Federated Learning under Distributed Concept Drift

Ellango Jothimurugesan
Carnegie Mellon University

Kevin Hsieh
Microsoft Research

Jianyu Wang[1]
Carnegie Mellon University

Gauri Joshi
Carnegie Mellon University

Phillip B. Gibbons
Carnegie Mellon University

Abstract

Federated Learning (FL) under distributed concept drift is a largely unexplored area. Although concept drift is itself a well-studied phenomenon, it poses particular challenges for FL, because drifts arise staggered in time and space (across clients). To the best of our knowledge, this work is the first to explicitly study data heterogeneity in both dimensions. We first demonstrate that prior solutions to drift adaptation that use a single global model are ill-suited to staggered drifts, necessitating multiple-model solutions. We identify the problem of drift adaptation as a time-varying clustering problem, and we propose two new clustering algorithms for reacting to drifts based on local drift detection and hierarchical clustering. Empirical evaluation shows that our solutions achieve significantly higher accuracy than existing baselines, and are comparable to an idealized algorithm with oracle knowledge of the ground-truth clustering of clients to concepts at each time step.

1 INTRODUCTION

Federated learning (FL) (Konečný et al., 2016; McMahan et al., 2017) is a popular machine learning (ML) paradigm that enables collaborative training without sharing raw training data. FL is crucial in the era of pervasive computing, where massive IoT and mobile phones continuously generate relevant ML data that cannot be easily shared due to privacy and communication constraints. FL also enables different organizations such as hospitals (Rieke et al., 2020) and retail stores (Yang et al., 2019) to jointly obtain valuable insights while preserving data privacy. FL has become an important technology in the real world with massive deployments (500+ million installations on Android devices) as well as a growing market with many solution providers (MarketsAndMarkets, 2021).

Existing FL solutions generally assume the training data comes from a *stable* underlying distribution, and the training data in the past is sufficiently similar to the test data in the future. Unfortunately, this assumption is often violated in the real world, where the underlying data distribution is non-stationary and constantly evolves. For instance, user sentiment and preference change drastically due to external environments such as the pandemic and macroeconomics (Koh et al., 2021; Garg et al., 2021). Data collected by cameras are also subject to various data changes such as unexpected weather and novel objects, which can lead to significant ML model performance losses (Suprem et al., 2020; Bhardwaj et al., 2022; Khani et al., 2023).

This *concept drift* problem (defined in §2.1) has been studied extensively in a centralized learning environment (Gama et al., 2014; Tahmasbi et al., 2021; Mallick et al., 2022). These centralized solutions, however, cannot address the fundamental challenges of concept drifts in FL where data is heterogeneous over *time* and across different *clients*. When different clients experience the data drift *at different times*, no single global model can perform well for all clients. Similarly, when *multiple concepts exist simultaneously*, no centralized training decision works well for all clients. Several recent works have recognized the problem of FL under concept drift and proposed solutions that adapt learning rates or add regularization terms (Chen et al., 2021; Manias et al., 2021; Casado et al., 2021; Guo et al., 2021). Although these solutions perform better than drift-oblivious algorithms such as FedAvg (McMahan et al., 2017), the solutions still use a single global model for all clients, and hence fail to address the aforementioned fundamental challenges of heterogeneity over time and across clients. Meanwhile, centralized ensemble methods that use multiple models for adapting to drift also suffer—in response to a localized data drift, a newly

[1]This work was done when Jianyu Wang was at CMU. He is now with Apple.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

created global model is trained over a mixture of concepts. The models in an ensemble are distinguished solely over time, and do not account for heterogeneity across clients.

In this work, we present the first FL solutions that employ multiple models to address FL under distributed concept drift. Our solutions aim to create one model for each new concept so that all clients under the same concept can train that model collaboratively, similar to what is done for *personalized* or *clustered* FL (Ghosh et al., 2019, 2020; Mansour et al., 2020; Sattler et al., 2020; Duan et al., 2021). We introduce two new algorithms for model creation and client clustering so that our solution addresses all the challenges of distributed concept drift. Our first algorithm, FedDrift-Eager, is a specialized algorithm that creates models based on drift detection. FedDrift-Eager is effective if new concepts are introduced one at a time. Our second algorithm, FedDrift, is a general algorithm that leverages hierarchical clustering to adaptively determine the appropriate number of models. FedDrift isolates drifted clients and conservatively merges clients via hierarchical clustering, so that FedDrift can effectively handle general cases where an unknown number of new concepts emerge simultaneously.

We empirically evaluate our solution using four popular concept drift datasets, and we compare our solution against state-of-the-art centralized concept drift solutions (KUE (Cano and Krawczyk, 2020) and DriftSurf (Tahmasbi et al., 2021)) and a recent FL solution that adapts to concept drifts (Adaptive-FedAvg (Canonaco et al., 2021)). Our results show that (i) FedDrift-Eager and FedDrift consistently achieve much higher and more stable model accuracy than existing baselines (average accuracy 93% vs. 90% for the best baselines, across six dataset/drift combinations); (ii) FedDrift performs much better than FedDrift-Eager when multiple new concepts are introduced at the same time; and (iii) our solution achieves a similar model accuracy as Oracle (94% accuracy), an idealized algorithm that knows the timing and distribution of concept drifts. On the real-world drift in the FMoW dataset (Koh et al., 2021), FedDrift achieves 64% accuracy vs. 58% accuracy for the best baselines. We make our source code and datasets publicly available to facilitate further research on this problem.[2]

2 BACKGROUND AND MOTIVATION

2.1 Problem Setup

We consider a FL setting with P clients, assumed to be stateful and participating at each round, and a central server that coordinates training across the clients. Training data are decentralized and arriving over time. The data at each client $c = 1, \ldots, P$ and each time $t = 1, 2, \ldots$ are sampled from a distribution (concept) $\mathcal{P}_c^{(t)}(x, y)$. We consider that data may be non-IID in two dimensions, varying across clients and across time. We say that there is a *concept drift* at time t and at client c if $\mathcal{P}_c^{(t)} \neq \mathcal{P}_c^{(t-1)}$ (the standard definition of drift with respect to a single node (Gama et al., 2014)). Under *distributed concept drift*, the time of change-points as well as the source or target distributions can differ across clients.

We seek a solution for adaptation to concept drift, generally involving any change in $\mathcal{P}(x, y)$. In contrast, by decomposing the joint distribution $\mathcal{P}(x, y) = \mathcal{P}(x)\mathcal{P}(y|x) = \mathcal{P}(y)\mathcal{P}(x|y)$, we distinguish from the special cases where $\mathcal{P}(y|x)$ is invariant (called covariate shift or virtual drift (Shimodaira, 2000; Tsymbal, 2004; Kairouz et al., 2021)) and $\mathcal{P}(x|y)$ is invariant (called label shift or target shift (Zhang et al., 2013; Azizzadenesheli et al., 2019)). (The datasets we consider in our evaluation (§5) contain general concept drifts with changes in the conditional distributions, with the exception of the FMoW dataset where the concept drift is specifically label shift.)

A single-model solution is to learn a single global model h (which is a function of time but is notationally suppressed) that is used for inference at all clients. The objective is to minimize over all time t, $\sum_{c=1}^{P} \mathbb{E}_{(x,y)\sim\mathcal{P}_c^{(t)}}[\ell(h(x), y)]$, where ℓ is the loss function. However, the optimal single model may not be well-suited in the presence of concept drifts. While the optimal single model can perform well under cases like covariate shift in which the feature-to-label mapping $\mathcal{P}(y|x)$ is fixed (although achieving fast convergence still requires a specialized strategy; e.g., FedProx (Li et al., 2020)), lower loss can often be obtained under the latter case by using specialized models for different concepts.

The multiple-model option is to learn a set of global models $\{h_m\}$ for $m \in [M]$ concepts, and a time-varying clustering of clients. We denote the cluster identities by one-hot vectors $w_c^{(t)}$, where $w_{c,m}^{(t)} = 1$ when the client c at time t uses model h_m for inference; we denote $h_{w_c^{(t)}}$ to represent the unique model h_m where $w_{c,m}^{(t)} = 1$. The objective is to minimize over all time t, $\sum_{c=1}^{P} \mathbb{E}_{(x,y)\sim\mathcal{P}_c^{(t)}}[\ell(h_{w_c^{(t)}}(x), y)]$.

2.2 Motivation

The prior work on drift adaptation in FL only consider restrictive settings such as (i) drifts occurring simultaneously in time (e.g., Figure 1(left)), where a centralized approach works well (Canonaco et al., 2021), or (ii) drifts with only minor deviations from a majority concept (e.g., Figure 1(right)), where updates from drifting clients are suppressed and the minority concept goes unlearned (Chen et al., 2021; Manias et al., 2021). Our work is the first to explicitly study the more general settings arising in distributed drifts, with heterogeneous data across clients and over time.

Consider the distributed drift pattern depicted in Figure 2. This is representative of an emerging trend (e.g., a breaking

[2] https://github.com/microsoft/FedDrift

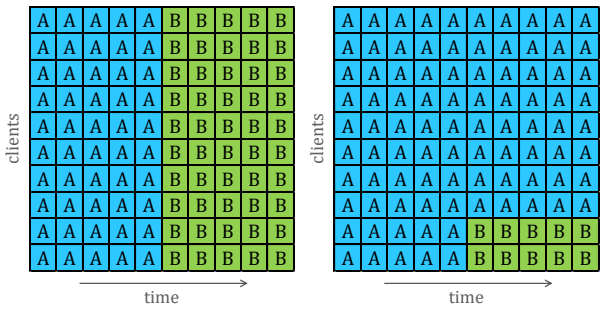


Figure 1: Simplistic drifts studied in prior work. (left) Simultaneous timing. (right) One majority concept.

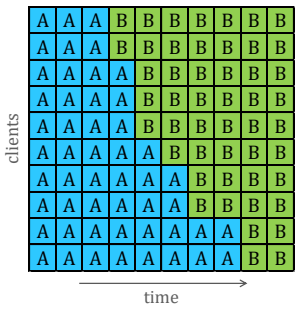


Figure 2: Distributed drift pattern (2 concepts).

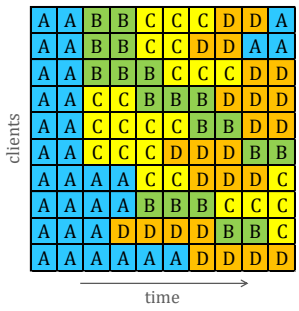


Figure 3: Distributed drift pattern (4 concepts).

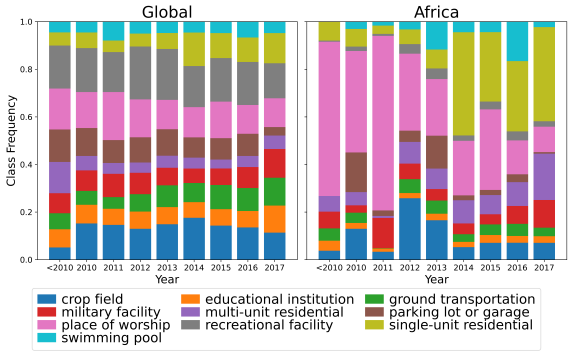


Figure 4: Class distribution over time in FMoW. The data drift viewed globally (left) is small relative to the localized data drift for Africa (right).

news event) that effects different clients at different times (e.g., due to their lag in learning of the news). For example, consider a next word prediction app in the period when "war" emerges as the popular next word after "Ukraine" or "slap" emerges after "Will Smith". Even for this simple case of a single staggered transition between two concepts, prior work results in significant accuracy loss. In particular, their use of a single global model (and at best a single global drift detection test) results in poor accuracy during the transition period (time steps 4–8 in Figure 2, see Figure 5(left) in §5).

We also consider more challenging cases, as depicted in Figure 3, where multiple concepts emerge at the same time and concept drifts may be recurring (a.k.a. periodic).

To demonstrate the challenge of distributed drift in real-world data, we consider the Functional Map of the World (FMoW) dataset adapted from the WILDS benchmark (Christie et al., 2018; Koh et al., 2021). The task is to classify the building type or land use from a satellite image, where images are over five major geographical regions (Africa, Americas, Asia, Europe, and Oceania) and across 16 years. Class distribution changes over time due to human activity and environmental processes. For the 10 most common classes, Figure 4 shows how the class distribution in Africa changes more rapidly over time, such as a reduction in places of worship and an increase in single-unit residential buildings. However, the global class distribution is relatively slow-changing. Our evaluation shows that the model trained on the global dataset only achieves 48% accuracy on Africa after the major drift at 2014, compared to 66% on the rest of the world. This real-world example highlights the necessity to mitigate data drift differently across regions, and existing centralized solutions cannot address this fundamental challenge.

