FIELDING: Clustered Federated Learning with Data Drift

Minghao Li
Harvard University

Dmitrii Avdiukhin
Logical Intelligence

Rana Shahout
Harvard University

Nikita Ivkin
Amazon

Vladimir Braverman
Johns Hopkins University, Google

Minlan Yu
Harvard University

Abstract

Federated Learning (FL) trains deep models across edge devices without centralizing raw data. However, client heterogeneity slows down convergence and limits global model accuracy. Clustered FL (CFL) mitigates this by grouping clients with similar representations and training a separate model for each cluster. In practice, client data evolves over time – a phenomenon we refer to as *data drift* – which breaks cluster homogeneity and degrades performance. Data drift can take different forms depending on whether changes occur in the output values, the input features, or the relationship between them. We propose FIELDING, a CFL framework for handling diverse types of data drift with low overhead. FIELDING detects drift at individual clients and performs selective re-clustering to balance cluster quality and model performance, while remaining robust to varying levels of heterogeneity. Experiments show that FIELDING improves final model accuracy by 2.4–6.9% and achieves target accuracy 1.38x–3.10x faster than existing state-of-the-art CFL methods.

1 INTRODUCTION

Federated Learning (FL) is a distributed machine learning paradigm that enables multiple clients to collaboratively train a shared model while keeping raw data local (Bonawitz et al., 2019; McMahan et al., 2017). This makes it particularly appealing in applications such as mobile text prediction (Yang et al., 2018), energy consumption forecast (Abdulla et al., 2024), and medical imaging (Zhou et al., 2023).

In practice, FL involves clients with diverse hardware, usage patterns, and data sources, leading to variation in both dataset sizes and distributions: known as data heterogeneity (Li et al., 2022a; Kim et al., 2024). This heterogeneity slows model convergence and degrades model performance (Li et al., 2020a; Zhao et al., 2018; Sattler et al., 2020). Clustered FL (CFL) addresses this by grouping clients with similar data characteristics and training a separate model per cluster (Ghosh et al., 2020; Duan et al., 2021; Liu et al., 2023; Jothimurugesan et al., 2023), improving learning efficiency and accuracy by reducing intra-cluster heterogeneity.

However, real-world deployments face data drifts, i.e. change in data distribution over time. Representative application scenarios include the following.

- Cellular service providers train troubleshooting models based on network conditions (Shi et al., 2022). Data heterogeneity arises from regional differences in aggregated demand, geography, carrier types, etc. Data drift arises from large events that alter traffic distribution, equipment updates, and site addition or removal.
- Autonomous driving uses FL for semantic segmentation (Fantauzzo et al., 2022). Vehicles collect heterogeneous data due to differences in population density, weather, lighting, etc., while data drift occurs as they move between urban and rural areas and encounter changing conditions.
- FL is used for device personalization, such as keyboard suggestions (Yang et al., 2018). Heterogeneity stems from differences in language use, vocabulary, and interests, while data drift occurs as users adopt new slang, trends, or interests.

Data drift typically falls into three categories (Huyen,

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

2022): (1) *label shift*, where the output categories frequencies change but the characteristics associated with each category remain stable (Lipton et al., 2018; Garg et al., 2020); (2) *covariate shift*, where the distribution of the input features changes but the feature-label relationship remains constant (Ganguly and Aggarwal, 2023; Mallick et al., 2022); (3) *concept drift*, where the underlying input-output relationship changes, rendering a model trained on past data inaccurate on new data (Casado et al., 2022; Jothimurugesan et al., 2023).

As data drift accumulates, intra-cluster heterogeneity increases, and clusters can become as diverse as the full client set (Jothimurugesan et al., 2023; Moreno-Torres et al., 2012). Effective drift handling must address two key challenges: **adapting to varying drift magnitudes** while maintaining practical efficiency, and **supporting diverse drift types**. Our work addresses these challenges.

The first challenge arises when drifts have varying magnitudes. When many clients experience drift, re-clustering based on outdated cluster characteristics becomes invalid, often requiring global re-clustering. FlexCFL (Duan et al., 2021) and IFCA (Ghosh et al., 2020) attempt to adapt incrementally by moving clients one at a time while keeping the number of clusters fixed. They, however, struggle under large-scale shifts as they reassign drifted clients using clusters' current average gradient or weight, which shift significantly after massive reassignments. Global re-clustering reinitializes all clusters to reflect current data, but incurs significant communication and computation costs: clients might download multiple model replicas and upload new gradients; moreover, the newly formed clusters typically suffer an accuracy dip. Hence, while global re-clustering works for all drift magnitudes, incremental reassignment is more desirable under small drifts for practical reasons. FedDrift (Jothimurugesan et al., 2023) adopts global re-clustering and pays an amplified cost by having each client train on all cluster models. Auxo (Liu et al., 2023) and FedAC (Zhang et al., 2024) reduce overhead by only reassigning and adjusting cluster counts with clients selected for training, but ignore drifted clients that are not selected, reducing overall effectiveness (see Section 2.1).

The second challenge is to detect and adapt to various drifts (label shift, covariate shift, and concept drift as mentioned above). FL frameworks rely on client representations—client-side information used for clustering—to capture drifts. However, there is no single representation that is lightweight and works uniformly well for all drift types. We demonstrate representations' trade-offs and our framework's capability of handling diverse drifts in Section 2.1.

Our Contributions In this work, we present FIELDING, a CFL framework that handles multiple types of data drift with low overhead. FIELDING addresses the above challenges by making two key design choices: it combines per-client adjustment with selective global re-clustering, adapting to varying drift magnitudes without incurring the full cost of global re-clustering at every step; and it re-clusters all drifted clients using lightweight, drift-aware representations, keeping client-side computation and communication costs low while enabling efficient information collection from all clients. Our theoretical framework supports all representation choices and provides per-round utility guarantees along with convergence bounds, allowing FIELDING to effectively handle all three types of data drift.

To demonstrate the practicality of our approach, we extended the FedScale engine (Lai et al., 2022) to support streaming data and built a FIELDING prototype on top. We evaluated our framework on four image streaming traces with up to 5,078 clients under diverse drift patterns. FIELDING improves the final accuracy by up to 6.9% and reaches target accuracy as much as 3.10× faster than standard CFL approaches. Moreover, we show that FIELDING flexibly accommodates different drift types via pluggable client representations (e.g., label distribution vectors, input embeddings, and gradients), integrates seamlessly with a range of client selection and aggregation schemes, and remains robust against varying degrees of heterogeneity.

Finally, we provably bound the loss across iterations for FIELDING. In comparison, guarantees for existing results with both clustering and drift handling have various drawbacks, such as no convergence guarantees (Jothimurugesan et al., 2023) or guarantees for over-simplified settings (Duan et al., 2021). [1]

2 FIELDING

FIELDING is a CFL framework that handles multiple types of data drift with low overhead. Effective data drift management requires identifying clients that need reassignment, determining their appropriate clusters, and collecting information to support drift detection and client movement. We first outline the challenges, then present the design decisions of FIELDING, and conclude with a system overview.

2.1 Motivation and Design Choices

We analyze the limitations of existing clustering strategies under real-world drift to motivate the design of FIELDING. Our analysis uses the Functional Map of

[1] FIELDING source code and datasets are available at `https://github.com/SophiaLi06/FIELDING`.

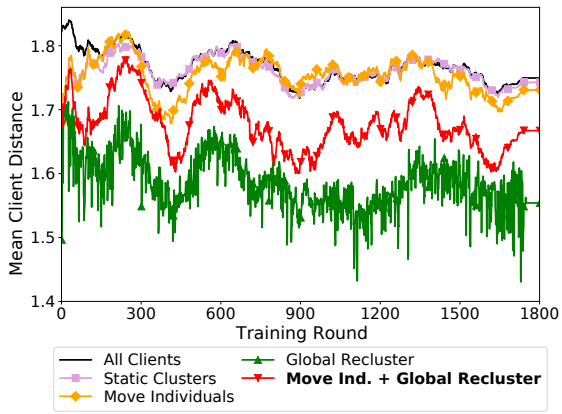
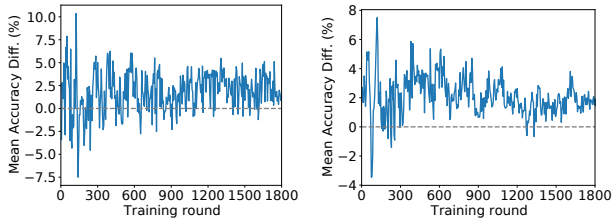


Figure 1: Client Heterogeneity of the Global Set and Clusters Generated Through Label-based Clustering.



