sum_{s=1}^{S_k} \sum_{h=0}^{H_s-1} \|\nabla F_k(v_h)\|^2$$

$$\leq \frac{\mathbb{E}[F_k(w_k^0) - F_k(w_k^{S_k})]}{\beta_0 \gamma \epsilon S_k H_{min}} + \frac{\left(\frac{L}{2} + \rho H_{max} + \frac{\rho H_{max}^2}{2}\right) \gamma H_{max} V_2}{\epsilon H_{min}} \qquad (25)$$

$$+ \frac{\sqrt{V_1} \left(2 \sum_{i=1}^K a_i + (2K+1)a_k + K\right) \Delta}{\gamma \epsilon H_{min}}$$

$$+ \frac{\left(\frac{L_k}{2} + \rho\right) \left(2 \sum_{i=1}^K a_i + (2K+1)a_k + K\right)^2 \Delta^2}{\gamma \epsilon H_{min}}$$

C Additional Experiment Results

Figure 9: Timeflow accuracy of clients and clusters on FashionMNIST (1cluster). Average accuracy of clients is shown for equal times of upload-download cycles.
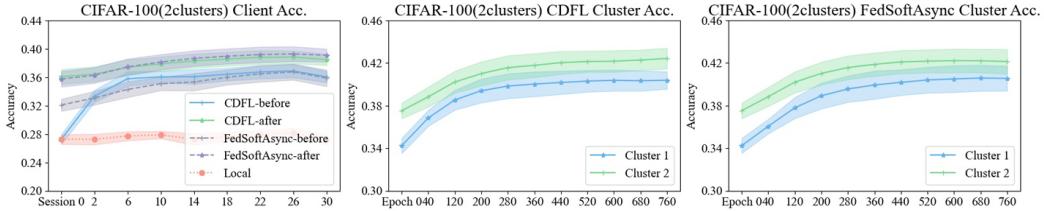

Figure 10: Timeflow accuracy of clients and clusters on FashionMNIST (2clusters). Average accuracy of clients is shown for equal times of upload-download cycles.
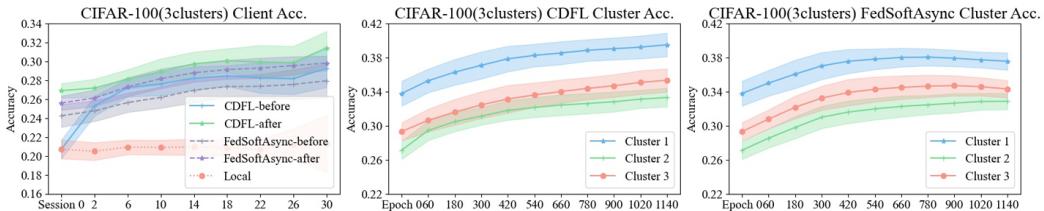

Figure 11: Timeflow accuracy of clients and clusters on FashionMNIST (4clusters). Average accuracy of clients is shown for equal times of upload-download cycles.

Figure 12: Timeflow accuracy of clients and clusters on FashionMNIST (6clusters). Average accuracy of clients is shown for equal times of upload-download cycles.

Figure 13: Timeflow accuracy of clients and clusters on CIFAR-100 (1cluster). Average accuracy of clients is shown for equal times of upload-download cycles.

Figure 14: Timeflow accuracy of clients and cluseters on CIFAR-100 (2clusters). Average accuracy of clients is shown for equal times of upload-download cycles.
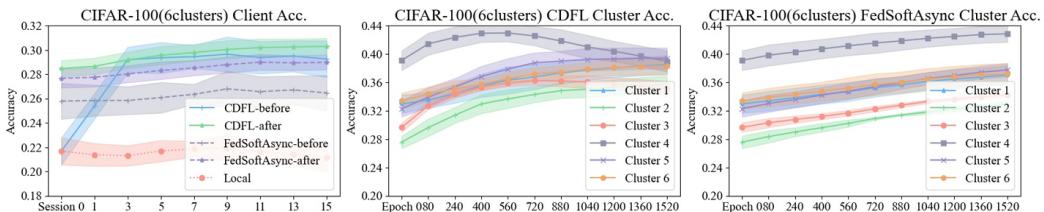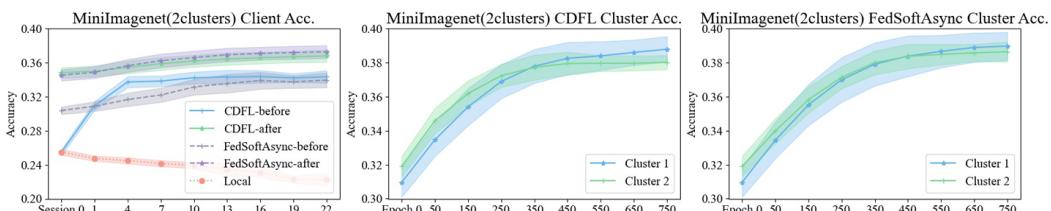

Figure 15: Timeflow accuracy of clients and clusters on CIFAR-100 (3clusters). Average accuracy of clients is shown for equal times of upload-download cycles.

Figure 16: Timeflow accuracy of clients and clusters on CIFAR-100 (6clusters). Average accuracy of clients is shown for equal times of upload-download cycles.

