

Federated Learning Clients Clustering with Adaptation to Data Drifts

Minghao Li^{††}, Dmitrii Avdiukhin[†], Rana Shahout^{††}, Nikita Ivkin[¶],
 Vladimir Braverman[§], Minlan Yu^{††}
 Harvard University^{††}, Northwestern University[†], Amazon[¶]
 , Rice University[§]

ABSTRACT

Federated Learning (FL) enables deep learning model training across edge devices and protects user privacy by retaining raw data locally. Data heterogeneity in client distributions slows model convergence and leads to plateauing with reduced precision. Clustered FL solutions address this by grouping clients with statistically similar data and training models for each cluster. However, maintaining consistent client similarity within each group becomes challenging when data drifts occur, significantly impacting model accuracy. In this paper, we introduce Fielding, a clustered FL framework that handles data drifts promptly with low overheads. Fielding detects drifts on all clients and performs selective label distribution-based re-clustering to balance cluster optimality and model performance, remaining robust to malicious clients and varied heterogeneity degrees. Our evaluations show that Fielding improves model final accuracy by 1.9%-5.9% and reaches target accuracies 1.16×-2.61× faster.

1 INTRODUCTION

Federated Learning (FL) is a distributed machine learning paradigm that enables multiple clients to train one model collaboratively while retaining raw data locally [6, 35]. As FL circumvents the communication costs of data movement and mitigates privacy leakage [17, 32], it has drawn attention across various fields and seen adoption in applications such as Google Keyboard [46], energy consumption forecasting [1], and medical image processing [49].

The main challenge in federated learning is data heterogeneity. Since federated learning involves many devices of diverse settings, there exists substantial dataset size and distribution divergences between clients from distant geographical locations and with distinct personal habits [19, 29]. Such data heterogeneity leads to slow model convergence and plateauing with degraded precision [28, 40, 48]. A wide range of clustering mechanisms have been proposed to group clients with similar data distribution into more homogeneous clusters and train one model for each cluster [12, 14, 18, 30].

Data drift occurs frequently in federated learning systems. It is defined as temporal changes in the label distribution (i.e., the proportion of samples assigned to each label) of clients' local datasets. Such drifts arise from varying sampling rates, hardware configurations, sensor modalities, and limitations in client storage capacities [9, 34, 36]. Existing clustering mechanisms, however, are susceptible to data drift. Prior works address this issue but often either inadequately adjust clusters or incur prohibitive costs. IFCA and FlexCFL [12, 14] adjust clusters by relocating shifted clients while maintaining a fixed number of clusters, operating under the assumption of low drift magnitude. Auxo [30] re-clusters clients participating in the current training round using gradient information; however, since only a subset of clients is selected each round, it delays adjustments for drifted clients until they are chosen for training. FedDrift [18] re-clusters all clients and adaptively determines the number of clusters by transmitting all cluster models to every client each round to obtain training loss on local data, leading to significant communication and computation costs.

In this work, we propose **Fielding**, a novel clustering-based FL framework that retains cluster client similarity despite frequent data drifts. Fielding focuses on re-clustering clients as data drifts, dynamically adjusting the number of clusters while stabilizing responses to minor data drifts. Fielding combines per-client migration with selective global clustering to capture significant distribution patterns without compromising accuracy.

We assume the existence of a *coordinator*, typically deployed on a powerful central server to manage synchronization and advance global model training [4, 6]. Our solution handles all drifted clients by clustering them according to their local distribution vector, a piece of information readily available all the time. We present a theoretical framework supporting this approach. Fielding is designed to be lightweight; it introduces no additional computation on the client side and requires clients only to transmit their distribution vector (an array with up to a few thousand values) when they register with the coordinator.

We implement Fielding by extending the widely adopted FL engine FedScale [22] to support streaming data and our clustering mechanisms. We evaluated Fielding with up to 5078 clients on four image streaming traces with data drifts. Compared to existing approaches, Fielding improves model accuracy by 1.9% to 5.9% and achieves the target accuracy $1.16\times$ to $2.61\times$ faster. We also show that Fielding is compatible with various client selection and aggregation algorithms, and robust to malicious clients and varied heterogeneity degrees.

Our main contributions are:

- We introduce Fielding, a clustered FL framework that handles data drifts promptly with low overheads.
- We show that Fielding imposes minimal overhead by eliminating additional client-side computations and reducing communication costs through the use of readily available data, and we provide a theoretical framework to support our approach.
- We extend the FedScale FL engine to implement Fielding and demonstrate its efficacy on four real-world datasets with up to 5,078 clients, achieving up to 5.9% accuracy improvement and faster convergence compared to existing methods.
- We show that Fielding is compatible with various client selection and aggregation algorithms and is robust against malicious clients and different degrees of data heterogeneity.

2 BACKGROUND

In this section, we provide the background of clustered federated learning and the impacts of data drifts.

2.1 Clustered Federated Learning

FL deployment often involves hundreds to thousands of available clients participating in the training [6]. For example, Google sets 100 as the target number of clients for a round of training when improving keyboard search suggestions [46] and [16] have developed mortality and stay time predictor for the eICU collaborative research database [37] containing data of 208 hospitals and more than 200,000 patients. Although thousands of devices may be available in a given round, FL systems typically select a subset to participate in training. This selection process ensures a reasonable training time and accounts for diminishing returns where including more clients doesn't accelerate convergence [44].

Data heterogeneity is a major challenge in FL. Varying volumes and distribution of local data among participants is known to cause slow convergence and poor model accuracy [43, 45]. To mitigate the impact of heterogeneity, an effective approach is clustering, which groups statistically similar clients. Clients in the same cluster train one cluster

model collaboratively and also have their inference requests served by this model. Previous works such as Auxo [30], FlexCFL [12], FedDrift [18], and FedAC [47] have shown that clustering can achieve high accuracy and fast model convergence.

2.2 Data Drifts Affect Clustering

The key challenges for clustered federated learning are *data drifts*. Data drifts refer to the changes in the percentage of clients' local samples associated with a specific label. Drifts happen naturally when clients have access to non-stationary streaming data (e.g., cameras on vehicles and virtual keyboards on phones) [5, 20]. For example, when training an FL model for keyboard suggestions, drifts occur when many clients simultaneously start searching for a recent event (widespread drift), a few clients pick up a niche hobby (concentrated intense drift), or a new term appears when a product is launched (new label added).

Data drifts make clustered FL ineffective as it increases the data heterogeneity in the clusters. We demonstrate this problem with the Functional Map of the World (FMoW) dataset [10]. This dataset contains time-stamped satellite images labeled according to the function (e.g., recreational facility, crop field, etc.) of the captured land. We keep images taken on or after January 1, 2015, and create one client per unique UTM zone metadata value (302 clients in total). A day's worth of data becomes available every two training rounds, and clients maintain images they received over the last 100 rounds.

We use *mean client distance* as the intra-cluster heterogeneity metric [23]. For each client, we calculate the average pairwise L1 distance between its data distribution vector and those of all other clients in the same cluster. We then use the mean of these per-client distances to quantify the overall heterogeneity after clustering. Note that instead of first averaging within each cluster and then finding the mean of those cluster averages, we use the overall mean across all clients. This helps us avoid biased results arising from imbalanced cluster sizes. Note that in the baseline case without any clustering, we consider all clients as members of a "global cluster".

We cluster clients based on their label distribution vectors at the starting round and track the overall heterogeneity by measuring the mean client distance as described earlier. We cluster based on label distribution because data drifts occurs when label distribution changes. Figure 1 shows that at the beginning (e.g., round 1), clustering reduces the per-cluster heterogeneity compared with no clustering (i.e., putting all clients in a global set). However, as data distribution shifts

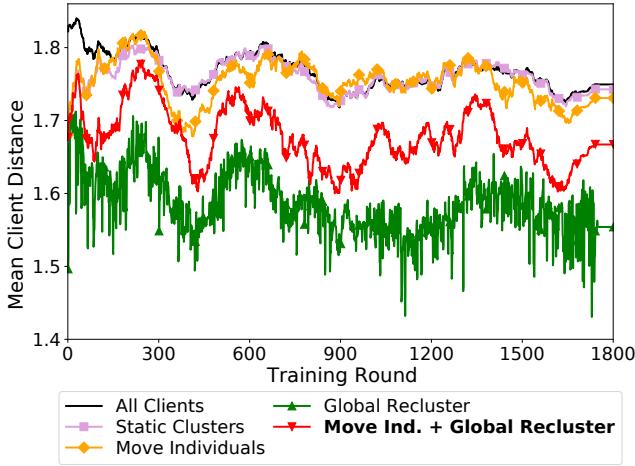


Figure 1: Client heterogeneity of the global set and clusters generated through label-based clustering.

over time, static clustering increases per-cluster heterogeneity and soon gets close to the no clustering case at round 322.