(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.

Table 1: Drift-aware CFL Systems Comparison.

System	Which clients are re-clustered	Client representation
IFCA	Drifted	Loss
FlexCFL	Drifted	Model Updates
Auxo	Selected	Gradients
FedAC	Selected	Parameters
FedDrift	Global	Loss
Fielding	Drifted with Selective Global	User-defined

the World (FMoW) dataset (Christie et al., 2018), containing 302 clients (one per unique UTM zone) with time-stamped satellite images labeled by land use (e.g., airport, crop field). Two training rounds correspond to one day. More details on FMoW appear in Section 3. To quantify intra-cluster heterogeneity, we use *mean client distance* (Lai et al., 2021): for each client, we compute the average pairwise L1 distance between its data distribution and that of clients in the same cluster. We then take the mean across clients.

Per-client adjustment fails under many drift. Per-client adjustment methods, such as IFCA (Ghosh et al., 2020) and FlexCFL (Duan et al., 2021), maintain a fixed number of clusters and reassign drifted clients individually based on gradients or model weights. As mentioned in Section 1, such assignments become unstable and lead to poor clustering when many clients drift simultaneously. Figure 1 ("Move Individuals") highlights this behavior. Between rounds 600–730, a significant drift event increases intra-cluster heterogeneity, and from rounds 740–1340, per-client adjustment fails to restore effective clustering. In fact, heterogeneity exceeds that of the unclustered client set.

Global re-clustering is unstable during small drifts. Global re-clustering resets all client assignments to match current data and yields the lowest heterogeneity in our experiments (Figure 1, "Global Recluster"). However, due to the random initialization of k-means (Arthur and Vassilvitskii, 2006; Ahmed et al., 2020), resulting clusters can vary considerably across rounds even under minor drifts. Combined with cluster warm-up, this leads to test accuracy fluctuations. Figure 2a shows the accuracy difference between our selective re-clustering approach (described in Section 2.2) and a baseline that re-clusters globally after every drift event. The baseline exhibits unstable accuracy, sometimes matching ours but falling short by up to 5%.

Design choice: combine per-client adjustment with selective global re-clustering. Per-client adjustment breaks in highly dynamic settings with widespread drift. Meanwhile, global re-clustering can harm model accuracy by destabilizing cluster assignments and slowing convergence, particularly during minor drifts. To balance these trade-offs, Fielding combines both strategies: starting with individual client migration and triggering global re-clustering only when needed (see Section 2.2 for details). As shown by the red curve ("Move Ind. + Global Recluster") in Figure 1, this approach maintains low intra-cluster heterogeneity throughout training (see Section 3).

Clustering only selected clients reduces accuracy. Methods such as Auxo (Liu et al., 2023) and FedAC (Zhang et al., 2024) use training outputs—e.g., gradients or model parameters—from selected clients to re-cluster them without any additional cost. However, unselected clients do not provide such information, making it unclear whether they have drifted or where they should be reassigned to. As outlined in Section B.1, typical FL settings sample only a subset of clients in each round. When unselected clients drift but remain in their previous clusters, they may receive models trained on data with different distributions, reducing accuracy.

Figure 2b compares the mean test accuracy when re-clustering only selected clients versus all drifted clients in each round. Re-clustering all drifted clients yields a 1.5%–4% accuracy gain across most rounds—a meaningful improvement given the final FMoW accuracy of 52.4%. These results show that re-clustering only selected clients produces imperfect clusters containing misaligned clients.

Global re-clustering based on training outputs incurs high overheads. Handling all drifted clients requires input from every client, not just those selected for training. Depending on the representation, this method might incur significant overhead. In FlexCFL (Duan et al., 2021), clients train the global model locally and report gradients to the coordinator. In our experiments using ResNet-18 (He et al., 2016) on FMoW, clients spent on average 116.7 seconds for model download and gradient upload, and an additional 50.4 seconds running forward and backward passes on local data.

Design choice: efficient re-clustering of all drifted clients across drift types. To handle drift comprehensively and efficiently, FIELDING re-clusters all drifted clients, regardless of training participation and training outputs, using lightweight representations collected with minimal overhead.

Under a static feature–label relationship $P(y \mid x)$, input feature $P(x)$ shifts often align with label distribution $P(y)$ shifts, and hence label-distribution vectors capture both label and covariate shifts. For concept drift, prior work distinguishes *synchronized drift*, where all clients share a new class conditional distribution $P(y \mid x)$, and *distributed drift*, where clients can shift to different $P(y \mid x)$ (Panchal et al., 2023). Under synchronized drifts, changes in $P(y \mid x)$ also alter $P(y)$ when $P(x)$ is fixed, making it detectable through the shifts of label distribution $P(y)$. Given its versatility, we use the label distribution vector as the primary client representation in our evaluations. In rare scenarios when concept drift doesn't lead to label drift, clients can track training loss and request re-clustering when loss increases across rounds. This approach is adopted by FedDrift (Jothimurugesan et al., 2023) and supported by our theoretical result Theorem 4, which bounds the loss function increase under proper client clustering. Alternatively, and to capture distributed concept drift, one can use a representation that captures concept drift, and natural proxies are to track gradients or, as a lightweight proxy for concept representation, the input feature distribution per label.

Each representation has trade-offs: label and embedding vectors have low compute and storage overhead but might struggle with distributed concept drift; while gradient-based features demand extra computation, are noise-sensitive, and evolve with model updates: Table 7 shows that gradient-based clustering improves as training progresses and the model stabilizes. Selecting or combining these representations enables more reliable detection in all types of drift (see Section E.2).

Data privacy Reporting representations strongly correlated with local data characteristics – such as label distributions and input embeddings – poses potential privacy concerns and is preferable for domains in which such information is not security sensitive, e.g., when the model is not publicly available, the model usage and outputs are limited and controlled, and the central server is trusted. A trusted centralized server setting is common, since more secure differentially private solutions that protect user privacy even from the central server performing the clustering (i.e., locally-private) incur a significant loss (Chang and Kamath, 2021). For example, every locally-private algorithm for the k-means has an additive error no less than the square root of the number of data points (Stemmer, 2020).

Additionally, privacy concerns can be alleviated via anonymization techniques; this is especially effective in scenarios when the information held by different parties is disjoint and not inferable from public records (e.g., medical images). Finally, since FIELDING is agnostic to client representations, developers can choose differentially private ones. Both our framework and the theoretical guarantees allow such representations.

2.2 System Overview

FIELDING consists of two components: a centralized coordinator that manages client representations, clustering, and training orchestration, and a client module that tracks local representations and reports drift. As discussed in Section 2.1, neither per-client adjustment nor global re-clustering alone performs well across all drift scenarios. FIELDING adopts a hybrid approach: it performs per-client adjustment by default and triggers global re-clustering only when needed based on an adaptive intra-cluster heterogeneity threshold Δ. When clients detect drift, they send updated representations to the coordinator, which reassigns each drifted client to the nearest cluster without updating cluster centers to ensure deterministic results regardless of client processing order. After all adjustments, it recomputes pairwise distances within each cluster; if any exceed Δ, global re-clustering is triggered, otherwise the assignments are finalized. Figure 3 illustrates this clustering process using label-distribution vectors as representation. Clients submit their representations upon registration and whenever drift is detected. The coordinator updates client metadata and assigns each client to the nearest cluster. It then checks pairwise dis-

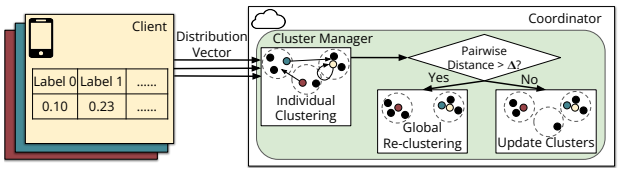


Figure 3: Clustering in FIELDING with Label Distribution as Representations. (Clients send distribution vectors to the coordinator, which moves drifted clients to the nearest cluster and triggers global re-clustering if any cluster has clients separated by more than Δ.)

tances before selecting participants for the next round to determine if global re-clustering is needed.

