On the Local Cache Update Rules in Streaming Federated Learning

Heqiang Wang, Jieming Bian, and Jie Xu, *Senior Member, IEEE*

***Abstract*—In this study, we address the emerging field of streaming federated learning (SFL) and propose local cache update rules to manage dynamic data distributions and limited cache capacity. Traditional federated learning (FL) relies on fixed data sets, whereas in SFL, data is streamed, and its distribution changes over time, leading to discrepancies between the local training data set and long-term distribution. To mitigate this problem, we propose three local cache update rules—first-in–first-out (FIFO), static ratio selective replacement (SRSR), and dynamic ratio selective replacement (DRSR)—that update the local cache of each client while considering the limited cache capacity. Furthermore, we derive a convergence bound for our proposed SFL algorithm as a function of the distribution discrepancy between the long-term data distribution and the client's local training data set. We then evaluate our proposed algorithm on two data sets: 1) a network traffic classification data set and 2) an image classification data set. Our experimental results demonstrate that our proposed local cache update rules significantly reduce the distribution discrepancy and outperform the baseline methods. Our study advances the field of SFL and provides practical cache management solutions in FL.**

***Index Terms*—Cache update rules, federated learning (FL), streaming data.**

I. Introduction

FEDERATED learning (FL) is a distributed machine learning paradigm that enables a set of clients with decentralized data to collaborate and learn a shared model under the coordination of a centralized server. In FL, data is stored on edge devices in a distributed manner, which reduces the amount of data that needs to be uploaded and decreases the risk of user privacy leakage. While FL has gained popularity in the field of distributed deep learning, most research on FL has been conducted under ideal conditions and has not fully accounted for real-world constraints and features. Given that the client in FL is typically an edge device, we highlight two features that are more aligned with reality. The first feature, called *Streaming Data*, acknowledges that clients often consist of edge devices that continually receive and record data samples on-the-fly. Therefore, FL must operate on dynamic data sets that are built on incoming streaming data, rather than static ones. The second feature, called *Limited Storage*, recognizes that edge devices, such as network routers and IoT devices have limited storage space allocated for each service and application. As a result, only a restricted amount of space can be reserved for FL training without compromising the quality of other services. For instance, many smart home routers possess a storage capacity of only 9–32 MB [1], allowing them to store merely a few tens to hundreds of training data samples. This article aims to address the lack of consideration for these two real-world features in current FL research.

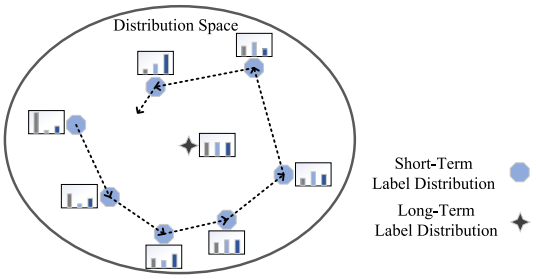


Fig. 1. Long-term label distribution and the trajectory of the short-term label distribution.

To address the problem presented above, we investigate a new FL problem, called streaming FL (SFL), where the local models of clients are trained based on dynamic data sets rather than static ones. In SFL, the relationship between the feature and label is fixed, but the distribution of features may vary across different rounds. SFL involves three different types of data distributions. The first one is the long-term (underlying) label distribution, which pertains to the data distribution of the client following a prolonged period of streaming data reception. This distribution cannot be anticipated during the training process and, due to storage capacity constraints, it is unfeasible to obtain an accurate long-term distribution by recording the whole data stream. The second type is the short-term (empirical) label distribution, which corresponds to the distribution of the client's currently received data. Short-term distributions are noisy and may vary over time, and they may not necessarily approximate the long-term distribution. The discrepancy between the long-term distribution and short-term distributions is illustrated in Fig. 1. The third type is the cached label distribution, which is the distribution of the data set currently stored in the client and is governed by the local cache update rule. The aforementioned three

Manuscript received 29 August 2023; revised 29 September 2023; accepted 21 October 2023. Date of publication 25 October 2023; date of current version 7 March 2024. This work was supported in part by the National Science Foundation under Grant 2006630, Grant 2033681, Grant 2029858, Grant 2044991, and Grant 2319780. *(Corresponding author: Heqiang Wang.)*

The authors are with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33146 USA (e-mail: hxw563@miami.edu; jxb1974@miami.edu; jiexu@miami.edu).

Data is available on-line at https://github.com/ystex/SFL-Appendix.git.
Digital Object Identifier 10.1109/JIOT.2023.3327316

distributions suggest that the primary challenge of SFL is the discrepancy between a priori unknown long-term distribution and the distribution of cached data for training, as the training data is continually gathered from the stream. As a result, a proper local data set update rule is essential to produce a cached distribution based on the short-term distributions so that it captures the long-term distribution as accurately as possible, thereby enhancing the learning performance. Our main contributions are summarized as follows.

1) We formulate the SFL problem and propose new FL algorithms for SFL. Unlike conventional FL, training in SFL must be conducted on a dynamic data set based on streaming data rather than a static data set. Clients also only have limited storage capacity, making storing all incoming data impossible.
2) We propose and study three different local data set update rules and theoretically analyze the discrepancy between the cached distributions and the long-term distribution. Based on this discrepancy analysis, we further prove a convergence bound of our proposed SFL algorithm.
3) We apply SFL to address a practical problem, namely online training of network traffic classifiers. Our experiments, which use both a network traffic classification (NTC) data set, the FMNIST data set, demonstrate that our proposed update rules outperform benchmarks in the SFL framework.

The remainder of this article is organized as follows. In Section II, we discuss related works on FL and NTC. Section III presents the system model and formulates the SFL problem. In Section IV, we introduce the SFL workflow, propose three local data set update rules, and analyze discrepancies. Section V presents the convergence analysis of SFL on non-i.i.d. data. The experimental results of SFL are presented in Section VI. Finally, Section VII concludes this article. All proofs can be found in the online supplementary material.

II. Related Work

In recent years, FL has emerged as a promising framework for decentralized deep learning. See [2], [3], [4] for comprehensive surveys. Among the various challenges in FL, the convergence analysis of FedAvg and its variants stands out as particularly crucial. Early works focused on FL under the assumptions of i.i.d. data sets and full client participation [5], [6], [7]. However, this assumption may not always hold in real-world FL scenarios, leading to an increasing number of works [8], [9], [10], [11], [12] investigating convergence proofs of FedAvg and its variants under non-i.i.d. data sets. Although the proposed SFL differs from conventional FL, our proof is primarily inspired by [12], which is based on nonconvex functions and non-i.i.d. data sets.