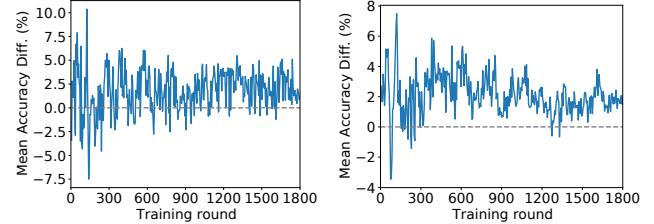
3 FIELDING DESIGN DECISIONS

Addressing data drifts is crucial for preserving the similarity of clients within a cluster. To handle data drifts, we must identify which clients should be moved, determine the appropriate clusters for these clients, and specify the information to collect for drift detection and client movements.

3.1 Combine Per-client Adjustment and Re-clustering

Existing clustered FL works rely on per-client adjustment or re-clustering to handle data drifts. However, neither approach works well on all levels of drifts.

Per-client adjustment fails in many-drift scenarios. Per-client adjustment keeps a fixed number of clusters and moves drifted clients individually to another cluster with similar gradients or model weights [12, 14]. However, in highly dynamic environments where many clients change their distributions at the same time [18], we have to move all these clients across clusters and the cluster membership changes dramatically. It is hard to decide which cluster the client should move to because the basis used by previous works to match client to the closest cluster (i.e., cluster’s average model weights or gradients) is changing [12, 14]. We demonstrate this problem in Figure 1 (“Move Individuals”). In rounds 1-40 and rounds 410-615, individual movements help keep the mean client distance below that of the global set even though clients are becoming more heterogeneous



(a) Selective global re-clustering vs. default global re-clustering. Positive numbers mean selective re-clustering is better.

(b) Global re-clustering vs. Selected re-clustering. Positive numbers mean re-clustering all drifted clients is better.

Figure 2: Accuracy difference between different re-clustering approaches.

(signaled by the increasing heterogeneity value). However, as the global set of all clients experiences major drifts during rounds 600-730 (signaled by the heterogeneity spike), many clients start moving across clusters simultaneously. Per-client adjustment then results in poor clusters and an overall heterogeneity value higher than the global set in rounds 740-1340.

Moreover, maintaining a fixed number of clusters can lead to sub-optimal clustering when new distribution patterns emerge during data drifts. For instance, we want to create another cluster when a new label appears for clients with a substantial number of samples of that label, but keeping the cluster number static will deem such creation infeasible.

Re-clustering makes model unstable during small drifts In Figure 1, we show an ideal case where we get optimal clusters by globally re-clustering all clients using label distribution vectors when any client drifts (“Global recluster”). Our implementation bounds the maximum number of clusters to 10, as previous works empirically show that having more clusters can negatively affect model convergence due to limited training resources [30]. As expected, this approach yields the lowest heterogeneity value. However, we will next show that such low heterogeneity does not necessarily lead to superior model accuracy.

Common clustering algorithms such as k-means have randomized initialization, leading to distinct results in multiple runs [2, 3]. This behavior is also observed in our experiment. In Figure 1, the “Global Recluster” curve exhibits more dramatic changes between rounds than other curves. Following the global re-clustering process, we need to determine the model each cluster should continue with. In our implementation, we align with previous works and use the average of the clients’ models within each cluster [31, 47]. When most clients remain in the same cluster, the new cluster models would be similar to the previous ones. But when there

are significant changes in the clusters, either due to large drifts or bad centroid initialization, the new models will differ substantially and require additional time to stabilize. This leads to sudden accuracy drops as the new clusters warm up. In Figure 2a, the x-axis denotes training rounds and the y-axis shows the mean accuracy we get with our selective global re-clustering (details in Section 3) subtracted by that of triggering global re-clustering by default. We observe that default global re-clustering leads to fluctuating test accuracy, which oscillates between meeting our approach’s accuracy and being as much as 5% worse.

Summary. Per-client adjustment fails in highly dynamic environments where many clients experience drifts. In contrast, global re-clustering may harm model accuracy because it increases the instability of cluster memberships and increases the model convergence times, especially during minor drifts. When cluster memberships remain largely stable, global re-clustering is not only excessive but also requires additional steps to warm up the new clusters (e.g., finding optimal cluster model parameters). On the other hand, when many clients experience drifts, moving individual clients can cause substantial shifts in cluster centers and miss the opportunity to create or remove clusters as distribution patterns evolve.

Based on the prior analysis, we opt for a design that starts with client individual migration and triggers global re-clustering selectively (Section 4 explains the detailed design). Our method is visualized as the red curve (“Move Ind.+Global Recluster”) in Figure 1. We retain a low heterogeneity value throughout training and achieve high model accuracy faster (Section 6).

3.2 Re-clustering All Drifted Clients

When we rerun the clustering algorithm every round, we can only easily recluster selected clients [30]. Since selected clients already get the gradients or model parameters during training, it is easy to compare with the existing clusters and decide whether they experience drifts and need change. In contrast, since unselected clients do not report gradients, we cannot decide whether and where to move them.

However, as mentioned in Section 2.1, pedagogical FL settings typically select only a subset of clients to contribute in each round. When unselected clients experience data drifts and are still stuck in their previous cluster, they might be served by the cluster model trained on data of a different distribution. We plot the difference in clients’ mean test accuracy of re-clustering only selected clients versus all clients that experience data drifts every round in Figure 2b. Re-clustering all drifted clients offers a 1.5%-4% higher accuracy in most rounds, which is a significant difference given that the final accuracy we achieve on FMoW is 52.4%. This trend

Table 1: Heterogeneity of all clients and intra-cluster heterogeneity after gradient-based clustering (lower is better). At the initial stages, gradient-based clustering doesn’t sufficiently decrease heterogeneity due to the model being not well-trained. In contrast, label-based clustering provides a consistent decrease in heterogeneity.

TOTAL ROUNDS	UN- CLUSTERED	GRADIENT-BASED CLUSTERING	LABEL-BASED CLUSTERING
100	1.81	1.82(+0.6%)	1.65(-8.8%)
200	1.80	1.82(+1.1%)	1.64(-8.9%)
500	1.78	1.71(-3.9%)	1.62(-9.0%)
1000	1.74	1.66(-4.6%)	1.56(-10.3%)
1500	1.76	1.63(-7.4%)	1.60(-9.1%)

demonstrates that only moving selected clients leads to imperfect clusters containing diverging but unselected clients.

Global re-clustering based on training outputs incur high overheads. Considering the suboptimal accuracy of only re-clustering the selected clients, our goal is to move all drifted clients promptly. One effective method to learn information from all clients for re-clustering is to train a global model with selected clients and then distribute it to all clients. For example, in FlexCFL, each client performs local training with the global model weights on their own data and reports gradients to the central server, where clustering happens [12]. The coordinator can then re-cluster all clients according to the gradients.

This approach incurs high communication and computing overheads. For example, in our experiments training ResNet-18 [15] for image classification tasks, devices on average take 116.7 seconds to download the 11 million parameters and upload gradient coordinates. They also need an additional 50.4 seconds to run forward and backward propagation.

3.3 Label-based Instead of Gradient-based Clustering

Another issue with gradient-based re-clustering is that the clustering effectiveness is sensitive to the model quality. Table 1 presents the resulting mean client distance when we perform gradient-based clustering after training a global model for various numbers of rounds. The numbers in parentheses indicate the change in mean client distance relative to that of the global set. A negative value in the parentheses indicates that the generated clusters are overall less heterogeneous than the global set. When we perform gradient-based clustering with the global model at round 100 and round 200, the resulting clusters are more heterogeneous. As we

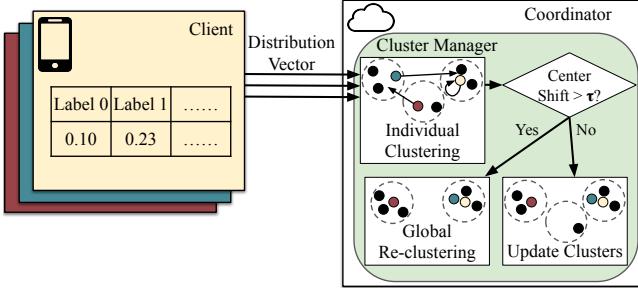


Figure 3: Clustering overview of Fielding. Clients send distribution vectors to the coordinator; the coordinator moves drifted clients to the closest cluster; global re-clustering is triggered if any cluster center shifts by a distance larger than τ .

postpone clustering to later rounds, gradient-based clustering achieves an increasingly larger reduction in mean client distance. This trend indicates that gradient-based clustering doesn't perform well in earlier rounds, when the global model used to collect gradients from all clients hasn't converged.

We propose using label distribution vectors for clustering (we refer to this approach as "label-based"). Label distribution vectors are available on all clients and enable us to promptly detect any drift by checking for distribution changes. Compared to gradient-based clustering, label-based doesn't demand parameter or gradient transmission, doesn't introduce additional computation tasks, and is not sensitive to the timing of client clustering. In Table 1, label-based clustering provides consistent heterogeneity reduction across rounds. Furthermore, as a label distribution vector typically has tens or hundreds of coordinates compared to millions in a full gradient, label-based clustering incurs significantly lower computational overhead. In our experiment of training ResNet-18 on the FMoW dataset, clustering all 302 clients using our label-based solution takes only 0.05 seconds while the gradient-based solution FlexCFL takes 37.92 seconds (note that FlexCFL already accelerates this process through dimensionality reduction using truncated Singular Value Decomposition).

