Federated Learning under Distributed Concept Drift

Ellango Jothimurugesan
Carnegie Mellon University
ejothimu@cs.cmu.edu

Kevin Hsieh
Microsoft Research
kevin.hsieh@microsoft.com

Jianyu Wang
Carnegie Mellon University
jianyuw1@andrew.cmu.edu

Gauri Joshi
Carnegie Mellon University
gaurij@andrew.cmu.edu

Phillip B. Gibbons
Carnegie Mellon University
gibbons@cs.cmu.edu

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). Our work is the first to explicitly study data heterogeneity in both dimensions. We first demonstrate that prior solutions to drift adaptation, with their 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) [25, 31] 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. 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. 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 [14, 24]. 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 [2, 38].

This *concept drift* problem [42] in streaming data has been studied extensively in a centralized learning environment [13, 39]. 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 [8, 9, 17, 29]. Although these solutions perform better than drift-oblivious algorithms such as FedAvg [31], 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.

Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

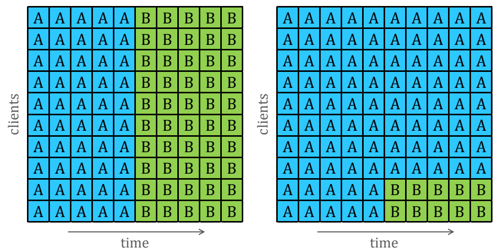


Figure 1: Simplistic drifts studied in prior work. (left) Simultaneous timing. (right) One majority concept.

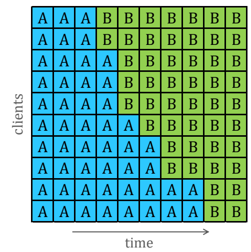


Figure 2: Distributed drift pattern (2 concepts).

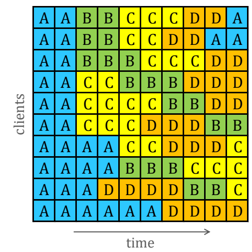


Figure 3: Distributed drift pattern (4 concepts).

We present the first FL solution that employs multiple models to address FL under distributed concept drift. Our solution aims 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 [11, 15, 16, 30, 35]. We introduce two new algorithms for client clustering to addresses 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 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, comparing against state-of-the-art centralized concept drift solutions (KUE [6] and DriftSurf [39]) and a recent FL solution that adapts to concept drifts (Adaptive-FedAvg [7]). 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. 88% for the best baseline, 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 [24], FedDrift achieves 64% accuracy vs. 58% accuracy for the best baseline.

2 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 $S_c^{(t)}$ at each client $c = 1, 2, \dots, P$ and each time $t = 1, 2, \dots$ 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 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 multiple-model FL problem is to learn a set of global models, and a time-varying clustering of clients. For notation, we denote the global models as h_m for $m \in [M]$, where M is the number of models at a given time (and can vary over time). 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,

$$\sum_{c=1}^{P} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{P}_c^{(t)}} [\ell(h_{w_c^{(t)}}(x), y)]. \tag{1}$$

In presenting our multiple-model solution, we assume that there is one concept and one model at time 1, and show in §4 how to learn the cluster identities $w_c^{(\tau)}$ for each time $\tau > 1$ as new data arrive. Given the cluster identities, learning the set of global models from the clients' data is by continuously retraining via FedAvg [31] within each cluster, which is described precisely in Appendix B.

3 Motivation

The prior work on drift adaptation in FL has considered only restrictive settings such as (i) drifts occurring simultaneously in time (e.g., Figure 1(left)), where a centralized approach works well [7], 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 [9, 29]. 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 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 [10, 24]. The task is to classify the building type or land use from a satellite image, where images are over 5 major geographical regions (Africa, Americas, Asia, Europe, and Oceania) and across 16 years. Concept drift due to human activity and environmental processes degrades predictive accuracy over time. 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 class distribution viewed globally 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 concept drift differently across regions, and existing centralized solutions cannot address this fundamental challenge.

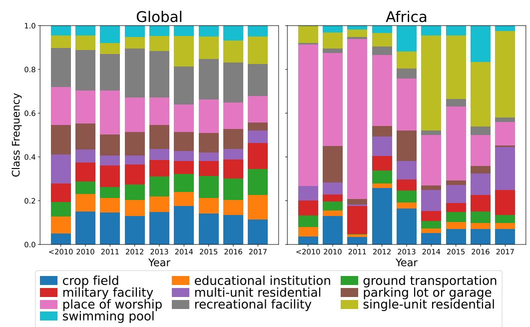


Figure 4: Class distribution over time in FMoW. The drift viewed globally (left) is small relative to the localized drift for Africa (right).

4 Clustering Algorithms

In §4.1 we handle the case where only one new concept emerges at a time. Then in §4.2 we give an algorithm that handles the general case where multiple new concepts may emerge simultaneously.

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. In Algorithm 1 (FedDrift-Eager), we apply a drift detection test locally at each client, and create a new cluster for all clients that detect a new concept.

There are many drift detection tests in the literature [1, 12, 18, 23, 32, 34, 39]. The particular test is not our focus and for simplicity we consider a test of the form $\ell_{c,m}^{(\tau)} > \ell_{c,m}^{(\tau-1)} + \delta$ where the loss degrades by a set threshold δ. However, the desired condition for creating a *new model* should check only for drifts that correspond to a concept *previously unobserved* and *ill-suited* for all existing models. Hence, in Algorithm 1, the drift detection test compares against the *best performing* model.

Algorithm 1 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
  $w_{c,*}^{(\tau)} \leftarrow \mathbf{0}$; $w_{c,M}^{(\tau)} \leftarrow 1$

Algorithm 1 groups all drifting clients in one cluster. However, under the drift in Figure 3 in which concepts B and C emerge simultaneously, this sub-optimally trains a single model for both concepts.