2.3 Related Work

Concept drift has been studied extensively in the centralized setting for decades. We refer the reader to the surveys by Gama et al. (2014) and Lu et al. (2018). As previously discussed, applying these centralized algorithms to FL is not well-suited for distributed concept drifts with heterogeneous data across time and clients. We demonstrate this in our experimental evaluation (§5), where we compare against state-of-the-art algorithms such as KUE (Cano and Krawczyk, 2020) and DriftSurf (Tahmasbi et al., 2021), and include concrete examples showing why their performance is worse when multiple concepts exist simultaneously.

Drift in FL, on the other hand, has so far seen only preliminary study. One line of work considers the setting where there is one concept in the system to be learned (either like the example in Figure 1(right) when a minority of clients drift, or when clients observe the main concept under random noise), and seek to speed up the convergence of a model for that one concept by suppressing clients with heterogeneous data via regularization (Guo et al., 2021; Chen et al., 2021) or drift detection (Manias et al., 2021). When it comes to adapting to a new concept over time, we are only aware of two works, and both only consider drifts with uniform timing (Figure 1(left)). First, Casado et al. (2021) consider only the *covariate shift* setting (where the labeling $\mathcal{P}(y|x)$ is fixed and only $\mathcal{P}(x)$ changes) and uses drift detection to partition data from distinct concepts, in order to train a single model accurately in the course of revisiting each partition (i.e., rehearsal). Second, Canonaco et al. (2021) propose Adaptive-FedAvg, in which the server tunes the learning rate used by all clients based on the variability across updates, with the goal of reacting fast when drift occurs while also achieving stable performance in the absence of drift. We compare against Adaptive-FedAvg in our evaluation.

Table 1: Table of Symbols

τ	current time (prior time indexed by t)
P	# clients (indexed by c)
M	# global models (indexed by m)
R	# communication rounds per time (by i)
K	# local steps per model per round (by j)
$S_c^{(t)}$	new data arriving at client c at time t
$N_c^{(t)}$	$= \lvert S_c^{(t)} \rvert$
B	minibatch size
η	step size
h_m	global model m
$h_{c,m}$	local update of h_m by client c
$w_{c,m}^{(t)}$	is $S_c^{(t)}$ used to update h_m?

Our solution to drift in FL (§3, §4) relies on learning multiple models, which has been studied in prior work on *personalized FL* and *clustered FL*. Clients with similar data can be grouped into clusters, where each cluster trains its global model (Ghosh et al., 2019, 2020; Mansour et al., 2020; Sattler et al., 2020; Briggs et al., 2020; Duan et al., 2021). As we extend the problem of data heterogeneity in FL with the dimension of time, we train multiple models with the algorithm in §3, which is inspired by the prior clustering algorithms IFCA (Ghosh et al., 2020) and HypCluster (Mansour et al., 2020). This serves as the starting point of our solution, where our main contribution is the creation of new clusters as new concepts arrive over time. Our solution in §4 to handle an unknown number of concepts relies on hierarchical clustering, which has been studied in static FL by Briggs et al. (2020). In this prior work, it is unclear how to set the distance threshold at which to stop merging clusters. In contrast, our approach has the advantage that the stop merging criterion is identical to the drift detection threshold, which has an intuitive interpretation of performance loss.

3 MULTIPLE-MODEL TRAINING IN FL

As discussed above, distributed concept drift often means that multiple concepts are present simultaneously. Hence, our proposed solution learns multiple global models, where each model is trained by a cluster of clients for each distinct concept. In this section, we present Algorithm 1 for multiple-model training in FL for a given input clustering, which may vary over time as drifts occur. Then in §4, we will show how to learn the necessary input clustering, and how new clusters can be created to adapt to newly appearing concepts.

We define a time step as the granularity at which new data may arrive at a client. A time step may consist of multiple communication rounds. The set of data arriving at client c and time t is denoted by $S_c^{(t)}$. The global models being trained are denoted by h_m for $m \in [M]$, where M is the total number of models at a given time. Each model is trained by a cluster of clients, where the clustering may vary over time as concept drifts occur. The cluster identities $w_{c,m}^{(t)}$ (§2.1) indicate whether the data $S_c^{(t)}$ that arrived at client c at time t are sampled when computing a local update to the global model h_m. Further, the cluster identity of a client at a given time indicates which model is used for inference.

Algorithm 1 Multiple-model training at time τ

Input: Cluster identities $w_{c,m}^{(t)}$
  for each round $i = 1, 2, \ldots, R$ **do**
    for each client $c = 1, 2, \ldots, P$ in parallel **do**
      for each model $m = 1, 2, \ldots, M$ in parallel **do**
        $h_{c,m} \leftarrow \text{LocalUpdate}(c, h_m, \{w_{c,m}^{(t)}\}_{t=1}^{\tau})$
    for each model $m = 1, 2, \ldots, M$ **do**
      $h_m \leftarrow \frac{\sum_{c=1}^{P} h_{c,m} \sum_{t=1}^{\tau} w_{c,m}^{(t)} N_c^{(t)}}{\sum_{c=1}^{P} \sum_{t=1}^{\tau} w_{c,m}^{(t)} N_c^{(t)}}$

  LocalUpdate$(c, h_m, \{w_{c,m}^{(t)}\}_{t=1}^{\tau})$:
  for each local step $j = 1, 2, \ldots, K$ **do**
    $b \leftarrow$ random minibatch of size B from $\cup_{t: w_{c,m}^{(t)}=1} S_c^{(t)}$
    $h_m \leftarrow h_m - \eta \nabla \ell(h_m; b)$
  return h_m

Algorithm 2 Clustering to the lowest loss

$\ell_{c,m}^{(\tau)} \leftarrow$ loss of h_m on client data $S_c^{(\tau)}$
$w_{c,m}^{(\tau)} \leftarrow \mathbf{1}\{m = \arg\min_{m'} \ell_{c,m'}^{(\tau)}\}$
Run Algorithm 1

Within each time, the training of the global models in Algorithm 1 is equivalent to Federated Averaging (McMahan et al., 2017), since the aggregation weight of each client within each cluster is fixed at time τ. So the convergence of Algorithm 1 can be guaranteed by directly using previous analyses for Federated Averaging, such as (Li et al., 2020; Wang and Joshi, 2021). The difference here is that the objective function that clients are minimizing at time τ is replaced by the following:

$$\tilde{F}_m^{(\tau)}(h_m) = \sum_{c=1}^{P} \tilde{w}_{c,m}^{\tau} F_c^{(\tau)}(h_m) \tag{1}$$

where $F_c^{(\tau)}$ denotes the local objective function on client c, and the normalized weight is defined as

$$\tilde{w}_{c,m}^{\tau} = \frac{\sum_{t=1}^{\tau} w_{c,m}^{(t)} N_c^{(t)}}{\sum_{c=1}^{P} \sum_{t=1}^{\tau} w_{c,m}^{(t)} N_c^{(t)}}. \tag{2}$$

In the ideal case where each cluster maps to one concept in the system, each h_m is specialized for each concept that is sampled from a unique data distribution ($\mathcal{P}(x, y)$), and these h_m form a strong solution to our overall objective in §2.1. This ideal solution is the Oracle algorithm in our evaluation in §5, and we empirically demonstrate that our proposed solutions achieve comparable accuracy.

Note that, as stated, each client c in Algorithm 1 retains its complete history of both the cluster indicators $w_{c,m}^{(t)}$ and the local data arrivals $S_c^{(t)}$. To reduce this overhead, each client could instead maintain just a sliding window of the most recent time steps, as long as the window suffices for the minibatch sampling in LOCALUPDATE.

Thus, we have separated the problem of concept drift in FL into two components: (i) determining the time-varying clustering of clients in response to concept drifts, which is then used as input for (ii) the multiple-model training in Algorithm 1. Suppose, hypothetically, that there is a global model already initialized for each concept up to some moderate accuracy. In this restrictive setting, Algorithm 2 can be used to determine the cluster identities for each new time step. Each client tests the global models from the previous time step over its newly arrived data and chooses to identify with the model with the best loss (breaking ties randomly).[3] This restrictive setting covers time steps involving drifts that occur between concepts known to the system; e.g., the later stages of a staggered drift from concept A to concept B after some clients have already observed concept B (Figure 2). However, Algorithm 2 does not have any mechanism to spawn new clusters or determine the number of clusters. In §4, we will show how to determine the input for Algorithm 1 with clustering algorithms that can spawn clusters over time to react to drifts to *new* concepts.

4 CLUSTERING ALGORITHMS

Under concept drift in FL, data are heterogeneous both over time and across clients. The concept at each time and client is the ground-truth clustering that we seek to learn. Ideally, the models trained by each cluster correspond 1-to-1 to the concepts present in the system. Specifically, we want to avoid two miss-clustering problems: (**P1**) spawning *multiple clusters* that correspond to a *single concept*, because then each model would be trained over only a subset of the relevant data, not taking full advantage of collaborative training, and (**P2**) merging clients corresponding to *multiple concepts* into a *single cluster* (model poisoning).

We present two clustering algorithms for adapting to concept drift. First, in §4.1 we handle the case where only one new concept emerges at a time, which includes the example drift pattern in Figure 2, by incorporating a straightforward drift detection algorithm. Second, in §4.2 we give a general algorithm that handles the general case where multiple new concepts may emerge simultaneously, which includes the example drift pattern in Figure 3, by incorporating a bottom-up technique that *isolates clients* that detect drift (addressing **P2**) and *iteratively merges* clusters corresponding to the same concept (addressing **P1**).

[3] If there are no new data at a particular client, then we say its cluster identity is carried over from the previous time step so the model used for inference is well-defined.

Algorithm 3 FedDrift-Eager at time τ

$\ell_{c,m}^{(\tau)} \leftarrow$ loss of model h_m on client data $S_c^{(\tau)}$
$w_{c,m}^{(\tau)} \leftarrow \mathbf{1}\{m = \arg\min_{m'} \ell_{c,m'}^{(\tau)}\}$
if $\min_m \ell_{c,m}^{(\tau)} > \min_m \ell_{c,m}^{(\tau-1)} + \delta$ at any client c **then**
  // create one model for all drifted clients
  $M \leftarrow M + 1$
  Initialize a new global model h_M
  $w_{c,*}^{(\tau)} \leftarrow \mathbf{0}$; $w_{c,M}^{(\tau)} \leftarrow 1$
Run Algorithm 1

In the rest of this section, we assume that the first time step starts with one concept and one model, and that our clustering is run for each time step $\tau > 1$ as new data arrive.

4.1 Special Case: One New Concept at a Time

When a new concept emerges, the clients that observe the drift should be split off to a new cluster to start training a new model. Drift detection has been well-studied in the centralized, non-FL, setting (Gama et al., 2004; Baena-García et al., 2006; Bifet and Gavaldà, 2007; Harel et al., 2014; Pesaranghader and Viktor, 2016; Pesaranghader et al., 2018; Tahmasbi et al., 2021). As we noted in §2.2, for staggered drifts in FL, trying to apply a drift detection test *globally* at the server over the aggregate error results in poor performance during the transition period. Instead, in Algorithm 3, we apply drift detection *locally* at each client.