Recently, there has been a growing interest in exploring nonstationary and continually evolving data sets that are known as concept drift problems [13], [14]. In traditional streaming, the majority of works [15], [16], [17] employed the window-based approach to deal with the data, which essentially is the same as our first-in–first-out (FIFO) update rule. However, since the focus was mostly on dealing with the concept drift issue [18], many studies [16], [17] adopted the adaptive window approach where the window size can be adjusted. In our problem, since the storage space is limited and fixed, we consider only the fixed-window FIFO rule. Besides FIFO, we propose two new cache update rules, namely dynamic ratio selective replacement (DRSR) and static ratio selective replacement (SRSR), to mitigate the discrepancy between long-term and local cached distribution.

There have been some works under the FL setting to address the concept drift problem, using techniques, such as adjusting learning rates [19], incorporating regularization terms [20], [21], or training multiple models separately [22]. However, although SFL considered in this article has some connection to this literature, there is a fundamental difference. In the SFL problem, the global objective function remains constant depending on the constant albeit unknown long-term label distribution. However, in the concept drift problem, the underlying distribution changes, leading to a change in the global objective function. Apart from the concept drift problems, two works [23] and [24] consider a similar streaming data structure that is more relevant to our proposed SFL. In [23], an online approach is proposed to control local model updates on streaming data and global model aggregations of FL, in order to prevent training load congestion and spread the model training out with the arrival of streaming data. In [24], an online data selection framework is introduced for FL with streaming data, which regulates the data distribution of all clients to approach an i.i.d. distribution. Our study focuses on exploring the discrepancy between long-term label distribution and cached label distribution that arises from FL with streaming data and limited storage.

In terms of application, we applied SFL to online training of network traffic classifiers, which are used to categorize network traffic data into different types or classes based on certain characteristics of the network data. Such classifiers can be built by using either the traditional methods [25] or machine-learning-based methods [26], [27]. Nonetheless, these approaches typically rely on centralized deep learning models, which may not be the optimal choice for distributed scenarios involving edge devices such as routers. Some recent works [28], [29] used FL to address the issue of traffic classification, but their solutions do not consider the gradual arrival of network data or the limited storage capacity of edge devices.

III. Problem Formulation

A. System Model

Let us consider a network consisting of one server and K clients. Unlike conventional FL frameworks that use static data sets, every client in the considered system gradually acquires data from its online data source, and each of these online data sources has a long-term (underlying) label distribution. To facilitate exposition, we discrete time into periods (each of which corresponds to a learning round as we will define shortly) and assume that each client $k \in \{1, \ldots, K\}$ receives a

set $\mathcal{S}_t^k$ of B_s labeled data samples from its online data source in each period t. Each client k has a finite cache $\mathcal{L}^k$ of size $B > B_s$. For analytical simplicity, we assume that B is a multiple of B_s and denote $M = (B/B_s) \in \mathbb{Z}_+$. Because the client cache is limited, not all labeled data samples can be stored and used for learning at the same time.

Each client k has a *long-term label distribution* $\pi^k = [\pi^{k,1}\ \pi^{k,2}\ \cdots\ \pi^{k,R}]$, where R is the total number of label classes and $\pi^{k,r}$ represents the probability that class r appears in client k, which is unknown by the client beforehand. However, the short-term (empirical) label distribution can be different from the long-term label distribution and nonstationary over time as shown in Fig. 1. For example, in NTC, productivity applications may take up a large portion of network traffic in the daytime while entertainment applications are more popular at night. As a result, the application label distribution of $\mathcal{S}_t^k$ in one period is noisy and biased due to not only the finite number of instances but also the nonstationary application usage patterns. Furthermore, the short-term label distribution often does not change abruptly but exhibits temporal correlations. In other words, the application label distributions in the received labeled data set may be similar in adjacent periods. Let $n_t^{k,r}$ be the number of instances with label r in $\mathcal{S}_t^k$ and we denote $\boldsymbol{u}_t^k = [u_t^{k,1}\ u_t^{k,2}\ \cdots\ u_t^{k,R}]$ as the *short-term label distribution* of $\mathcal{S}_t^k$ where $u_t^{k,r} = n_t^{k,r}/B_s$. We make the following assumptions on $\boldsymbol{u}_t^k$.

Assumption 1 (Limited Temporal Correlation): There exists an integer $\Gamma > 0$ such that 1) for any $\tau \leq \Gamma$, we have $0 < \max_{k,r,t} \mathbb{E}[(u_t^{k,r} - \pi^{k,r})(u_{t-\tau}^{k,r} - \pi^{k,r})] \leq \delta^2$ for some constant δ^2; 2) for any $\tau > \Gamma$, $\mathbb{E}[(u_t^{k,r} - \pi^{k,r})(u_{t-\tau}^{k,r} - \pi^{k,r})] = 0$.

Assumption 1 states that the temporal correlation of the label distribution is confined in a neighborhood of Γ periods. For analytical simplicity, we assume the same δ^2 for any $\tau \leq \Gamma$ but practically it makes sense that δ^2 is larger for smaller τ since closer periods exhibit stronger correlation. Given that a diminishing correlation over time is a typical occurrence, this assumption is intuitive.

Because the client has a finite cache, we also define the *cached label distribution* at client k in period t, denoted by $\boldsymbol{v}_t^k = [v_t^{k,1}\ v_t^{k,2}\ \cdots\ v_t^{k,R}]$, as the label distribution of data currently in the cache. The cached label distribution is a joint result of both the short-term distribution and the local cache update rule.

To better understand these concepts, consider the NTC problem. Each local-area network (LAN) k connects to the network via a router/access point k, which monitors the application usage in the LAN. Instead of having the router directly upload raw data to the server for model training, we employ the FL framework, and these routers act as the clients in FL. Suppose there are a total number of R possible applications and NTC aims to identify the application $y \in \{1, \ldots, R\}$ based on the data packet feature x. In our problem, we consider that labeled data packets continuously arrive at the routers depending on the application usage pattern in the LAN for training the DL-based network traffic classifier. The labeled data packets may be manually labeled with delay and the number is kept small relative to the total data traffic in order to reduce the labeling overhead and complexity. It is important to note that the network traffic classifier problem represents only one instance of the broader SFL problem. In utilizing the network traffic classifier problem to illustrate SFL, our aim is simply to aid the reader's comprehension of the problem.