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 or multiple concepts (or even zero in the event of false positives in detection). We designed Algorithm 2 (FedDrift) for clustering in the face of this uncertainty. For each client that detects drift to a new concept, Algorithm 2 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 [21, 36].

Hierarchical clustering is generally specified by a distance function and a stopping criterion. The distance between clusters $D(i, j)$ based on the loss degradation of model h_i over a subsample of the data associated with the cluster for model h_j (and vice versa for symmetry). This distance corresponds to the magnitude of drift between the concepts for each cluster, analogous to the drift detection condition for splitting clusters. Hence, we naturally re-use the drift detection threshold δ to represent the tolerance level up to which clusters can be merged, avoiding the addition of another hyperparameter.

Algorithm 2 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 subsample 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) // *subroutine described in Appendix C*

One subtlety to Algorithm 2 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, but as the models converge over time, the distance may drop below δ, which iterative merging accounts for. A more complete description of Algorithm 2 is described in Appendix C.

5 Experimental Results

We empirically demonstrate that FedDrift-Eager and FedDrift are more effective than prior centralized drift adaptation and achieve high accuracy comparable to an oracle algorithm in the presence of distributed drifts. Our evaluation covers the synthetic drifts in Figures 2 and 3, which (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). The 2-concept and 4-concept drift patterns are generated for the datasets SINE [33], CIRCLE [33], SEA [3], and MNIST [26], described in Appendix D. We also evaluate on the real-world drift in the FMoW dataset (§3).

We compare against the following baselines. First, the Oblivious algorithm learns a single model with FedAvg. Second, we consider drift adaptation algorithms applied at the server on top of FedAvg: the drift detection method DriftSurf [39], two ensemble methods KUE [6] and AUE [5], and a Window method that forgets data older than one time step. Third, Adaptive-FedAvg [7] is an FL algorithm that learns a single model and adapts to drifts by centrally tuning the learning rate based on the variability across updates. Fourth, we compare to static FL clustering algorithms IFCA [16] and CFL [35], which we extend to the time-varying setting by adding a window method (more variations reported in Appendix E). Fifth, Oracle has oracle access to the ground-truth clustering at training time.

In Table 1, we report the average test accuracy across clients and time. Across the 2-concept drift datasets, we observe that the multiple-model algorithms FedDrift-Eager and FedDrift outperform prior centralized solutions. In Figure 5(left), the accuracy is broken down per time step on CIRCLE-2,

Table 1: Average accuracy (%) across all clients and time (over 5 trials, intervals represent 1 std dev)

	SINE-2	CIRCLE-2	SEA-2	MNIST-2	SEA-4	MNIST-4	FMoW
Oblivious	50.44 ± 1.52	88.36 ± 0.27	86.37 ± 0.34	87.25 ± 0.14	85.38 ± 0.28	82.97 ± 0.04	58.46 ± 0.08
DriftSurf	83.90 ± 1.01	92.54 ± 0.67	87.27 ± 0.34	91.71 ± 1.60	85.48 ± 0.28	82.99 ± 0.05	58.42 ± 0.16
KUE	87.05 ± 0.12	93.83 ± 0.04	87.62 ± 0.42	89.74 ± 0.07	85.53 ± 0.12	79.78 ± 0.16	37.46 ± 7.95
AUE	86.06 ± 0.60	92.74 ± 0.51	87.46 ± 0.12	92.19 ± 0.07	85.55 ± 0.08	81.29 ± 0.19	54.22 ± 0.14
Window	86.42 ± 0.74	93.67 ± 0.15	88.08 ± 0.10	92.15 ± 0.34	85.76 ± 0.16	81.16 ± 0.46	58.79 ± 0.14
Adaptive-FedAvg	78.02 ± 10.73	86.26 ± 0.00	86.69 ± 0.39	92.16 ± 0.04	85.32 ± 0.25	81.62 ± 0.07	52.76 ± 0.23
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	95.15 ± 0.32	95.62 ± 1.14	87.66 ± 0.36	90.53 ± 0.81	85.67 ± 0.21	79.99 ± 0.58	58.70 ± 0.13
FedDrift-Eager	98.46 ± 0.03	97.86 ± 0.20	88.35 ± 0.37	**95.99 ± 0.06**	88.08 ± 0.24	89.21 ± 2.02	61.62 ± 0.45
FedDrift	98.48 ± 0.01	**97.88 ± 0.17**	**88.65 ± 0.43**	95.93 ± 0.01	**88.41 ± 0.29**	**94.09 ± 0.08**	**64.91 ± 0.31**
Oracle	98.46 ± 0.01	97.57 ± 0.59	88.53 ± 0.23	96.00 ± 0.02	88.75 ± 0.20	94.60 ± 0.04	-

where we observe that centralized algorithms particularly suffer during the transition period—when both concepts simultaneously exist, there is no single model that is an accurate fit at 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. On the other hand, while the clustering algorithms IFCA and CFL could flexibly employ specialized models across clients, they do sub-optimally and their accuracy is overall behind (details in Appendix E). For the case of CFL, its iterative cluster splitting reacts quickly, but creates excessive models for a staggered concept drift without unification.

For the 4-concept drift, the accuracy per time step on MNIST-4 is shown in Figure 5(right). We refer to the clustering learned by FedDrift in Figure 6, in which each cell indicates the model ID, and the background color indicates the ground-truth concept. We observe that at time 3, a local model is created for 5 of the 6 drifted clients, and by time 5, under iterative hierarchical clustering, a distinct model is learned for each concept with no excess models. The accuracy of FedDrift is close to Oracle throughout, with a gap at time 3 when each local model is created. Meanwhile, for FedDrift-Eager, just one model is initially created for both the yellow and green concepts, and its accuracy takes longer to recover. The performance of the centralized baselines never recover.