There are many drift detection tests in the literature, but the particular test is not our focus and for simplicity we consider a test of the following form. A drift is signaled at client c at time τ with respect to a model h_m if the loss of the model over the newly arrived data, denoted as $\ell_{c,m}^{(\tau)}$, degrades by a threshold δ relative to the loss measured at time $\tau - 1$:

$$\ell_{c,m}^{(\tau)} > \ell_{c,m}^{(\tau-1)} + \delta. \tag{3}$$

This test checks for any drift that incurs performance degradation with respect to a given model. However, the desired condition for creating a *new model* should check only for concept drifts that correspond to a concept *previously unobserved* and *ill-suited* for all existing models. For other drifts, such as the later stage of the staggered drift from concept A to concept B in Figure 2 (after concept B has already been detected and an appropriate model created), a client should join an existing cluster (in this case, the cluster for B). Hence, in Algorithm 3, the drift detection test for model creation compares against the *best performing* model:

$$\min_m \ell_{c,m}^{(\tau)} > \min_m \ell_{c,m}^{(\tau-1)} + \delta. \tag{4}$$

We note that detection tests that compare across multiple models have been previously studied in centralized learning in the context of adapting to recurring drifts (Katakis et al.,

2010). The clustering in Algorithm 3 (FedDrift-Eager) applies this multiple-model drift detection test at each client, and creates a new cluster for all the clients that detect a new concept; otherwise, each client identifies with the cluster with the best-performing model. This algorithm relies on the assumption that only one new concept occurs at a time by assigning the drifted clients to a single cluster. Despite this limitation, Algorithm 3 still merits interest as it experimentally performs well on the non-trivial case of the staggered drift in Figure 2 that has not been addressed by the prior work, as shown in §5. However, for the drift in Figure 3 in which concepts B and C emerge simultaneously at different clients, this algorithm creates only one cluster and sub-optimally tries to train a single model for both new concepts (problem **P2** above). Next, we extend this algorithm to address the general case where an unknown number of new concepts can occur at a time.

4.2 General Case

When drifts to new concepts are detected at multiple clients, in general we do not know whether the drifts all correspond to one concept or multiple concepts (or even zero concepts in the event of false positives in detection). We designed Algorithm 4 (FedDrift) for clustering in the face of this uncertainty. For each client that detects drift to a new concept, Algorithm 4 conservatively isolates the clients to individual clusters, and then merges clusters corresponding to the same concept slowly and safely over time by iteratively applying classical hierarchical agglomerative clustering (Shalev-Shwartz and Ben-David, 2014).

The generic hierarchical clustering procedure is specified by a distance function over the set of elements to be clustered and a stopping criterion, and at each step until the stopping criterion is met, merges the two closest clusters, where the distance between clusters of multiple elements is commonly defined to be the maximum distance between their constituents (known as a max-linkage clustering). In Algorithm 4, the MERGE subroutine combines two clusters i and j by averaging their models with weight proportional to the size of each model's training dataset (over all clients) and unifying the cluster identities.

To specify a distance function for hierarchical clustering, Algorithm 4 first aggregates at the server the loss estimates L_{ij} of the model h_i evaluated over a subsample of the data associated with the cluster for model h_j.[4] Then the distances between each cluster are initialized as $D(i,j) \leftarrow \max(L_{ij} - L_{ii}, L_{ji} - L_{jj}, 0)$.[5] $L_{ij} - L_{ii}$ measures the loss degradation of model h_i when evaluated over the data associated with h_j, relative to the loss over its own data. We informally interpret this difference as the magnitude of drift between the concept associated with h_i to the concept associated with h_j, analogous to the drift detection condition in Eq (3) (although not identical due to the bias of L_{ii} measuring a model's accuracy over its own training data). The term $D(i,j)$ is defined to be symmetric by also accounting for the magnitude of the drift $L_{ji} - L_{jj}$ in the reverse direction from concept j to concept i.

In addition to defining the cluster distances $D(i,j)$, employing hierarchical clustering also requires setting a stopping criterion. Typically, that corresponds to specifying either the desired number of clusters (which in our case is unknown), or an upper limit on the distance between clusters to stop merging. By our identification of the cluster distance as a magnitude of drift, we naturally re-use the drift detection threshold δ to also represent the tolerance level up to which clusters can be merged, eliminating one hyperparameter.

In Algorithm 4, both creating new clusters and merging existing clusters are based on the observed difference of the models' accuracy across two samples of data. For the clustering to accurately distinguish concepts, we assume that relevant changes in the concepts are manifested in the degradation of a model's predictive accuracy, and that the local sample size is sufficient for statistical significance—the same assumptions necessary for prior drift detection tests (Harel et al., 2014; Pesaranghader and Viktor, 2016; Pesaranghader et al., 2018; Tahmasbi et al., 2021).

Algorithm 4 FedDrift at time τ

$\ell_{c,m}^{(\tau)} \leftarrow$ loss of model h_m on client data $S_c^{(\tau)}$
for each client $c = 1, 2, \ldots, P$ in parallel **do**
  if $\min_m \ell_{c,m}^{(\tau)} > \min_m \ell_{c,m}^{(\tau-1)} + \delta$ **then**
    Initialize a local model at client c to be added to the set of global models at $\tau + 1$, and assign client c to its own cluster
  else
    $w_{c,m}^{(\tau)} \leftarrow \mathbf{1}\{m = \arg\min_{m'} \ell_{c,m'}^{(\tau)}\}$
for each i, j from $1, 2, \ldots, M$ in parallel **do**
  $L_{ij} \leftarrow$ loss of model h_i on sample of $\cup_{c,t:w_{c,j}^{(t)}=1} S_c^{(t)}$
Cluster distances $D(i,j) \leftarrow \max(L_{ij} - L_{ii}, L_{ji} - L_{jj}, 0)$
while $\min_{i \neq j} D(i,j) < \delta$ **do**
  MERGE(i, j, D)
Run Algorithm 1

MERGE(i, j, D):
Add a new model $h_k \leftarrow \frac{h_i \sum_{c,t} w_{c,i}^{(t)} N_c^{(t)} + h_j \sum_{c,t} w_{c,j}^{(t)} N_c^{(t)}}{\sum_{c,t} w_{c,i}^{(t)} N_c^{(t)} + \sum_{c,t} w_{c,j}^{(t)} N_c^{(t)}}$
$w_{c,k}^{(t)} \leftarrow w_{c,i}^{(t)} + w_{c,j}^{(t)}$ for all c, t
$D(k,l) = \max(D(i,l), D(j,l))$ for all l
Delete models h_i, h_j

[4] More precisely, at client c, the data clustered to h_j are subsampled proportionate to the size of the local dataset relative to the global dataset for h_j, $\sum_t w_{c,j}^{(t)} N_c^{(t)} / \sum_{c'} \sum_t w_{c',j}^{(t)} N_{c'}^{(t)}$.

[5] We note that $D(i,j)$ is not necessarily a true distance function as there is no guarantee that it satisfies the triangle inequality.

One subtlety to Algorithm 4 is that the hierarchical clustering is iteratively run at every time step, because the cluster distances vary with time. A simpler alternative would be to only try merging newly created clusters of local models after one time step of training. However, at that one time step, even models corresponding to the same concept may fail to merge given the limited sample size and limited number of training iterations. In other words, while the models are still warming-up, they may still be separated by a distance exceeding δ. As the models converge over time, the distance may drop below δ, which Algorithm 4 accounts for by iteratively attempting to merge.

The hierarchical clustering strategy of Algorithm 4 allows it to adaptively determine the appropriate number of clusters even when an unknown number of new concepts emerge at a time, but it also incurs additional computational resources relative to Algorithm 3. Algorithm 4 creates more global models M, adding to the communication cost of sending $O(MP)$ models. Additionally, the hierarchical clustering adds an $O(M^2 \log M)$ time complexity at the server at every time step (using a heap data structure for finding the minimum pairwise distance). In Appendix B, we discuss how we can restrict Algorithm 4 to create fewer overall models for higher efficiency. Also, similar to Algorithm 1, each client c could maintain $w_{c,m}^{(t)}$ and $S_c^{(t)}$ for just a sliding window of the most recent time steps, as long as the window suffices for Algorithm 4's subsampling step.

5 EXPERIMENTAL RESULTS

We empirically demonstrate that FedDrift-Eager and FedDrift are more effective than prior centralized drift adaptation and achieve high accuracy that is comparable to an oracle algorithm in the presence of distributed concept drifts. Prior work on FL under drifts is limited to simple cases such as in Figure 1, as noted in §2.2. Our evaluation covers the synthetic drifts in Figures 2 and 3, which represent more complex scenarios where drifts (i) occur across clients with staggered timing, (ii) correspond to different concept changes across different clients, and (iii) involve recurring concepts (e.g., the sequence A–B–C–D–A). We also evaluate on the real-world drift in the FMoW dataset (§2.2), which shows gradual concept changes staggered across clients.

The synthetic drift patterns are studied with respect to the following datasets: SINE (Pesaranghader et al., 2016), CIRCLE (Pesaranghader et al., 2016), SEA (Bifet et al., 2010), and MNIST (LeCun et al., 1998). SINE and CIRCLE each have 2 defined concepts, and we generate partitions of the data under the 2-concept staggered drift of Figure 2, while SEA and MNIST have more defined concepts, and we generate partitions under both the 2-concept and 4-concept drift patterns of Figures 2 and 3 for 10 clients and 10 time steps. For the real drift in FMoW, we evaluate on a subset of the data including the 10 most common classes, and identify each of the 5 major regions as one client and each new year as one time step. Appendix A has further dataset details.

We compare our algorithms FedDrift-Eager and FedDrift against the following baselines. First, the Oblivious algorithm learns a single model with FedAvg and has no mechanism for drift adaptation. Second, we consider traditional (non-FL) drift adaptation algorithms applied centrally at the server on top of FedAvg. Drift adaptation is typically classified into three categories, and we compare against algorithms representative of each: the drift detection method DriftSurf (Tahmasbi et al., 2021), two ensemble methods KUE (Cano and Krawczyk, 2020) and AUE (Brzezinski and Stefanowski, 2013)[6], and a Window method that forgets data older than one time step (more are reported in Appendix B). Third, Adaptive-FedAvg (Canonaco et al., 2021) is an FL algorithm that learns a single model and adapts to drifts by centrally tuning the learning rate used by all clients as a function of the variability across updates. Fourth, we compare to static FL clustering algorithms IFCA (Ghosh et al., 2020) and CFL (Sattler et al., 2020), which we extend to the time-varying setting by adding a window method (more variations reported in Appendix B). Fifth, Oracle is an idealized algorithm that has oracle access to the concept ID at training time and runs the multiple-model training of Algorithm 1 with the ground-truth clustering.

We run our experiments using the FedML framework (He et al., 2020). At each time step, each client observes a new batch of training data. For all the experiments on synthetic datasets, the models trained under each algorithm are fully connected neural networks with a single hidden layer of size $2d$ where d is the number of features. On the FMoW dataset, each algorithm trains ResNet18 models pretrained on ImageNet (He et al., 2016). After training for each time step, we test each algorithm over the batch of data arriving at the following time step, for all time steps. Each experiment is run for 5 trials, and we report the mean and the standard deviation. Additional algorithm details are in Appendix A.

In Table 2, we report the test accuracy averaged across all clients and all time steps except for the times of drifts (for synthetic datasets). We omit the times of drift because there is no chance for a client to adapt to the drift yet, and we eliminate the noise from beneficial clustering mistakes if by chance a client were clustered to the model appropriate for the test data after the drift. (For completeness, Appendix B shows results averaged over all time steps including drifts.)

Across all the 2-concept datasets under the staggered drift, we observe that the multiple-model algorithms FedDrift-Eager and FedDrift outperform the prior centralized solutions. In Figure 5, the accuracy is broken down per time

[6]By comparing against ensemble methods, we also account for the factor that multiple-model algorithms have higher capacity than single-model algorithms. AUE and KUE make predictions using a weighted vote over 5 and 10 models, respectively.