We design the heterogeneity threshold Δ to be adaptive via multiplicative increase/additive decrease with the consideration of avoiding unnecessary global re-clustering. We start with $\Delta = c$ (we set $c = 0.1$, which works well empirically). After each data drift event, we check whether global re-clustering has been triggered by two consecutive drift events. If so, we update Δ to $m \cdot \Delta$; otherwise, we update Δ to $\max(c, \Delta - c)$. This mechanism is inspired by congestion control algorithms (Chiu and Jain, 1989), and recent ML systems works have adopted similar ideas (Qiu et al., 2024). The multiplicative increase of Δ under consecutive global re-clustering makes FIELDING react fast to overly aggressive global re-clustering when the threshold is too low, and linear decrease, on one hand, maintains the correct Δ estimation for many iterations, and, on the other hand, eventually adapts if the Δ needs to be decreased. The Δ value we estimate remains within a factor of m of true Δ. We present results of the ablation study on m in Section E.1, and we choose $m = 2$ for our experiments.

2.3 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, with the distance between clients measured based on their representations. In particular, for label-based clustering, 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 $p_1, \ldots, p_l$ and $q_1, \ldots, q_l$ is defined as $\sum_{i=1}^{l} |p_i - q_i|$.

We assume that the algorithm runs for T global iterations. Our goal is, by the end of each iteration, to maintain a good model for each cluster, with respect to its clients' data. At each iteration, the label distribution of each client might change, e.g., by adding or removing data points, and the first step of FIELDING is

Algorithm 1 CFL with data drift

Partition clients into clusters using k-means clusters based on client representations
Initialize cluster models $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(K)}$
for *every iteration* $t = 0, \ldots, T-1$ **do**
  Handle data drift using Algorithm 2
  Let $C_1, \ldots, C_K$ be the clusters
  Let M be the number of machines sampled per iteration
  for *every cluster* $k = 1, \ldots, K$ *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 map

to handle data drift (Algorithm 2). In this step, each drifted client is assigned to the closest cluster. If this causes a significant change in intra-cluster heterogeneity – namely, if there exists two clients which belong to the same cluster and have pairwise distance exceeding Δ (see Section 2.2 for the discussion on how to adaptively choose Δ) – we recluster all clients from scratch. This condition avoids frequent reclustering, which might require excessive resources and can adversely change the clients' losses. Importantly, we recluster all clients, not just available clients, which, as shown in Section 2.1, improves accuracy. We choose the number of clusters that gives the highest silhouette score.

After global reclustering, we compute new cluster models: let $\mathbf{x}_i$ be the old cluster model for client i; 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 models of its clients. After the clusters are initializes, we train the cluster models for R rounds. In each round, we sample clients from the cluster and run L local iterations of gradient descent. After local iterations, we set the cluster model as the average of the models of all participating clients.

2.4 Convergence

In Section A, we analyze the performance of a simplified version of our framework by bounding the average of clients' loss functions at every iteration. Our proof treats Δ (the intra-cluster pairwise distance upper bound in our proof) as a constant to improve the presentation, but the proof could naturally handle different

Algorithm 2 Handling data drift

for *each drifting client* x **do**
  Assign x to the closest cluster center
  Recompute the centers of the affected clusters
Let $r^{(1)}, \ldots, r^{(N)}$ be clients' representations
if *there exist clients* i, j *from the same cluster with* $\|r^{(i)} - r^{(j)}\| > \Delta$ **then**
  for *each client* i **do**
    Let $\mathbf{x}_i$ be the model corresponding to the i'th client's cluster
  Let $C_1, \ldots, C_K$ be k-means clusters based on client representations
  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 based on their representations
Choose K with the largest silhouette score
Cluster the clients using the K-means clustering

Δ at different iterations. The major challenge in the analysis is bounding the adverse effects 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 can potentially be negative (see Figure 2b) due to the mixing of models from different clusters.

We make the following assumptions (see Section A for the precise statements and the discussion of the assumptions). First, we assume that the distance between client representations – for example, 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 clients are clusterable; that is, there exists K clusters so that representations of clients within each cluster are similar.

Assumption A. *Each* $f_t^{(i)}$ *– the local objective for client* i *at iteration* t *– is* L*-smooth and satisfies* μ*-Polyak-Łojasiewicz condition, 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 ratio between the difference between the objective function values and the distance between representations is at most* θ*.*

Under these assumptions, data drifts can affect each client representation – and hence the objective we optimize – by at most a fixed value. Moreover, by the clustering assumption, clients within each cluster have similar objectives after the reclustering, allowing us to bound the increase in the loss function due to reclustering. In Section A, we show the following result.

Table 2: The Min, Max, Mean, and Median Drift Magnitudes Before and After Each Drift Event. We Use L_1 Distance for Labels and L_2 Distance for Embeddings, with 2 Being the Maximum Possible Distance Value.

Dataset	Metric	Min	Max	Mean	Median
Cityscapes	Label	0.127	0.155	0.139	0.139
	Embedding	0.058	0.070	0.063	0.063
FMoW	Label	0.032	0.667	0.302	0.288
	Embedding	0.065	0.623	0.359	0.357
Waymo	Label	0.118	0.160	0.139	0.135
	Embedding	0.071	0.085	0.077	0.077
OpenImage	Label	1.130	1.484	1.179	1.135
	Embedding	0.493	0.504	0.496	0.495

Theorem 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*

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

Intuitively, at every iteration, we provide a "regret" bound comparing the loss at each cluster with the best loss we could have at each cluster. The bound has two terms: an exponentially decaying term corresponding to the initial loss, and a non-vanishing term corresponding to stochastic noise and the loss due to clustering and data drift. Ultimately, early on, the first term dominates, and we observe rapid improvement in objective value. On the other hand, at later stages, the first term vanishes, and the behavior is dictated by the stochastic and sampling noise, and severity of data drifts.

3 EVALUATION

We evaluate FIELDING on four FL tasks with label distribution vectors as client representations (see Section E.2 for results with distributed concept drift and other representations). End-to-end wise, FIELDING improves final test accuracy by 2.4%-6.9% and is more stable than prior CFL methods. It works well with complementary FL optimizations and still accelerates model convergence under low heterogeneity.

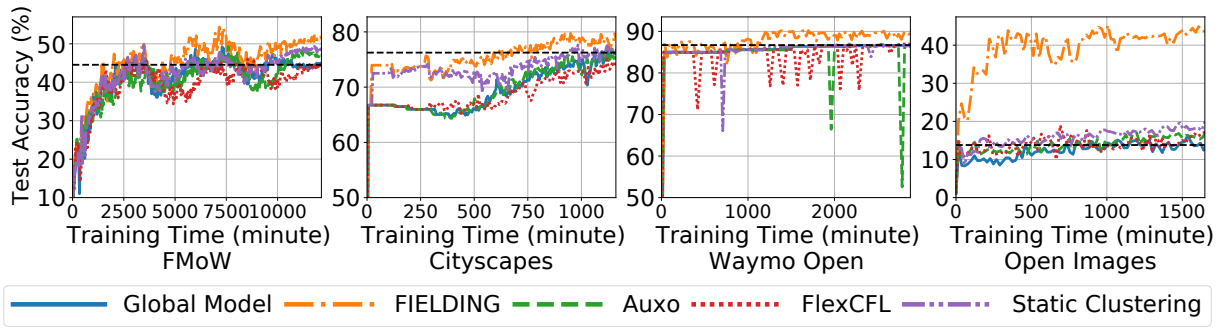


Figure 4: Time to Accuracy (TTA) Comparison Over Four Tasks.

Environment. We emulate large-scale FL training with two GPU servers, each with two NVIDIA A100 GPUs (80 GB memory) and two AMD EPYC 7313 16-core CPUs. We use FedScale's device datasets for realistic device computing and network capacity profiles while following the standardized training setup in the original paper (Lai et al., 2022). Per-device computation time is estimated from device computing speed, batch size, and number of local iterations; communication time is estimated based on downloaded and uploaded data volume and network bandwidth.

Datasets and models. We evaluate on one satellite dataset (FMoW (Christie et al., 2018)) and two video datasets (Waymo Open (Sun et al., 2020), Cityscapes (Cordts et al., 2016)) that exhibit natural drift. For FMoW, we keep images taken after January 1, 2015, and create one client per UTM zone. Two training rounds correspond to one day. For Waymo Open and Cityscapes, we use video segments pre-processed by Ekya (Bhardwaj et al., 2022), creating one client per camera per Waymo Open segment (212 clients) and one client per 100 consecutive frames per Cityscapes segment (217 clients). Since both datasets have video frame IDs within segments but lack global timestamps, we sort samples by frame ID and split them into 10 intervals. We stream in one data interval every 30 rounds for Cityscapes and every 20 rounds for Waymo Open. We train ResNet-18 (He et al., 2016) on Cityscapes and FMoW, and VisionTransformer-B16 (Dosovitskiy et al., 2021) on Waymo Open.

