DYNAMITE: Dynamic Interplay of Mini-Batch Size and Aggregation Frequency for Federated Learning with Static and Streaming Dataset

Weijie Liu, *Student Member, IEEE,* Xiaoxi Zhang, *Member, IEEE,* Jingpu Duan, *Member, IEEE,* Carlee Joe-Wong, *Senior Member, IEEE,* Zhi Zhou, *Member, IEEE,* and Xu Chen, *Senior Member, IEEE*

Abstract—Federated Learning (FL) is a distributed learning paradigm that can coordinate heterogeneous edge devices to perform model training without sharing private data. While prior works have focused on analyzing FL convergence with respect to hyperparameters like batch size and aggregation frequency, the joint effects of adjusting these parameters on model performance, training time, and resource consumption have been overlooked, especially when facing dynamic data streams and network characteristics. This paper introduces novel analytical models and optimization algorithms that leverage the interplay between batch size and aggregation frequency to navigate the trade-offs among convergence, cost, and completion time for dynamic FL training. We establish a new convergence bound for training error considering heterogeneous datasets across devices and derive closed-form solutions for co-optimized batch size and aggregation frequency that are consistent across all devices. Additionally, we design an efficient algorithm for assigning different batch configurations across devices, improving model accuracy and addressing the heterogeneity of both data and system characteristics. Further, we propose an adaptive control algorithm that dynamically estimates network states, efficiently samples appropriate data batches, and effectively adjusts batch sizes and aggregation frequency on the fly. Extensive experiments demonstrate the superiority of our offline optimal solutions and online adaptive algorithm.

Index Terms—Federated Learning, Edge Computing, Batch Size, Resource-Constrained

1 Introduction

Federated Learning (FL) [1]–[3] has gained much attention as it enables distributed model training by multiple collaborative devices without exposing their raw data. In the meanwhile, with the increasing amount of data generated from different geographical locations and the proliferation of edge computing technologies [4], [5], deploying FL at edge devices has become a promising computation paradigm to facilitate data-driven applications (e.g., smart surveillance and personalized healthcare) while preserving data privacy. Unlike traditional distributed machine learning (DML) [6], [7], FL allows each training device (a.k.a. worker) to perform multiple local updates before uploading their model parameters to the central server in each aggregation round, and it does not require partitioning a central pool of data across distributed workers.

Despite its advantages, FL still faces two major challenges: 1) skewed distributions and unbalanced sizes of training data at different devices (*statistical challenge*), and 2) heterogeneous and limited edge resources (*system challenge*). The former is also referred to as non-independent-and-identical (non-i.i.d.) data, which has been analyzed for representative FL algorithms, especially FedAvg [3]. Studies to address the system challenge have mainly focused on improving the learning efficiency by mitigating the impact of slow "straggler" devices on the wall-clock time of training and communication [5], [8]. In addition, the cost due to either the energy consumed over a long training period [9], [10] or operational charge paid to incentivize participating clients [11], [12] can be prohibitive for FL at the edge [13]. Thus, taking both time and cost into consideration when configuring training tasks on heterogeneous devices is of vital importance for FL algorithms. To address these challenges simultaneously, we call for a full-fledged FL algorithm that can capture the three-way trade-off between convergence, training time, and cost expenditure. Recent works have analyzed the model convergence when varying different controls, e.g., balancing the number of local updates and aggregation rounds [8], or adjusting workers' mini-batch sizes under a time budget [5], but these metrics are generally considered separately. In contrast, we propose to jointly optimize the aggregation frequency and mini-batch sizes, as they are the hyperparameters that determine the amount of data processed in each aggregation round and thus most affect these performance metrics.