Table 2: Average accuracy (%) across all clients and time (over 5 trials)

	SINE-2	CIRCLE-2	SEA-2	MNIST-2	SEA-4	MNIST-4	FMoW
Oblivious	52.11 ± 1.79	88.38 ± 0.17	86.46 ± 0.22	87.37 ± 0.16	85.40 ± 0.09	82.95 ± 0.03	58.57 ± 0.07
DriftSurf	84.18 ± 1.40	92.34 ± 0.38	87.20 ± 0.27	93.26 ± 0.52	85.55 ± 0.13	82.97 ± 0.09	58.45 ± 0.19
KUE	86.86 ± 0.17	93.71 ± 0.14	87.25 ± 0.94	90.44 ± 0.44	85.09 ± 0.86	79.89 ± 0.26	33.11 ± 6.09
AUE	86.00 ± 0.95	92.84 ± 0.19	87.48 ± 0.07	92.22 ± 0.05	85.47 ± 0.12	82.07 ± 0.47	54.23 ± 0.14
Window	86.28 ± 0.64	93.72 ± 0.14	87.94 ± 0.10	92.34 ± 0.07	85.72 ± 0.13	81.43 ± 0.44	58.88 ± 0.15
Adaptive-FedAvg	74.10 ± 10.03	86.26 ± 0.00	86.77 ± 0.53	92.18 ± 0.05	85.25 ± 0.27	81.64 ± 0.04	52.82 ± 0.21
IFCA+Window	**98.49 ± 0.13**	94.31 ± 1.62	**88.04 ± 0.17**	91.76 ± 0.50	86.17 ± 1.00	81.27 ± 0.43	49.40 ± 0.76
CFL+Window	96.92 ± 1.84	96.04 ± 1.56	87.81 ± 0.32	90.66 ± 0.35	86.06 ± 0.11	80.51 ± 0.72	58.82 ± 0.11
FedDrift-Eager	97.53 ± 0.13	**97.82 ± 0.17**	87.51 ± 0.88	**95.52 ± 0.11**	87.61 ± 1.26	90.69 ± 1.20	61.77 ± 0.51
FedDrift	97.43 ± 0.06	**97.82 ± 0.19**	87.29 ± 0.75	95.48 ± 0.08	**88.13 ± 0.76**	**93.80 ± 0.08**	**64.84 ± 0.33**
Oracle	98.45 ± 0.03	97.84 ± 0.22	87.76 ± 0.98	95.54 ± 0.11	88.79 ± 0.41	94.30 ± 0.08	-

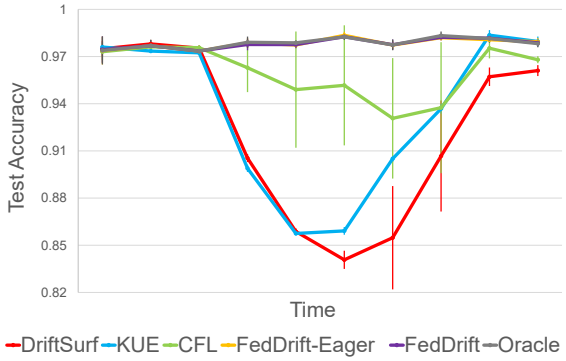


Figure 5: Accuracy at each time (averaged across clients) on CIRCLE-2.

step on CIRCLE-2, where we observe that centralized algorithms particularly suffer during the transition period. The fundamental issue is that when both concepts simultaneously exist, no single model can accurately fit for all clients. Even the ensemble algorithm (KUE) has poor performance because any new model added is updated by each client, and during the transition period, there is no model trained solely over data from the second concept. FedDrift-Eager and FedDrift learn models specialized for the second concept immediately after it emerges, and learn to apply the appropriate model at each client during the transition, matching the performance of Oracle.

Another challenge that the 2-concept staggered drift poses for DriftSurf, KUE, AUE, and Adaptive-FedAvg is that their adaptation strategies are a function of estimators that, from the central server's perspective, are aggregated over some clients that are drifting and others that are not. It is muddy whether drift is truly occurring, and even the unsophisticated window-based algorithm performs slightly better.

The clustering algorithms IFCA and CFL with a window perform relatively well on the 2-concept staggered drifts because they can flexibly employ a model specialized for the second concept during the transition period, but are overall behind FedDrift and FedDrift-Eager. We observe IFCA's success in adapting to drift is dependent on its random parameter initialization for its clusters, and works well particularly for the sharp drift on SINE-2.[7] For CFL, we observe that its iterative cluster splitting reacts quickly to drift, but creates excessive models for a concept over time without unifying clients under staggered drift. Appendix B has more details.

Regarding the 4-concept drift, Table 2 shows that all baselines are ill-suited, while FedDrift performs close to Oracle, and that FedDrift-Eager has intermediate performance (due to its false unification of simultaneously emerging concepts). To understand the performance of FedDrift, see Figure 6. In the ideal case (Oracle), there would be exactly one model for each concept. For FedDrift, at time 3 one new model is created for 5 of the 6 clients that drifted, and one false negative where a drifted client stays on the original model. With hierarchical clustering applied at the beginning of time 4, the 3 clusters corresponding to the green concept are correctly merged, while all clients on the yellow concept cluster to model 4 which had the lowest test loss over the new data. Also at time 4, model 6 is created for the new orange concept. Then at time 5, hierarchical clustering merges models 4 and 5 (due its iterative application in FedDrift, as the distance decreases after model 4 is further trained). After time 5, FedDrift has a distinct model for each concept, and no excess models.

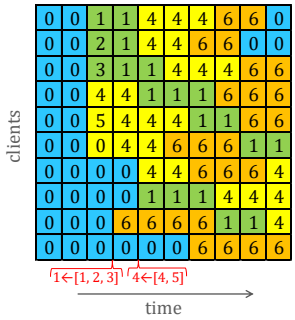


Figure 6: The clustering learned by FedDrift on MNIST-4. Each cell indicates the model ID at each client and time step, and the background color indicates the ground-truth concept.

One drawback of FedDrift is that it can create more models compared to FedDrift-Eager, adding to the communication cost. Appendix B shows that restricting FedDrift to just one new global model per time step (additional local models are still permitted) decreases its accuracy by only 0.92% on the MNIST-4 dataset, while saving communication.

[7] The accuracy of IFCA is higher than Oracle in a few cases but within the standard deviation, which we attribute to randomness in the model initialization and training.

Finally, we discuss the drift in the real-world FMoW dataset where we observe FedDrift has superior performance. The authors of the WILDS benchmark primarily make note of the performance loss of a globally trained model on data from Africa over time (Koh et al., 2021). We observe FedDrift successfully adapts to the local drift, switching the model applied at Africa at 2014 when there is a significant increase in single-unit residential buildings in Figure 4 in §2.2. Instead of creating a new model at 2014, we find FedDrift joins the cluster for Oceania where a local model was previously created, and stays at that cluster for 2014 and 2015, before then splitting into a new individual cluster for 2016 and 2017. We also observe that FedDrift detects a drift at 2015 for both Europe and the Americas, creating two more local models that contribute to higher accuracy.

Meanwhile, FedDrift-Eager similarly adapts to the change in Africa yielding a performance benefit, but it does not adapt well to the simultaneous drift for Europe and the Americas. Both FedDrift and FedDrift-Eager outperform the centralized adaptation baselines which fail to adapt to the drift when viewed globally (c.f. Figure 4). Finally, the low accuracy of IFCA is explained by its random initialization of model parameters for its clusters, in lieu of the pretrained ImageNet initialization under the rest of the algorithms, and the low accuracy of KUE is explained by its ineffective random subspace projections of the data for this task.

6 DISCUSSION

In this work, we present FedDrift-Eager and FedDrift, the first FL solutions designed to address the challenges of distributed concept drifts staggered in time and space (across clients). We empirically confirm the proposed solutions achieve significantly higher accuracy over existing baselines. We discuss the assumptions, limitations and future direction of our work here.

Privacy considerations. The clustering algorithm of FedDrift shares the local model learned by a single client with all clients, which could raise privacy concerns. For privacy-sensitive applications, our methods could be combined with other privacy-preserving techniques, e.g., model perturbation (Kairouz et al., 2021, §4) in future work.

Drift detection methods. For simplicity, we use a basic drift detection test (Eq (3) in §4) for a change in the loss that exceeds a given threshold. For production use, it would be beneficial to use a state-of-the-art detection test that is more statistically grounded and yields a quantitative statement on the assumption (§4.2) that the size of local data samples is large enough for statistical significance when creating and merging clusters. In particular, tests based on loss degradation by a proportional threshold (Baena-García et al., 2006; Barros et al., 2017) rather than an absolute threshold may be better suited for the multiple-model algorithm (FedDrift), as different models can have different loss magnitudes. We leave the exploration of combining various drift detection tests with our proposed solutions as future work.

Concept drifts and anomalies. We assume all observed concept drifts should be considered. But in the case of anomalies, it may be desirable not to react. One line of related work focuses on adapting only to "true" drifts while also exhibiting robustness in the presence of anomalies (Togbe et al., 2021; Sankararaman et al., 2022). Future work might investigate extending clustering algorithms like FedDrift to include anomaly detection in order to exclude outliers in isolated clusters and prevent false merges that could result in model poisoning.

Clustering algorithm alternatives. A design choice of our clustering algorithms is that we identify each client with the best-performing global model at each time step. An alternative approach is soft-clustering, previously explored by Li et al. (2021) in the context of static clustering in FL, in which a client fractionally identifies with multiple global models and takes the average for inference. We choose to not use soft-clustering because our preliminary experiments with soft-clustering show no benefit in performance, while increasing communication costs for additional local updates.

Model averaging alternatives. In FedDrift, the initial model parameters for a cluster after merging is the average of the constituent models, weighted by the size of each model's training dataset. An alternative approach to investigate in future work is to use weights that incorporate each model's loss over a sample of the aggregate dataset (already computed with the L_{ij}'s) so that more accurate models are weighted higher, analogous to weighted majority voting.

7 CONCLUSION

Federated learning under distributed concept drift is a largely unexplored area, posing particular challenges because drifts can arise staggered in time and space (across clients). This paper presented FedDrift-Eager and FedDrift, the first algorithms explicitly designed to mitigate these challenges. Empirical evaluation on a variety of dataset/drift combinations showed that these algorithms achieve significantly higher accuracy than existing baselines, and are comparable to an idealized algorithm with oracle knowledge of the ground-truth clustering. We hope that our solution spurs further research to this emerging problem, as well as addressing the privacy implications of clustering clients.

Acknowledgements

We thank the anonymous AISTATS reviewers for their valuable and constructive suggestions. This work was supported in part by NSF grants CCF-1919223, CCF-2045694, CNS-2112471, CNS-2211882, ONR N00014-23-1-2149, U.S. Army W911NF20D0002, and a Google Research Collaboration gift award.

References

Azizzadenesheli, K., Liu, A., Yang, F., and Anandkumar, A. (2019). Regularized learning for domain adaptation under label shifts. In *International Conference on Learning Representations (ICLR)*.

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., and Morales-Bueno, R. (2006). Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams (StreamKDD)*.

Barros, R. S., Cabral, D. R., Gonçalves Jr, P. M., and Santos, S. G. (2017). RDDM: Reactive drift detection method. *Expert Systems with Applications*, 90:344–355.

Bhardwaj, R., Xia, Z., Ananthanarayanan, G., Jiang, J., Shu, Y., Karianakis, N., Hsieh, K., Bahl, P., and Stoica, I. (2022). Ekya: Continuous learning of video analytics models on edge compute servers. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

Bifet, A. and Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of International Conference on Data Mining (ICDM)*.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). MOA: Massive online analysis. *J. Mach. Learn. Res. (JMLR)*, 11:1601–1604.

Briggs, C., Fan, Z., and Andras, P. (2020). Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *International Joint Conference on Neural Networks (IJCNN)*.

Brzezinski, D. and Stefanowski, J. (2013). Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Trans. Neural Netw. Learn. Syst*, 25(1):81–94.

Cano, A. and Krawczyk, B. (2020). Kappa updated ensemble for drifting data stream mining. *Machine Learning*, 109(1):175–218.

Canonaco, G., Bergamasco, A., Mongelluzzo, A., and Roveri, M. (2021). Adaptive federated learning in presence of concept drift. In *International Joint Conference on Neural Networks*, pages 1–7.

Casado, F. E., Lema, D., Criado, M. F., Iglesias, R., Regueiro, C. V., and Barro, S. (2021). Concept drift detection and adaptation for federated and continual learning. *Multimedia Tools and Applications*, pages 1–23.

Chen, Y., Chai, Z., Cheng, Y., and Rangwala, H. (2021). Asynchronous federated learning for sensor data with concept drift. In *IEEE International Conference on Big Data (BigData)*.

Christie, G., Fendley, N., Wilson, J., and Mukherjee, R. (2018). Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Duan, M., Liu, D., Ji, X., Wu, Y., Liang, L., Chen, X., Tan, Y., and Ren, A. (2021). Flexible clustered federated learning for client-level data distribution shift. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2661–2674.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence-SBIA*, pages 286–295.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4).

Garg, A., Shukla, N., Marla, L., and Somanchi, S. (2021). Distribution shift in airline customer behavior during COVID-19. *arXiv preprint arXiv:abs/2111.14938*.

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. (2020). An efficient framework for clustered federated learning. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Ghosh, A., Hong, J., Yin, D., and Ramchandran, K. (2019). Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*.

Guo, Y., Lin, T., and Tang, X. (2021). Towards federated learning on time-evolving heterogeneous data. *arXiv preprint arXiv:2112.13246*.