To evaluate Fielding under highly dynamic drift, we construct a synthetic trace using the FedScale Open Images (Kuznetsova et al., 2020) benchmark. We find the top 100 most frequent classes and retain clients with samples from at least 10 of these 100 classes (5078 clients in total). Each client randomly partitions local data *labels* into 10 buckets, with each bucket then containing all samples of its labels. We stream in one bucket every 50 rounds and use ShuffleNet v2 (Ma et al., 2018) for this task. For all four datasets, clients start with 100 rounds worth of data and retain samples from the most recent 100 rounds. The four datasets naturally contain both label and covariate shifts. We quantify drift magnitude using the average per-client label-distribution shift distance (L_1 distance) and embedding shift distance (L_2 distance between normalized embeddings) before and after each drift event and report them in Table 2. Dataset and training configurations are summarized in Table 4 and Table 5.

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 (Liu et al., 2023) as the continuous re-clustering baseline and FlexCFL (Duan et al., 2021) as the individual movement baseline. We employ FedProx (Li et al., 2020b), a federated optimization algorithm that tackles client heterogeneity, for all approaches.

Metrics. 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–the average client test accuracy when training one global model (black dashed lines in Figure 4). We report final average test accuracy across settings to demonstrate Fielding's sensitivity and robustness.

3.1 End-to-end Training Performance

Figure 4 shows that Fielding improves average client test accuracy by 6.9%, 2.4%, and 2.7% in FMoW, Cityscapes, and Waymo Open, respectively. These gains are substantial given the baseline accuracies of 45% to 85% and are comparable to those reported in prior CFL works on *static data*. When considering the global model's final test accuracy as the target accuracy, Fielding offers a 1.38×, 1.44×, and 3.10× training speed-up, respectively (we define the point at which Fielding's accuracy consistently surpasses the target accuracy as the moment it achieves the target accuracy). On the highly heterogeneous and dynamic Open Images trace, Fielding boosts accuracy by 26.4-

Table 3: Final Accuracy Gaps Between Static Clustering and FIELDING

Dataset	Static Clustering	FIELDING
Cityscapes	76.3%	78.8%
FMoW	48.4%	52.4%
Waymo	86.6%	89.4%
OpenImage	19.9%	44.0%

31.1% in the final 100 rounds. The effectiveness of drift handling is shown in Table 3, which reports the final accuracy gap between FIELDING and a static clustering baseline where clients are clustered once in the first round based on label distributions and the clusters remain static. FIELDING achieves 24.1% improvement on the OpenImage dataset.

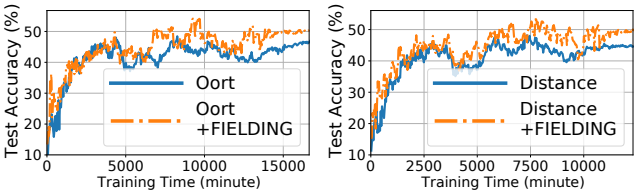
Auxo performs comparably or slightly better than the global model baseline, with up to 0.9% accuracy gain and 1.05× convergence speedup, but underperforms relative to the results in the original paper on static data, indicating reduced clustering effectiveness. This degradation stems from re-clustering only the selected participants, ignoring drifted but unselected clients. This also causes abrupt accuracy drops on the biased Waymo Open dataset (over 80% samples are cars) due to sudden model shifts when divergent clients are selected. FlexCFL shows no gains in accuracy or convergence across real-world datasets, as it migrates clients individually without adjusting the number of clusters. As Figure 1 shows, this approach leads to cluster heterogeneity approaching that of the global client set.

FedDrift Comparison. FedDrift (Jothimurugesan et al., 2023) is another clustered FL method designed to handle drift. However, it is impractical for large-scale settings, as it (1) requires every client to train every cluster model in each round and (2) assumes all clients remain online and available for training. Therefore, we construct a small-scale setting with the FMoW dataset and 50 clients (the number of clients we selected per round in our original experiment). Figure 7 shows that FIELDING incurs much shorter round time and reaches FedDrift's final accuracy 1.67× faster.

3.2 Compatibility with Other Optimizations

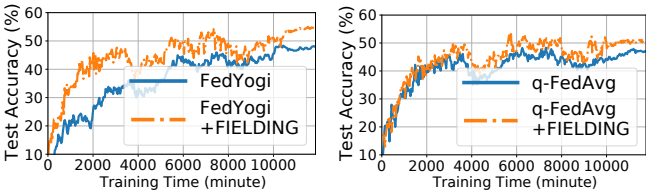
To demonstrate that FIELDING is agnostic to client selection and model aggregation strategies, we evaluate it on FMoW together with various complementary optimizations. In Figure 5 and Figure 6, we use the algorithm name alone to denote the baseline where we train one global model and use "+ FIELDING " to denote running the algorithm atop FIELDING.

We use the Oort selection algorithm (Lai et al., 2021)



(a) Oort selection (b) Distance-based selection

Figure 5: FIELDING with Client Selection Strategies.



(a) FedYogi (b) q-FedAvg

Figure 6: FIELDING with FL Algorithms.

and a distance-based algorithm prioritizing clients closer to the distribution center as client selection examples. As Figure 5 shows, FIELDING improves the final average test accuracy by 3.9% and 4.7% and reaches the target accuracy 1.32× and 1.15× faster. Note that Oort selection actively incorporates clients otherwise filtered out due to long response time, leading to a longer average round time. Figure 6 shows that FIELDING works well with aggregation algorithm FedYogi (Reddi et al., 2021) and q-FedAvg (Li et al., 2020c). FIELDING improves the final accuracy by 6.5% and 4.0% while giving a 1.20× and 1.24× speedup respectively.

3.3 Robustness and Sensitivity Analysis

Varying data heterogeneity. Our analysis on FIELDING's sensitivity to data heterogeneity degrees builds on the idea of mitigating non-IID data impacts with a small shared dataset (Zhao et al., 2018). We construct three datasets shared with all clients: one sample for each of the least represented 50% labels, one sample per label, and two samples per label. A larger shared set implies a smaller degree of heterogeneity. As shown in Figure 8, both FIELDING and the baseline global model benefit from this sharing, with FIELDING improving the final average test accuracy by 2.8% to 4.1%.

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

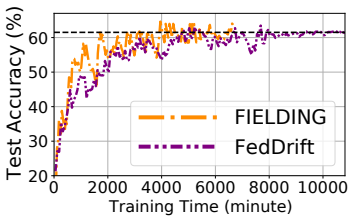


Figure 7: Small-scale Comparison Against FedDrift on FMoW.

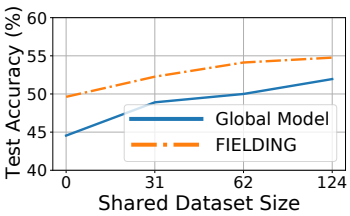


Figure 8: FIELDING Offers Gains Across Heterogeneity Degrees.

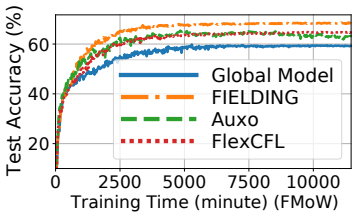


Figure 9: FIELDING Improves Model Accuracy on Static Data.

available throughout the training. As shown in Figure 9, 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.

4 RELATED WORK

Heterogeneity-aware FL. Recent works have also 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 (Lai et al., 2021). PyramidFL adjusts the number of local iterations and parameter dropouts to optimize participants' data and system efficiency (Li et al., 2022b). DSS-Edge-FL dynamically determines the size of local data used for training and selects representative samples, optimizing resource utilization while considering data heterogeneity (Serhani et al., 2023). MOON corrects local updates on clients by maximizing the similarity between representations learned by local models and the global model (Li et al., 2021).

Drifts-aware FL. Adaptive-FedAvg handles concept drift by extending FedAvg with a learning rate scheduler that considers the variance of the aggregated model between two consecutive rounds (Canonaco et al., 2021). CDA-FedAvg detects concept drifts and extends FedAvg with a short-term and a long-term memory for each client. It applies rehearsal using data in the long-term memory when drifts happen (Casado et al., 2022). Master-FL proposes a multi-scale algorithmic framework that trains clients across multiple time horizons with adaptive learning rates (Ganguly and Aggarwal, 2023). They are complementary to our work and can improve cluster models when minor drifts that don't trigger global re-clustering happen.