Further, we have the following intuitions about these performance metrics. As illustrated in Figure 1-Left, increasing either the mini-batch size (interchangeably used with batch size in this paper) or the number of local updates can lead to more training samples processed and thus

- *Weijie Liu, Xiaoxi Zhang, Zhi Zhou and Xu Chen are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China. (Corresponding author: Xiaoxi Zhang.)*
  E-mail: liuwj55@mail2.sysu.edu.cn, {zhangxx89, zhouzhi9, chenxu35}@mail.sysu.edu.cn.
- *Jingpu Duan is with the Institute of Future Networks, Southern University of Science and Technology, Shenzhen 518055, China, and also with the Department of Communications, Peng Cheng Laboratory, Shenzhen 518066, China.*
  E-mail: duanjp@sustech.edu.cn.
- *Carlee Joe-Wong is with Department of Electrical and Computer Engineering, Carnegie Mellon University, CA 94035.*
  E-mail: cjoewong@andrew.cmu.edu.

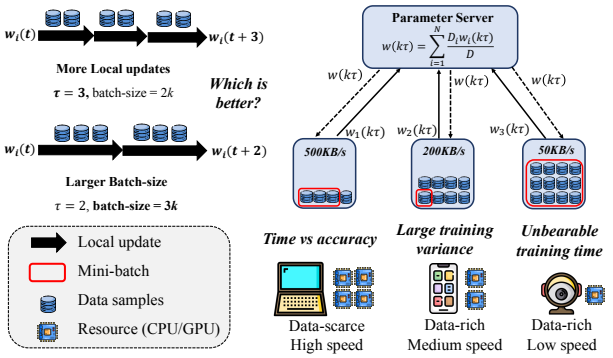


Fig. 1. Left: Interplay of batch size and aggregation frequency; Right: Different batch sizes among clients with various computation and communication capabilities.

improve the local model accuracy. However, doing so can also increase the consumed cost and training time. Moreover, a larger number of local updates (lower aggregation frequency) may result in a larger gap between the local and global models [3], hindering convergence, though this effect may depend on the batch size at each device. Therefore, we ask: *what is the best way to improve the FL model training when we can control both of these variables?* To the best of our knowledge, this work proposes the first attempt to co-optimize mini-batch size and global aggregation frequency under dynamic edge networks, considering performance metrics of model accuracy, training time, and resource cost.

This work also reveals that strategically choosing *different mini-batch sizes among clients* is crucial to improve model accuracy, time, and cost expenditure. A motivating example might be performing an FL task for object detection on heterogeneous edge devices using their locally captured pictures. As illustrated in Figure 1-Right, the generally accepted "no-straggler" principle [5], which assigns the batch sizes of different FL devices for ensuring a uniform time per aggregation round [5], [14], may not be optimal for this scenario. Specifically, the laptop with high training speed but relatively few data samples will have a large mini-batch while other data-rich devices such as the smartphone can only have a small mini-batch due to the relatively slow training speed. This could severely impede the convergence rate, as a small mini-batch size could introduce a high variance to the stochastic gradients (see Section 4). On the other hand, if we neglect the clients' heterogeneous computing capacities by simply setting a uniform batch size as what FL practitioners usually do [3], [13], [15], the straggler effects can be severe. Batch sizes, however, cannot help to limit battery usage and communication latency during model synchronization. Therefore, jointly choosing the aggregation frequency in the meanwhile is also important for balancing the energy cost, training time, and model accuracy. To achieve this, we make the following **technical contributions**:

1) *New convergence bound with respect to batch size and global aggregation frequency (Section 4).* We extend the FedAvg [3] FL framework by allowing different clients to use different mini-batch sizes. We capture FL clients' heterogeneity in the sizes and distributions of their datasets, based on which we then derive a novel convergence upper bound for the global model training, with respect to the aggregation frequency and batch sizes. Prior theoretical works usually assume a *full-batch* training setting to achieve bounded convergence rates, but practical FL deployments generally adopt the mini-batch approach. Our error bound can help bridge this inconsistency by quantifying the impacts of batch sizes considering the clients' heterogeneous data characteristics.

2) *Novel closed-form results and algorithm design for co-optimizing the batch size and aggregation frequency (Section 5).* We propose an optimization model to capture the complex trade-offs among accuracy, completion (computation plus communication) time, and cost. Driven by our derived convergence bound, we provide closed-form solutions that co-optimize the batch size and aggregation frequency uniformly across clients. These results capture the interplay between these two control variables and can be easily adopted by FL developers. We also propose an efficient algorithm to optimize *heterogeneous* batch sizes for different clients, which can further increase the model accuracy.