Harel, M., Crammer, K., El-Yaniv, R., and Mannor, S. (2014). Concept drift detection through resampling. In *Proceedings of the International Conference on Machine Learning (ICML)*.

He, C., Li, S., So, J., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M., and Avestimehr, S. (2020). FedML: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Kairouz, P., McMahan, H. B., et al. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, 22(3):371–391.

Khani, M., Ananthanarayanan, G., Hsieh, K., Jiang, J., Netravali, R., Shu, Y., Alizadeh, M., and Bahl, V. (2023). RECL: Responsive resource-efficient continuous learning for video analytics. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips,

R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B., Haque, I., Beery, S. M., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. (2021). WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv: abs/1610.05492*.

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

Li, C., Li, G., and Varshney, P. K. (2021). Federated learning with soft clustering. *IEEE Internet of Things Journal*, 9(10):7773–7782.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., and Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363.

Mallick, A., Hsieh, K., Arzani, B., and Joshi, G. (2022). Matchmaker: Data drift mitigation in machine learning for large-scale systems. In *Proceedings of Machine Learning and Systems (MLSys)*.

Manias, D. M., Shaer, I., Yang, L., and Shami, A. (2021). Concept drift detection in federated networked systems. *arXiv preprint arXiv:2109.06088*.

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

MarketsAndMarkets (2021). Federated learning solutions market by application (drug discovery, industrial iot), vertical (healthcare and life sciences, bfsi, manufacturing, retail and ecommerce, energy and utilities), and region - global forecast to 2028. https://www.marketsandmarkets.com/Market-Reports/federated-learning-solutions-market-151896843.html.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Pesaranghader, A. and Viktor, H. L. (2016). Fast Hoeffding drift detection method for evolving data streams. In *ECML PKDD*, pages 96–111.

Pesaranghader, A., Viktor, H. L., and Paquet, E. (2016). A framework for classification in data streams using multi-strategy learning. In *ICDS*, pages 341–355.

Pesaranghader, A., Viktor, H. L., and Paquet, E. (2018). McDiarmid drift detection methods for evolving data streams. In *International Joint Conference on Neural Networks (IJCNN)*.

Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H. R., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K., et al. (2020). The future of digital health with federated learning. *NPJ digital medicine*, 3(1).

Sankararaman, A., Narayanaswamy, B., Singh, V. Y., and Song, Z. (2022). Fitness:(fine tune on new and similar samples) to detect anomalies in streams with drift and outliers. In *International Conference on Machine Learning*. PMLR.

Sattler, F., Müller, K.-R., and Samek, W. (2020). Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2).

Suprem, A., Arulraj, J., Pu, C., and Ferreira, J. (2020). ODIN: Automated drift detection and recovery in video analytics. *Proceedings of the VLDB Endowment*, 13(11).

Tahmasbi, A., Jothimurugesan, E., Tirthapura, S., and Gibbons, P. B. (2021). DriftSurf: Stable-state / reactive-state learning under concept drift. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Togbe, M. U., Chabchoub, Y., Boly, A., Barry, M., Chiky, R., and Bahri, M. (2021). Anomalies detection using isolation in concept-drifting data streams. *Computers*, 10(1):13.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2).

Wang, J. and Joshi, G. (2021). Cooperative SGD: A unified framework for the design and analysis of local-update SGD algorithms. *Journal of Machine Learning Research*, 22(213):1–50.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2).

Zhang, K., Schölkopf, B., Muandet, K., and Wang, Z. (2013). Domain adaptation under target and conditional shift. In *Proceedings of the International Conference on Machine Learning (ICML)*.

A DATASETS AND EXPERIMENTAL PARAMETERS

We consider synthetic distributed drifts with respect to the following datasets previously used in the concept drift and personalized FL literature (Brzezinski and Stefanowski, 2013; Tahmasbi et al., 2021; Briggs et al., 2020; Canonaco et al., 2021; Manias et al., 2021): SINE and CIRCLE (Pesaranghader et al., 2016) which each have 2 defined concepts, and SEA (Bifet et al., 2010) and MNIST (LeCun et al., 1998), which have up to 4 concepts. In SINE, the first concept is a decision boundary of the sine curve $x_2 < \sin(x_1)$ for data points sampled from the unit square, and the second concept reverses the direction (swapping the labels). In CIRCLE, the two concepts are each decision boundaries of two different circles in the unit square, representing a smaller concept change than SINE. The first circle is centered at (0.2, 0.5) with radius 0.15 and the second circle is centered at (0.6, 0.5) with radius 0.25. In SEA, each concept corresponds to a shifted hyperplane. Each point in SEA has three attributes in [0, 10], where the label is determined by $x_1 + x_2 \leq \theta_j$ where j corresponds to 4 concepts, $\theta_A = 9, \theta_B = 8, \theta_C = 7, \theta_D = 9.5$. (The third attribute x_3 is not correlated with the label.) In SEA, at every concept there is noise in the observed labels, where the label is swapped with 10% chance for each data point independently. In MNIST, concept A corresponds to the original labeling of the hand-drawn digits, and under each other concept, the labels of two of the digits are swapped (B swaps digits 1 and 2, C swaps digits 3 and 4, and D swaps digits 5 and 6).

For each of the synthetic drift datasets in our experiments, the training data are distributed across 10 clients and arrive over 10 time steps. The partition of the data at each client and time is a constant 500 number of samples from the concept corresponding to the concept drift patterns in Figures 2 and 3 in §2.2. In our experimental results, after training at each time τ we report the test accuracy over the data at $\tau + 1$. For clarification, in reporting the accuracy at the last time step 10, we test over an 11th sample of data at each client that is from the same concept observed during training at time 10.

We also evaluate on the real-world drift in the Functional Map of the World (FMoW) dataset included in the WILDS benchmark (Christie et al., 2018; Koh et al., 2021). The learning task is to classify the land use or building type from satellite images, which has significant practical relevance, "aiding policy and humanitarian efforts in applications such as deforestation tracking, population density mapping, crop yield prediction, and other economic tracking applications" (Koh et al., 2021). Each image is RGB and square with a width of 224 pixels. The WILDS benchmark is not explicitly posed as a drift *adaptation* problem that we study in this paper, but instead as a drift *robustness* problem, and so they originally partitioned the data into train/validation/test splits. For our evaluation, we re-partition the dataset, distributing training data across 5 clients arriving over 9 time steps, using the metadata annotation of each image by region (Africas, Americas, Asia, Europe, Oceania) and year. The first 8 years from 2002–2009 have much fewer images collected, which we group into one time step, and then we treat each year from 2010–2017 as one time step each. The partition of the data at each client and time step is a subsample of up to 1000 images at the 10 classes that are the most common (counting across all regions and years). The test data evaluated for the last time step are a disjoint subsample also from the same year 2017 as the training data. Figure 4 in §2.2 depicts how the data drifts gradually over time, where the development of new infrastructure is a result of social, political, economic, and environmental factors. Viewed globally, the drift is small. Koh et al. (2021) write: "intriguingly, a large subpopulation shift across regions only occurs with a combination of time and region shift." Further, they call for solutions that "can leverage the structure across both space and time" and also hypothesize a benefit to "potentially transfer knowledge of other regions with similar economies and infrastructure" which we empirically confirm where FedDrift clusters Africa and Oceania together for years 2014–2015.

Across all algorithms we evaluate, the algorithms that learn a single model use FedAvg for training, and the clustering algorithms that learn multiple models use Algorithm 1 in §3 for training (which reduces to FedAvg when there is one cluster). The training parameters used in our experiments are shown in Table 3. For efficiency of the larger FMoW experiments, we reduce to 10 rounds and batch size 32—we observe that this suffices by convergence of the training accuracy.

Table 3: Training parameters

Parameter	Description	Experimental setting (all synthetic drifts)	Experimental setting (FMoW)
R	# communication rounds	100	10
K	# local steps per model per round	50	50
B	minibatch size	50	32
η	step size	varies	varies

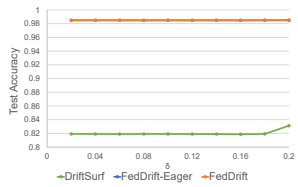
Regarding the learning rate selection, first we discuss all algorithms excluding Adaptive-FedAvg. We searched for learning rates of the form 10^{-a} for $a = 1, 2, 3, 4$, for each dataset, and found that $\eta = 10^{-2}$ was the best for SINE-2, CIRCLE-2, SEA-2, and SEA-4, that $\eta = 10^{-3}$ was best for MNIST-2 and MNIST-4, and that $\eta = 10^{-4}$ was best for FMoW. (This held for both of the two extremes among our baselines, Oblivious and Oracle, and we apply the same learning rate across all the algorithms. For FMoW, there is no known Oracle, so we searched only using the Oblivious baseline.) Also note that for computing the LocalUpdate at each client, we use the implementation of Adam in PyTorch with the options weight decay = 10^{-3} and amsgrad = True. We treat Adaptive-FedAvg separately, because it uses SGD with its own internal learning rate scheduler as its mechanism to react to drifts. We found that the initial learning rate of 10^{-2} was the best for each dataset with the exception of SINE-2, instead using 10^{-1}. (This higher learning rate explains the high standard deviation in the reported accuracy of Adaptive-FedAvg on SINE-2.)

Next, we report the selection of the drift detection threshold δ in the algorithms DriftSurf, FedDrift-Eager, and FedDrift. While the optimal δ is expected to vary across datasets, even for a fixed dataset, different algorithms can peak in performance at varying δ. The performance of each of these three algorithms for each dataset across δ in the range $0.02, 0.04, \ldots, 0.20$ is shown in Figure 7. To not bias towards any one algorithm, the experimental results are reported for each algorithm and dataset using its best δ. (The δ used for the FedDrift-C variant discussed in Appendix B is identical to that used for FedDrift.) However, using a fixed $\delta = 0.04$ for FedDrift-Eager and FedDrift makes at most a 1 pp difference in the results reported in Table 2 (on one trial).

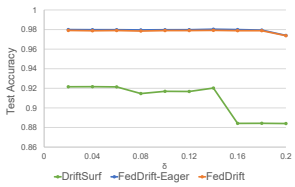
For all other hyperparameters of the algorithms we evaluate, we follow the parameter choices stated in the original papers, with the following exceptions: for DriftSurf we use $r = 3$ (which performed better than their suggested $r = 4$); for CFL we use $\gamma = 0.1$ (for which there is no default, but is shown to be a good setting from Theorem 1 and Figure 3 of their paper (Sattler et al., 2020) given that the number of distinct concepts at a time is at most 5 across all evaluated datasets); and for AUE we use $K = 5$ as the total ensemble size (compared to the $K = 10$ in their paper they consider over a significantly longer time horizon). In reporting FMoW results, for training efficiency, we further restrict to a total ensemble size of 4 for AUE and KUE.

Furthermore, for the FMoW dataset, which has more than one distinct data distribution at the initial time step unlike the remaining datasets, we use a different initialization of IFCA variants and FedDrift. For IFCA variants, clients initially self-select among 5 cluster centers instead of being all assigned to a single cluster. For FedDrift, clients are initialized to a local model each, which can be merged starting at the next time step. (If we instead initialize all clients to a single model that can later be split, we observed the average test accuracy of FedDrift is 64.46%, or 0.38% worse.)

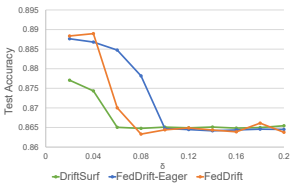
Finally, regarding the model training in Algorithm 1 at time τ, we apply one optimization for efficiency to only train models that are currently clustered to. (Although note that any such models are still retained by FedDrift-Eager and FedDrift in order to react to recurring drifts even if they are not actively being trained.)



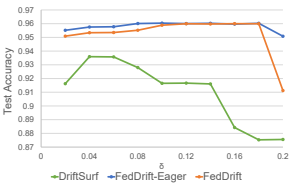
(a) SINE-2



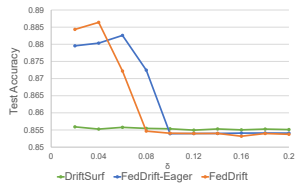
(b) CIRCLE-2



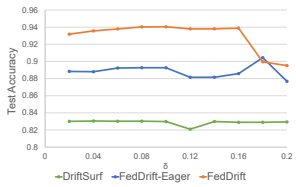
(c) SEA-2



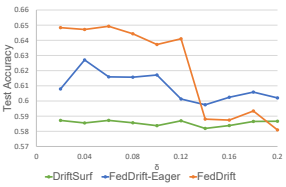
(d) MNIST-2



(e) SEA-4



(f) MNIST-4



(g) FMoW

Figure 7: Average accuracy of each drift detection-based algorithm under varying thresholds δ.