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

Clustered FL. FlexCFL (Duan et al., 2021) and IFCA (Ghosh et al., 2020) adjust clusters by relocating shifted clients while maintaining a fixed number of clusters, operating under the assumption of low drift magnitude. Auxo (Liu et al., 2023) and FedAC (Zhang et al., 2024) re-cluster selected clients using gradient information or local models, delaying drift adjustments for unselected clients. FedDrift (Jothimurugesan et al., 2023) re-clusters all clients by broadcasting all cluster models every round to compute client local training loss, leading to significant costs. Section B discusses other prior CFL works without drift handling and the impact of drifts on static clusters.

5 CONCLUSION

We presented FIELDING, a clustering-based FL framework for efficient handling of diverse data drifts. FIELDING combines per-client adjustment with selective global re-clustering, re-clusters all drifted clients, dynamically adjusts the number of clusters, and stabilizes the system's response to minor drifts. It is robust to varying levels of heterogeneity and is compatible with a range of client selection and aggregation strategies. In FIELDING, clients register with a centralized coordinator that maintains metadata and assigns clusters based on client representations. Global re-clustering is triggered only when intra-cluster heterogeneity exceeds a threshold, significantly reducing the overall re-clustering cost and mitigating loss spikes after re-clustering. Experiments show that FIELDING improves final test accuracy by 2.4%–6.9% and provides greater stability under real-world data drift.

As a potential future direction, FIELDING can be enhanced from the privacy perspective as discussed in Section 2.1.

6 ACKNOWLEDGEMENTS

We thank the AISTATS reviewers for their invaluable feedback. This work was supported in part by NSF Expedition 2326605 and ACE, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

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. doi:10.1016/j.segan.2024.101342

Charu C Aggarwal et al. 2015. *Data mining: the textbook*. Vol. 1. Springer.

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.

David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding*. Technical Report. Stanford.

Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. 2022. Ekya: Continuous Learning of Video Analytics Models on Edge Compute Servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 119–135. https://www.usenix.org/conference/nsdi22/presentation/bhardwaj

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 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.

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.

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* 81, 3 (2022), 3397–3419. doi:10.1007/s11042-021-11219-x

Alisa Chang and Pritish Kamath. 2021. Practical Differentially Private Clustering. Google Research Blog. https://research.google/blog/practical-differentially-private-clustering Accessed: 2026-03-23.

Yujing Chen, Yue Ning, Martin Slawski, and Huzefa 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.

Dah-Ming Chiu and Raj Jain. 1989. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems* 17, 1 (1989), 1–14.

Yae Jee Cho, Pranay Sharma, Gauri Joshi, Zheng Xu, Satyen Kale, and Tong Zhang. 2023-07-23/2023-07-29. On the Convergence of Federated Averaging with Cyclic Client Participation. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 5677–5721.

Yae Jee Cho, Jianyu Wang, and Gauri Joshi. 2020. Client Selection in Federated Learning: Convergence Analysis and Power-of-Choice Selection Strategies. arXiv:2010.01243 [cs, stat] doi:10.48550/arXiv.2010.01243

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.

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)*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy

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.

Lidia Fantauzzo, Eros Fanì, Debora Caldarola, Antonio Tavera, Fabio Cermelli, Marco Ciccone, and Barbara Caputo. 2022. Feddrive: Generalizing federated learning to semantic segmentation in autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 11504–11511.

Bhargav Ganguly and Vaneet Aggarwal. 2023. Online Federated Learning via Non-Stationary Detection and Adaptation Amidst Concept Drift. *IEEE/ACM Transactions on Networking* (2023).

Saurabh Garg, Yifan Wu, Sivaraman Balakrishnan, and Zachary Lipton. 2020. A unified view of label shift estimation. *Advances in Neural Information Processing Systems* 33 (2020), 3290–3300.

Avishek 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.

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.

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.

Chip Huyen. 2022. *Designing Machine Learning Systems*. O'Reilly Media, USA.

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.

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.

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.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, 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.

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.

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.

Hunmin Lee, Yueyang Liu, Donghyun Kim, and Yingshu Li. 2021. Robust convergence in federated learning through label-wise clustering. *arXiv preprint arXiv:2112.14244* (2021).

Hunmin Lee and Daehee Seo. 2023. FedLC: Optimizing Federated Learning in Non-IID Data via Label-Wise Clustering. *IEEE Access* 11 (2023), 42082–42095. doi:10.1109/ACCESS.2023.3271517

Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. 2022b. 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.

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.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020b. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze (Eds.), Vol. 2. 429–450.

Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2020c. Fair Resource Allocation in Federated Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=ByexElSYDr

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020a. 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.

Zonghang Li, Yihong He, Hongfang Yu, Jiawen Kang, Xiaoping Li, Zenglin Xu, and Dusit Niyato. 2022a. Data heterogeneity-robust federated learning via group client selection in industrial IoT. *IEEE Internet of Things Journal* 9, 18 (2022), 17844–17857.

Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151. doi:10.1109/18.61115

Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. 2018. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*. PMLR, 3122–3130.

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.

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.

Ankur Mallick, Kevin Hsieh, Behnaz Arzani, and Gauri Joshi. 2022. Matchmaker: Data drift mitigation in machine learning for large-scale systems. *Proceedings of Machine Learning and Systems* 4 (2022), 77–94.

Y. Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three Approaches for Personalization with Applications to Federated Learning. *ArXiv* abs/2002.10619 (2020). https://api.semanticscholar.org/CorpusID:211296702

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.

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.

Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. 2012. A unifying view on dataset shift in classification. *Pattern recognition* 45, 1 (2012), 521–530.

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.

Kunjal Panchal, Sunav Choudhary, Subrata Mitra, Koyel Mukherjee, Somdeb Sarkhel, Saayan Mitra, and Hui Guan. 2023. Flash: Concept Drift Adaptation in Federated Learning. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 26931–26962. https://proceedings.mlr.press/v202/panchal23a.html

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.

Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew Kalbarczyk, Tamer Başar, and Ravishankar K Iyer. 2024. Power-aware deep learning model serving with {μ-Serve}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 75–93.

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*.

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.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2021. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (2021), 3710–3722. doi:10.1109/TNNLS.2020.3015958

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. doi:10.1109/TNNLS.2019.2944481

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. doi:10.1109/OJCOMS.2023.3313257

Xiaofeng Shi, Matthew Osinski, Chen Qian, and Jia Wang. 2022. Towards automatic troubleshooting for

user-level performance degradation in cellular services. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 716–728.

Uri Stemmer. 2020. Locally private k-means clustering. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms* (Salt Lake City, Utah) *(SODA '20)*. Society for Industrial and Applied Mathematics, USA, 548–559.

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)*.

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.

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.

Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu. 2021. Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data. In *Proceedings of the Web Conference 2021*. 935–946.

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).

Jie Zhang, Zhiqi Li, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Chao Wu. 2022. Federated Learning with Label Distribution Skew via Logits Calibration. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 26311–26329. `https://proceedings.mlr.press/v162/zhang22p.html`

Jianfei Zhang and Shuaishuai Lv. 2021. FedLabCluster: A Clustered Federated Learning Algorithm Based on Data Sample Label. In *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*. 423–428. doi:`10.1109/EIECS53707.2021.9588126`

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).

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).

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.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [**Yes**/No/Not Applicable] Mathematical setting, assumptions, proof, and algorithms are provided in Section 2.3, Section 2.4, and Appendix A.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [**Yes**/No/Not Applicable] The algorithm complexity analysis is in Section C.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [**Yes**/No/Not Applicable] Our code is available at `https://anonymous.4open.science/r/AISTATS-A7B9/` with installation instructions (including dependencies and external libraries).