3) *Online adaptive joint optimization algorithm (Sections 6 and 7).* We design an adaptive control algorithm to dynamically choose the number of local updates and heterogeneous batch sizes among different clients, accommodating the online estimates of model convergence and system statistics across distributed training devices. Our algorithm can augment practical FL training strategies for both the cases of using static local datasets and dynamic data streams, with or without relying on limited data storage in a fluctuating edge network. Extensive experiments under different testbed settings demonstrate the superiority of our algorithms in terms of the accuracy, cost, and training time.

2 Related Work

Convergence analysis for FL has been extensively studied in recent years. For instance, [3] analyzes the convergence of the classic *FedAvg* algorithm on non-i.i.d. data and establishes an $O(1/T)$ convergence bound for strongly convex and smooth problems. A refined FL framework *FedProx* [2] has accounted for clients' different amounts of partial work, with provable convergence guarantees. Further, [16] proves that the asynchronous *FedAvg* has near-linear convergence to the global optimum for strongly convex optimization problems. A few other works propose FL algorithms and analysis for non-convex optimizations [17]–[19]. These FL convergence analysis works mainly focus on the effect of the number of local updates or total number of iterations.

Improving the FL efficiency has been studied in several directions, such as gradient compression [7], [20]–[22] and hyperparameter selection [5], [8], [23]. This work is orthogonal to the former (i.e., it can be combined with gradient compression), and falls in the latter regime, since we also aim to choose the best hyperparameters (i.e., batch-size and aggregation frequency). To optimize the learning speed, most studies choose hyperparameters to mitigate the effect of "straggler" devices, such as device sampling [13], [24], [25], client selection [11], [12], and staleness control [4], [26], [27]. Alternatively, recent works [5], [28], [29] also consider optimizing batch sizes to improve FL efficiency by equalizing the epoch time for each device to mitigate the

straggler effect. However, their works either lack theoretical analysis [28] or neglect data heterogeneity and resource constraints across clients [5], [29], which are important characteristics in edge systems. Several studies propose new strategies to handle streaming data for model training [30]–[33]. However, they do not focus on balancing the tradeoff among cost, accuracy, and training time.

Controlling FL under resource constraints has risen as the main challenge for edge-enabled FL training. An increasing number of studies have been proposed to improve FL accuracy under resource budgets, accounting for either completion time [34]–[36] or operational cost [9], [10]. Luo et al. [37] propose a cost-effective FL design to choose the number of participants and local updates for total training cost minimization, respectively. Wang et al. [8] derive a tractable convergence bound with an arbitrary number of local updates and design an algorithm for dynamically adjusting the aggregation frequency. Our work additionally analyzes the joint effect of mini-batch size on convergence, time, and cost metrics. A few recent works also consider choosing the mini-batch size. E.g, Ma et al. [5] propose a synchronous FL algorithm to adjust the batch size, and Liu et al. [22] jointly optimize the batch size, gradient compression ratio, and spectrum allocation for wireless FL. Our work provides a new convergence bound with respect to *heterogeneous* clients' batch sizes and provides *both closed-form optimal solutions and online adaptive controls* for jointly selecting the aggregation frequency and batch sizes.

3 Preliminaries and Problem Formulation

3.1 Federated Learning

We consider a parameter-server (PS) architecture, which consists of a set (defined as $\mathcal{N}$) of clients with $N = |\mathcal{N}|$ distributed edge devices (clients) and a centralized PS for global aggregation. Each device $i \in \mathcal{N}$ has a local data set $\mathcal{D}_i$ with D_i data samples $\mathbf{x}_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ..., \mathbf{x}_{i,D_i}]$, and $\mathcal{D}_i$ is non-i.i.d. across i. We define the loss function for each sample $\mathbf{x}_{i,j}$ as $f(\mathbf{w}, \mathbf{x}_{i,j})$ and the local loss function of device i as:

$$F_i(\mathbf{w}) = \frac{1}{D_i} \sum_{j \in \mathcal{D}_i} f(\mathbf{w}, \mathbf{x}_{i,j}). \tag{1}$$

The ultimate goal is to train a shared (global) model $\mathbf{w}$ that minimizes the global loss function, defined as:

$$F(\mathbf{w}) = \sum_{i \in \mathcal{N}} \frac{D_i}{D} F_i(\mathbf{w}), \tag{2}$$

where D is defined as $D = \sum_{i \in \mathcal{N}} D_i$.