B ADDITIONAL EXPERIMENTAL RESULTS

We present additional experimental results on more baseline algorithms and on variants of our algorithms restricted to limited memory or communication.

Additional Baseline Algorithms. The additional algorithms presented in this appendix are:

- **Four traditional drift adaptation algorithms.** AUE-PC is a variation of the ensemble method AUE with the ensemble weights set *per-client*. Window-2 is a window method like Window, except that it forgets data older than two time steps instead of one. Weighted-Linear and Weighted-Exp also forget older data like window methods, but do so more gradually by down-weighting older data with either linear or exponential decay.
- **The FL clustering algorithm** CFL **(Sattler et al., 2020).** In extending the original static algorithm to our time-varying setting, we also consider a variant CFL-W, in which during training, each client samples only from the window of the newest data arriving at each time.
- **Three variations of the** IFCA **clustering algorithm (Ghosh et al., 2020)** that we considered for extending the original algorithm to the time-varying setting. First, IFCA(T) is exactly Algorithm 2 in §3, which defines cluster identities for each client and each time, in order to associate the data within a client that are heterogeneous over time across multiple clusters. IFCA(T) chooses the cluster identity once per *time step* (where time steps consist of multiple communication rounds)—this differs from the original algorithm described by Ghosh et al. (2020), which recomputes the cluster identity once per *round*. Second, IFCA does the per-round clustering; more precisely, for each time step τ, the cluster identity $w_{c,m}^{(\tau)}$ is recomputed at every round under the same equation used at the beginning of the time step in Algorithm 2. Third, IFCA-W is a variant of IFCA that trains only over the most recent data arrivals at each time, and the cluster identities of data from previous time steps are forgotten. In general, the IFCA-based algorithms require the number of clusters as input, which we provide as oracle knowledge—either 2 or 4 depending on the total number of concepts over time in each dataset. This gives IFCA-based algorithms an advantage over all other algorithms we evaluate, which do not know the number of clusters a priori. For the initialization of all three variations, at time 1 and round 1, all clients are assigned to a single cluster, matching the assumption we made for FedDrift and FedDrift-Eager in §4. The exception to this initialization strategy is on FMoW, where the total number of concepts is not known, and the concept at time 1 across clients is not identical; for this dataset, we instead initialize all IFCA-based algorithms with a total of 5 clusters (matching the number of regions), and where each client identifies with the best-performing randomly initialized model (same as the original paper).
- **A more communication-efficient variant of** FedDrift**.** FedDrift-C is the algorithm referred to in the last paragraph of §4 that is restricted to introducing one new global model per time step. More details on this algorithm are described later in this section.
- **Sliding window variants of** FedDrift-Eager **and** FedDrift**.** FedDrift-Eager-W and FedDrift-W are restricted to using only the most recent time step of data $S_c^{(t)}$ and cluster identities $w_{c,m}^{(t)}$.
- **A baseline sliding window variant** Oracle-W, which has oracle access to the ground-truth clustering but only uses the most recent time step of data in training.

In general, we use the -W suffix in the name of an algorithm to indicate a limited memory of a window of one time step. This memory restriction reduces the number of samples used for training at a time and might reduce the accuracy achievable under ground-truth clustering (Oracle-W vs. Oracle). Yet, the window is not strictly a drawback: (i) forgetting the older data builds in a passive adaptation to drift and (ii) in our setting it also guarantees that each client's training data at a step are all drawn from the same distribution—this is why we also investigate -W variants when extending the prior static clustering algorithms CFL and IFCA to our setting when data arrive over time.

Test Accuracy Results. Table 4 (extending Table 2 in §5) shows the test accuracy of all algorithms, averaged across all clients and time steps, but omitting the times of drifts. As noted in §5, we omit the times of drift when all algorithms suffer from the performance loss. For completeness, the test accuracy averaged over all time steps including drifts is shown in Table 5. In this latter table, note that Oracle and Oracle-W suffer a performance loss too at the time of drift. Under the test-then-train evaluation, Oracle has access to the concept ID of the data at training time but not at test time, where at each client, the model used for inference corresponds to the observed concept in the most recently arrived training data. Note

Table 4: Average test accuracy (%) across clients and time, omitting drifts (5 trials)

	SINE-2	CIRCLE-2	SEA-2	MNIST-2	SEA-4	MNIST-4	FMoW
Oblivious	52.11 ± 1.79	88.38 ± 0.17	86.46 ± 0.22	87.37 ± 0.16	85.40 ± 0.09	82.95 ± 0.03	58.57 ± 0.07
DriftSurf	84.18 ± 1.40	92.34 ± 0.38	87.20 ± 0.27	93.26 ± 0.52	85.55 ± 0.13	82.97 ± 0.09	58.45 ± 0.19
KUE	86.86 ± 0.17	93.71 ± 0.14	87.25 ± 0.94	90.44 ± 0.44	85.09 ± 0.86	79.89 ± 0.26	33.11 ± 6.09
AUE	86.00 ± 0.95	92.84 ± 0.19	87.48 ± 0.07	92.22 ± 0.05	85.47 ± 0.12	82.07 ± 0.47	54.23 ± 0.14
AUE-PC	88.67 ± 0.73	92.82 ± 0.23	87.55 ± 0.20	92.24 ± 0.03	86.67 ± 0.05	81.84 ± 0.33	54.15 ± 0.10
Window	86.28 ± 0.64	93.72 ± 0.14	87.94 ± 0.10	92.34 ± 0.07	85.72 ± 0.13	81.43 ± 0.44	58.88 ± 0.15
Window-2	85.97 ± 0.94	93.28 ± 0.15	87.62 ± 0.33	92.80 ± 0.44	86.58 ± 0.15	82.22 ± 0.30	59.44 ± 0.23
Weighted-Linear	72.93 ± 2.05	89.87 ± 0.54	87.02 ± 0.46	89.86 ± 0.36	86.42 ± 0.10	82.74 ± 0.04	58.05 ± 0.17
Weighted-Exp	82.52 ± 1.78	92.38 ± 0.32	87.11 ± 0.34	92.52 ± 0.20	86.60 ± 0.07	82.51 ± 0.09	58.49 ± 0.09
Adaptive-FedAvg	74.10 ± 10.03	86.26 ± 0.00	86.77 ± 0.53	92.18 ± 0.05	85.25 ± 0.27	81.64 ± 0.04	52.82 ± 0.21
CFL	57.57 ± 8.87	86.59 ± 3.42	86.46 ± 0.24	86.54 ± 0.43	86.24 ± 0.15	80.97 ± 0.78	57.92 ± 0.32
CFL-W	96.92 ± 1.84	96.04 ± 1.56	87.81 ± 0.32	90.66 ± 0.35	86.06 ± 0.11	80.51 ± 0.72	58.82 ± 0.11
IFCA(T)	98.45 ± 0.03	91.72 ± 5.19	86.46 ± 0.23	87.33 ± 0.15	85.41 ± 0.20	82.90 ± 0.05	47.76 ± 1.98
IFCA	98.46 ± 0.02	92.20 ± 5.32	86.45 ± 0.25	87.55 ± 0.25	85.35 ± 0.09	82.89 ± 0.04	48.17 ± 1.30
IFCA-W	**98.49 ± 0.13**	94.31 ± 1.62	**88.04 ± 0.17**	91.76 ± 0.50	86.17 ± 1.00	81.27 ± 0.43	49.40 ± 0.76
FedDrift-Eager	97.53 ± 0.13	**97.82 ± 0.17**	87.51 ± 0.88	**95.52 ± 0.11**	87.61 ± 1.26	90.69 ± 1.20	61.77 ± 0.51
FedDrift	97.43 ± 0.06	**97.82 ± 0.19**	87.29 ± 0.75	95.48 ± 0.08	88.13 ± 0.76	**93.80 ± 0.08**	**64.84 ± 0.33**
FedDrift-C	97.91 ± 0.70	97.61 ± 0.19	87.52 ± 0.91	95.45 ± 0.13	88.26 ± 0.80	92.88 ± 0.39	61.86 ± 0.30
FedDrift-Eager-W	97.95 ± 0.67	97.56 ± 0.24	87.32 ± 1.02	93.41 ± 1.14	86.99 ± 0.40	89.59 ± 0.38	61.94 ± 0.38
FedDrift-W	97.86 ± 0.59	97.52 ± 0.22	87.30 ± 1.01	93.85 ± 0.06	**88.56 ± 0.39**	91.34 ± 0.06	64.22 ± 0.60
Oracle	98.45 ± 0.03	97.84 ± 0.22	87.76 ± 0.98	95.54 ± 0.11	88.79 ± 0.41	94.30 ± 0.08	-
Oracle-W	98.53 ± 0.15	97.81 ± 0.13	87.31 ± 0.75	93.91 ± 0.05	88.41 ± 0.57	91.75 ± 0.05	-

that for the real-world gradual drifts in FMoW, the ground-truth is unknown, so we omit results for Oracle. Furthermore, because drifts occur gradually and there is no oracle knowledge of their timing, we report identical test accuracy results on FMoW in Tables 4 and 5, averaging across all clients and time steps.

Based on these tables, we make the following observations on the additional algorithms. The AUE-PC variant of AUE extends the model weights in the ensemble method to be individualized per-client, based on the performance of each model over each client's local data (as opposed to weights chosen based on the aggregate performance at the server). This additional flexibility leads to only a marginal accuracy improvement over AUE across all datasets. While it is generally valuable for clients at different stages of a staggered drift to use different models for inference, the more fundamental obstacle is that each global model trained by AUE-PC is updated by all clients. In the course of the 2-concept staggered drift, all of the models in the ensemble are trained either over a mixture of data from both concepts or solely from the first concept, and there is no accurate model available that is a good fit for the second concept.

The Window-2 algorithm and the weighted sampling algorithms Weighted-Linear and Weighted-Exp are techniques for forgetting older data, but less abruptly compared to Window-1, and in general they all perform similarly. On the sharp drift of SINE-2, the fastest forgetting algorithm Window performs the best of these. On the other hand, on the 4-concept drift of MNIST-4 in which the time axis does not well separate different concepts, the slowest forgetting algorithm Weighted-Linear performs best. Meanwhile, the performance of all four algorithms are close on the SEA datasets, which have greater overlap between the concepts.

The clustering algorithms CFL and CFL-W start with each client in one cluster, and recursively split clusters over rounds and over time based on the intra-cluster similarity of their local updates. We observe that the CFL-W variant is the better-performing of the two on each dataset except MNIST-4 (which is also the only dataset where Oblivious outperforms Window), and is a consequence of the passive drift adaptation of its sliding window which forgets older data. The performance of CFL-W is relatively high on SINE-2 and CIRCLE-2. As an example, the clustering learned on SINE-2 is shown in Figure 8. We observe that, for the first 6 time steps, it correctly distinguishes the two concepts by using distinct models. The disadvantage of the clustering of CFL-W is that it creates excess models for the same concept and does not take full advantage of collaborative training. At time 5, it is limited to splitting its cluster for model 0 when the green concept occurs, but cannot merge the drifted clients to the existing cluster created for the green concept at the previous time step. This limitation of only being able to subdivide existing clusters, but not merge clusters or re-assign clients to existing clusters results in poor performance on more complex drifts.