4 FIELDING OVERVIEW

To improve clustered FL performance, we develop Fielding, a clustered FL framework that efficiently reassigns clients after data drifts. As explained in Section 3, Fielding has three key design decisions: (1) reassign all drifted clients instead of only selected clients, (2) combine per-client adjustment and global re-clustering, and (3) cluster clients based on label distribution vectors.

4.1 System Overview

As discussed in Section 3.1, neither per-client adjustment nor global re-clustering works well across all drift scenarios. Therefore, we propose a combined approach that leverages both methods. Fielding employs a threshold τ to determine whether global re-clustering is necessary. When clients report data drifts, we first try re-clustering drifted clients individually by moving them to the cluster whose center is the closest to their new distribution vector. To ensure determinism across different orders of client movements, we do not update the cluster center throughout this per-client adjustment process. After individual movements, we recalculate the cluster centers and trigger global re-clustering over all clients only when any center moves by a distance larger than τ . In our implementation, we set τ as one-third of the average distance between a pair of cluster centers, which is a value that works empirically well across evaluation settings.

Figure 3 illustrates the clustering mechanism of Fielding. When clients register with the centralized coordinator, they send their local data distribution vector, either during initial registration or following a recent data drift. Upon receiving this vector, the coordinator updates the client's metadata and assigns the client to the nearest cluster. Before selecting participants for the upcoming training round, the coordinator finds the distance each cluster center has shifted. It then initiates a global clustering process if any cluster exceeds the threshold τ . Otherwise, it finalizes the updated cluster membership.

Our clustering approach incurs additional communication costs, as clients are expected to upload their data distribution vector. However, this vector is small ($4 \text{ bytes} \times N$ where N is the number of labels), and clients only need to send it when they first join or experience data drift. In Section 4.3, we prove the model convergence guarantees with label-based clustering.

4.2 Algorithmic Framework

In this section, we describe our algorithm – presented in Algorithm 1 – in detail. Initially, Fielding clusters the clients using the k -means algorithm (Algorithm 3). Importantly, we measure distance between data distributions, which is computed as the ℓ_1 -distance between their histograms: that is, assuming we have l labels, the distance between clients with label distribution p_1, \dots, p_l and q_1, \dots, q_l is defined as $\sum_{i=1}^l |p_i - q_i|$.

We assume that the algorithm is executed for T global iterations, and during each iteration, the label distribution of each client might change, for instance by adding or removing data points. The first step of Fielding is to handle the data drifts (Algorithm 2). Our goal is, by the end of each iteration, to maintain a good model for each cluster, with respect to

Algorithm 1 Clustered Federated Learning with data drift

Partition clients into clusters using Algorithm 3
Initialize cluster models $\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(K)}$
for every iteration $t = 0, \dots, T - 1$ **do**
 Handle data drifts using Algorithm 2
 Let C_1, \dots, C_K be the clusters
 Let M be the number of machines sampled per iteration
 for every cluster $k = 1, \dots, K$ in parallel **do**
 for R rounds **do**
 Sample a set S of M/K clients from C_k based on the selection strategy
 for each client $i \in S$ in parallel **do**
 Initialize $\mathbf{x}_i = \mathbf{c}^{(k)}$
 for L local iterations **do**
 $\mathbf{x}_i \leftarrow \mathbf{x}_i - \eta G_t^{(i)}(\mathbf{x}_i)$
 $\mathbf{c}^{(k)} \leftarrow \text{avg}_{i \in S} \mathbf{x}_i$
 report cluster models and client-to-cluster assignment

Algorithm 2 Handling data drifts

for each drifting client x **do**
 Assign x to the closest cluster center
 Recompute the centers of the affected clusters
Let θ be the average distance between the cluster centers
if some center moved by at least $\theta/3$ **then**
 for each client i **do**
 Let \mathbf{x}_i be the model corresponding to the i 'th client's cluster
 Let C_1, \dots, C_K be clusters returned by Algorithm 3
 for each cluster C_k **do**
 $\mathbf{c}^{(k)} \leftarrow \text{avg}_{i \in C_k} \mathbf{x}_i$

Algorithm 3 Clustering

Define the distance between clients as the ℓ_1 -distance between their label histograms
Choose K with the largest silhouette score
Cluster the clients using the K -means clustering

its clients' data. In this step, each drifted client is assigned to the closest cluster. If this causes a significant change in cluster centers – namely, if there exists a cluster moving by more than $\tau/3$, where τ is the average distance between cluster centers – we use Algorithm 3 to recluster clients from scratch. This condition avoids frequent reclustering, which might require excessive resources and can adversely change the clients' losses. Importantly, we recluster all the clients, not just available clients, which, as we show in Section 2, improves the accuracy.

For the global reclustering, we choose the number of clusters that gives the highest silhouette score. After reclustering, we compute new cluster models: for each client i , let \mathbf{x}_i be its old cluster model, and then for each new cluster C_k , we define its model as an average of \mathbf{x}_i for $i \in C_k$. That is, we average the old client models.

After the clusters are computed, we train the cluster models for R rounds. During each round, we sample the clients from the cluster and run L local iterations of gradient descent. After the local iterations, we set the cluster model as the average of the models of all participating clients.

4.3 Convergence

In Appendix A, we analyze the performance of our framework by bounding the average of clients' loss functions at every iteration. Formally, for every cluster C_k , let $\mathbf{c}^{(k)}$ be the model corresponding to the cluster by the end of the iteration, and let $\mathbf{c}^{(k,*)}$ be the optimal model for this cluster, that is, the minimizer of $\sum_{i \in C_k} f_t^{(i)}$. Then, we bound the average loss of the clients with respect to the best possible cluster models:

$$\frac{1}{N} \sum_{k=1}^K \sum_{i \in C_k} (f_t^{(i)}(\mathbf{c}^{(k)}) - f_t^{(i)}(\mathbf{c}^{(k,*)})) \quad (1)$$

Our analysis has two ingredients, corresponding to two main stages in our algorithm: gradient descent steps and reclustering. For the gradient descent part, we use the standard convergence analysis of SGD, which for strongly convex function f shows that for stochastic gradient descent update the following holds:

$$\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] \leq (1 - \eta\mu)^T (f(\mathbf{x}_0) - f(\mathbf{x}^*)) + \frac{L\eta}{2\mu} \sigma^2$$

The right-hand side has two terms, where the first term decays exponentially, and the second term corresponds to the total contribution of the stochastic noise. In our analysis, we similarly have the $(f(\mathbf{x}_0) - f(\mathbf{x}^*)) (1 - \eta\mu)^T$ term which decays exponentially, and a non-vanishing term corresponding to the noise coming from the stochastic noise and the client sampling.

The major challenge in the analysis is bounding the effect of data drifts and reclustering. Data drifts change client datasets, hence changing their local objective functions. On the other hand, while reclustering is useful in the long run, its immediate effect on the objective (1) can potentially be negative (see Figure 2b) due to the mixing of models from different clusters.

To address these issues, we introduce natural assumptions. First, we assume that the distance between client representations – which in our algorithm correspond to label distributions – translates into the difference between their objective functions. Second, we assume that the effect of data drift

is bounded: that is, the representation r_i of client i changes by at most δ for each data drift. Finally, we assume that the clients are clusterable; that is, K clusters exist, so representations of clients within each cluster are similar.

ASSUMPTION A. $f_t^{(i)}$ – local objective for client i at iteration t – is μ -strongly convex and L -smooth, and we have access to a stochastic oracle with variance σ^2 . There exists clustering such that representations inside each cluster are Δ -close, the data drift changes representations by at most δ , and the difference between the objective functions is at most the distance between representations multiplied by θ .

Under these assumptions, the data drift can affect each client representation – and hence objective 1 – by at most a fixed value. Moreover, by the clustering assumption, the clients within each cluster have similar objective function; moreover, this holds both before and after the reclustering. This allows us to bound the increase in the loss function due to reclustering. In Appendix A, we prove the following theorem.

THEOREM 4.1. Let N be the number of clients and M be the total number of machines sampled per round. Let \mathbf{x}^* be the minimizer of $f_0 = \text{avg}_i f_0^{(i)}$. Let $\mathbf{c}_t^{(k,*)}$ be the minimizer for cluster k at iteration t . Then, under Assumption A, for $\eta \leq 1/L$, for any iteration t we have

$$\begin{aligned} & \frac{1}{N} \sum_{k \in [K]} \sum_{i \in C_k} \left(f_t^{(i)}(\mathbf{c}_{t+1}^{(k)}) - f_t^{(i)}(\mathbf{c}_t^{(k,*)}) \right) \\ & \leq (1 - \eta\mu)^{TR} (f_0(\mathbf{x}_0) - f_0(\mathbf{x}^*)) \\ & + \frac{L\eta}{2\mu} \left(\frac{\sigma^2 + 8L\theta\Delta}{M/K} + 3\theta(\Delta + \delta)(1 - \eta\mu)^R \right) \end{aligned}$$

That is, objective (1) has two terms: an exponentially decaying term corresponding to the initial loss, and a non-vanishing term corresponding to stochastic noise, as well as loss due to clustering and data drift.