B. Learning Objective

Our goal is to train a machine learning model using the limited number of labeled data samples received by the different clients. Without loss of generality, we assume that the data arrival rate to all clients is the same. Therefore, the long-term label distribution of the overall network is simply the average of that of each client, i.e., $\pi = (1/K)\sum_{k=1}^K \pi^k$. We define the loss function as $f(w) = \mathbb{E}_{\xi\sim\pi} F(w;\xi)$ where $F(w;\xi)$ is the objective function with data sample/s ξ, ξ represents the sample/s drawn from the long-term label distribution, and the loss function can further be decomposed into a weighted sum of local loss functions as follows:

$$f(w) = \frac{1}{K}\sum_{k=1}^{K} f^k(w) = \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}_{\xi\sim\pi^k} F^k(w;\xi) \tag{1}$$

where $f^k(w) = \mathbb{E}_{\xi\in\pi^k} F^k(w;\xi)$ is the local loss function of client k. Thus, training the machine learning model is equivalent to solving for the optimal parameter w that minimizes the loss function, i.e., $\min_w f(w)$.

Because of the distributed nature of the network, it is impractical to send all the labeled data samples to a central location to train the model. Privacy concerns can also be another reason that forbids clients from directly exchanging data with each other. In this article, we take the FL approach to train the machine learning model in a distributed manner assisted by a parameter server, where clients train local models based on their local data and periodically exchange the local models with a parameter server to derive the global model. However, compared to conventional FL systems where local models are trained on static local data sets, the online machine learning model must be trained on time-varying dynamic data. As the labeled data samples are received gradually over time at the clients, the clients do not have access to the long-term label distribution at the beginning but must continuously update their finite local cache for the incoming training instances. The cached label distribution in the local cache may diverge from the long-term label distribution because of the short-term nonstationarity, thereby degrading the FL performance.

In the next sections, we introduce the SFL architecture for online machine learning model training and study how different local cache updating rules affect learning performance.

IV. SFL and Local Cache Update Rules

A. SFL Architecture

In the proposed SFL system, learning is organized into a series of iterative learning rounds. As previously mentioned, one period corresponds to a learning round. Each learning round t comprises the following five steps.

1) *Global Model Download:* Each client k downloads the current global model w_t from the parameter server. *Local Cache Update:* Due to the unique characteristics of SFL, specifically streaming data and limited storage, every client is required to update its local cache $\mathcal{L}_t^k$ before training the local model.
2) *Local Model Update:* Each client k uses w_t as the initial model to train a new local model w_{t+1}^k based on the current training data samples in its local cache $\mathcal{L}^k$. Because the local cache is finite and usually small, we consider local training performs E steps of full-batch gradient descent (GD). Specifically, the local model is updated as

$$w_{t,0}^k = w_t \tag{2}$$

$$w_{t,\tau+1}^k = w_{t,\tau}^k - \eta_L g_{t,\tau}^k \quad \forall \tau = 1, \ldots, E \tag{3}$$

$$w_{t+1}^k = w_{t,E}^k \tag{4}$$

where $g_{t,\tau}^k = \nabla F^k(w_{t,\tau}^k; \mathcal{L}_t^k)$ is the gradient computed on the local data set currently stored in the local cache $\mathcal{L}_t^k$, and η_L is the local learning rate. Note that because of the short-term nonstationarity and finite cache space, $\nabla F^k(w; \mathcal{L}_t^k) = \mathbb{E}_{\xi \sim \pi^k} F(w; \xi)$ does not hold.
3) *Local Model Upload:* Clients then upload their local model updates to the server. Typically, instead of uploading the local model w_{t+1}^k, client k may upload only the local model update Δ_t^k, which is defined as the total model difference as follows:

$$\Delta_t^k = \frac{1}{\eta_L}\left(w_{t,E}^k - w_{t,0}^k\right) = -\sum_{\tau=0}^{E-1} g_{t,\tau}^k. \tag{5}$$

4) *Global Model Update:* The server updates the global model by using the aggregated local model updates from the clients

$$w_{t+1} = w_t + \eta\eta_L \Delta_t, \text{ where } \Delta_t = \frac{1}{K}\sum_{k=1}^{K} \Delta_t^k \tag{6}$$

where η is the global learning rate.

The above steps highlight that the key of SFL lies in the local cache updating rules with limited storage space.

B. Local Cache Update

The key difference between conventional FL and SFL is how the local model update is performed, specifically, what data the local model is trained on. In conventional FL, the local model is trained on a static local data set (using either all data or sampled data) whereas in SFL, the local data set must be continuously updated as new data is received and old data is removed. Therefore, the local cache update rule will affect what data is used for training local models and consequently the global learning performance.

We illustrate the streaming data arrival and local cache updating in Fig. 2. Between two consecutive local model updates, new labeled data is received by the clients. In particular, client k receives a labeled data set $\mathcal{S}_t^k$ by the local cache update step in round t. Then client k updates the local cache $\mathcal{L}_t^k$ using the new data $\mathcal{S}_t^k$ and the existing data in the local cache according to some update rule Φ as follows: $\mathcal{L}_t^k \leftarrow \Phi(\mathcal{L}_{t-1}^k, \mathcal{S}_t^k)$. The updated local cache is then used for local model training at client k. Next, we introduce several local cache update rules.

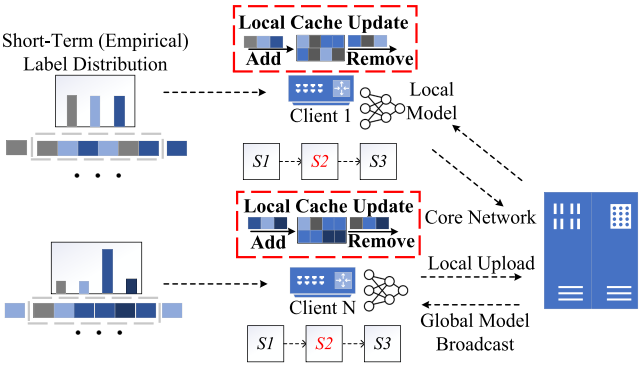


Fig. 2. SFL framework. There are three main stages in SFL: S_1 streaming data arrival. S_2 local cache updating. S_3 local model updating.

1) FIFO: A straightforward local cache update rule is FIFO, which is also used as a baseline for many other caching systems. Specifically, the FIFO update rule uses queuing logic to remove the oldest data so that a newly received data instance can be added. In our problem, client k simply removes the B_s oldest labeled data instances, denoted by $\mathcal{H}_{t-1}^k$, to make room for the new B_s labeled data instances in $\mathcal{S}_t^k$. Mathematically

$$\mathcal{L}_t^k \leftarrow \mathcal{L}_{t-1}^k \backslash \mathcal{H}_{t-1}^k \cup \mathcal{S}_t^k. \tag{7}$$

The cached label distribution changes as a result of the updated local cache as follows:

$$v_t^{k,r} = \frac{n_r(\mathcal{L}_{t-1}^k) - n_r(\mathcal{H}_{t-1}^k) + n_r(\mathcal{S}_t^k)}{B} \quad \forall r = 1, \ldots, R \tag{8}$$

where $n_r(\mathcal{X})$ is the number of data instances with label r in a set $\mathcal{X}$. We characterize the discrepancy between the cached distribution and the following long-term distribution.