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [**Yes**/No/Not Applicable] Assumptions are provided in Section 2.4 and Appendix A. All assumptions are clearly stated.
   (b) Complete proofs of all theoretical results. [**Yes**/No/Not Applicable] Proofs are provided in Section 2.4 and Appendix A.
   (c) Clear explanations of any assumptions. [**Yes**/No/Not Applicable] All assumptions are clearly explained.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [**Yes**/No/Not Applicable] Our code is available at `https://github.com/SophiaLi06/FIELDING` with installation instructions, data, and scripts to reproduce the results.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [**Yes**/No/Not Applicable] The training details are in Appendix D.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [**Yes**/No/Not Applicable] We describe the specific measure in Section 3.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [**Yes**/No/Not Applicable] We describe the infrastructure in Section 3.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [**Yes**/No/Not Applicable] All assets are properly cited.
   (b) The license information of the assets, if applicable. [**Yes**/No/Not Applicable] License information are provided in Appendix D.
   (c) New assets either in the supplemental material or as a URL, if applicable. [**Yes**/No/Not Applicable] We uploaded new assets to an anonymous Google Drive `https://drive.google.com/drive/folders/1KzyvdQzZh2ruRNv9X3vwnbG0v8x_OGWT`. Detailed descriptions of the assets are in our code base `https://github.com/SophiaLi06/FIELDING`.
   (d) Information about consent from data providers/curators. [Yes/No/**Not Applicable**]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/**Not Applicable**]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Yes/No/**Not Applicable**]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/**Not Applicable**]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/**Not Applicable**]

Supplementary Materials

A PROOF OF CONVERGENCE

Algorithm 4 Clustered Federated Learning with Data Drift

input: the number of clusters 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, reclustering threshold Δ

Initialize cluster models $\mathbf{c}_0^{(1)}, \dots, \mathbf{c}_0^{(K)}$ with the same model $\mathbf{x}_0$

for $t = 0, \dots, T-1$ **do**

  for *each client* i **do**

    Adopt data drift for client i

    Reassign client i to the closest cluster

    Let $\mathbf{x}_i$ be the model corresponding to the cluster containing client i

  Let $r_t^{(1)}, r_t^{(2)}, \dots$ be the client representations

  if *there exists a cluster* C *such that* $\rho(r_t^{(i)}, r_t^{(j)}) > \Delta$ *for clients* $i, j \in C$ **then**

    Let $C_1, \dots, C_K$ be the K-center clustering of all clients based on representations

  else

    Let $C_1, \dots, C_K$ be the current clusters

  for $k = 1, \dots, K$ **do**

    $\tilde{\mathbf{c}}_0^{(k)} \leftarrow \operatorname{avg}_{i \in C_k} \mathbf{x}_i$ `// Cluster model is the average of clients' models`

  for $\tau = 0, \dots, R-1$ **do**

    for $k = 1, \dots, K$ **do**

      Sample subset S from C_k of size M/K independently with replacement

      $\tilde{\mathbf{c}}_{\tau+1}^{(k)} \leftarrow \tilde{\mathbf{c}}_\tau^{(k)} - \eta \operatorname{avg}_{i \in S} G_t^{(i)}(\tilde{\mathbf{c}}_\tau^{(k)})$

  for $k = 1, \dots, K$ **do**

    $\mathbf{c}_{t+1}^{(k)} \leftarrow \tilde{\mathbf{c}}_R^{(k)}$

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. (2020a). To simplify the proof, in the algorithm we use the k-center clustering instead of k-means clustering.

We present the precise algorithm in Algorithm 4. After every data drift event, we first accept data changes for each client. After that, if the heterogeneity within some cluster exceeds a certain threshold, 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, for all i and t:*

- *$f_t^{(i)}$ is L-smooth: $\|\nabla f_t^{(i)}(\mathbf{x}) - \nabla f_t^{(i)}(\mathbf{x})\| \leq L \|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y}$*
- *$f_t^{(i)}$ satisfies μ-Polyak-Łojasiewicz (μ-PL) condition: $\|f_t^{(i)}(\mathbf{x})\|^2 \geq 2\mu(f_t^{(i)}(\mathbf{x}) - f_t^{(i)}(\mathbf{x}^*))$, where $\mathbf{x}^*$ is the minimizer of $f_t^{(i)}$.*

The μ-PL condition is a relaxation of the strong convexity assumption; both assumptions are standard in the

federated learning literature (Cho et al., 7 29; Li et al., 2020a; Cho et al., 2020). The following are standard assumptions on stochastic gradient.

Assumption C (Stochastic gradients)**.** *For each client i, at every data drift event t, let $G_t^{(i)}$ be the stochastic gradient oracle:*

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

The next assumption connects the objective value with client representations. Client representations can take various forms, such as label distributions, input embeddings, or gradients.

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 θ such that $\left|f_t^{(i)}(\mathbf{x}) - f_t^{(j)}(\mathbf{x})\right| \leq \theta \cdot \rho\left(r_t^{(i)}, r_t^{(j)}\right)$ for all i, j, t.*

We next give the intuition of why the assumption is natural for the label-based and the embedding-based distances. If the concept $P(y \mid x)$ doesn't change between clients, every model achieves the same expected error on each class. For each class, assuming that the class data for different clients is sampled from the same distribution, we get the same expected loss on this class for different clients. Hence, the overall difference in the loss function is largely determined by the fraction of each class in the clients' data. Similarly, if the client embedding faithfully captures information about its inputs, assuming the trained models depend on the inputs continuously, they have the same expected error on similar inputs.

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: $\rho\left(r_t^{(i)}, r_{t+1}^{(i)}\right) \leq \delta$ for all i, t.*

This assumption naturally holds in case when, for each client, only a small fraction of data points changes at every round. 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 $C_1, \ldots, C_K$ so that, for any $k \in [K]$ and any $i, j \in C_k$, we have $\rho\left(r_t^{(i)}, r_t^{(j)}\right) \leq \Delta$.*

Assumption Discussion To summarize, we have the following assumptions: standard convergence assumptions (B and C) and assumptions necessary to conceivably get advantage of clustering (D, E, and F).

Assumption B states two properties: smoothness and PL-condition. Smoothness is a ubiquitous assumption in both convex and non-convex settings; when it's not present in some works, it's replaced with comparably strong assumptions (such as bounded gradients). PL condition is a relaxation of strong convexity and allows achieving linear convergence rate w.r.t. the objective value; this assumption is not strictly required, but its absence results in slower convergence rates.

Assumption C is a standard assumption on stochastic gradients. Somewhat weaker assumptions are possible, but such complications are not the focus of the paper.

Assumption D claims that the close representations translate to close loss function values. Existing clustered FL algorithms rely on some form of representation (which might include gradients or the loss function itself), and the representation must be good to take advantage of clustering. In case when the assumption is not satisfied, the clustering quality would suffer; the model will still be able to train, but one would expect the convergence to be similar to that without clustering.

Assumption E states that the client data doesn't change wildly after the drift, which allows reclustering to utilize some information before the drift. If this assumption is not satisfied, then, in general, re-training after each drift might be no better than training from scratch.

Assumption F assumes that there is a non-trivial clustering structure among clients. If the assumption is not satisfied, our algorithm still works, but might provide no tangible benefit over approaches without clustering. This is expected since the problem has no structure for our (or other clustered FL) algorithm to utilize.

Finally, these assumptions are not strict requirements: our bounds degrade gracefully with the degree of their violation.

Proofs For completeness, we present the analysis of SGD convergence for functions satisfying μ-PL condition.

Lemma 2. *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*

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

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

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

where in the last inequality we used $\eta \leq 1/L$. Using the fact that f satisfies μ-PL condition, we have $\left\|\nabla f(\mathbf{x}_t)\right\|^2 \geq 2\mu(f(\mathbf{x}_t) - f(\mathbf{x}^*))$, giving

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

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

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

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

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

The above result demonstrates how the objective improves with each round: namely, after R iterations, 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 3. *Let N be the number of clients, and $t+1$ be a fixed data drift event. Let $C_1, \ldots, C_k$ be clusters $\mathbf{c}_1, \ldots, \mathbf{c}_K$ be cluster models, and $\mathbf{c}_1^*, \ldots, \mathbf{c}_K^*$ be the optimal cluster models at the end of processing data drift event t. Similarly, let $\bar{C}_1, \ldots, \bar{C}_\ell$ be clusters $\bar{\mathbf{c}}_1, \ldots, \bar{\mathbf{c}}_K$ be cluster models, and $\bar{\mathbf{c}}_1^*, \ldots, \bar{\mathbf{c}}_K^*$ be the optimal cluster models immediately after reclustering. Then, under Assumptions B-F, 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

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

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,

$$f_{t+1}^{(i)}(\bar{\mathbf{c}}_\ell) = f_{t+1}^{(i)}\Big(\operatorname*{avg}_{j\in\bar{C}_\ell} \mathbf{x}_j\Big) \leq \operatorname*{avg}_{j\in\bar{C}_\ell} f_{t+1}^{(i)}(\mathbf{x}_j), \tag{1}$$

where the last inequality follows by convexity of $f_t^{(i)}$. By Assumption F, since all clients $\bar{C}_\ell$ 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 (1) 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.

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. □

Combining the above statements leads to our main convergence result.