5 IMPLEMENTATION

We implemented Fielding in Python on top of FedScale [22], a state-of-the-art open-source FL engine. We follow FedScale’s design of having one centralized coordinator and client-specific executors. Figure 4 illustrates the end-to-end workflow of Fielding. At the beginning of each training round, ① client executors register with the coordinator to participate and report their local data distribution optionally. The coordinator’s cluster manager maintains clients’ distribution records and ② moves clients to the closest cluster when their reported distribution drifts. After moving drifted clients individually, the cluster manager measures center shift distances and either triggers global re-clustering when any cluster center shifts significantly or finalizes clusters membership. In the case of global clustering, we use

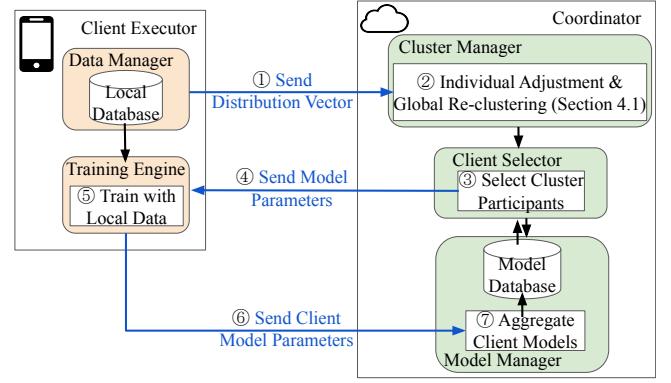


Figure 4: The system architecture and workflow of Fielding. Blue arrows represent communication between clients and the coordinator. Black arrows indicate the workflow among different components on the same machine.

K -means clustering and determine K using the silhouette method [39]. To stabilize new clusters and avoid creating poor clusters due to bad centroids initialization [2], we reuse previous cluster centers when the target K value is no greater than the number of existing clusters. The initial model of each newly created cluster is set as the average of its clients’ previous cluster model (see Algorithm 2).

When client clustering is done, the coordinator notifies the client selector to ③ select a subset of clients to contribute to each cluster and ④ communicate the latest model parameters to the selected clients. ⑤ Upon receiving a set of parameters, client executors conduct the training process over local data and ⑥ upload the updated model parameters to the coordinator. Finally, ⑦ the model manager aggregates individual models and stores the new cluster models. This process happens iteratively until the clients’ mean test accuracy reaches a target value. Fielding also regularly creates checkpoints for the models, clients’ metadata, and cluster memberships for future fine-tuning and failure recovery. Fielding’s overheads are mainly determined by the number and size of distribution vectors we need to re-cluster. In our largest evaluation setting where we train with 5078 clients on a dataset with 100 labels, per-client adjustment takes 2.0 seconds and global re-clustering takes 15.6 seconds on average. Storing the latest reported distribution vector for each client consumes $5078 \times 100 \times 4B \approx 1.9MB$ of memory.

6 EVALUATION

We evaluate Fielding’s effectiveness by running four FL tasks. We also demonstrate Fielding’s compatibility with various client selection strategies and aggregation algorithms. Our end-to-end results highlight that

- Fielding improves the final test accuracy by 1.9%-5.9% and is more stable than other clustered FL works when facing real-world data drifts.
- Fielding works well with complementary FL optimizations and offers consistent accuracy improvements across different settings.
- Fielding is robust to malicious clients who report corrupted distribution vectors and accelerates model convergence when clients are less heterogeneous.

Environment. We emulate large-scale FL training with four NVIDIA A100 GPUs. We use FedScale’s device datasets which provide realistic device computing and network capacity numbers while following the standardized training configurations in the original paper [22]. Per-device computation time estimation is based on device computing speed, batch size, and number of local iterations. And we estimate the communication time as the volume of downloaded and uploaded data divided by the network bandwidth.

Datasets and Models. We use a satellite image dataset (FMoW [10]) and two video datasets (Waymo Open [42] and Cityscapes [11]) that include natural data drifts. For FMoW, we keep images taken after January 1, 2015, and create one client for each UTM zone. Two training rounds correspond to one day in this dataset. For Waymo Open and Cityscapes, we use video segments pre-processed by Ekyia [5]. We create one client per camera per Waymo Open segment (212 clients in total), and one client per 100 consecutive frames per Cityscapes segment (217 clients in total). As these two datasets have video frame IDs within segments but lack global timestamps, we sort the samples on each client by frame ID and then partition the frames into 10 intervals. We stream in one data interval every 20 training rounds. We train ResNet-18 [15] on these three real-world datasets.

To demonstrate how Fielding behaves in highly dynamic settings, we construct an artificial trace using the FedScale Open Images [21] benchmark. We find the top 100 classes in terms of number of samples and keep clients whose local data labels include at least 10 of the top 100 classes (5078 clients in total). Each client randomly and independently partitions all local data labels into 10 buckets. Each bucket then contains all local samples of labels mapped to it. We stream in one bucket every 50 training rounds. We train ShuffleNet v2 [33] on Open Images. On all four datasets, clients retain samples they received over the last 100 rounds. The details of experiment settings and training parameters are included in Appendix B.

Baselines. Our non-clustering baseline trains one global model by randomly selecting a subset of participants from all available clients every round. To compare with other clustered FL works with data drift handling measures, we use

Auxo [30] as the continuous re-clustering baseline and FlexCFL [12] as the individual movement baseline. We employ FedProx [26], a federated optimization algorithm that tackles client heterogeneity, for all approaches.

Metrics. Our main evaluation metrics are Time-to-Accuracy (TTA) and final test accuracy. We define TTA as the training time required to achieve the target accuracy, which is the average test accuracy when training a single global model for all clients. We report the final average test accuracy across diverse settings to demonstrate the sensitivity and robustness of Fielding.

6.1 End-to-end Training Performance

Figure 5 shows that Fielding improves clients’ average test accuracy by 5.9%, 1.9%, and 2.1% on FMoW, Cityscapes, and Waymo Open respectively. When considering the global model’s final test accuracy as the target accuracy, Fielding offers a 1.27 \times , 1.16 \times , and 2.61 \times training speed up respectively (we define the moment when Fielding’s accuracy starts invariably exceeding the target accuracy as the point at which it achieves the target accuracy). On the highly heterogeneous and dynamic Open Images trace, Fielding achieves a significant accuracy improvement of 26%.

Although Auxo exhibits comparable or better performance than the global model baseline, the benefits are limited: accuracy improvement up to 0.9% and convergence speedup up to 1.05 \times . These results are less impressive than those reported in the original paper on static datasets, suggesting a decline in clustering effectiveness. This degradation stems from re-clustering only the selected participants after each round, which neglects clients that have drifted but remain unselected. Additionally, this approach can lead to sudden accuracy drops on the biased Waymo Open dataset, where over 80% samples are cars, as selecting diverging clients can cause dramatic model weight changes. FlexCFL fails to improve model accuracy or convergence speed on all three real-world datasets. This is because FlexCFL migrates clients individually without adjusting the number of clusters. As we observe in Figure 1, such an approach will eventually make clusters as heterogeneous as a global set containing all clients.

6.2 Compatibility with Other Optimizations

To demonstrate that Fielding is agnostic to client selection and model aggregation strategies, we evaluate Fielding’s performance on FMoW together with various complementary optimizations. In Figure 6 and Figure 7, we use the algorithm name alone to denote the baseline where we train one global model and use “+ Fielding” to denote running the algorithm atop Fielding.

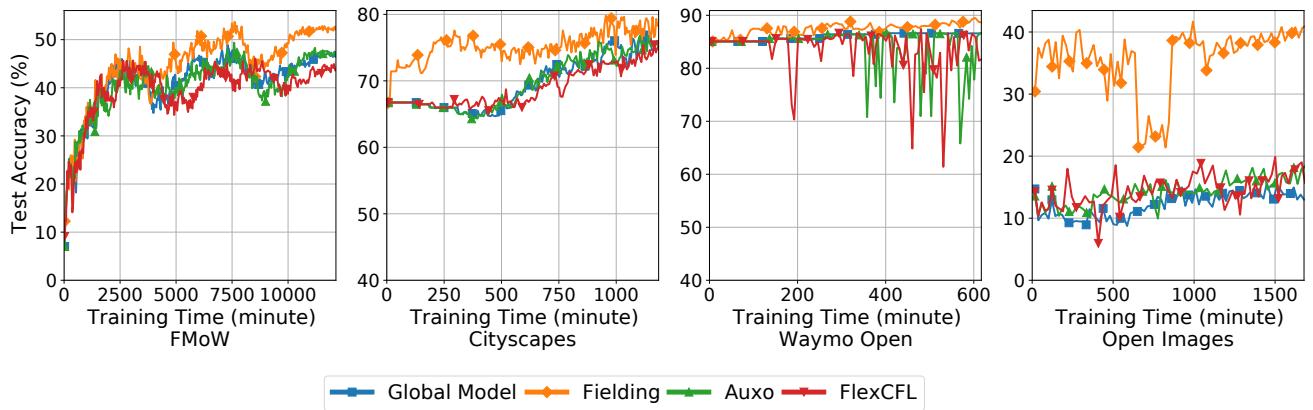


Figure 5: Time to accuracy (TTA) comparison over four tasks.