Proposition 1: The discrepancy between the cached label distribution and the long-term label distribution by using FIFO is bounded as follows:

$$\mathbb{E}\left[\left(v_t^{k,r} - \pi^{k,r}\right)^2\right] \le \frac{1}{M}(\min\{2\Gamma + 1, M\})\delta^2. \tag{9}$$

Proposition 1 shows that FIFO update rule can reduce the distribution discrepancy in the local cache by a factor at most $\min\{([2\Gamma + 1]/M), 1\}$ compared to the short-term label distribution depending on how the short-term label distributions are temporally correlated (i.e., Γ) and the size of the cache (i.e., M). In particular, if the short-term label distributions are independent across time (i.e., $\Gamma = 0$), then FIFO is able to reduce distribution discrepancy by $1/M$. Moreover, as the local cache size increases to infinity, the discrepancy diminishes asymptotically, i.e., $\lim_{M\to\infty} \mathbb{E}[(v_t^{k,r} - \pi^{k,r})^2] \to 0$.

An obvious issue with the FIFO update rule is that the cached distribution $\boldsymbol{v}_t^k$ can still fluctuate significantly because of the short-term nonstationarity and the finite cache size, especially when the short-term label distribution is strongly temporally correlated and the cache size is small. Next, we propose two new cache update rules tailored to SFL.

2) Static Ratio Selective Replacement: The reason why FIFO may result in a large fluctuation in the short-term label distribution is that it is unable to use historical data instances and their label distribution information. The goal of SRSR is to smooth out the short-term label distribution and make it approximate the long-term label distribution by using a moving average type of update rule. Specifically, SRSR comprises two steps.

Step 1: SRSR computes a weighted average of the number of data instances with label r in the local cache and that of the newly received data, i.e.,

$$\tilde{n}_r = \left(1 - \frac{B_s}{B}\theta\right) n_r\left(\mathcal{L}_{t-1}^k\right) + \theta n_r\left(\mathcal{S}_t^k\right). \tag{10}$$

This will be the target number of data instances with the label r in the updated local cache. Here, the scalar B_s/B ensures that the size constraint of the local cache is always satisfied since one can easily verify that for any $\theta \in [0, 1]$, we have

$$\sum_{r=1}^{R} \tilde{n}_r = \left(1 - \frac{B_s}{B}\theta\right) B + \theta B_s = B. \tag{11}$$

Step 2: SRSR performs selective replacement to meet the target label numbers while utilizing the new data as much as possible. Specifically, there are two cases depending on the values of $\tilde{n}_r$ and $n_r(\mathcal{S}_t^k)$.

1) *Case 1:* $\tilde{n}_r \leq n_r(\mathcal{S}_t^k)$. In this case, SRSR removes all data with label r in $\mathcal{L}_{t-1}^k$ and uniformly randomly selects $\tilde{n}_r$ data instances with label r from $\mathcal{S}_t^k$ to insert into the local cache.
2) *Case 2:* $\tilde{n}_r > n_r(\mathcal{S}_t^k)$. In this case, SRSR uniformly randomly removes $n_r(\mathcal{L}_{t-1}^k) + n_r(\mathcal{S}_t^k) - \tilde{n}_r$ existing data instances with label r from the local cache and inserts all data instances with label r from $\mathcal{S}_t^k$ into the local cache.

Since the target label numbers are met, the cached label distribution by using SRSR is thus

$$\begin{aligned} v_t^{k,r} &= \frac{1}{B}\left(\left(1 - \frac{B_s}{B}\theta\right) n_r\left(\mathcal{L}_{t-1}^k\right) + \theta n_r\left(\mathcal{S}_t^k\right)\right) \\ &= \frac{\theta}{B}\sum_{\tau=0}^{t}\left(1 - \frac{B_s}{B}\theta\right)^{\tau} n_r\left(\mathcal{S}_{t-\tau}^k\right). \end{aligned} \tag{12}$$

The above equation shows that the cached label distribution takes all historical data distribution into account but discounts old information at a rate $1 - (B_s/B)\theta$.

Proposition 2: The discrepancy between the cached label distribution and the long-term label distribution by using SRSR is bounded as follows:

$$\begin{aligned} \mathbb{E}\left[\left(v_t^{k,r} - \pi^{k,r}\right)^2\right] &\leq 2\left(1 - \frac{\theta}{M}\right)^{2t}\left(\pi^{k,r}\right)^2 \\ &\quad + 2\left(1 - (1 - \frac{\theta}{M})^{2t}\right)\frac{\left(1 - \frac{\theta}{M}\right)^{-\Gamma} - \left(1 - \frac{\theta}{M}\right)^{\Gamma+1}}{2 - \frac{\theta}{M}}\delta^2. \end{aligned} \tag{13}$$

Corollary 1: By choosing θ sufficiently small, the bound on $\mathbb{E}[(v_t^{k,r} - \pi^{k,r})^2]$ decreases over t. Moreover

$$\lim_{t\to\infty} \mathbb{E}\left[\left(v_t^{k,r} - \pi^{k,r}\right)^2\right] \leq 2\frac{\left(1 - \frac{\theta}{M}\right)^{-\Gamma} - \left(1 - \frac{\theta}{M}\right)^{\Gamma+1}}{2 - \frac{\theta}{M}}\delta^2.$$

Proposition 2 and Corollary 1 imply that by choosing a small θ and with a large cache size M, SRSR can achieve a small label distribution discrepancy after sufficiently many rounds. On the other hand, the convergence to that small discrepancy is slower with a smaller θ. Moreover, even in the limit $t \to \infty$, the discrepancy bound does not vanish unless $M \to \infty$, i.e., the local cache has an infinity capacity.

3) Dynamic Ratio Selective Replacement: Now, we propose the DRSR update rule that overcomes the drawbacks of FIFO and SRSR. The goal of DRSR is to maintain the cached label distribution in the cache as the time-average short-term label distribution up to the current period. To this end, DRSR first uses a dynamic weight to compute the target numbers of data instances with different labels following a formula similar to (10) in SRSR, i.e.,

$$\tilde{n}_r = \left(1 - \frac{B_s}{B}\theta_t\right) n_r\left(\mathcal{L}_{t-1}^k\right) + \theta_t n_r\left(\mathcal{S}_t^k\right) \tag{14}$$

where $\theta_1, \ldots, \theta_t$ is the sequence of dynamic weights. Once $\tilde{n}_r \ \ \forall r$ is computed, DRSR follows the exact same step 2 as in SRSR to perform the selective replacement.