As in the classic FedAvg [1] framework, clients divide their local data into mini-batches, perform multiple local updates, and upload their local models to the PS, which then broadcasts the updated global model to the clients by aggregating the local models. Prior works either assume using the whole dataset for each round (*full-batch training*) [3], [8] or simplify the effects of batch size on the convergence and training time in their analysis (e.g., [5]). Here we propose a more general FL setting by enabling customized batch sizes and the number of local updates.

TABLE 1: Main notations

K	The number of communication rounds
τ	The number of local update steps
T	The total number of iterations
B_i	The maximum buffer size of device i
$\mathcal{B}_i$	Buffer of device i
s_i	Batch size of device i
s_i^k	Batch size of device i at round k
$\mathbf{s_k}$	Batch size configuration at round k
p_i	Computational capacity of device i
t_{ci}	Computation time per update of device i
t_{ui}	Communication time per round of device i
$\mathcal{D}_i$	Local dataset of device i
$\mathcal{D}_i^k$	Local data stream of device i at round k
$\mathcal{D}$	Entire dataset over all devices
$\mathcal{D}_k$	Entire data stream over all devices at round k
$F(\mathbf{w})$	Global loss function
$F_i(\mathbf{w})$	Local loss function of device i
$F_{i,\mathcal{S}_i}(\mathbf{w})$	Local batch loss function of device i

3.2 Arbitrary batch size and aggregation frequency

To capture different batch sizes across clients, we define the loss function $F_{i,\mathcal{S}_i}(\mathbf{w})$ under a mini-batch instead of the original local loss function $F_i(\mathbf{w})$ for each end device i:

$$F_{i,\mathcal{S}_i}(\mathbf{w}) = \frac{1}{s_i} \sum_{j \in \mathcal{S}_i} f(\mathbf{w}, \mathbf{x}_{i,j}), \tag{3}$$

where $\mathcal{S}_i$ denotes a mini-batch randomly selected from $\mathcal{D}_i$, and s_i represents the size of $\mathcal{S}_i$. The full-batch training is a special case with $s_i = D_i$ and $F_{i,\mathcal{S}_i}(\mathbf{w}) = F_i(\mathbf{w})$. With a learning rate $\eta > 0$, the local update rule is defined as:

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) - \eta g_i(\mathbf{w}_i(t-1)), t \neq k\tau \tag{4}$$

where the batch gradient is $g_i(\mathbf{w}_i(t-1)) \triangleq \nabla F_{i,\mathcal{S}_i}(\mathbf{w}_i(t-1))$. We consider a total of K aggregation rounds (i.e., communication rounds) are performed in the FL training. The model update at each global aggregation step is:

$$\mathbf{w}(t) = \frac{\sum_{i=1}^{N} D_i \mathbf{w}_i(t)}{D}, \ t = k\tau, \tag{5}$$

where τ is the number of local updates in each aggregation round, meaning that the PS only performs (5) and sends the global model $\mathbf{w}(t)$ to the clients at $t = k\tau, k = 1, 2, ..., K$.

3.3 Accuracy-time-and-cost joint optimization model

Compared to data centers, mobile edge devices usually have limited computing resources such as CPUs and GPUs. Their limited battery lives also restrict the energy available for FL. Moreover, edge devices in FL training often establish the connection with the PS through the Wide Area Network [38], which could also incur high bandwidth costs in each communication round. It is therefore necessary to consider both computation and communication costs.