Theorem 4. *Let N be the number of clients, and let their objective functions satisfy Assumptions B-F. 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 4 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 3, 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

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

By Lemma 2, 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. □

B ADDITIONAL BACKGROUND

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

B.1 Clustered Federated Learning

FL deployment often involves hundreds to thousands of available clients participating in the training (Bonawitz et al., 2019). For instance, Huang et al. (2019) have developed mortality and stay time predictors for the eICU collaborative research database (Pollard et al., 2018) 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 only a subset for training. For example, Google sets 100 as the target number of clients per training round when improving keyboard search suggestions (Yang et al., 2018). This selection process ensures a reasonable training time and accounts for diminishing returns, where including more clients does not accelerate convergence (Xu et al., 2024).

Data heterogeneity is a major challenge in FL, as variations in local data volume and distribution among participants can lead to slow convergence and reduced model accuracy (Yang et al., 2021; Wen et al., 2023). Clustering is an effective strategy to mitigate the impact of heterogeneity by grouping statistically similar clients. Clients within the same cluster collaboratively train one cluster model, which then serves their inference requests. Prior works demonstrate that clustering achieve high accuracy and fast model convergence (Sattler et al., 2021; Mansour et al., 2020; Ghosh et al., 2020; Liu et al., 2023; Jothimurugesan et al., 2023).

Our experiments with the Fielding system prototype mainly use label distribution vectors as the client representation, as they handle both label and covariate shifts and are lightweight. Although previous studies have examined label distribution vectors as leverage to address heterogeneity (Zhang and Lv, 2021; Zhang et al., 2022; Lee and Seo, 2023; Lee et al., 2021), Fielding distinguishes itself from existing works. Federated learning via Logits Calibration (Zhang et al., 2022) and FedLC (Lee and Seo, 2023) are orthogonal and investigate building a better global model through client model aggregation and loss function improvements. FedLabCluster (Zhang and Lv, 2021) and Lee et al. (2021) do not explicitly discuss data drift handling. Moreover, we present a theoretical framework with a convergence guarantee under data drift and re-clustering.

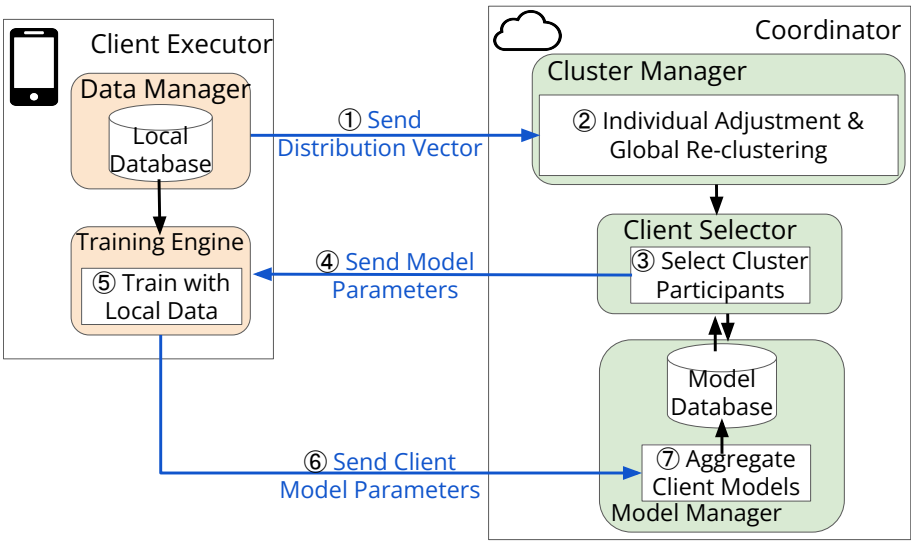


Figure 10: The System Architecture and Workflow of FIELDING. (Blue arrows represent communication between clients and the coordinator. Black arrows indicate the workflow among different components on the same machine.)

B.2 Data Drift Affects Static Clustering

Drifts happen naturally when clients have access to non-stationary streaming data (e.g., cameras on vehicles and virtual keyboards on phones) (Koh et al., 2021; Bhardwaj et al., 2022; Chen et al., 2020; Nandi and Xhafa, 2022; Marfoq et al., 2023). 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 drift reduces the effectiveness of clustered FL as it increases data heterogeneity within clusters. We demonstrate this problem with the Functional Map of the World (FMoW) dataset (Christie et al., 2018). Recall from Section 3 that our preprocessed FmoW dataset contains time-stamped satellite images labeled taken on or after January 1, 2015, and each unique UTM zone metadata value corresponds to one client (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.

As mentioned in Section 2, we use *mean client distance* as the intra-cluster heterogeneity. 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 described above. Figure 1 shows that initially (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 over time, static clustering increases per-cluster heterogeneity and soon gets close to the no clustering case at round 322.

C IMPLEMENTATION

We implemented FIELDING in Python on top of FedScale (Lai et al., 2022), a state-of-the-art open-source FL engine. We follow FedScale's design of having one centralized coordinator and client-specific executors. Figure 10 illustrates the end-to-end workflow of the FIELDING system prototype with label distribution vectors as client representations. 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 K-means clustering and determine K using the silhouette method (Rousseeuw, 1987). 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 on the coordinator are mainly determined by the number and size of client representations we need for clustering. In our largest evaluation setting where we use label distribution vectors as client representation and 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. This overhead is small relative to training time, with the simulated round durations (using real device compute and bandwidth profiles) ranging from 3.0 to 7.2 minutes on the OpenImage trace. Storing the latest reported distribution vector for each client consumes $5078 \times 100 \times 4\text{B} \approx 1.9\text{MB}$ of memory. The representation storage cost scales linearly with the number of coordinates per representation. Given that datasets typically have up to thousands of labels, and prior work has shown success in clustering clients with dimensionally reduced model updates (Duan et al., 2021), we assume that per-client representation has 10,000 coordinates. Then, with 256 GB memory, a typical RAM size of an enterprise production server, we can accommodate the representations of 6.4 million clients. The computational cost is dominated by the K-means clustering assignment step, where we compute the distance to each cluster center for every client and pay the overall computational complexity of $O(k \cdot n \cdot d)$ for k clusters, n clients, and representation dimensionality d (Aggarwal et al., 2015). Previous work shows that intra-cluster heterogeneity reduction sees diminishing returns as we increase k beyond 10 (Liu et al., 2023). In our evaluation, the largest k found through the search for the highest silhouette score is 9. Thus, we assume that the upper bound k is 10. Together with the previous assumption that a client representation has up to 10,000 coordinates, the computational complexity scales linearly with the number of clients n. On our testbed server with AMD EPYC 7313 16-core CPUs, single-thread K-means clustering with $k = 10$, $n = 100,000$, and $d = 10,000$ takes 477.8 seconds. A multithreaded implemention will further reduce this overhead.

D EXPERIMENT SETTINGS

We conduct experiments on public datasets Functional Map of the World (FMoW) (Christie et al., 2018), Cityscapes (Cordts et al., 2016), Waymo Open (Sun et al., 2020), and Open Images (Kuznetsova et al., 2020). In Table 4, 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 3, 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 5, 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 4: 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

The URL, version information, and license of datasets we used are as follows:

- FMoW: `s3://spacenet-dataset/Hosted-Datasets/fmow/fmow-rgb/`. fMoW-rgb version. This data is

Table 5: Training Parameters.

Dataset	Total Rounds	Learning Rate	Batch Size	Number of Local Steps	Participants 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

Table 6: Ablation Study on the Multiplicative Factor (Denoted as m) in Our Adaptive Approach.

Multiplicative factor	1.1	1.5	2.0	2.5	3.0
FMoW final accuracy	53.0%	53.2%	52.4%	52.1%	49.9%
Number of global clustering	479	177	92	65	52

licensed under the Functional Map of the World Challenge Public License.

- Cityscapes: `https://www.cityscapes-dataset.com/file-handling/?packageID=3` and `https://www.cityscapes-dataset.com/file-handling/?packageID=1`. Fine annotation version (5000 frames in total). The dataset is released under Cityscapes' custom terms and conditions.
- Waymo Open: `https://console.cloud.google.com/storage/browser/waymo_open_dataset_v_1_0_0`. Perception Dataset v1.0, August 2019: Initial release. The dataset is released under Waymo Dataset License Agreement for Non-Commercial Use.
- Open Images: `https://fedscale.eecs.umich.edu/dataset/openImage.tar.gz`. Open Images (V7) Preprocessed by FedScale. The original Open Images dataset annotations are licensed by Google LLC under CC BY 4.0 license.