Proposition 3: The discrepancy between the cached label distribution and the long-term label distribution by using DRSR with $\theta_t = (B/[B_s t])$ is bounded as follows:

$$\mathbb{E}\left[\left(v_t^{k,r} - \pi^{k,r}\right)^2\right] \leq \frac{2\Gamma + 1}{t}\delta^2. \tag{15}$$

Proposition 3 shows that the discrepancy decreases over time, and the cached label distribution converges to the long-term label distribution at a rate of $O(1/t)$.

V. Convergence Analysis

In this section, we analyze the convergence of SFL. Because of the mismatch between the long-term label distribution $\boldsymbol{\pi}^k$ and the cached label distribution $\boldsymbol{v}_t^k$ of the local cache $\mathcal{L}_t^k$, the gradient $g_{t,\tau}^k = \nabla F^k(w_{t,\tau}^k; \mathcal{L}_t^k)$ computed in the local model update steps differs from the desired gradient on the long-term label distribution, i.e., $\nabla f^k(w_{t,\tau}^k)$. Thanks to the full batch GD, we are able to characterize the difference between $g_{t,\tau}^k$ and $\nabla f^k(w_{t,\tau}^k)$ through an intermediate variable, which we name the *virtual local gradient* and denote as $\hat{g}_{t,\tau}^k$. Specifically, $\hat{g}_{t,\tau}^k$ is defined as follows:

$$\hat{g}_{t,\tau}^k = \sum_{r=1}^{R} \pi^{k,r} \nabla F_r^k\left(w_{t,\tau}^k; \mathcal{L}_t^{k,r}\right) \tag{16}$$

where $\nabla F_r^k(w_{t,\tau}^k; \mathcal{L}_t^{k,r})$ is the gradient computed on only the subset of data instances with label r, denoted by $\mathcal{L}_t^{k,r}$, in the current local cache $\mathcal{L}_t^k$. We note that $\hat{g}_{t,\tau}^k$ is only imaginary since neither it is actually computed nor it can be realistically computed. This is because our algorithm does *not* actually divide $\mathcal{L}_t^k$ into R subsets $\mathcal{L}^{k,1}, \ldots, \mathcal{L}^{k,R}$ and compute the gradients on each of these sets. Instead, only a single local gradient $\nabla F^k(w_{t,\tau}^k; \mathcal{L}_t^k)$ is computed. More critically, even with $\nabla F_r^k(w_{t,\tau}^k; \mathcal{L}_t^{k,r}) \ \ \forall r = 1, \ldots, R$, computing $\hat{g}_{t,\tau}^k$ requires the knowledge of the long-term label distribution $\boldsymbol{\pi}^k$, which is unknown by the algorithm.

Before we move on to establish the connection between $g_{t,\tau}^k$ and $\nabla f^k(w_{t,\tau}^k)$ and prove the convergence of the proposed SFL algorithm under different local cache update rules, we make the following standard assumptions.

Assumption 2 (Lipschitz Smoothness): The local objective function is Lipschitz smooth, i.e., $\exists L > 0$, such that $\|\nabla f^k(x) - \nabla f^k(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^d$ and $\forall k$.

Assumption 3 (Unbiased Gradient Estimator): For each client k, the label-wise local gradient is unbiased, i.e., $\mathbb{E}_{\mathcal{L}^{k,r}} \nabla F^k(x; \mathcal{L}^{k,r}) = \nabla f^{k,r}(x) \triangleq \mathbb{E}_{\xi^r} \nabla F(x; \xi^r)$ where ζ^r is a instance with label r.

Assumption 4 (Bounded Dissimilarity): There exists constants $\sigma_G > 0$ and $A \geq 0$ so that $\|f^k(x)\|^2 \leq (A^2+1)\|f(x)\|^2 + \sigma_G^2 \quad \forall x \quad \forall k$. When the local loss functions are identical, $A^2 = 0$ and $\sigma_G^2 = 0$.

Assumption 5 (Gradient Bound): The label-wise local gradient is bounded, $\mathbb{E}[\|\nabla F^k(x; \mathcal{L}^{k,r})\|^2] \leq \sigma_M^2 \quad \forall k \quad \forall r \quad \forall \mathcal{L}^{k,r}$.

Assumption 2 ensures that the gradient of the local objective function does not change too abruptly between any two points in its domain. Assumption 3 states that for each client and for each label, the expected value of the local gradient estimator is equal to the true gradient. Assumption 4 ensures that while individual clients' loss functions can differ from a global function, the extent of this difference is bounded. Finally, Assumption 5 states that the expected squared magnitude of the gradient of the local loss function for each label remains bounded. Similar assumptions are commonly used in both the nonconvex optimization and FL literature [10], [11], [12], [30]. It is worth noting that all assumptions are primarily introduced to facilitate the convergence analysis of SFL. The proposed solutions can be applied in practice without these assumptions.

In the previous section, we established the upper bound on the cached label distribution and the long-term label distribution for different local update rules. To facilitate the exposition, we introduce a unified notation λ_t to represent the upper bounds. Specifically

$$\mathbb{E}\left[\left(v_t^{k,r} - \pi^{k,r}\right)^2\right] \leq \lambda_t^2. \tag{17}$$

The specific forms of λ_t can be found in Propositions 1–3 for FIFO, SRSR, and DRSR, respectively.

We begin by introducing some necessary lemmas to help us with the theorem that follows.

Lemma 1: The expectations of the difference between the real local gradient $g_{t,\tau}^k$ and virtual local gradient $\hat{g}_{t,\tau}^k$ is upper bounded as: $\mathbb{E}[|g_{t,\tau}^k - \hat{g}_{t,\tau}^k|^2] \leq R^2\lambda_t^2\sigma_M^2$. The difference between the virtual local gradient $\hat{g}_{t,\tau}^k$ and expected gradient $\nabla f^k(w_{t,\tau}^k)$ is upper bounded as: $\mathbb{E}[|\hat{g}_{t,\tau}^k - \nabla f^k(w_{t,\tau}^k)|^2] \leq 2R^2\overline{\pi}^2\sigma_M^2$, where $\overline{\pi} = \max_{k,r} \pi^{k,r}$.

The following result is on the upper bound for the τ-step SGD in the full participation case with Lemma 1.

Lemma 2: For any step-size satisfying $\eta_L \leq (1/8LE)$, we have: $\forall \tau = 0, \ldots, E-1$

$$\mathbb{E}\left[\|w_{t,\tau}^k - w_t\|^2\right] \leq 5E\eta_L^2R^2\lambda_t^2\sigma_M^2 + 60E^2\eta_L^2R^2\overline{\pi}^2\sigma_M^2 + 30E^2\eta_L^2\sigma_G^2 + 30E^2\eta_L^2\left(A^2+1\right)\|\nabla f(x)\|^2. \tag{18}$$

By defining $\Delta_t = \bar{\Delta}_t + e_t$, where $\Delta_t = -(1/K)\sum_{k=1}^K \sum_{\tau=0}^{E-1} g_{t,\tau}^k$ and $\bar{\Delta}_t = -(1/K)\sum_{k=1}^K \sum_{\tau=0}^{E-1} \hat{g}_{t,\tau}^k$, we can obtain the convergence bound of SFL with full client participation as follows.