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 [24]. We observe FedDrift successfully adapts to the local drift, switching the model applied at Africa at year 2014, the time with a significant increase in single-unit residential buildings in Figure 4 in §3. Instead of creating a new model for 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).

To conclude, our evaluation under distributed drifts staggered in time and space demonstrates that FedDrift-Eager and FedDrift achieve accuracy significantly higher than existing baselines and comparable to an idealized algorithm with oracle knowledge of the ground-truth clustering.

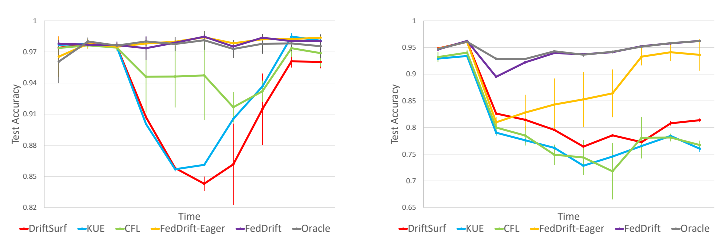


Figure 5: Accuracy at each time (averaged across clients) on CIRCLE-2 (left) and MNIST-4 (right).

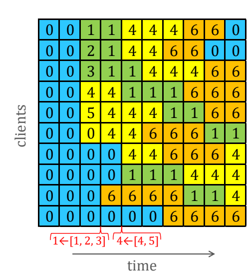


Figure 6: FedDrift on MNIST-4.

References

[1] Manuel Baena-García, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavaldà, and R Morales-Bueno. Early drift detection method. In *StreamKDD*, pages 77–86, 2006.

[2] Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Nikolaos Karianakis, Yuanchao Shu, Kevin Hsieh, Victor Bahl, and Ion Stoica. Ekya: Continuous learning of video analytics models on edge compute servers. *CoRR*, abs/2012.10557, 2020.

[3] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: Massive online analysis. *JMLR*, 11:1601–1604, 2010.

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

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

[6] Alberto Cano and Bartosz Krawczyk. Kappa updated ensemble for drifting data stream mining. *Machine Learning*, 109(1):175–218, 2020.

[7] Giuseppe Canonaco, Alex Bergamasco, Alessio Mongelluzzo, and Manuel Roveri. Adaptive federated learning in presence of concept drift. In *International Joint Conference on Neural Networks*, pages 1–7, 2021.

[8] Fernando E Casado, Dylan Lema, Marcos F Criado, Roberto Iglesias, Carlos V Regueiro, and Senén Barro. Concept drift detection and adaptation for federated and continual learning. *Multimedia Tools and Applications*, pages 1–23, 2021.

[9] Yujing Chen, Zheng Chai, Yue Cheng, and Huzefa Rangwala. Asynchronous federated learning for sensor data with concept drift. In *IEEE International Conference on Big Data*, pages 4822–4831, 2021.

[10] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018.

[11] Moming Duan, Duo Liu, Xinyuan Ji, Yu Wu, Liang Liang, Xianzhang Chen, Yujuan Tan, and Ao Ren. Flexible clustered federated learning for client-level data distribution shift. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[12] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence-SBIA*, pages 286–295, 2004.

[13] João Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), 2014.

[14] Abhinav Garg, Naman Shukla, Lavanya Marla, and Sriram Somanchi. Distribution shift in airline customer behavior during COVID-19. *CoRR*, abs/2111.14938, 2021.

[15] Avishek Ghosh, Justin Hong, Dong Yin, and Kannan Ramchandran. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*, 2019.

[16] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *NeurIPS*, pages 19586–19597, 2020.

[17] Yongxin Guo, Tao Lin, and Xiaoying Tang. Towards federated learning on time-evolving heterogeneous data. *arXiv preprint arXiv:2112.13246*, 2021.

[18] Maayan Harel, Koby Crammer, Ran El-Yaniv, and Shie Mannor. Concept drift detection through resampling. In *ICML*, pages 1009–1017, 2014.

[19] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

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

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

[24] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M. Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021.

[25] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492, 2016.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[28] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12): 2346–2363, 2018.

[29] Dimitrios Michael Manias, Ibrahim Shaer, Li Yang, and Abdallah Shami. Concept drift detection in federated networked systems. *arXiv preprint arXiv:2109.06088*, 2021.

[30] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[32] Ali Pesaranghader and Herna L Viktor. Fast hoeffding drift detection method for evolving data streams. In *ECML PKDD*, pages 96–111, 2016.

[33] Ali Pesaranghader, Herna L Viktor, and Eric Paquet. A framework for classification in data streams using multi-strategy learning. In *ICDS*, pages 341–355, 2016.

[34] Ali Pesaranghader, Herna L Viktor, and Eric Paquet. McDiarmid drift detection methods for evolving data streams. In *International Joint Conference on Neural Networks (IJCNN)*, 2018.

[35] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

[36] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

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

[38] Abhijit Suprem, Joy Arulraj, Calton Pu, and Joao Ferreira. ODIN: Automated drift detection and recovery in video analytics. *Proceedings of the VLDB Endowment*, 13(11), 2020.

[39] Ashraf Tahmasbi, Ellango Jothimurugesan, Srikanta Tirthapura, and Phillip B. Gibbons. DriftSurf: Stable-state / reactive-state learning under concept drift. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

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

[41] Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of local-update sgd algorithms. *Journal of Machine Learning Research*, 22(213):1–50, 2021.

[42] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1), 1996.