Table 5: Average test accuracy (%) across clients and time, including drifts (5 trials)

	SINE-2	CIRCLE-2	SEA-2	MNIST-2	SEA-4	MNIST-4	FMoW
Oblivious	47.36 ± 1.74	87.15 ± 0.15	86.22 ± 0.21	86.40 ± 0.15	85.16 ± 0.06	81.59 ± 0.02	58.57 ± 0.07
DriftSurf	79.45 ± 1.55	90.98 ± 0.36	86.91 ± 0.27	92.24 ± 0.64	85.19 ± 0.16	81.59 ± 0.05	58.45 ± 0.19
KUE	82.56 ± 0.18	92.45 ± 0.12	87.02 ± 0.92	89.59 ± 0.58	84.81 ± 0.68	77.84 ± 0.30	33.11 ± 6.09
AUE	81.24 ± 1.29	91.60 ± 0.17	87.23 ± 0.07	91.09 ± 0.04	85.09 ± 0.07	79.95 ± 0.63	54.23 ± 0.14
AUE-PC	83.65 ± 0.92	91.58 ± 0.21	87.38 ± 0.18	91.15 ± 0.06	86.30 ± 0.10	79.58 ± 0.47	54.15 ± 0.10
Window	81.77 ± 0.66	92.46 ± 0.12	87.72 ± 0.09	91.58 ± 0.07	85.30 ± 0.09	78.84 ± 0.26	58.88 ± 0.15
Window-2	81.46 ± 0.93	92.00 ± 0.15	87.43 ± 0.38	91.79 ± 0.56	86.18 ± 0.16	79.96 ± 0.49	59.44 ± 0.23
Weighted-Linear	67.34 ± 1.92	88.59 ± 0.52	86.77 ± 0.51	88.74 ± 0.36	86.13 ± 0.13	81.31 ± 0.04	58.05 ± 0.17
Weighted-Exp	76.86 ± 1.82	91.03 ± 0.31	86.91 ± 0.34	91.38 ± 0.20	86.26 ± 0.11	80.91 ± 0.09	58.49 ± 0.09
Adaptive-FedAvg	69.69 ± 10.13	85.60 ± 0.00	86.62 ± 0.50	91.33 ± 0.05	84.95 ± 0.26	79.49 ± 0.04	52.82 ± 0.21
CFL	51.98 ± 8.01	85.33 ± 3.35	86.19 ± 0.29	85.54 ± 0.41	85.95 ± 0.22	79.36 ± 0.94	57.92 ± 0.32
CFL-W	87.65 ± 1.27	94.00 ± 1.32	87.56 ± 0.32	89.73 ± 0.30	85.38 ± 0.14	78.15 ± 1.13	58.82 ± 0.11
IFCA(T)	88.77 ± 0.02	90.06 ± 4.62	86.22 ± 0.22	86.36 ± 0.14	85.10 ± 0.13	81.53 ± 0.05	47.76 ± 1.98
IFCA	88.78 ± 0.02	90.49 ± 4.73	86.21 ± 0.28	86.56 ± 0.21	85.06 ± 0.04	81.51 ± 0.03	48.17 ± 1.30
IFCA-W	**88.80 ± 0.12**	92.84 ± 1.19	**87.84 ± 0.14**	90.81 ± 0.67	85.52 ± 0.50	79.17 ± 0.39	49.40 ± 0.76
FedDrift-Eager	87.93 ± 0.12	**95.50 ± 0.14**	87.01 ± 0.72	**93.63 ± 0.10**	86.73 ± 0.64	83.99 ± 0.72	61.77 ± 0.51
FedDrift	87.84 ± 0.05	**95.50 ± 0.15**	86.85 ± 0.60	93.60 ± 0.07	86.95 ± 0.51	**85.44 ± 0.08**	**64.84 ± 0.33**
FedDrift-C	88.27 ± 0.61	95.30 ± 0.17	87.04 ± 0.70	93.58 ± 0.13	**86.98 ± 0.52**	85.30 ± 0.43	61.86 ± 0.30
FedDrift-Eager-W	88.31 ± 0.59	95.23 ± 0.23	86.90 ± 0.91	91.84 ± 0.16	86.50 ± 0.25	81.97 ± 0.21	61.94 ± 0.38
FedDrift-W	88.22 ± 0.52	95.20 ± 0.21	86.95 ± 0.89	91.85 ± 0.06	**86.98 ± 0.36**	83.14 ± 0.06	64.22 ± 0.60
Oracle	88.76 ± 0.02	95.51 ± 0.18	87.23 ± 0.93	93.65 ± 0.10	86.99 ± 0.40	85.81 ± 0.07	-
Oracle-W	88.83 ± 0.14	95.48 ± 0.11	86.89 ± 0.63	91.91 ± 0.05	86.56 ± 0.70	83.46 ± 0.03	-

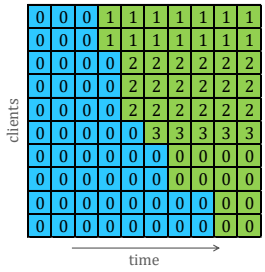


Figure 8: The clustering learned by CFL-W on SINE-2. Each cell indicates the model ID at each client and time step, and the background color indicates the ground-truth concept.

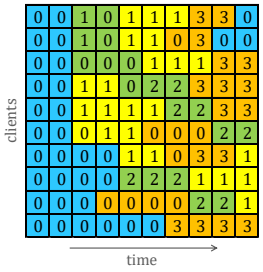


Figure 9: The clustering learned by FedDrift-Eager on MNIST-4. Each cell indicates the model ID at each client and time step, and the background color indicates the ground-truth concept.

For IFCA, IFCA-W, and IFCA(T), the clustering is pre-initialized with a random model for each concept that can occur over time for each dataset. In general, we observe that this is not a reliable method for reacting to drift. All the IFCA variants perform well under the sharp label-swap drift of SINE-2. When the new concept occurs, the drifted clients cluster to the second model, and the learned clustering matches the ground-truth. On CIRCLE-2, we found that IFCA and IFCA(T) learned the correct clustering in 2 out of 5 trials, and otherwise used only a single model in the other 3 trials. IFCA-W learned the correct clustering in 1 out of 5 trials. (Note the high standard deviation in Table 4.) Across the SEA and MNIST datasets, none of the three algorithms ever used more than a single model (with one exception—on SEA-4, in 1 out of 5 trials, IFCA-W used a distinct model for the yellow concept). For the SEA and MNIST datasets, we observe that the IFCA and IFCA(T) degrade to the Oblivious algorithm, and that IFCA-W degrades to the Window algorithm. Note that despite the degenerate clustering to a single model, matching the Window algorithm, IFCA-W achieves the highest accuracy in Table 4 on SEA-2 (but within the standard deviation of Window) due to randomness in the model initialization. In general, our experimental protocol fixes the same random seed for each trial across all all algorithms and where all created models within an algorithm use the same initialization. The IFCA variants are an exception to the rule because its initialization requires *distinct* random models. On the FMoW dataset, we observe again that random initialization can sometimes address drift, but unreliably: in 1 out of 5 trials each for all IFCA variants, a separate model is used for the Africa region at later time steps. (However, the IFCA variants are among the worst performing in our evaluation because their random initialization precludes the pre-trained

ImageNet initialization we use for other algorithms.) The authors of the original paper on IFCA note that the accuracy of the clustering is sensitive to the initialization of the models, and propose random restarts to address this issue, but restarts do not translate well to the time-varying setting we study. In our work, FedDrift-Eager and FedDrift address the initialization problem by using drift detection to deal with new concepts as they occur and to cultivate new clusters.

For FedDrift-Eager-W and FedDrift-W, restricting to a window has minimal impact on the accuracy for the SEA dataset. There is a significant loss of accuracy for the MNIST dataset relative to the non-windowed versions, but note that the same significant loss occurs when going from Oracle to Oracle-W, so this loss is a result of windowing, not specific to our algorithm. Indeed, the accuracy of FedDrift-W is quite close to Oracle-W.

The communication-efficient FedDrift-C**.** As noted in §4, one of the drawbacks of FedDrift is that it can create more models M compared to FedDrift-Eager, adding to the communication cost of sending $O(MP)$ models. The goal is to only use a number of global models close or equal to the number of distinct concepts, and while FedDrift can hierarchically merge created models of the same concept, FedDrift can observe temporary spikes in the number of global models. To mitigate this cost, we evaluate FedDrift-C, which differs from FedDrift in that, at each time after drift occurs, only one random client that drifted contributes its local model as a global model. In the case that multiple new concepts occur at a time, only one of the new concepts will be learned immediately, but clients that are still at an unlearned concept are eligible to detect drift again at the following time step and get another chance to contribute its local model. Meanwhile, while a concept goes unlearned globally, drifted clients do not contribute to any of the global models.

For the 4 concepts in MNIST-4, we observed that FedDrift learned a total of 7 global models (later merged down to 4) as shown in Figure 6 in §5. FedDrift-C more efficiently maintained a maximum of 4 global models across all time, at a penalty of 0.92% accuracy due to the delayed learning of one of the two simultaneously arising concepts. Meanwhile, FedDrift-Eager suffers a larger 3.11% penalty after it incorrectly merged the two simultaneous concepts, as shown in Figure 9—model 1 is initially trained over the green and yellow concepts, and while the clients at the green concept later abandon model 1 and eventually learn a separate model 2, the green concept training data still poison both model 0 and model 1.

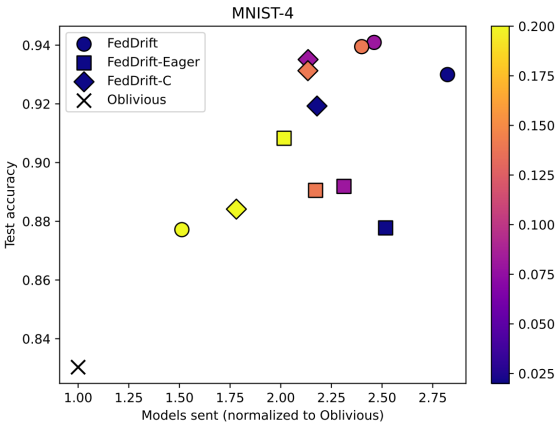


Figure 10: The accuracy-communication trade-off on MNIST-4 for FedDrift-Eager, FedDrift, and FedDrift-C. Each algorithm is evaluated under various selections of the splitting/merging threshold δ between 0.02 and 0.20, indicated by color. The vertical axis is the average test accuracy across clients and time, omitting drifts. (1 trial)

We quantify this accuracy-communication trade-off in Figure 10 where we show the average test accuracy and total number of models sent by FedDrift-Eager, FedDrift, and FedDrift-C under various selections of the drift detection threshold δ. Increasing the value of δ restricts cluster splitting (increases false negative detections) and promotes cluster merging, which reduces the number of models and concepts learned (at $\delta = 1$, each algorithm is identical to Oblivious). Empirically, we confirm that choosing larger settings of δ can trade-off accuracy for efficiency. (Choosing δ too small for FedDrift can also negatively affect accuracy due to increased false positive detections, but to a lesser degree because the hierarchical clustering of FedDrift can correct some false positives—see below on Impact of False Positives.) We observe that, generally, using FedDrift-C over FedDrift preserves most of the accuracy improvement over Oblivious while saving communication—with one exception at the largest $\delta = 0.20$ where both algorithms are susceptible to false merging, but FedDrift has more total models added to make the mistake of merging two concepts that FedDrift-C avoids. We also observe that the Pareto front is mostly configurations of FedDrift and FedDrift-C over FedDrift-Eager. Finally, we observe that all variants of FedDrift are

Table 6: Accuracy (%) on MNIST-R, omitting drifts

	MNIST-R
Oblivious	85.12 ± 1.37
DriftSurf	85.03 ± 1.36
KUE	81.56 ± 1.90
AUE	83.87 ± 1.64
AUE-PC	83.67 ± 1.66
Window	82.37 ± 1.94
Window-2	83.65 ± 1.83
Weighted-Linear	84.87 ± 1.34
Weighted-Exp	84.60 ± 1.44
Adaptive-FedAvg	83.17 ± 1.51
CFL	84.20 ± 1.54
CFL-W	82.24 ± 1.77
IFCA(T)	84.50 ± 1.21
IFCA	84.39 ± 1.45
IFCA-W	85.93 ± 3.35
FedDrift-Eager	89.85 ± 1.49
FedDrift	**94.06 ± 0.38**
FedDrift-C	92.76 ± 0.56
FedDrift-Eager-W	86.60 ± 2.27
FedDrift-W	90.83 ± 0.17
Oracle	95.03 ± 0.15
Oracle-W	91.66 ± 0.31

Table 7: Accuracy (%) on MNIST-R, including drifts

	MNIST-R
Oblivious	83.92 ± 1.23
DriftSurf	83.83 ± 1.21
KUE	79.77 ± 2.03
AUE	81.96 ± 1.03
AUE-PC	81.52 ± 1.43
Window	80.11 ± 1.45
Window-2	81.30 ± 1.58
Weighted-Linear	83.64 ± 1.21
Weighted-Exp	83.39 ± 1.29
Adaptive-FedAvg	81.41 ± 1.24
CFL	83.05 ± 1.37
CFL-W	80.58 ± 1.94
IFCA(T)	83.31 ± 1.11
IFCA	83.29 ± 1.29
IFCA-W	81.65 ± 0.67
FedDrift-Eager	85.26 ± 0.81
FedDrift	**86.77 ± 0.76**
FedDrift-C	86.65 ± 0.94
FedDrift-Eager-W	81.74 ± 1.60
FedDrift-W	83.70 ± 0.80
Oracle	87.32 ± 0.86
Oracle-W	84.29 ± 0.89

more efficient than ensemble algorithms—relative to Oblivious, FedDrift variants send 2–3x models compared to AUE which sends 5x—because for ensembles, clients contribute to every model at each communication round, compared to FedDrift where clients contribute only to the clusters they belong to (the broadcast of all models for clustering in FedDrift is only once per time step).