Theorem 1: Let constant local and global learning rates η_L and η be chosen as such that $\eta_L \leq \min([1/\sqrt{60(A^2+1)}EL], [1/8LE])$ and $\eta\eta_L \leq [1/4EL]$. Under Assumptions 2–5 with full client participation, the sequence of model w_t in the real sequence satisfies

$$\min_{t=0,\ldots,T-1} \mathbb{E}\|\nabla f(w_t)\|^2 \leq \frac{f_0 - f_*}{c\eta\eta_L ET} + \Phi_G + \Phi_M + \Phi_L \tag{19}$$

where c is a constant, $f_0 \triangleq f(w_0)$, $f_* \triangleq f(w_*)$, w_* is the optimal model, and

$$\Phi_G = \frac{30E^2\eta_L^2L^2}{c}\sigma_G^2 \tag{20}$$

$$\Phi_M = \frac{60\eta_L^2E^2L^2R^2\overline{\pi}^2}{c}\sigma_M^2 \tag{21}$$

$$\Phi_L = \frac{\left(5\eta_L^2EL^2 + 3\eta\eta_LLE + 1\right)R^2\sigma_M^2}{cT}\sum_{t=0}^{T-1}\lambda_t^2. \tag{22}$$

The above convergence bound contains four parts: a vanishing term $([f_0 - f_*]/[c\eta\eta_LET])$ as T increases, a constant term Φ_G whose size depends on the problem instance parameters and is independent of T, a third term Φ_M is affected by the number of classes R and maximum ratio $\overline{\pi}$, and a final term Φ_L that depends on the cumulative gap between the real and virtual sequences. The key insight derived by Theorem 1 is that the SFL convergence bound depends on two additional terms Φ_M and Φ_L when compared to the conventional FL. For each client, if we could use the long-term label distribution, the cumulative ratio gap $(1/T)\sum_{t=0}^{T-1}\lambda_t^2 = 0$. Consequently, the convergence bound is simply $([f_0 - f_*]/[c\eta\eta_LET]) + \Phi_G + \Phi_M$. However, this gap cannot be eliminated since the client cannot directly use the long-term label distribution in the local model updating. By applying the specific learning rate, with $T \to \infty$, we can get the following corollary for the general convergence rate.

Corollary 2: With learning rates $\eta_L = (1/\sqrt{TE})$ and $\eta = \sqrt{EK}$, the convergence rate of the general case under full client participation is

$$\mathcal{O}\left(\frac{1}{\sqrt{EKT}}\right) + \underbrace{\mathcal{O}\left(\frac{\sigma_G^2}{T}\right)}_{\Phi_G} + \underbrace{\mathcal{O}\left(\frac{R^2\overline{\pi}^2\sigma_M^2}{T}\right)}_{\Phi_M} + \underbrace{\mathcal{O}\left(\frac{\sigma_M^2\sum_{t=0}^{T-1}\lambda_t^2}{T}\right)}_{\Phi_L}.$$

Based on the corollary 2 above, Φ_L is the major factor that determines whether the results converge to a stationary point without any constant terms. By substituting the values of λ_t^2 for the three update rules mentioned earlier, we can derive the corresponding final convergence rates. Furthermore, it is shown that under the *DRSR* update rule the SFL can eventually converge to a stationary point.

VI. Experiments

Setup: Our experiments are based on two data sets: 1) FMNIST and 2) the NTC data set extracted from ISCXVPN2016 as in [27]. FMNIST is a commonly used data set for image classification tasks, while NTC is a specialized data set for NTC. It contains 45 000 network packets that are divided into ten classes, each representing a different application, such as YouTube or Skype, which are encrypted traffic samples using various methods. The packet vectors can be reshaped to 39×39 bytes gray images. Further details and hyperparameters for the experiment are provided in the online supplementary materials [31], [32]. All experiment results reported are the average of 10 independent runs.

Data Stream Generation: The SFL system consists of 10 clients and every client receives training data samples from C classes in the long-term distribution, which is non-i.i.d between clients. The distribution of each class within every client is determined by random values ranging between $[0.05, 0.95]$ and the sum of these values for C classes equals 1. It is crucial to note that the value of parameter C can effectively adjust the degree of noniidness. A lower C value typically results in a higher degree of noniidness, as it might lead to each client having a distinct class of data samples. Conversely, a larger C value implies a lower degree of noniidness, and if C is equal to the total number of classes (e.g., $C = 10$), the scenario approximates an i.i.d case. To simulate the time-varying short-term distributions, we generate ten possible distributions for each client. In each time slot, the client receives one distribution as the short-term distribution. To capture the temporal correlation of the short-term distribution, the transition between any two short-term distributions is governed by a probability determined by the Kullback–Leibler (K-L) divergence [33] between these two label distributions. The probability of distribution transition between two distributions is higher when the K-L divergence between them is lower. The long-term distribution is obtained as the stationary distribution of these short-term distributions, based on the transition matrix, which is unknown to the client in advance. The test data set is established prior to training through sampling that corresponds to the long-term distribution and remains consistent throughout the FL rounds.

Benchmarks: In the experiment, the following two benchmarks are used for performance comparison.

1) *Full Information (FULL):* In this ideal scenario, each client has a local training data set with a distribution the same as the long-term distribution.
2) *Lazy Updates (LAZY):* In this scenario, the client keeps the initial training data set in the cache and does not update its data set. The client then conducts local training by utilizing this static local data set.

As we have analyzed in Section IV, both FIFO and SRSR update rules can converge to a stationary point with infinite cache capacity. However, since infinite cache capacity is impractical in real-world scenarios, we will only conduct experiments under finite cache capacity.

Label Distribution: We begin our analysis by examining the label distribution across clients, specifically in the context

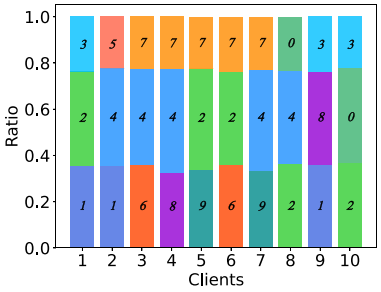


Fig. 3. Underlying Label Distribution on Clients ($C = 3$).