Figure 17: Timeflow accuracy of clients and clusters on MiniImagenet-100(2clusters). Average accuracy of clients is shown for equal times of upload-download cycles.
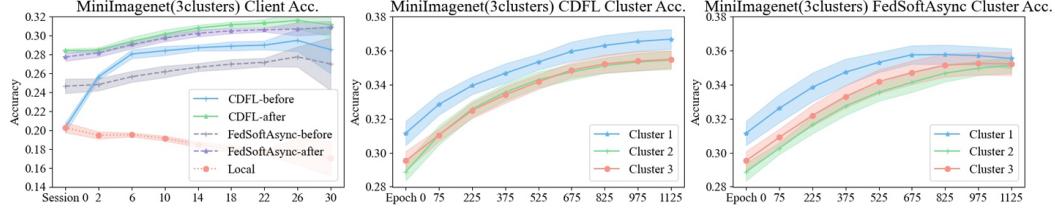
Figure 18: Timeflow accuracy of clients and clusters on MiniImagenet-100 (3clusters). Average accuracy of clients is shown for equal times of upload-download cycles.
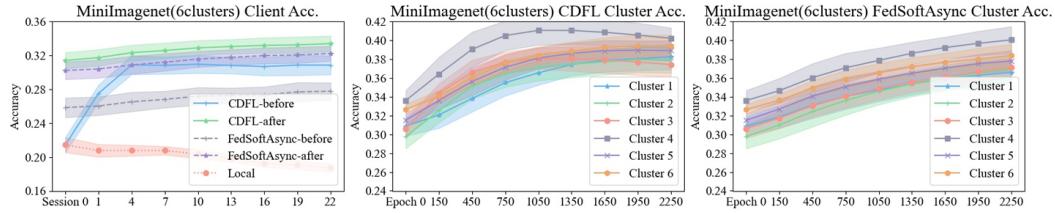

Figure 19: Timeflow accuracy of clients and clusters on MiniImagenet-100 (6clusters). Average accuracy of clients is shown for equal times of upload-download cycles.

C.1 More Ablation Studies

We further conduct ablation studies on different sets of hyper-parameters in experiments. All the ablation experiments are done within 4 clusters and 100 clients undergoing 2000 global epochs (FashionMNIST) and within 4 clusters and 100 clients undergoing 1000 global epochs (CIFAR-100).

Number of Clients. We conduct experiments with varying numbers of clients of 100, 250, 500, 1000 on FashionMNIST(4clusters) and CIFAR100(4clusters) and in Figure 20. Remarkably, the average accuracy of both clients and clusters exhibited minimal variation across different client counts. This observation underscores the robustness of our system.
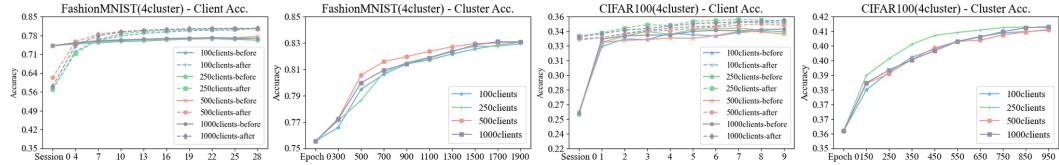

Figure 20: Client and Cluster Accuracies of Different Number of Clients on FashionMNIST(4 clusters) and CIFAR100(4 clusters)

Value of ρ. We experiment with the value of the ρ set to 0.01, 0.1, 0.5, and 1 on FashionM-NIST(4clusters) CIFAR100(4clusters) and in Figure 21. The results suggest that on both datasets, smaller ρ leads to better cluster accuracy. However, smaller ρ values, as observed in CIFAR-100 experiments, lead to reduced accuracy on clients' local models before uploading. This may result from the limited amount of local data, and an underutilization of the cluster models with a smaller weight on the regularization term.
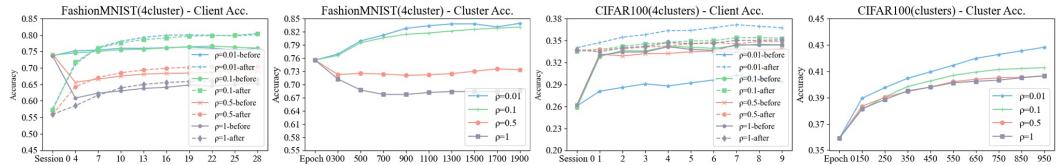

Figure 21: Client and Cluster Accuracies of Different ρ on FashionMNIST(4 clusters) and CI-FAR100(4 clusters)

Different a and b We have added experiments on different choice of a and b in Algorithm 2 on FashionMNIST(4clusters) and CIFAR100(4 clusters) and in Figure 22. From the experiment, we can see the influence of the different combination of a and b.
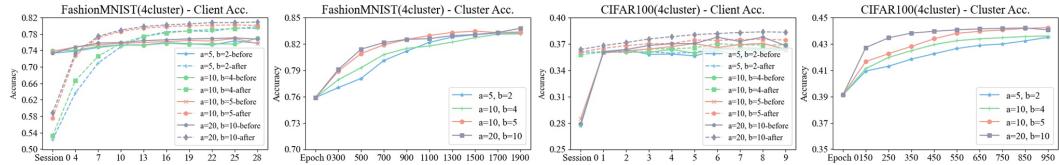

Figure 22: Client and Cluster Accuracies of Different a and b on FasionMNIST(4 clusters) and CIFAR100(4 clusters)