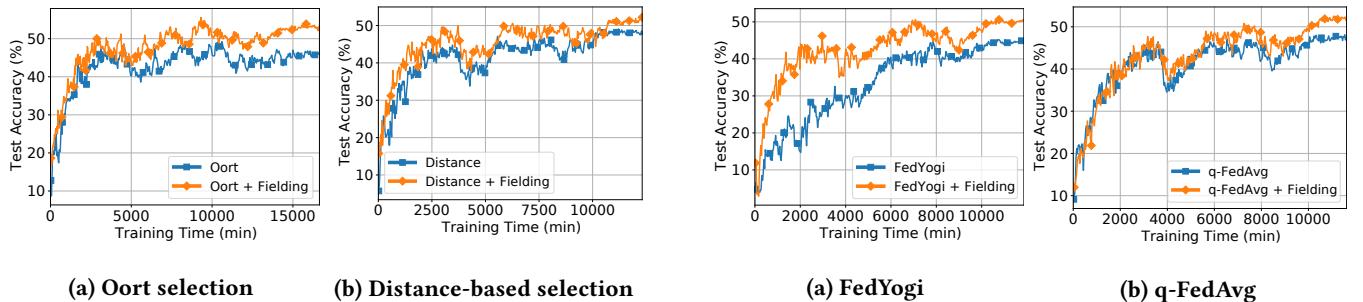


Figure 6: Fielding with different client selection strategies on FMoW dataset.

We use the Oort selection algorithm [23] and a distance-based algorithm prioritizing clients closer to the distribution center as client selection examples. As shown in Figure 6, Fielding improves the final average test accuracy by 6.0% and 4.5% while reaching the target accuracy 2.62 \times and 1.17 \times faster. Note that Oort selection actively incorporates clients otherwise filtered out due to long response time, leading to a longer average round time.

We also show that Fielding works well with aggregation algorithm FedYogi [38] and q-FedAvg [27] in Figure 7. Fielding improves the final accuracy by 5.9% and 4.9% while giving a 1.34 \times and 1.25 \times speedup respectively.

6.3 Robustness and Sensitivity Analysis

Malicious clients. We study the impact of having malicious clients who intentionally report wrong label distribution vectors. We simulate such behavior by randomly selecting a subset of clients who permute the coordinates of their distribution vectors when registering with the coordinator. As reported in Figure 8, Fielding consistently outperforms the baseline global model accuracy across different percentages of malicious clients.

Varying data heterogeneity. Our analysis on Fielding’s sensitivity to data heterogeneity degrees is inspired by the

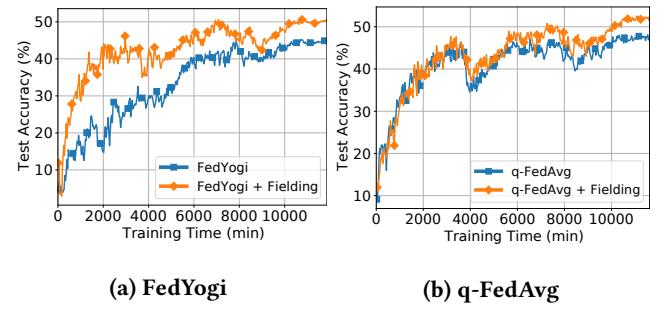


Figure 7: Fielding with different FL algorithms on FMoW dataset.

idea of improving training performance on non-IID data through a small shared dataset [48]. We inject new training samples by creating three datasets shared with all clients: one sample for each of the least represented 50% labels, one sample for each label, and two samples for each label. A larger shared set implies a smaller degree of heterogeneity. As shown in Figure 9, both Fielding and the baseline global model benefit from this sharing approach, with Fielding improving the final average test accuracy by 3.2% to 4.4%.

Static data. Finally, we demonstrate that Fielding still improves model convergence when clients have static local data and no drifts happen. We rerun the experiment on the FMoW dataset with all data samples available throughout the training. As shown in Figure 10, Fielding offers the largest gain and improves the final average test accuracy by 9.2%. Auxo and FlexCFL achieve an improvement of 3.9% and 5.6% respectively. Note that Auxo experiences an accuracy drop towards the end of training. This is likely due to Auxo re-clustering all selected clients after each round by default, causing unnecessary cluster membership changes when the data is completely static.

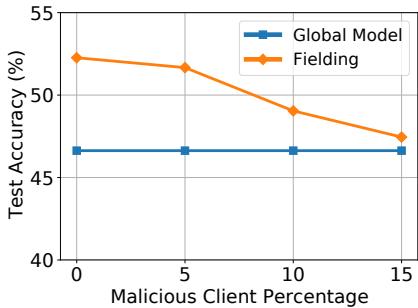


Figure 8: Fielding is robust to malicious clients.

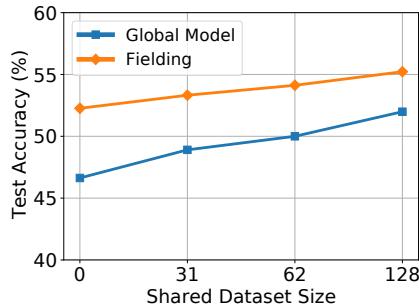


Figure 9: Fielding offers gains across heterogeneity degrees.

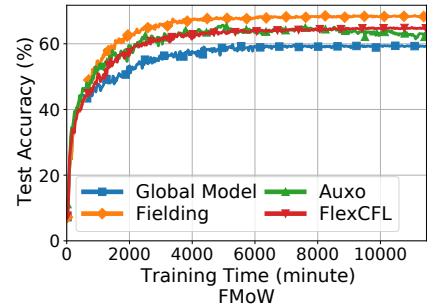


Figure 10: Fielding offers gains across heterogeneity degrees.

7 RELATED WORK AND DISCUSSION

Clustered FL Previous clustering solutions [12, 14, 18, 30, 47] have achieved success at bounding intra-cluster heterogeneity and improving model performance. However, their improvements are compromised by real-world data drifts as they either assume stable environments or handle drifts insufficiently.

Heterogeneity-aware FL. Recent works have tackled heterogeneity challenges through client selection, workload scheduling, and update corrections. Oort considers both statistical and system utility and designs an exploration-exploitation strategy for client selection [23]. PyramidFL adjusts the number of local iterations and parameter dropouts to optimize participants’ data and system efficiency [24]. DSS-Edge-FL dynamically determines the size of local data used for training and selects representative samples, optimizing resource utilization while considering data heterogeneity [41]. MOON corrects local updates on clients by maximizing the similarity between representations learned by local models and the global model [25].

Drifts-aware FL. Adaptive-FedAvg is an FL algorithm that adapts client learning rate based on the model variability between consecutive rounds [7]. CDA-FedAvg detects concept drifts and extends FedAvg for learning under such drifts. It defines a short-term and a long-term memory for each client and applies rehearsal using data in the long-term memory when drifts happen [8]. Master-FL proposes a multi-scale algorithmic framework that instructs clients to train over multiple time horizons with a suitable learning rate for each time chunk [13].

Fielding focuses on dynamic client clustering that is robust against potential data drifts. It is agnostic to other FL optimizations and should work seamlessly with the FL aggregation, selection, and scheduling algorithms above.

8 CONCLUSION

We have presented Fielding, a novel clustering-based Federated Learning framework designed to handle data drifts with low overhead. Fielding addresses the challenges of data drift by re-clustering clients based on their label distribution vectors as their data distributions change, dynamically adjusting the number of clusters, and stabilizing responses to minor data shifts. Fielding is robust against malicious clients and varying degrees of heterogeneity, and it is compatible with various client selection and aggregation algorithms. In Fielding, clients register with a centralized coordinator that updates their metadata and assigns them to the nearest cluster based on their local data distribution. Before each training round, the coordinator determines how much each cluster center has shifted. If any cluster exceeds a predefined threshold, a global re-clustering process is initiated; otherwise, the updated cluster memberships are finalized. Experimental results demonstrate that Fielding improves the final test accuracy by 1.9% to 5.9% compared to existing methods and provides greater stability when facing real-world data drifts across different settings.