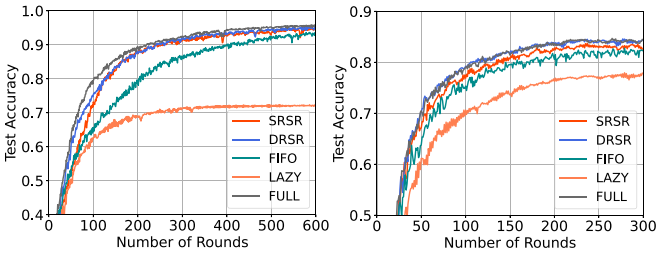
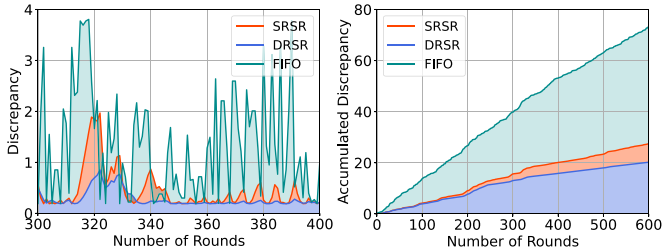


Fig. 4. Performance comparison of the proposed update rules and benchmarks with full participation. (a) NTC Non-iid ($C = 3$). (b) FMNIST Non-iid ($C = 3$).

where $C = 3$. Fig. 3 illustrates the long-term label distribution on clients. The figure presents the fact that the label distribution is non-i.i.d among the clients.

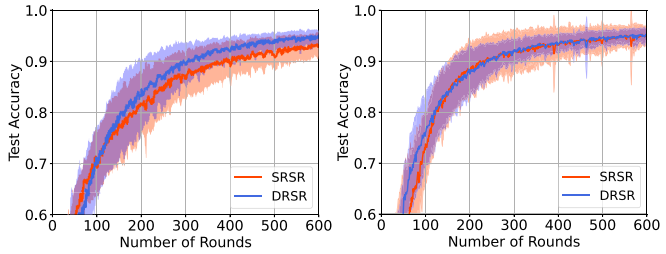
Performance Comparison: We first compare the convergence performance between our proposed update rules and benchmarks with parameters $\{B = 300, B_s = 150, C = 3, \theta = (2/3)\}$. Fig. 4(a) and (b) plots the convergence curves on the NTC data set and the FMNIST data set with full client participation, respectively. Several observations are made as follows. First, DRSR, SRSR, and FIFO outperform LAZY in terms of test accuracy and convergence speed, particularly in the later stages. DRSR and SRSR achieve performance close to FULL on the NTC data set due to their ability to gradually approximate the long-term label distribution. Second, DRSR and SRSR outperform FIFO in the entire learning process, mainly attributed to their ability to retain the knowledge of past data streams. Third, the learning performance of DRSR, SRSR, and FIFO is significantly better than LAZY on the NTC data set, while the performance gain is less significant on the FMNIST data set. Overall, DRSR and SRSR are better than FIFO on both data sets. More comparisons between DRSR and SRSR will be given later.

Distribution Discrepancy: The learning performance of SFL depends on how well it can approximate the long-term label distribution. In this part, we examine how the distribution discrepancy changes during the training process for different local data set update rules. The per-slot discrepancy is defined as $\psi_t = \sum_{k\in K}\sum_{r\in R}(v_t^{k,r} - \pi^{k,r})^2$ and the accumulated discrepancy is defined as $\psi = \sum_{t\in T}\sum_{k\in K}\sum_{r\in R}(v_t^{k,r} - \pi^{k,r})^2$. Fig. 5 shows the per-slot discrepancy and the accumulated discrepancy for the different update rules. From Fig. 5(a), we



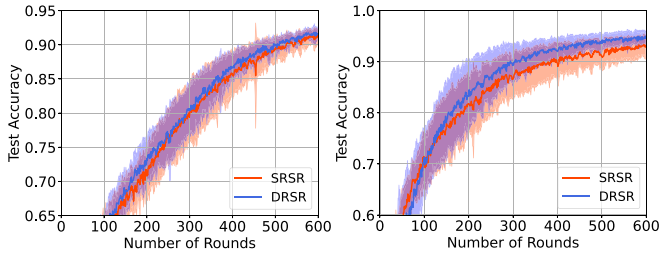
(a) (b)

Fig. 5. Discrepancy between the cached label distribution $v_t^{k,r}$ and the long-term label distribution ratio $\pi^{k,r}$. (a) Per-slot discrepancy ψ_t. (b) Accumulated discrepancy ψ.



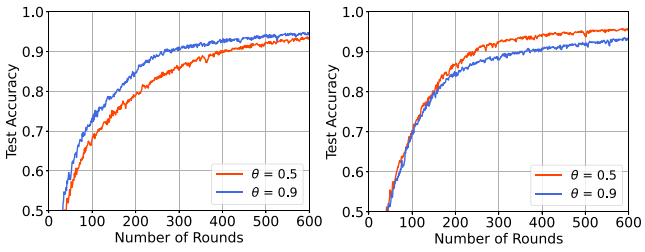
(a) (b)

Fig. 6. Performance comparison of SRSR and DRSR with different B_s. (a) NTC Non-iid ($B_s = 30$). (b) NTC Non-iid ($B_s = 150$).



(a) (b)

Fig. 7. Performance comparison of SRSR and DRSR with different B. (a) NTC Non-iid ($B = 100$). (b) NTC Non-iid ($B = 300$).



(a) (b)

Fig. 8. Performance comparison of SRSR with different θ. (a) NTC Non-iid ($B_s = 30$). (b) NTC Non-iid ($B_s = 150$).

can see that DRSR and SRSR exhibit lower discrepancy and fewer fluctuations compared to FIFO, and the discrepancy of DRSR decreases over time. Fig. 5(b) demonstrates that DRSR has the lowest cumulative discrepancy throughout the learning process and performs better than both SRSR and FIFO.

Impact of Streaming Data Size B_s: The proposed update rules, DRSR and SRSR were examined for their learning performance when trained with different values of B_s and a constant $B = 300$. A larger value of B_s corresponds to a larger streaming packet per round. The NTC data set was used with two different streaming data sizes, $B_s \in \{30, 150\}$, to investigate the effects of varying B_s. The results, shown in Fig. 6, indicate that both DRSR and SRSR are capable of achieving the desired level of test accuracy. The variance of DRSR decreases with training, particularly in the later stages, where it is significantly smaller than the variance of SRSR.

Impact of Cache Capacity B: In this set of experiments, we investigate the impact of varying the cache capacity, represented by B, on the learning performance while keeping the ratio (B_s/B) constant at 0.1. We use the NTC data set and tested two different values of $B \in \{100, 300\}$. Our results, as shown in Fig. 7, suggest that increasing B leads to higher test accuracy and faster training rates. For example, at 200 rounds, the test accuracy is 0.72 for $B = 100$ and 0.83 for $B = 300$. However, in practical situations, the cache capacity of a client is often limited, despite the potential for better performance with larger values of B.