E ADDITIONAL EXPERIMENTS

E.1 Ablation Study on the Adaptive Intra-cluster Heterogeneity Threshold

Recall that we design the intra-cluster heterogeneity threshold Δ of FIELDING to be adaptive. It is initialized as $\Delta = c$ (we set $c = 0.1$, which works well empirically). After each drift event, if global re-clustering has been triggered in two consecutive events, we update Δ to $m \cdot \Delta$; otherwise, we update Δ to $\max(c, \Delta - c)$. The multiplicative factor m in our adaptive approach is tunable. We set it as $m = 2$, which works empirically well in our evaluations. A smaller m leads to a lower worst-case deviation from the true value of Δ, but also causes more global re-clusterings. We conducted an additional ablation study on m using the FMoW dataset. As the results in Table 6 show, a smaller m generates a higher final accuracy, but notably increases the number of global clusterings. Compared to $m = 1.1$ and $m = 1.5$, setting $m = 2$ has an up to 0.8% lower accuracy but triggers global re-clustering 5.2× and 1.9× less frequently, respectively.

E.2 Representations Comparison

In this section, we empirically evaluate our framework with various choices of representations.

E.2.1 Gradients as Representation

An issue with gradient-based re-clustering is that the clustering effectiveness is sensitive to the model quality. Table 7 presents the resulting average pairwise distribution vector L1 distance and embedding squared Euclidean distance when we perform gradient-based clustering after training a global model for various numbers of rounds on FMoW. The numbers in parentheses indicate the change in average 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 as heterogeneous as the global set. As we postpone clustering to later rounds,

Table 7: Heterogeneity of All clients and Intra-cluster Heterogeneity After Clustering (lower is better) on FMoW at Various Rounds. (The left value denotes the pairwise L1 distance of distribution vectors, while the right represents the pairwise embedding squared Euclidean distance. Early on, gradient-based clustering is less effective due to the model being unstable, whereas label-based clustering decreases heterogeneity consistently. Notably, smaller distribution distances generally align with smaller embedding distances, suggesting that optimizing the label-based clustering objective also generates good embedding-based clusters.)

Total Rounds	Un-Clustered		Gradient-Based Clustering		Label-Based Clustering	
100	1.81	46.33	1.82(+0.6%)	46.30(-0.06%)	1.65(-8.8%)	43.02(-7.14%)
200	1.80	42.75	1.82(+1.1%)	42.53(-0.51%)	1.64(-8.9%)	41.52(-2.88%)
500	1.78	36.99	1.71(-3.9%)	35.82(-3.16%)	1.62(-9.0%)	35.55(-3.89%)
1000	1.74	31.85	1.66(-4.6%)	30.70(-3.61%)	1.56(-10.3%)	29.97(-5.90%)
1500	1.76	32.35	1.63(-7.4%)	30.68(-5.16%)	1.60(-9.1%)	30.76(-4.91%)

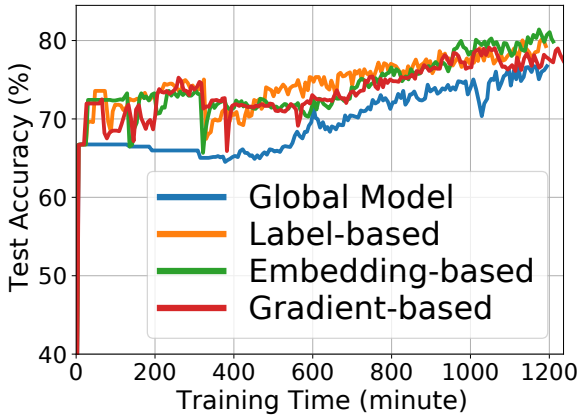


Figure 11: Fielding Achieves High Accuracy Across Client Representations on Cityscapes.

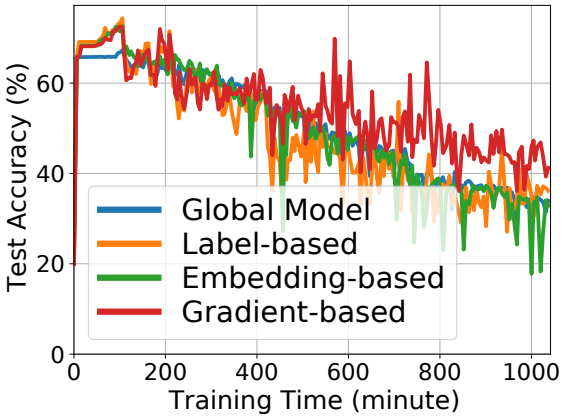


Figure 12: Gradient-based Clustering Handles Concept Drift Better.

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 we use to collect gradients from all clients has not converged.

As mentioned in Section 2, we propose using label distribution vectors as client representations when we deal with label and covariate shifts. 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 7, 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 model update (which has the same number of coordinates as the gradient)-based solution FlexCFL takes 37.92 seconds (note that FlexCFL already accelerates this process through dimensionality reduction using truncated Singular Value Decomposition).

In terms of overhead, both embedding- and gradient-based clustering introduce client-side computation, which can be mitigated by deploying a smaller shared model for representation collection, eliminating extra model download time. We adopt this approach when we gather the time-to-accuracy results of training ResNet-18 on Cityscapes with various representations (shown in Figure 11). We first construct a small training set by randomly sampling 200 images from each class. We then train a ResNet-18 model on this dataset for 300 epochs and broadcasts it so that clients store it locally and use it for gradient and embedding generation. Note that gradient-based Fielding still takes longer to finish 200 training rounds than other variants in Figure 11 due to the longer computational time of back propagation and longer transmission time of gradient vectors.

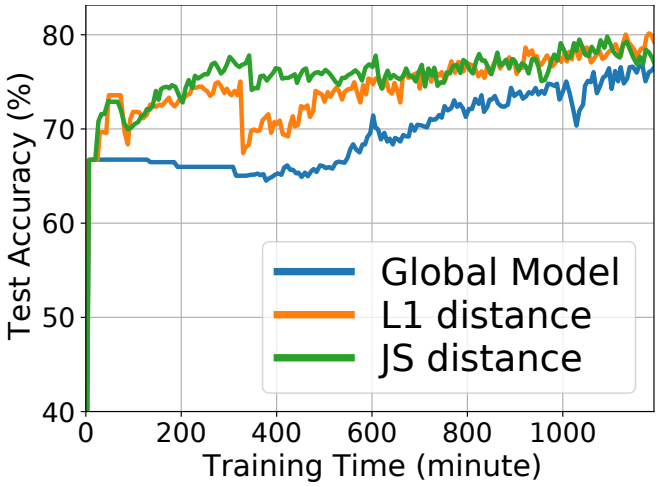


Figure 13: FIELDING Performance with Different Distance Metrics.

Handling concept drifts where $P(y|x)$ changes requires loss-based representations, making gradients more appropriate than label distribution vectors or embeddings. Figure 12 shows the time-to-accuracy results with synthesized concept drifts on Cityscapes. Here we make all data samples available throughout the training, but introduce drift events by randomly choosing 50% clients and having each chosen client randomly pick two labels and swap their samples (i.e., if a client picks label A and B, then all samples previously labeled A are now labeled B, and samples labeled B now have label A). Gradient-based FIELDING manages to retain test accuracy under such aggressive concept drift, while label and embedding-based FIELDING have similar results as the baseline without any clustering. This result highlights that gradient has the potential of addressing concept drifts while label distribution and embedding don't.

E.2.2 Input Embeddings as Representations

Since covariate shift is defined as input distribution ($P(x)$) changing while the feature-to-label mapping ($P(y|x)$) remains constant, input embedding distance should be a proxy for label distribution distance. Hence, minimizing the distance between input embeddings naturally minimizes the label distribution distance as well. Table 7 supports this intuition by showing a correlation between the label distribution distance and the embedding distance. This observation suggests that using label distribution vectors or input embeddings as representations should have comparable performance on labeled datasets. As shown in Figure 11, both label distribution vectors and embeddings enable FIELDING to outperform the no-clustering baseline (the sudden accuracy drop of the embedding-based curve around 150 minutes is the result of global re-clustering). Input embeddings have the potential of capturing label and covariate shifts on unlabeled data, which label distribution vectors are incapable of.

E.3 Supporting Diverse Distance Metric

To demonstrate that FIELDING is not tied to specific distance metric, we run the Cityscapes task with label distribution vectors as client representations, and both L1 and Jensen–Shannon distance (Lin, 1991) as the distance metric. Figure 13 shows that FIELDING is compatible with different distance metrics chosen for a given client representation.