A Related Work

Concept drift refers to a change over time in the joint distribution $\mathcal{P}(x, y)$. By decomposing the joint distribution $\mathcal{P}(x, y) = \mathcal{P}(x)\mathcal{P}(y|x)$, we distinguish between drifts where only $\mathcal{P}(x)$ changes versus drifts where the feature-to-label mapping $\mathcal{P}(y|x)$ changes. Under the former case (which goes by the names virtual drift [40], covariate drift [37], and feature-distribution skew [22]) a single model can perform well despite change (although achieving fast convergence in an FL setting still requires a specialized strategy; e.g., FedProx [27]). But under the latter case where the feature-to-label mapping changes (real drift or concept drift [40]), lower loss can often be obtained by using specialized models for different concepts.

Concept drift has been studied extensively in the centralized setting for decades. We refer the reader to the surveys by Gama et al. [13] and Lu et al. [28]. As discussed in §1, directly applying centralized adaptation to a single global model in FL is not well-suited for distributed concept drifts with heterogeneous data across time and clients. Furthermore, centralized ensemble methods that use multiple models for adapting to drift also suffer, because the models in an ensemble are distinguished solely over time, and do not account for heterogeneity across clients—in response to a localized data drift, any newly created global model is trained over a mixture of concepts. We demonstrate this in our experimental evaluation, where we compare against state-of-the-art algorithms such as DriftSurf [39] and KUE [6].

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 [9, 17] or drift detection [29]. 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 like the example in Figure 1(left). First, Casado et al. [8] consider only the *virtual drift* 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. [7] propose Adaptive-FedAvg, in which the server tunes the learning rate used by all clients as a function of the variability across updates, with the goal of reacting fast when drift occurs while also achieving stable performance in the absence of drift. In our experimental evaluation, we compare against Adaptive-FedAvg.

Our solution to drift in FL 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 of clients is associated with a global model that they collaboratively train [4, 11, 15, 16, 30, 35]. As we extend the problem of data heterogeneity in FL with an additional dimension of time, we train multiple models with the algorithm in Appendix B, which is heavily inspired by the prior clustering algorithms IFCA [16] and HypCluster[30]. 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. Finally, our solution in §4 to handle an unknown number of concepts relies on hierarchical clustering, which has been studied in FL (in the static case) previously by Briggs et al. [4]. In the prior work, the clustering is based on clients' local updates, and it is unclear how to set the distance threshold at which to stop merging. In contrast, an advantage of our approach is that the stopping criterion is identical to the drift detection threshold, which has an intuitive interpretation of performance loss.

B Multiple-Model Training in FL

Distributed concept drift often means that multiple concepts are present simultaneously, necessitating the need for multiple-model training. In §4, we presented algorithms to learn a time-varying clustering of clients, where each cluster is associated with a global model. In this section, we show how train models for each cluster in Algorithm 3.

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 identity of client c at time t is denoted by the one-hot vector $w_c^{(t)}$, where $w_{c,m}^{(t)} = 1$ when assigned to the cluster associated with model h_m and 0 otherwise. The cluster identities $w_{c,m}^{(t)}$ 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.

Within each time, the training of the global models in Algorithm 3 is equivalent to Federated Averaging [31], since the aggregation weight of each client within each cluster is fixed at time τ. So the convergence of Algorithm 3 can be guaranteed by directly using previous analyses for Federated Averaging, such as [27, 41]. The difference here is that the objective function that clients are minimizing at time τ is replaced by the following:

$$\tilde{F}_m^{(\tau)}(h_m) = \sum_{c=1}^{P} \tilde{w}_{c,m}^{\tau} F_c^{(\tau)}(h_m) \tag{2}$$

where $F_c^{(\tau)}$ denotes the local objective function on client c, and the normalized weight is defined as $\tilde{w}_{c,m}^{\tau} = \sum_{t=1}^{\tau} w_{c,m}^{(t)} N_c^{(t)} / \sum_{c=1}^{P} \sum_{t=1}^{\tau} w_{c,m}^{(t)} N_c^{(t)}$.

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. This ideal solution is the Oracle algorithm in our evaluation in §5, and we empirically demonstrate in §5 that our proposed solutions achieve comparable accuracy.

Algorithm 3 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 $c = 1, 2, \ldots, P$
    and each model $m = 1, 2, \ldots, M$ in parallel **do**
      $h_{c,m} \leftarrow \textsc{LocalUpdate}(c, h_m, \{w_{c,m}^{(t)}\}_{t=1}^{\tau})$
    for each model $m = 1, 2, \ldots, M$ **do**
      $h_m \leftarrow \frac{\sum_{c=1}^{P} h_{c,m} \sum_{t=1}^{\tau} w_{c,m}^{(t)} N_c^{(t)}}{\sum_{c=1}^{P} \sum_{t=1}^{\tau} w_{c,m}^{(t)} N_c^{(t)}}$

  $\textsc{LocalUpdate}(c, h_m, \{w_{c,m}^{(t)}\}_{t=1}^{\tau})$:
  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

τ	current time (prior time indexed by t)
P	# clients (indexed by c)
M	# global models (indexed by m)
R	# communication rounds (indexed 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?

Algorithm 4 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 3

Note that, as stated, each client c in Algorithm 3 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.

In setting up the problem of distributed concept drift in FL (§2), we separated it 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 3. Suppose, hypothetically, that there is a global model already initialized for each concept up to some moderate accuracy. In this restrictive setting, Algorithm 4 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). The setting considered encompasses 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 4 does not have any mechanism to spawn new clusters or determine the number of clusters. In §4, we presented clustering algorithms that can spawn clusters over time to react to drifts to *new* concepts.

C Detailed Description of FedDrift

In this section, we give a more complete presentation of Algorithm 2 (FedDrift) first described in §4.2.