There are potential directions to further enhance Fielding. Our current approach requires labeled data, which may not be available in all contexts. Additionally, clients actively report their label distribution vectors to the coordinator, which introduces latent privacy concerns. In future work, we will focus on alternatives, such as inferring distribution vectors from gradients.

REFERENCES

- [1] Nawaf Abdulla, Mehmet Demirci, and Suat Ozdemir. 2024. Smart meter-based energy consumption forecasting for smart cities using adaptive federated learning. *Sustainable Energy, Grids and Networks* 38 (2024), 101342. <https://doi.org/10.1016/j.segan.2024.101342>
- [2] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. 2020. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* 9, 8 (2020), 1295.

- [3] David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding*. Technical Report. Stanford.
- [4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*. PMLR, 2938–2948.
- [5] Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. 2022. Ekyu: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 119–135.
- [6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingberman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roslander. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia (Eds.), Vol. 1. 374–388.
- [7] Giuseppe Canonaco, Alex Bergamasco, Alessio Mongelluzzo, and Manuel Roveri. 2021. Adaptive federated learning in presence of concept drift. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–7.
- [8] Fernando E Casado, Dylan Lema, Marcos F Criado, Roberto Iglesias, Carlos V Regueiro, and Senén Barro. 2022. Concept drift detection and adaptation for federated and continual learning. *Multimedia Tools and Applications* (2022), 1–23.
- [9] Yujing Chen, Yue Ning, Martin Slawski, and Huzeifa Rangwala. 2020. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 15–24.
- [10] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. 2018. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6172–6180.
- [11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [12] Moming Duan, Duo Liu, Xinyuan Ji, Yu Wu, Liang Liang, Xianzhang Chen, Yujuan Tan, and Ao Ren. 2021. Flexible clustered federated learning for client-level data distribution shift. *IEEE Transactions on Parallel and Distributed Systems* 33, 11 (2021), 2661–2674.
- [13] Bhargav Ganguly and Vaneet Aggarwal. 2023. Online Federated Learning via Non-Stationary Detection and Adaptation Amidst Concept Drift. *IEEE/ACM Transactions on Networking* (2023).
- [14] Avishhek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An Efficient Framework for Clustered Federated Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 19586–19597.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [16] Li Huang, Andrew L Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. 2019. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of biomedical informatics* 99 (2019), 103291.
- [17] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. 2022. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems* 4 (2022), 814–832.
- [18] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B Gibbons. 2023. Federated learning under distributed concept drift. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 5834–5853.
- [19] Gyudong Kim, Mehdi Ghasemi, Soroush Heidari, Seungryong Kim, Young Geun Kim, Sarma Vrudhula, and Carole-Jean Wu. 2024. HeteroSwitch: Characterizing and Taming System-Induced Data Heterogeneity in Federated Learning. *Proceedings of Machine Learning and Systems* 6 (2024), 31–45.
- [20] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*. PMLR, 5637–5664.
- [21] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallochi, Alexander Kolesnikov, et al. 2020. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision* 128, 7 (2020), 1956–1981.
- [22] Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. 2022. Fedscale: Benchmarking model and system performance of federated learning at scale. In *International conference on machine learning*. PMLR, 11814–11827.
- [23] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient Federated Learning via Guided Participant Selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. USENIX Association, 19–35.
- [24] Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. 2022. PyramidFL: A fine-grained client selection framework for efficient federated learning. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 158–171.
- [25] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10713–10722.
- [26] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze (Eds.), Vol. 2. 429–450.
- [27] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2020. Fair Resource Allocation in Federated Learning. In *International Conference on Learning Representations*.
- [28] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the Convergence of FedAvg on Non-IID Data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- [29] Zonghang Li, Yihong He, Hongfang Yu, Jiawen Kang, Xiaoping Li, Zenglin Xu, and Dusit Niyato. 2022. Data heterogeneity-robust federated learning via group client selection in industrial IoT. *IEEE Internet of Things Journal* 9, 18 (2022), 17844–17857.
- [30] Jiachen Liu, Fan Lai, Yinwei Dai, Aditya Akella, Harsha V Madhyastha, and Mosharaf Chowdhury. 2023. Auxo: Efficient federated learning via scalable client clustering. In *Proceedings of the 2023 ACM Symposium on Cloud Computing*. 125–141.
- [31] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. 2023. Multi-center federated learning: clients clustering for better personalization. *World Wide Web* 26, 1 (2023), 481–500.
- [32] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S. Yu. 2024. Privacy and Robustness in Federated Learning: Attacks and Defenses. *IEEE Transactions on Neural Networks and Learning Systems* 35, 7 (2024), 8726–8746.

- <https://doi.org/10.1109/TNNLS.2022.3216981>
- [33] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*. 116–131.
 - [34] Othmane Marfoq, Giovanni Neglia, Laetitia Kameni, and Richard Vidal. 2023. Federated learning for data streams. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 8889–8924.
 - [35] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
 - [36] Arijit Nandi and Fatos Xhafa. 2022. A federated learning method for real-time emotion state classification from multi-modal streaming. *Methods* 204 (2022), 340–347.
 - [37] Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific data* 5, 1 (2018), 1–13.
 - [38] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. 2021. Adaptive Federated Optimization. In *International Conference on Learning Representations*.
 - [39] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.
 - [40] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2020. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems* 31, 9 (2020), 3400–3413. <https://doi.org/10.1109/TNNLS.2019.2944481>
 - [41] Mohamed Adel Serhani, Haftay Gebreslasie Abreha, Asadullah Tariq, Mohammad Hayajneh, Yang Xu, and Kadhim Hayawi. 2023. Dynamic Data Sample Selection and Scheduling in Edge Federated Learning. *IEEE Open Journal of the Communications Society* 4 (2023), 2133–2149. <https://doi.org/10.1109/OJCOMS.2023.3313257>
 - [42] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. 2020. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
 - [43] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. 2023. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics* 14, 2 (2023), 513–535.
 - [44] Mengwei Xu, Dongqi Cai, Yaozong Wu, Xiang Li, and Shangguang Wang. 2024. {FwdLLM}: Efficient Federated Finetuning of Large Language Models with Perturbed Inferences. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 579–596.
 - [45] Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigu Bian, Yunxin Liu, and Xuanze Liu. 2021. Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data. In *Proceedings of the Web Conference 2021*. 935–946.
 - [46] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).
 - [47] Yuxin Zhang, Haoyu Chen, Zheng Lin, Zhe Chen, and Jin Zhao. 2024. FedAC: A Adaptive Clustered Federated Learning Framework for Heterogeneous Data. *arXiv preprint arXiv:2403.16460* (2024).
 - [48] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).
 - [49] Lisang Zhou, Meng Wang, and Ning Zhou. 2023. Distributed Federated Learning-Based Deep Learning Model for Privacy MRI Brain Tumor Detection. *Journal of Information, Technology and Policy* (2023), 1–12.

A PROOF OF CONVERGENCE

Algorithm 1: Clustered Federated Learning with Data Drift

```

1 input: the number of clients  $K$ , the number of data drift events  $T$ , the number of sampled clients per round  $M$ , the
   number of rounds per data drift event  $R$ 
2 Initialize cluster models  $\mathbf{c}_0^{(1)}, \dots, \mathbf{c}_0^{(K)}$  with the same model  $\mathbf{x}_0$ 
3 for  $t = 0, \dots, T - 1$  do
4   for each client  $i$  do
5     Adopt data drift for client  $i$ 
6     Reassign client  $i$  to the closest cluster
7     Let  $\mathbf{x}_i$  be the model corresponding to the cluster containing client  $i$ 
8     Let  $C_1, \dots, C_K$  be the  $K$ -center clustering of all clients based on representations from model  $\tilde{\mathbf{x}}_t$ 
9     for  $k = 1, \dots, K$  do
10     $\tilde{\mathbf{c}}_0^{(k)} \leftarrow \text{avg}_{i \in C_k} \mathbf{x}_i$ 
11    for  $\tau = 0, \dots, R - 1$  do
12      for  $k = 1, \dots, K$  do
13        Sample subset  $S$  from  $C_k$  of size  $M/K$  independently with replacement
14         $\tilde{\mathbf{c}}_{\tau+1}^{(k)} \leftarrow \tilde{\mathbf{c}}_\tau^{(k)} - \eta \text{avg}_{i \in S} G_t^{(i)}(\tilde{\mathbf{c}}_\tau^{(k)})$ 
15    for  $k = 1, \dots, K$  do
16       $\mathbf{c}_{t+1}^{(k)} \leftarrow \tilde{\mathbf{c}}_R^{(k)}$ 

```

In this section, we analyze the performance of our framework in the case when the number of clusters K is known and the number of local iterations is 1. The analysis can be modified to handle local iterations and other variations based on the previous work, see e.g. Li et al. [28]. In the algorithm, we recluster clients after every data drift event and use the k -center clustering instead of k -means clustering to simplify the proof.