Impact of Parameter θ: In this section, we examine the impact of θ, which tunes the amount of incoming data to put in the cache, on the learning performance of SRSR. A higher θ indicates that a greater portion of the current streaming data is chosen and incorporated into the local cache. If $\theta = 1$, every new data sample is incorporated into the local cache. Conversely, if $\theta = 0$, none of the new data samples are added to the local cache. The rest parameters are $\{B = 300, B_s = [30, 150], C = 3\}$. The experiment results are shown in Fig. 8. When B_s is large, a smaller θ leads to better learning performance. This occurs because larger θ can cause a substantial shift in the cached label distribution, leading to an increase in variation [as seen in Fig. 8(b)]. However, when B_s is small, choosing a small value for θ leads to a degradation of learning performance. The reason for this is that the ratio in the cached label distribution changes at a slow pace at the beginning [as seen in Fig. 8(a)]. Consequently, determining the appropriate θ beforehand is a challenging task. However, this issue can be resolved using the DRSR update rule, which reduces θ gradually over time.

VII. Conclusion

This article presents a novel FL framework named SFL, which differs from traditional FL by operating on a dynamic data set. This dynamic nature of the data, coupled with the limited cache capacity on clients, results in discrepancies between the local training data set and the long-term data distribution. We propose three update rules for the local cache update process in the SFL problem and provide a thorough theoretical analysis and experimental comparison to support our work. Future research will focus on developing more effective update rules for SFL to accelerate the training convergence speed and explore the potential of applying SFL to other practical scenarios.

REFERENCES

[1] S. Warner, "Buffer." 2021. [Online]. Available: https://people.ucsc.edu/warner/buffer.html

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[3] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[4] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1342–1397, 2nd Quart., 2021.

[5] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. Int. Conf. Learn. Rep.*, 2018, pp. 1–17.

[6] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5693–5700.

[7] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," in *Proc. ICML Workshop Coding Theory Mach. Learn.*, 2019, pp. 1–50.

[8] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

[9] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-IID data," 2020, *arXiv:1907.02189*.

[10] H. Yang, X. Zhang, P. Khanduri, and J. Liu, "Anarchic federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 25331–25363.

[11] D. Jhunjhunwala, P. Sharma, A. Nagarkatti, and G. Joshi, "FedVARP: Tackling the variance due to partial client participation in federated learning," in *Proc. Uncertainty Artif. Intell.*, 2022, pp. 906–916.

[12] H. Yang, M. Fang, and J. Liu, "Achieving linear speedup with partial worker participation in non-IID federated learning," in *Proc. Int. Conf. Learn. Rep.*, 2020, pp. 1–23.

[13] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2018.

[14] T. R. Hoens, R. Polikar, and N. V. Chawla, "Learning from streaming data with concept drift and imbalance: An overview," *Prog. Artif. Intell.*, vol. 1, no. 1, pp. 89–101, 2012.

[15] G. Widmer and M. Kubat, "Effective learning in dynamic environments by explicit context tracking," in *Proc. Mach. Learn. ECML-93 Eur. Conf. Mach. Learn.*, 1993, pp. 227–243.

[16] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *IEEE Trans. pattern Anal. Mach. Intell.*, vol. 16, no. 9, pp. 920–932, Sep. 1994.

[17] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*. 2007, pp. 443–448.

[18] J. Gama, I.*v*Zliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[19] G. Canonaco, A. Bergamasco, A. Mongelluzzo, and M. Roveri, "Adaptive federated learning in presence of concept drift," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–7.

[20] Y. Chen, Z. Chai, Y. Cheng, and H. Rangwala, "Asynchronous federated learning for sensor data with concept drift," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2021, pp. 4822–4831.

[21] F. E. Casado, D. Lema, M. F. Criado, R. Iglesias, C. V. Regueiro, and S. Barro, "Concept drift detection and adaptation for federated and continual learning," *Multimedia Tools Appl.*, vol. 81, no. 3, pp. 3397–3419, 2022.

[22] E. Jothimurugesan, K. Hsieh, J. Wang, G. Joshi, and P. B. Gibbons, "Federated learning under distributed concept drift," 2022, *arXiv:2206.00799*.

[23] Y. Jin, L. Jiao, Z. Qian, S. Zhang, and S. Lu, "Budget-aware Online control of edge federated learning on streaming data with stochastic inputs," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3704–3722, Dec. 2021.

[24] C. Gong, Z. Zheng, F. Wu, B. Li, Y. Shao, and G. Chen, "ODE: A data sampling method for practical federated learning with streaming data and limited buffer," 2022, *arXiv:2209.00195*.

[25] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1135–1156, 2nd Quart., 2013.

[26] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 1171–1179.

[27] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for DL-based traffic classifier update," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2020, pp. 397–405.

[28] H. Mun and Y. Lee, "Internet traffic classification with federated learning," *Electronics*, vol. 10, no. 1, p. 27, 2020.

[29] Y. Peng, M. He, and Y. Wang, "A federated semi-supervised learning approach for network traffic classification," 2021, *arXiv:2107.03933*.

[30] H. Wang and J. Xu, "Friends to help: Saving federated learning from client dropout," 2022, *arXiv:2205.13222*.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobilenetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

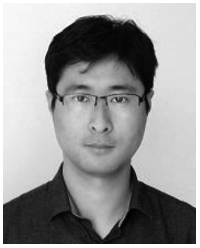
[33] J. M. Joyce, "Kullback-leibler divergence," *International Encyclopedia of Statistical Science*. Heidelberg, Germany: Springer, 2011, pp. 720–722.

Heqiang Wang received the B.S. degree in electrical and computer engineering from the University of Kentucky, Lexington, KY, USA, in 2016, and the M.S. degree in electrical and computer engineering from the University of Connecticut, Storrs, CT, USA, in 2019. He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of Miami, Coral Gables, FL, USA.

His research interests include resource allocation and bias mitigation federated learning problems.

Jieming Bian received the B.A. degree in economics from the University of Colorado Denver, Denver, CO, USA, in 2019, and the M.S. degree in operations research from Columbia University, New York, NY, USA, in 2021. He is pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of Miami, Coral Gables, FL, USA.

His research interests include communication efficiency and client scheduling federated learning problems.



Jie Xu (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2008 and 2010, respectively, and the Ph.D. degree in electrical engineering from the University of California at Los Angeles, Los Angeles, CA, USA, in 2015.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL, USA. His research interests include mobile-edge computing/intelligence, machine learning for networks, and network security.

Dr. Xu received the NSF CAREER Award in 2021.