Under distributed drift in FL, data are heterogeneous both over time and across clients, where 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).

Algorithm 2 incorporates a bottom-up technique that *isolates clients* that detect drift (addressing **P2**) and *iteratively merges* clusters corresponding to the same concept (addressing **P1**) by leveraging hierarchical clustering.

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). Algorithm 2 merges clusters as shown in Algorithm 5, combining 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.

Algorithm 5 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

To specify a distance function for hierarchical clustering, Algorithm 2 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. [1] Then the distances between each cluster are initialized as $D(i, j) \leftarrow \max(L_{ij} - L_{ii}, L_{ji} - L_{jj}, 0)$.[2] The first term $L_{ij} - L_{ii}$ 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 (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 re-use the drift detection threshold δ to also represent the tolerance level up to which clusters can be merged in Algorithm 2, which avoids introducing another hyperparameter.

In Algorithm 2, 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 [18, 32, 34, 39].

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

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

The hierarchical clustering strategy of Algorithm 2 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 1. Algorithm 2 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 E, we discuss how we might restrict Algorithm 2 to create fewer overall models for higher efficiency.

Similar to Algorithm 3, 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 2's subsampling step.

D Datasets and Experimental Parameters

We consider concept drift with respect to the following datasets previously used in the concept drift and personalized FL literature [4, 5, 7, 29, 39]: SINE and CIRCLE [33] which each have 2 defined concepts, and SEA [3] and MNIST [26], 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 attributees 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).

Experiments are run using the FedML framework [19]. For each of the synthetic 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 §1. SINE and CIRCLE each have two 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.

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 report the accuracy averaged across all clients and all time steps (omitting the time of except for the times of drifts. We omit the times of drift because there is no chance for a client to adapt to the drift yet, and all algorithms suffer from the inevitable performance loss. By omitting the time of drift, we eliminate the noise from beneficial clustering mistakes if by chance a client was clustered to the model appropriate for the test data after the drift. For completeness, the results averaging over all time steps including drifts are in Appendix E. Each experiment is run for 5 trials, and we report the mean and the standard deviation.

We also evaluate on the real-world drift in the Functional Map of the World (FMoW) dataset included in the WILDS benchmark [10, 24]. 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" [24]. 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 §3 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. [24] 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 3 in Appendix B for training (which reduces to FedAvg when there is one cluster). 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 [20]. The training parameters used in our experiments are shown in Table 2. 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.

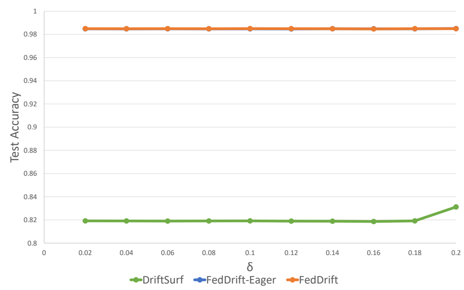
Regarding the learning rate selection, first we discuss all algorithms excluding Adaptive-FedAvg. We searched for learning rates of the form 10^{-a} for $a = 1, 2, 3, 4$, for each dataset, and found that $\eta = 10^{-2}$ was the best for SINE-2, CIRCLE-2, SEA-2, and SEA-4, that $\eta = 10^{-3}$ was best 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 $0.02, 0.04, \dots, 0.20$ 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 E 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 1 (on one trial).

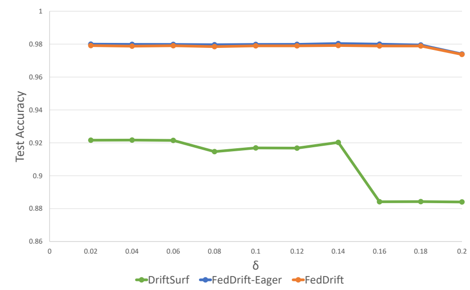
For all other hyperparameters of the algorithms we evaluate, we follow the parameter choices stated in the original papers, with the following exceptions: for DriftSurf we use $r = 3$ (which performed better than their suggested $r = 4$); for CFL we use $\gamma = 0.1$ (for which there is no default, but is shown to be a good setting from Theorem 1 and Figure 3 of their paper [35] 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.

Table 2: Training parameters

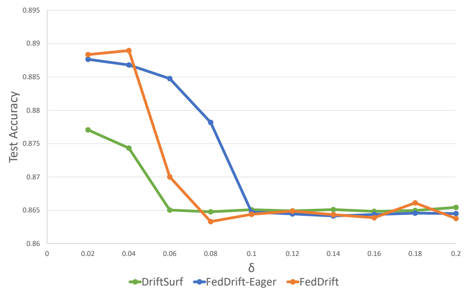
Parameter	Description	Experimental setting (all synthetic drifts)	Experimental setting (FMoW)
R	# communication rounds	100	10
K	# local steps per model per round	50	50
B	minibatch size	50	32
η	step size	varies	varies