We present the algorithm in Algorithm 1. After every data drift event, we first accept data changes for each client. After that, we recluster the clients, and assign to each cluster a model by averaging models of all clients in the cluster. After that, for several rounds, we perform standard Federated Averaging updates on each cluster: we sample M clients, compute the stochastic gradient for each client, and update the cluster model with the sampled gradients.

ASSUMPTION B (OBJECTIVE FUNCTIONS). Let $f_t^{(i)}$ be the local function at each client i at data drift event t . Then $f_t^{(i)}$ is μ -strongly convex and L -smooth (has an L -Lipschitz gradient) for all i and t .

The following are the standard assumptions on stochastic gradient.

ASSUMPTION C (STOCHASTIC GRADIENTS). For each client i , at every data drift event t , we have access to stochastic gradient oracle $G_t^{(i)}$:

- $\mathbb{E} \left[G_t^{(i)}(\mathbf{x}) \right] = f_t^{(i)}(\mathbf{x})$ for all \mathbf{x} ;
- $\mathbb{E} \left\| G_t^{(i)}(\mathbf{x}) - f_t^{(i)}(\mathbf{x}) \right\|^2 \leq \sigma^2$ for all \mathbf{x} .

The next assumption connects the objective value with client representations. Client representations can take various forms, and in our paper, we assume that the representation of the cluster is its label distribution.

ASSUMPTION D (REPRESENTATION). For a metric space (X, ρ) , let $r_t^{(i)} \in X$ be a representation of client i at time t . Then there exists a constant θ such that $|f_t^{(i)}(\mathbf{x}) - f_t^{(j)}(\mathbf{x})| \leq \theta \cdot \rho(r_t^{(i)}, r_t^{(j)})$ for all i, j, t .

Clearly, if data on clients can change arbitrarily at every data drift event, in general, there is no benefit in reusing previous iterates. Hence, our next assumption bounds how much the client changes after the drift.

ASSUMPTION E (DATA DRIFT). At every data drift event, the representation of each client changes by at most δ : $\rho(r_t^{(i)}, r_{t+1}^{(i)}) \leq \delta$ for all i, t .

Finally, we assume that the clients are clusterable: there exist K clusters so that within each cluster all points have similar representation.

ASSUMPTION F (CLUSTERING). At every data drift event, the clients can be partitioned into K clusters C_1, \dots, C_K so that, for any $k \in [K]$ and any $i, j \in C_k$, we have $\rho(r_t^{(i)}, r_t^{(j)}) \leq \Delta$.

For completeness, we present the SGD convergence analysis for strongly convex functions. The following analysis is well-known, but, compared with the most standard analysis, it works directly with the loss functions instead of distances to the optimum, thus avoiding losing the condition number in the following bounds.

LEMMA A.1. Let f be L -smooth and μ -strongly convex. Let g be an unbiased stochastic gradient oracle of f with variance σ^2 . Let \mathbf{x}^* be the minimizer of f . Then, for the stochastic gradient update rule $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta g(\mathbf{x}_t)$ with $\eta \leq 1/L$, we have

$$\mathbb{E} [f(\mathbf{x}_T) - f(\mathbf{x}^*)] \leq (1 - \eta\mu)^T (f(\mathbf{x}_0) - f(\mathbf{x}^*)) + \frac{L\eta}{2\mu} \sigma^2.$$

PROOF. Let $\mathbb{E}_t [\cdot]$ be expectation conditioned on \mathbf{x}_t . By the Descent Lemma, we have:

$$\begin{aligned} \mathbb{E}_t [f(\mathbf{x}_{t+1})] &\leq f(\mathbf{x}_t) + \mathbb{E}_t \langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L}{2} \mathbb{E}_t \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ &= f(\mathbf{x}_t) + \mathbb{E}_t \langle \nabla f(\mathbf{x}_t), -\eta g(\mathbf{x}_t) \rangle + \frac{L}{2} \mathbb{E}_t \|\eta g(\mathbf{x}_t)\|^2 \\ &= f(\mathbf{x}_t) - \eta \|\nabla f(\mathbf{x}_t)\|^2 + \frac{L\eta^2}{2} (\|\nabla f(\mathbf{x}_t)\|^2 + \sigma^2) \\ &\leq f(\mathbf{x}_t) - \frac{\eta}{2} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{L\eta^2}{2} \sigma^2, \end{aligned}$$

where in the last inequality we used $\eta \leq \frac{1}{L}$. Using the fact that f is μ -strongly-convex, we have $\|\nabla f(\mathbf{x}_t)\|^2 \geq 2\mu(f(\mathbf{x}_t) - f(\mathbf{x}^*))$, giving

$$\mathbb{E}_t [f(\mathbf{x}_{t+1})] \leq f(\mathbf{x}_t) - \eta\mu(f(\mathbf{x}_t) - f(\mathbf{x}^*)) + \frac{L\eta^2}{2} \sigma^2.$$

Subtracting $f(\mathbf{x}^*)$ from both parts, we have

$$\mathbb{E}_t [f(\mathbf{x}_{t+1}) - f(\mathbf{x}^*)] \leq (1 - \eta\mu) (f(\mathbf{x}_t) - f(\mathbf{x}^*)) + \frac{L\eta^2}{2} \sigma^2.$$

By telescoping, taking the full expectation, and using $\sum_{i=0}^{\infty} (1 - \eta\mu)^i = \frac{1}{\eta\mu}$, we get

$$\mathbb{E} [f(\mathbf{x}_T) - f(\mathbf{x}^*)] \leq (1 - \eta\mu)^T (f(\mathbf{x}_0) - f(\mathbf{x}^*)) + \frac{L\eta}{2\mu} \sigma^2. \quad \square$$

The above result shows how the objective improves with each round: namely, after R , the non-stochastic part of the loss improves by a factor which depends on R exponentially. However, each data drift and each reclustering might potentially hurt the objective. Our next result bounds their effect on the loss.

LEMMA A.2. Let N be the number of clients, and $t + 1$ be a fixed data drift event. Let C_1, \dots, C_k be clusters $\mathbf{c}_1, \dots, \mathbf{c}_K$ be cluster models, and $\mathbf{c}_1^*, \dots, \mathbf{c}_K^*$ be the optimal cluster models at the end of processing data drift event t . Similarly, let $\bar{C}_1, \dots, \bar{C}_\ell$ be clusters $\bar{\mathbf{c}}_1, \dots, \bar{\mathbf{c}}_K$ be cluster models, and $\bar{\mathbf{c}}_1^*, \dots, \bar{\mathbf{c}}_K^*$ be the optimal cluster models immediately after reclustering. Then, under Assumptions B-D, the following holds:

$$\frac{1}{N} \sum_{\ell \in [K]} \sum_{i \in \bar{C}_\ell} \left(f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell) - f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell^*) \right) \leq \frac{1}{N} \sum_{k \in [K]} \sum_{i \in C_k} \left(f_t^{(i)}(\mathbf{c}_k) - f_t^{(i)}(\mathbf{c}_k^*) \right) + 3\theta(\Delta + \delta)$$

PROOF. For a client i , let k and ℓ be such that $i \in C_k \cap \bar{C}_\ell$. We rewrite $f_t^{(i)}(\bar{\mathbf{c}}_\ell) - f_t^{(i)}(\bar{\mathbf{c}}_\ell^*)$ as

$$f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell) - f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell^*) = (f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell) - f_{t+1}^{(i)}(\mathbf{c}_k^*)) + (f_{t+1}^{(i)}(\mathbf{c}_k^*) - f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell^*))$$

and bound each term separately.

Bounding the first term. Let \mathbf{x}_j be the model of client j before reclustering, i.e. $\mathbf{x}_j = \mathbf{c}_{k'}$ where $j \in C_{k'}$. Then,

$$f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell) = f_{t+1}^{(i)}\left(\operatorname{avg}_{j \in \bar{C}_\ell} \mathbf{x}_j\right) \leq \operatorname{avg}_{j \in \bar{C}_\ell} f_{t+1}^{(i)}(\mathbf{x}_j), \quad (2)$$

where the last inequality follows by convexity of $f_t^{(i)}$. By Assumption F, since all clients \bar{C}_ℓ belong to the same cluster, their representations differ by at most Δ , and hence by Assumption D their local objectives differ by at most $\theta\Delta$. Therefore,

$$\operatorname{avg}_{j \in \bar{C}_\ell} f_{t+1}^{(i)}(\mathbf{x}_j) \leq \operatorname{avg}_{j \in \bar{C}_\ell} (f_{t+1}^{(j)}(\mathbf{x}_j) + \theta\Delta) = \operatorname{avg}_{j \in \bar{C}_\ell} f_{t+1}^{(j)}(\mathbf{x}_j) + \theta\Delta$$

Summing over all $i \in \bar{C}_\ell$, we get

$$\sum_{i \in \bar{C}_\ell} (\operatorname{avg}_{j \in \bar{C}_\ell} f_{t+1}^{(i)}(\mathbf{x}_j) + \theta\Delta) = \sum_{j \in \bar{C}_\ell} (f_{t+1}^{(j)}(\mathbf{x}_j) + \theta\Delta)$$

By definition, for each $i \in C_k$ we have $\mathbf{x}_i = \mathbf{c}_k$. So,

$$\sum_{\ell \in [K]} \sum_{i \in \bar{C}_\ell} f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell) \leq \sum_{k \in [K]} \sum_{i \in C_k} f_{t+1}^{(i)}(\mathbf{c}_k)$$

Hence, the sum of the first terms in Equation (2) over all i can be bounded as

$$\sum_{k \in [K]} \sum_{i \in C_k} (f_{t+1}^{(i)}(\mathbf{c}_k) - f_{t+1}^{(i)}(\mathbf{c}_k^*) + \theta\Delta) \leq \sum_{k \in [K]} \sum_{i \in C_k} (f_t^{(i)}(\mathbf{c}_k) - f_t^{(i)}(\mathbf{c}_k^*) + \theta(\Delta + 2\delta)),$$

where we used Assumptions D and E to bound the change of each objective after the data drift as $\theta\delta$.