Limited budget for the total cost expenditure. Formally, we suppose that a units of computation cost are incurred (such as the cost of energy consumption) for processing a single sample, and b units of bandwidth cost are consumed in each global aggregation step. Let $s_{tot} = \sum_{i \in \mathcal{N}} s_i$

represent the sum of batch sizes per iteration over all the devices. We consider that the total cost incurred by the entire training process cannot exceed a constant R, i.e., $K(a\tau s_{tot} + b) \leq R$, which conforms to the definition of model training cost in [39]. Here, R can represent a cost budget of the energy consumption if the devices are owned by the FL owner, or the total rental fee of the edge devices if they are rented from another party. It can also be the budget of total monetary reward sent to participating clients [11], e.g., for compensating clients' battery consumption and/or privacy losses [40].

Heterogeneous system capacities. In practice, different edge devices can have heterogeneous computation and communication capacities, and the training time in each round is determined by the slowest device (straggler). Let p_i denote the computation speed (number of samples processed per time) of device i. We then define t_{ci} as the computation time of i for a single local update and assume that it is proportional to the batch size, i.e., $t_{ci} = s_i/p_i$. Further, t_{ui} is the communication time of each device i incurred by synchronizing her local model with the PS. These definitions are consistent with practical system modelings for FL training [5], [12]. Suppose that the FL task owner has an expected completion time deadline θ. We have the constraint on the completion time[1]:

$$\max_{i\in\mathcal{N}} K(\tau t_{ci} + t_{ui}) \leq \theta. \tag{6}$$

Our goal is to find the optimal batch sizes $\mathbf{s}^* = [s_1, s_2, ..., s_N]$ and the number of local update steps τ^* to minimize the gap between the expected global loss function $\mathbb{E}[F(\mathbf{w}(K\tau))]$ and the optimum F^* after performing K communication rounds, while satisfying the cost and completion time constraints. We define $[X] \triangleq \{1, \cdots, X\}$. Here we formulate the optimization problem as follows:

$$\textbf{Minimize}_{\mathbf{s},\tau} \quad \mathbb{E}[F(\mathbf{w}(K\tau))] - F^* \quad \textbf{(Training error)} \tag{7}$$

$$\textbf{S.t.} \quad \max_{i\in\mathcal{N}} K(\tau t_{ci} + t_{ui}) \leq \theta \quad \textbf{(Completion time)} \tag{8}$$

$$K(a\tau s_{tot} + b) \leq R \quad \textbf{(Cost)} \tag{9}$$

$$s_i \in [D_i], \forall i, \tau \in [\tau_{max}] \quad \textbf{(Feasibility)} \tag{10}$$

To solve the above optimization problem, we need to first navigate the complex trade-offs among the expected error, completion time, and total cost incurred by the training process, via controlling our decision variables $\mathbf{s}$ (mini-batch size) and τ (the number of local updates). We emphasize that, in addition to $\tau > 1$ unlike centralized DML, edge FL faces heterogeneous distributions and sizes of local datasets (D_i), and thus may yield heterogeneous optimal mini-batch sizes s_i across workers, which we shall show in Section 5. In contrast, the number of local updates τ needs to be uniform across clients, as unequal aggregation frequencies for different clients can cause objective inconsistency, i.e., the model converges to a mismatched objective function. While it is possible to address such inconsistencies during the aggregation process [41], we do not consider such scenarios for the sake of simplicity. Our first challenge is then to simultaneously quantify the effects of the $\mathbf{s}$ and τ in the training error, formalized in our next section.

1. To capture the randomness of computation/communication times of devices, (6) can be re-written as the constraint on the expected completion time: $K\mathbb{E}[\max_i (\tau t_{ci} + t_{ui})] \leq \theta$; but for simplicity we consider that the variance of each device's runtime is small, compared with their differences among the devices, so that it suffices to consider posing the constraint on a deterministic form of the runtime, e.g., defining t_{ci} and t_{ui} as expected runtimes in the first place.

4 Training Error Bound Analysis

In this section, we derive a new convergence bound to approximate (7), considering the effects of mini-batch sizes s_i and the number of local update steps τ. We first list our assumptions posed on the training model, which are generally adopted in pioneering FL works [42], [43]. We also evaluate the efficacy of our algorithm for training models that do not satisify these assumptions in Section 7.