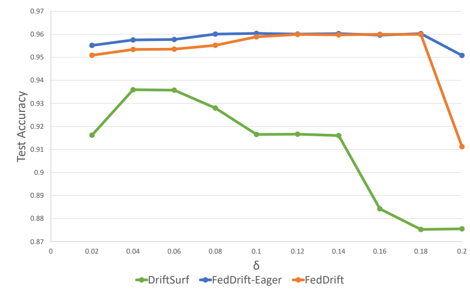
(a) SINE-2



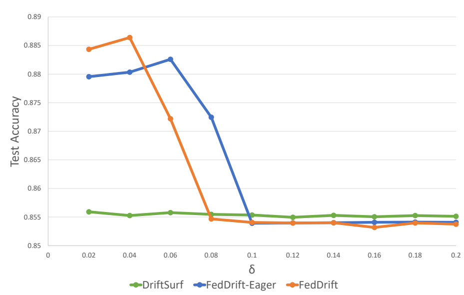
(b) CIRCLE-2



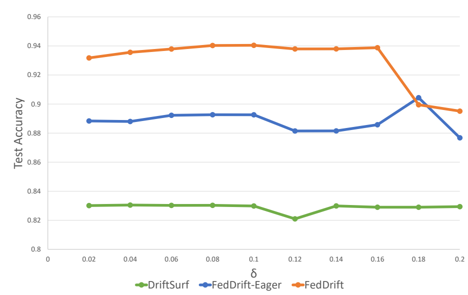
(c) SEA-2



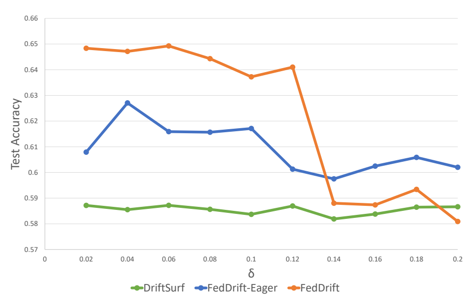
(d) MNIST-2



(e) SEA-4



(f) MNIST-4



(g) FMoW

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

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 cluster that can later be split, we observed the average test accuracy of FedDrift is 64.46%, or 0.45 pp worse.)

Finally, regarding the model training in Algorithm 3 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.)

E 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 [35].** 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 [16]** that we considered for extending the original algorithm to the time-varying setting. First, IFCA(T) is exactly Algorithm 4 in §B, 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. [16], 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 4. 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.

Table 3: 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	50.44 ± 1.52	88.36 ± 0.27	86.37 ± 0.34	87.25 ± 0.14	85.38 ± 0.28	82.97 ± 0.04	58.46 ± 0.08
DriftSurf	83.90 ± 1.01	92.54 ± 0.67	87.27 ± 0.34	91.71 ± 1.60	85.48 ± 0.28	82.99 ± 0.05	58.42 ± 0.16
KUE	87.05 ± 0.12	93.83 ± 0.04	87.62 ± 0.42	89.74 ± 0.07	85.53 ± 0.12	79.78 ± 0.16	37.46 ± 7.95
AUE	86.06 ± 0.60	92.74 ± 0.51	87.46 ± 0.12	92.19 ± 0.07	85.55 ± 0.08	81.29 ± 0.19	54.22 ± 0.14
AUE-PC	87.67 ± 1.70	93.05 ± 0.19	87.61 ± 0.08	92.22 ± 0.09	85.60 ± 0.05	81.43 ± 0.22	54.15 ± 0.10
Window	86.42 ± 0.74	93.67 ± 0.15	88.08 ± 0.10	92.15 ± 0.34	85.76 ± 0.16	81.16 ± 0.46	58.79 ± 0.14
Window-2	85.21 ± 1.67	93.03 ± 0.46	87.71 ± 0.33	92.54 ± 0.37	85.67 ± 0.16	82.16 ± 0.32	59.44 ± 0.23
Weighted-Linear	72.78 ± 1.23	89.91 ± 0.65	87.00 ± 0.01	89.70 ± 0.12	85.49 ± 0.17	82.79 ± 0.05	58.05 ± 0.17
Weighted-Exp	82.77 ± 0.64	92.69 ± 0.25	87.59 ± 0.15	92.19 ± 0.17	85.59 ± 0.09	82.55 ± 0.06	58.49 ± 0.09
Adaptive-FedAvg	78.02 ± 10.73	86.26 ± 0.00	86.69 ± 0.39	92.16 ± 0.04	85.32 ± 0.25	81.62 ± 0.07	52.76 ± 0.23
CFL	60.27 ± 4.82	88.39 ± 0.40	86.36 ± 0.28	86.97 ± 0.40	85.33 ± 0.26	81.95 ± 0.55	57.92 ± 0.32
CFL-W	95.15 ± 0.32	95.62 ± 1.14	87.66 ± 0.36	90.53 ± 0.81	85.67 ± 0.21	79.99 ± 0.58	58.70 ± 0.13
IFCA(T)	98.45 ± 0.03	91.72 ± 5.19	86.46 ± 0.23	87.33 ± 0.15	85.44 ± 0.14	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	98.46 ± 0.03	97.86 ± 0.20	88.35 ± 0.37	**95.99 ± 0.06**	88.08 ± 0.24	89.21 ± 2.02	61.62 ± 0.45
FedDrift	98.48 ± 0.01	**97.88 ± 0.17**	**88.65 ± 0.43**	95.93 ± 0.01	**88.41 ± 0.29**	**94.09 ± 0.08**	**64.91 ± 0.31**
FedDrift-C	98.51 ± 0.11	97.42 ± 0.57	88.30 ± 0.53	95.85 ± 0.05	87.46 ± 0.42	93.22 ± 0.44	61.86 ± 0.30
FedDrift-Eager-W	98.51 ± 0.12	97.34 ± 0.76	88.43 ± 0.23	94.05 ± 0.02	87.90 ± 0.25	89.31 ± 0.38	61.94 ± 0.38
FedDrift-W	**98.58 ± 0.17**	97.68 ± 0.09	88.43 ± 0.22	93.95 ± 0.02	88.17 ± 0.39	91.47 ± 0.07	64.22 ± 0.60
Oracle	98.46 ± 0.01	97.57 ± 0.59	88.53 ± 0.23	96.00 ± 0.02	88.75 ± 0.20	94.60 ± 0.04	-
Oracle-W	98.47 ± 0.03	97.84 ± 0.11	88.70 ± 0.17	94.04 ± 0.02	88.74 ± 0.13	91.89 ± 0.05	-