Random Drift Patterns. Throughout this paper, we have considered the 4-concept drift pattern in Figure 3 in §2.2 as a specific concrete example in order to depict the challenges in distributed concept drift, motivate the design of FedDrift, and discuss the experimental performance by comparing the learned clustering matrix to the ground-truth. To examine the performance more generally, we consider a family of datasets MNIST-R with random concept changes. Using the same four concepts as in MNIST-4, MNIST-R is generated with all clients at the first concept to start, and then each client independently randomly observes one of the four concepts every two time steps (as opposed to every time step which is not possible to adapt to). Across 5 random seeds, the average accuracy is shown in Table 6 (and in Table 7 for all time including drifts). We generally observe the same relative performances of each algorithm as on the previously specified MNIST-4 drift. The performance of FedDrift is close to that of Oracle, FedDrift-C is close behind, FedDrift-Eager is lower given that it is likely to have multiple new concepts occurring simultaneously in MNIST-R, and then all prior baselines follow.

Adaptation under Label Shift. In this work, we focus on the general case of concept drift as opposed to special cases like covariate shift or label shift. All of the synthetic drift datasets studied above involve a change in the decision boundary. But our solutions are also applicable under specific cases of drift. The real drift in the FMoW dataset is an example of label shift (defined in §2.1). Here we consider another label shift dataset, MNIST-L-4, in which the drift is synthetically generated to follow the same distributed drift pattern as MNIST-4 in the ground-truth clustering, but differs in the concept definitions so that classes are incrementally introduced over time: concept A is only over digits 0/1, concept B is only over digits 2/3, concept C is only over digits 4/5, and concept D is only over digits 6/7.

Table 8 shows the average accuracy omitting time steps of drift, and Table 9 shows the average accuracy over all time including drift times. Unlike all previous datasets (including FMoW with label shift), we observe a significant difference between the two tables. On the metric omitting drifts, FedDrift-Eager and FedDrift attain 99% accuracy comparable to Oracle, then the IFCA variants attain similarly high accuracy compared to the remaining baselines. However, on the metric including drifts, we observe Adaptive-FedAvg performs best, and even the Oblivious algorithm outperforms the multiple-model algorithms like FedDrift, IFCA, and Oracle. The reason is that for the concepts in MNIST-L-4, it is possible for a single model to fit multiple concepts (different labels) accurately as long as a concept was previously seen at another client. On the

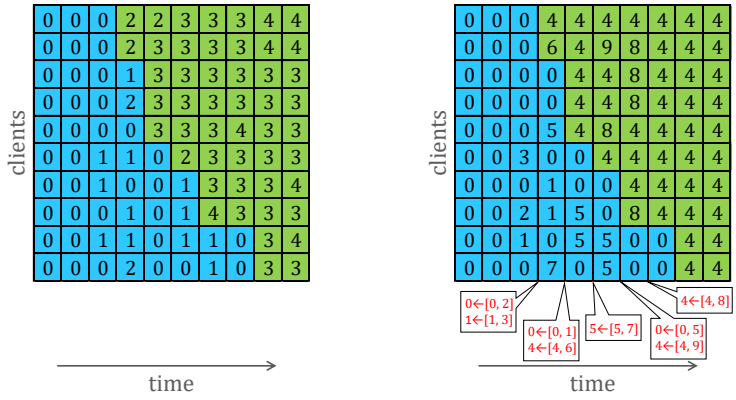
Table 8: Accuracy (%) on MNIST-L-4, omitting drifts

	MNIST-L-4
Oblivious	90.07 ± 0.23
DriftSurf	91.79 ± 0.64
KUE	90.68 ± 1.03
AUE	92.76 ± 1.29
AUE-PC	92.08 ± 1.40
Window	87.28 ± 1.26
Window-2	90.10 ± 0.33
Weighted-Linear	91.13 ± 0.56
Weighted-Exp	91.87 ± 0.31
Adaptive-FedAvg	96.74 ± 0.04
CFL	87.92 ± 1.52
CFL-W	92.00 ± 1.25
IFCA(T)	98.86 ± 1.01
IFCA	99.48 ± 0.34
IFCA-W	99.02 ± 0.62
FedDrift-Eager	99.17 ± 0.09
FedDrift	**99.56 ± 0.01**
FedDrift-C	97.85 ± 0.86
FedDrift-Eager-W	98.16 ± 0.80
FedDrift-W	99.20 ± 0.02
Oracle	99.65 ± 0.01
Oracle-W	99.29 ± 0.01

Table 9: Accuracy (%) on MNIST-L-4, including drifts

	MNIST-L-4
Oblivious	81.60 ± 0.29
DriftSurf	83.48 ± 1.24
KUE	81.44 ± 1.07
AUE	83.58 ± 1.22
AUE-PC	83.08 ± 1.23
Window	77.14 ± 1.30
Window-2	81.34 ± 0.22
Weighted-Linear	82.97 ± 0.66
Weighted-Exp	83.89 ± 0.33
Adaptive-FedAvg	**89.44 ± 0.03**
CFL	77.08 ± 0.81
CFL-W	81.67 ± 1.59
IFCA(T)	71.99 ± 2.20
IFCA	71.98 ± 2.79
IFCA-W	71.64 ± 2.54
FedDrift-Eager	75.15 ± 1.30
FedDrift	70.69 ± 0.01
FedDrift-C	69.79 ± 0.60
FedDrift-Eager-W	74.94 ± 1.23
FedDrift-W	70.43 ± 0.01
Oracle	70.75 ± 0.01
Oracle-W	70.49 ± 0.01

other hand, the multi-model approach suffers from poor performance at the time of drift as the newly created model has not seen the labels in the test data. Under the 4-concept drift pattern, the single model learned by Adaptive-FedAvg has high test accuracy on concepts B (green) and C (yellow) after the first occurrence in the system, while Oracle has low accuracy at the time of drift by employing a specialized model trained solely on concept A (blue).



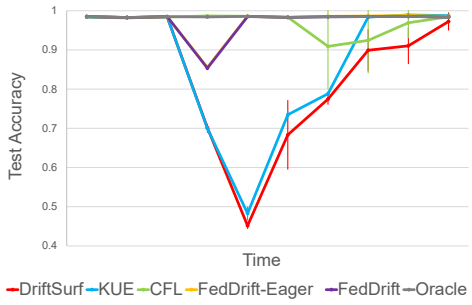
(a) FedDrift-Eager (b) FedDrift

Figure 11: The clustering learned on SINE-2 when $\delta = 0.01$. Each cell indicates the model ID at each client and time step, and the background color indicates the ground-truth concept.

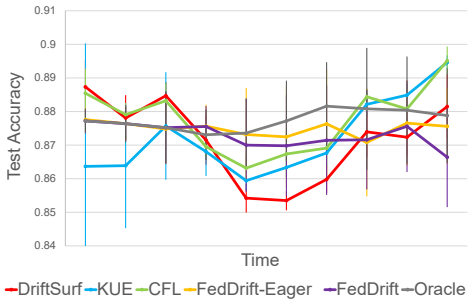
Impact of False Positives. To demonstrate the application of the hierarchical clustering in FedDrift, in §5 we discussed the example of the learned clustering for MNIST-4 in Figure 6. Here in Figure 11 we present another example on SINE-2 at a small $\delta = 0.01$ (corresponding to more aggressive detection) to demonstrate an example of how hierarchical clustering can be beneficial even in the case of a 2-concept drift in mitigating false positives. At time 3, in both FedDrift-Eager and FedDrift there are three false positives, where in FedDrift-Eager, the new model 1 is retained but its underlying data forgotten, while in FedDrift, although initially 3 redundant models are created, they are all merged back with model 0 within 2 time steps, averaging their parameters and reincorporating their clustered data. The advantage of hierarchical clustering is also evident at time 4 when 2 false positives and 2 true positives occur together. In FedDrift-Eager, one new model is created for

all the clients, but this new model is "poisoned" by contributions from the blue concept and does not work well at time 5, resulting in another drift detection to create model 3 (and forgetting about the data associated with model 2). FedDrift, on the other hand, creates models solely trained over either the blue and green concepts, and eventually merges all models of an identical concept, recovering all of the data. While the false positive mitigation demonstrated in this example is not a significant contributor to the observed higher accuracy of FedDrift in our evaluation because we use higher δ values as noted in Appendix A, it is relevant when there is greater uncertainty in selecting the threshold hyperparameter.

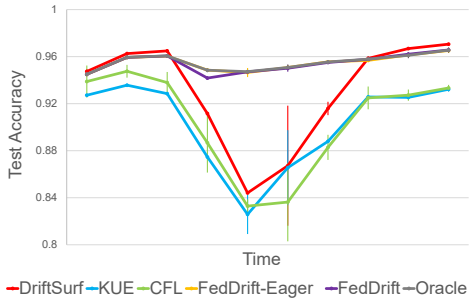
Test Accuracy Over Time. Finally, in Figure 12, we include plots that we omit from the main paper due to space constraints. The figure shows the accuracy over time for FedDrift-Eager, FedDrift, and selected baselines representing drift detection, ensembles, and clustered FL, supplementing Figure 5 in §5. (Note the varying scales of the y-axes.) We observe the same general trends: (i) the centralized drift adaptation algorithms suffer in performance, particularly during the transition period when no one model works well across all clients; (ii) CFL can react to the drift early on SINE-2 as with CIRCLE-2 before, but its performance degrades with excessive further splits; (iii) for the 4-concept drift in SEA-4 and MNIST-4 centralized baselines and CFL never recover in performance with multiple concepts present; and (iv) on SEA-4 and MNIST-4, FedDrift is close to Oracle except for a gap at time 3 when it uses local models prior to merging, while FedDrift-Eager lags behind FedDrift when it creates a single model for the 2 simultaneously arising concepts but can slowly recover with further detections.



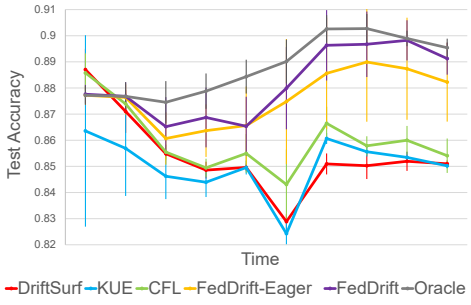
(a) SINE-2



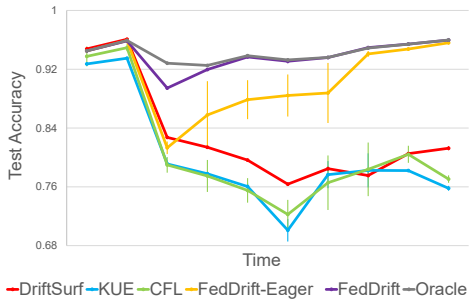
(b) SEA-2



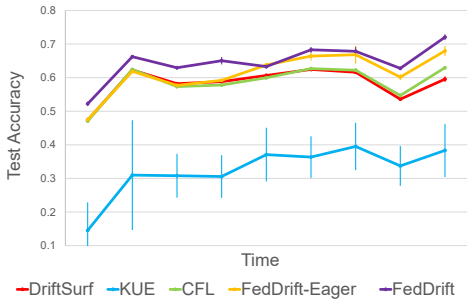
(c) MNIST-2



(d) SEA-4



(e) MNIST-4



(f) FMoW

Figure 12: Test accuracy of selected algorithms at each time on SINE-2, SEA-2, MNIST-2, SEA-4, MNIST-4, and FMoW. Vertical lines represent standard deviations.