Bounding the second term. Let $\mathbf{x}_t^{(i,*)}$ be the minimizer of $f_t^{(i)}$ and $\mathbf{x}_{t+1}^{(i,*)}$ be the minimizer of $f_{t+1}^{(i)}$. Then, by Assumptions D and F we have

$$\begin{aligned} f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell) &\geq f_{t+1}^{(i)}(\mathbf{x}_{t+1}^{(i,*)}) && (\mathbf{x}_{t+1}^{(i,*)} \text{ is the minimizer of } f_{t+1}^{(i)}) \\ &\geq f_t^{(i)}(\mathbf{x}_{t+1}^{(i,*)}) - \theta\delta && (\text{Assumptions D and E}) \\ &\geq f_t^{(i)}(\mathbf{x}_t^{(i,*)}) - \theta\delta && (\mathbf{x}_t^{(i,*)} \text{ is the minimizer of } f_t^{(i)}) \\ &\geq f_t^{(i)}(\mathbf{c}_k^*) - \theta(\Delta + \delta), \end{aligned}$$

where the last inequality holds since otherwise $f_t^{(i)}(\mathbf{c}_k^*) > f_t^{(i)}(\mathbf{x}_t^{(i,*)}) + \theta\Delta$, which is impossible since \mathbf{c}_k^* is the minimizer of $\operatorname{avg}_{j \in C_k} f_t^{(j)}$ and $\operatorname{avg}_{j \in C_k} f_t^{(j)}(\mathbf{x}_t^{(i,*)}) \leq f_t^{(i)}(\mathbf{x}_t^{(i,*)}) + \theta\Delta$ by Assumptions D and 3.

Finally, we have $|f_t^{(i)}(\mathbf{c}_k^*) - f_{t+1}^{(i)}(\mathbf{c}_k^*)| < \theta\delta$ by Assumptions D and E. Combining the bounds concludes the proof. \square

Combining these results leads to our main convergence result.

THEOREM A.3. *Let N be the number of clients, and let their objective functions satisfy Assumptions B-D. Let M be the total number of machines sampled per round. Let \mathbf{x}^* be the minimizer of $f_0 = \operatorname{avg}_i f_0^{(i)}$. Let $\mathbf{c}_t^{(k,*)}$ be the minimizer for cluster k at data drift event t . Then, for $\eta \leq 1/L$, for any data drift event t of Algorithm 1 we have*

$$\frac{1}{N} \sum_{k \in [K]} \sum_{i \in C_k} \left(f_t^{(i)}(\mathbf{c}_{t+1}^{(k)}) - f_t^{(i)}(\mathbf{c}_t^{(k,*)}) \right) \leq (1 - \eta\mu)^{TR} (f(\mathbf{x}_0) - f(\mathbf{x}^*)) + \frac{L\eta}{2\mu} \left(\frac{\sigma^2 + 8L\theta\Delta}{M/K} + 3\theta(\Delta + \delta)(1 - \eta\mu)^R \right)$$

PROOF. First, we express the objective in terms of objectives for individual clusters:

$$\frac{1}{N} \sum_{k \in [K]} \sum_{i \in C_k} \left(f_t^{(i)}(\mathbf{c}_t^{(k)}) - f_t^{(i)}(\mathbf{c}_t^{(k,*)}) \right) = \frac{1}{N} \sum_{k \in [K]} |C_k| \operatorname{avg}_{i \in C_k} \left(f_t^{(i)}(\mathbf{c}_t^{(k)}) - f_t^{(i)}(\mathbf{c}_t^{(k,*)}) \right)$$

We then analyze the convergence in each cluster. For any L -smooth function h with minimizer \mathbf{x}^* , it holds that $\|\nabla h(\mathbf{x})\| \leq \sqrt{2L(h(\mathbf{x}) - h(\mathbf{x}^*))}$. First, note that, inside each cluster, the objective functions differ by at most $\theta\Delta$. Hence, for any two clients i and j from the same cluster, defining $h = f_t^{(i)} - f_t^{(j)}$, by $2L$ -smoothness of h , for any \mathbf{x} we have

$$\|\nabla h(\mathbf{x})\| \leq \sqrt{4L(h(\mathbf{x}) - h(\mathbf{x}^*))} \leq \sqrt{8L\theta\Delta}$$

Hence, sampling clients uniformly from the cluster introduces variance at most $8L\theta\Delta$. Since we sample M/K clients from a cluster, the total variance – including the stochastic variance – is at most $\frac{\sigma^2 + 8L\theta\Delta}{M/K}$.

Next, by Lemma A.2, the total objective increases by at most $3\theta(\Delta + \delta)$ after reclustering. Since during R rounds the non-stochastic part of each objective decreases by a factor of $(1 - \eta\mu)^R$, we have

$$\begin{aligned} & \frac{1}{N} \sum_{k \in [K]} |C_k| \operatorname{avg}_{i \in C_k} \left(f_{t+1}^{(i)}(\mathbf{c}_{t+1}^{(k)}) - f_{t+1}^{(i)}(\mathbf{c}_{t+1}^{(k,*)}) \right) \\ & \leq \frac{1}{N} \sum_{k \in [K]} |C_k| \left(\operatorname{avg}_{i \in C_k} \left(f_t^{(i)}(\mathbf{c}_t^{(k)}) - f_t^{(i)}(\mathbf{c}_t^{(k,*)}) \right) + 3\theta(\Delta + \delta) \right) (1 - \eta\mu)^R + \sum_{i=0}^R (1 - \eta\mu)^i \cdot \frac{\sigma^2 + 8L\theta\Delta}{M/K} \\ & = (1 - \eta\mu)^R 3\theta(\Delta + \delta) + (1 - \eta\mu)^R \frac{1}{N} \sum_{k \in [K]} |C_k| \operatorname{avg}_{i \in C_k} \left(f_t^{(i)}(\mathbf{c}_t^{(k)}) - f_t^{(i)}(\mathbf{c}_t^{(k,*)}) \right) + \sum_{i=0}^R (1 - \eta\mu)^i \cdot \frac{\sigma^2 + 8L\theta\Delta}{M/K} \end{aligned}$$

By Lemma A.1, the last (stochastic) term accumulates over all data drift events as $\frac{L\eta}{2\mu} \cdot \frac{\sigma^2 + 8L\theta\Delta}{M/K}$. By a similar reasoning, $3\theta(\Delta + \delta)$ accumulates as $\frac{L\eta}{2\mu} \cdot 3\theta(\Delta + \delta)(1 - \eta\mu)^R$. And finally, $f(\mathbf{x}_0) - f(\mathbf{x}^*)$ term is multiplied by $(1 - \eta\mu)^R$ at every data drift event, giving factor of $(1 - \eta\mu)^{TR}$ over T data drift events. \square

B EXPERIMENT SETTINGS

We conduct experiments on public datasets Functional Map of the World (FMoW) [10], Cityscapes [11], Waymo Open [42], and Open Images [21]. In Table 2, we list the total number of clients we create on each dataset and the number of rounds in between clients getting new samples (e.g., as mentioned in Section 6, on Cityscapes we partition each client’s data into 10 intervals and stream in one interval every 30 rounds; so the rounds between new data arrivals are 30). In Table 3, we specify the training parameters including the total number of training rounds, learning rate, batch size, number of local steps, and the total number of clients selected to contribute in each round.

Table 2: Dataset configurations.

DATASET	NUMBER OF CLIENTS	ROUNDS BETWEEN NEW DATA ARRIVALS
FMoW	302	2
CITYSCAPES	217	30
WAYMO OPEN	212	20
OPEN IMAGES	5078	50

Table 3: Training parameters.

DATASET	TOTAL ROUNDS	LEARNING RATE	BATCH SIZE	NUMBER OF LOCAL STEPS	CLIENTS SELECTED PER ROUND
FMoW	2000	0.05	20	20	50
CITYSCAPES	200	0.001	20	20	20
WAYMO OPEN	100	0.001	20	20	20
OPEN IMAGES	400	0.05	20	20	200