Table 4: 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	45.77 ± 1.52	87.12 ± 0.26	86.12 ± 0.35	86.28 ± 0.12	85.11 ± 0.24	81.60 ± 0.03	58.46 ± 0.08
DriftSurf	79.19 ± 0.88	91.16 ± 0.68	87.00 ± 0.35	90.55 ± 1.68	85.13 ± 0.19	81.62 ± 0.04	58.42 ± 0.16
KUE	87.05 ± 0.12	93.83 ± 0.04	87.62 ± 0.42	89.74 ± 0.07	85.53 ± 0.12	79.78 ± 0.16	37.46 ± 7.95
AUE	81.28 ± 0.81	91.50 ± 0.46	87.21 ± 0.11	91.07 ± 0.07	85.15 ± 0.07	79.65 ± 0.25	54.22 ± 0.14
AUE-PC	82.18 ± 2.01	91.75 ± 0.17	87.34 ± 0.08	91.07 ± 0.09	85.16 ± 0.04	79.70 ± 0.24	54.15 ± 0.10
Window	81.92 ± 0.88	92.40 ± 0.11	87.86 ± 0.08	91.35 ± 0.43	85.33 ± 0.10	78.88 ± 0.62	58.79 ± 0.14
Window-2	80.35 ± 2.02	91.73 ± 0.49	87.45 ± 0.34	91.47 ± 0.47	85.24 ± 0.15	80.06 ± 0.61	59.44 ± 0.23
Weighted-Linear	67.20 ± 1.43	88.67 ± 0.64	86.77 ± 0.02	88.56 ± 0.12	85.16 ± 0.11	81.38 ± 0.04	58.05 ± 0.17
Weighted-Exp	76.80 ± 0.88	91.30 ± 0.26	87.34 ± 0.16	91.05 ± 0.18	85.19 ± 0.06	80.96 ± 0.07	58.49 ± 0.09
Adaptive-FedAvg	73.82 ± 10.75	85.60 ± 0.00	86.55 ± 0.35	91.31 ± 0.05	85.01 ± 0.21	79.45 ± 0.06	52.76 ± 0.23
CFL	54.41 ± 4.33	87.08 ± 0.31	86.10 ± 0.30	86.00 ± 0.38	85.00 ± 0.25	80.45 ± 0.64	57.92 ± 0.32
CFL-W	86.83 ± 0.55	93.72 ± 0.93	87.36 ± 0.42	89.47 ± 0.74	85.25 ± 0.17	77.35 ± 0.81	58.70 ± 0.13
IFCA(T)	88.77 ± 0.02	90.06 ± 4.62	86.22 ± 0.22	86.36 ± 0.14	85.12 ± 0.09	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	88.76 ± 0.01	95.51 ± 0.18	87.86 ± 0.33	**94.09 ± 0.05**	86.64 ± 0.18	83.58 ± 0.79	61.62 ± 0.45
FedDrift	88.77 ± 0.02	**95.54 ± 0.15**	**88.13 ± 0.39**	94.03 ± 0.02	**86.68 ± 0.20**	**85.72 ± 0.07**	**64.91 ± 0.31**
FedDrift-C	88.82 ± 0.09	95.12 ± 0.50	87.78 ± 0.44	93.97 ± 0.05	86.21 ± 0.40	85.62 ± 0.47	61.86 ± 0.30
FedDrift-Eager-W	88.82 ± 0.12	95.05 ± 0.67	87.87 ± 0.23	92.04 ± 0.03	86.44 ± 0.20	82.15 ± 0.32	61.94 ± 0.38
FedDrift-W	**88.88 ± 0.15**	95.35 ± 0.08	87.95 ± 0.15	91.93 ± 0.03	86.46 ± 0.31	83.29 ± 0.06	64.22 ± 0.60
Oracle	88.77 ± 0.01	95.25 ± 0.52	87.99 ± 0.20	94.11 ± 0.02	86.89 ± 0.17	86.10 ± 0.03	-
Oracle-W	88.77 ± 0.03	95.51 ± 0.10	88.15 ± 0.14	92.03 ± 0.01	86.83 ± 0.06	83.58 ± 0.03	-

Test Accuracy Results. Table 3 (extending Table 1 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 4. 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 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 3 and 4, 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,

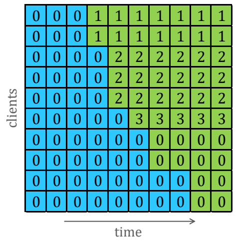


Figure 8: Clustering learned by CFL-W on SINE-2. Each cell indicates the model ID at each client and time step, and the background color indicates the ground-truth concept.

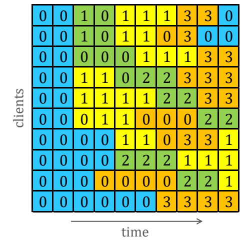


Figure 9: Clustering learned by FedDrift-Eager on MNIST-4. Each cell indicates the model ID at each client and time step, and the background color indicates the ground-truth concept.

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.

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 3.) 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. 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 0.87% accuracy due to the delayed learning of one of the two simultaneously arising concepts. Meanwhile, FedDrift-Eager suffers a larger 4.88% 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.

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 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 §3 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

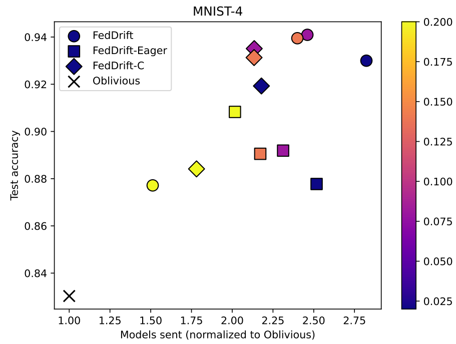


Figure 10: The accuracy-communication trade-off on MNIST-4 for FedDrift-Eager, FedDrift, and FedDrift-C. Each algorithm is evaluated under various selections of the splitting/merging threshold δ between 0.02 and 0.20, indicated by color. The vertical axis is the average test accuracy across clients and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

Table 5: Average accuracy (%), omitting drifts

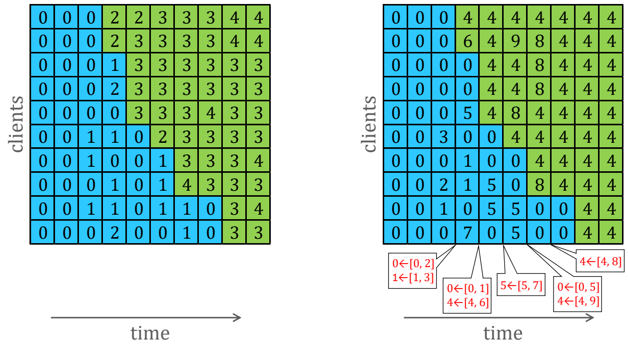
	MNIST-R
Oblivious	85.12 ± 1.37
DriftSurf	85.03 ± 1.36
KUE	81.56 ± 1.90
AUE	83.87 ± 1.64
AUE-PC	83.67 ± 1.66
Window	82.37 ± 1.94
Window-2	83.65 ± 1.83
Weighted-Linear	84.87 ± 1.34
Weighted-Exp	84.60 ± 1.44
Adaptive-FedAvg	83.17 ± 1.51
CFL	84.20 ± 1.54
CFL-W	82.24 ± 1.77
IFCA(T)	84.50 ± 1.21
IFCA	84.39 ± 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 6: Average accuracy (%), 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

shown in Table 5 (and in Table 6 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.

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

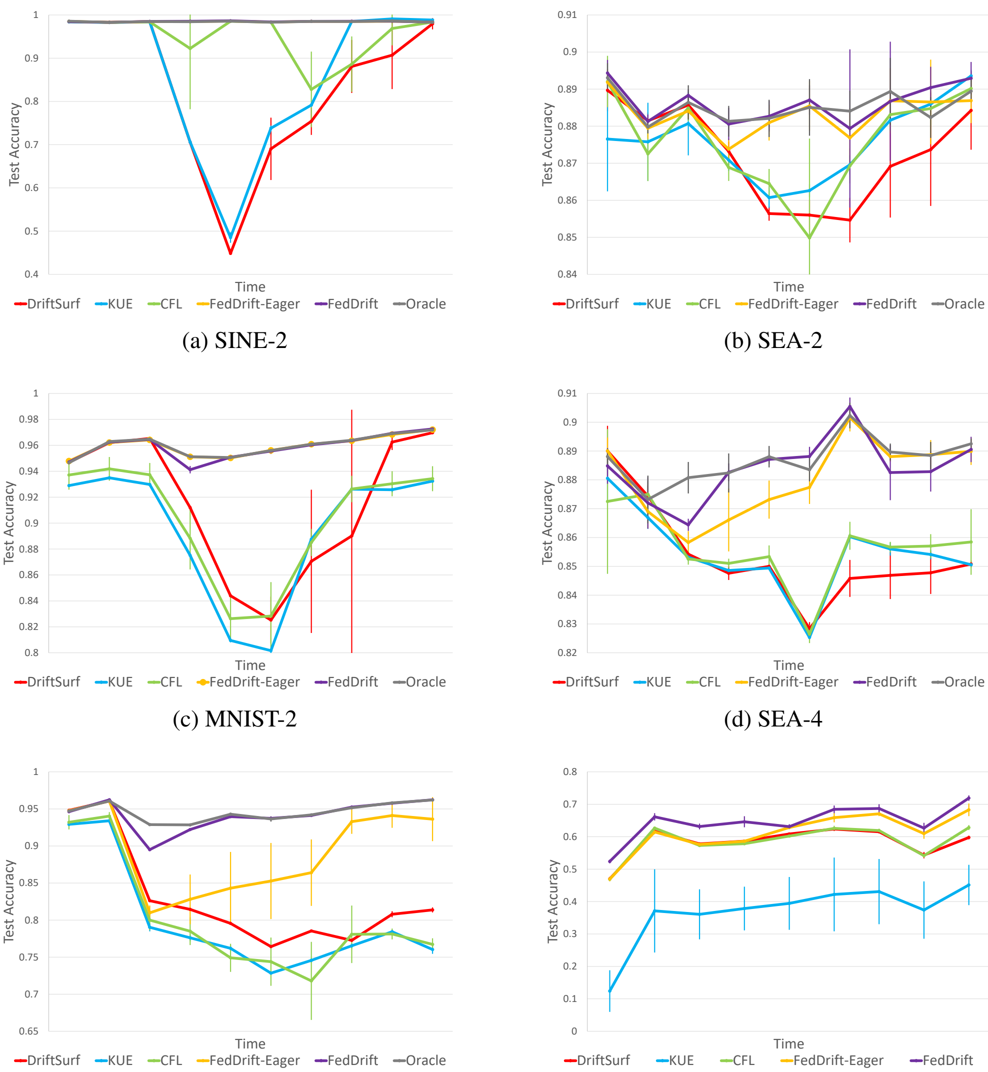


(a) FedDrift-Eager (b) FedDrift

Figure 11: 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.

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 D, it is relevant when there is greater uncertainty in selecting the threshold hyperparameter.

Test Accuracy Over Time. Finally, in Figure 12, we include plots omitted for space from the body of the paper on 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.) Similarly, here 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.