Assumption 1. *ρ-quadratic-continuous: For each client $i \in \mathcal{N}$ and some constant $\rho > 0$, the batch loss function $F_{i,\mathcal{S}_i}$ satisfies: $\|F_{i,\mathcal{S}_i}(\mathbf{w}_1) - F_{i,\mathcal{S}_i}(\mathbf{w}_2)\| \leq \rho \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2$ for all $\mathbf{w}_1, \mathbf{w}_2$.*

Assumption 2. *β-smooth: For each client $i \in \mathcal{N}$ and some constant $\beta > 0$, the batch loss function $F_{i,\mathcal{S}_i}$ satisfies: $\|\nabla F_{i,\mathcal{S}_i}(\mathbf{w}_1) - \nabla F_{i,\mathcal{S}_i}(\mathbf{w}_2)\| \leq \beta \|\mathbf{w}_1 - \mathbf{w}_2\|$ for all $\mathbf{w}_1, \mathbf{w}_2$.*

The local and global loss function satisfy the above assumptions straightforwardly due to the definition of $F_i(\cdot)$ and $F(\cdot)$.

Assumption 3. *Polyak-Łojasiewicz condition [44]: There exists some constant c that $0 < c \leq \beta$ and $c \leq 2\rho$, and for each device $i \in \mathcal{N}$, the global loss function $F(\mathbf{w})$ satisfies: $\|\nabla F(\mathbf{w})\|_2^2 \geq 2c(F(\mathbf{w}) - F^*),\ \forall \mathbf{w}$.*

Assumption 4. *First and Second Moment Limits: For some scalars $\mu_G \geq \mu > 0$ and $M_i > 0$, under any given model $\mathbf{w}$ and batch of data samples ξ_t randomly selected from $\cup_i \mathcal{D}_i$ at step t, the global batch-gradient $g(\mathbf{w}, \xi_t)$ and the variance of the gradient under any single data $\mathbf{x}_{i,j} \in \mathcal{D}_i$ of each client i, denoted as $\mathbb{V}_{\mathbf{x}_{i,j}}[\nabla f(\mathbf{w}, \mathbf{x}_{i,j})]$, satisfy:*

$$\begin{aligned} \nabla F(\mathbf{w})^T \mathbb{E}_{\xi_t}[g(\mathbf{w}, \xi_t)] &\geq \mu \|\nabla F(\mathbf{w})\|_2^2, \\ \|\mathbb{E}_{\xi_t}[g(\mathbf{w}, \xi_t)\|_2 &\leq \mu_G \|\nabla F(\mathbf{w})\|_2, \\ \mathbb{V}_{\mathbf{x}_{i,j}}[\nabla f(\mathbf{w}, \mathbf{x}_{i,j})] &\leq M_i,\ \forall i \in \mathcal{N}. \end{aligned}$$

Assumption 5. *Bounded Gradient Divergence (non-i.i.d. degrees): Let $g(\mathbf{w})$ denote the global gradient under the dataset $\cup_i \mathcal{D}_i$. For some bounded scalar $\delta_i > 0$, the local gradient of each client i under her full dataset $\mathcal{D}_i$ satisfies:*

$$\|g_i(\mathbf{w}) - g(\mathbf{w})\| \leq \delta_i,\ \forall \mathbf{w}, i.$$

Based on the above assumptions, we show our first main result, an upper-bound of the training error with different batch sizes and uniform local update steps across devices, in the following theorem.

Theorem 1 (Error bound with heterogeneous batch sizes s_i)**.** *Suppose that the loss functions satisfy Assumptions 1–5. Assuming $F^* \geq 0$, given a fixed learning rate $0 \leq \eta \leq \frac{\mu}{\beta\mu_G^2}$ and the initial global parameter $\mathbf{w}(0)$, the expected error after K*

aggregation rounds with τ local updates per round is:

$$\mathbb{E}[F(\mathbf{w}(K\tau))] - F^* \leq q^{K\tau}\left[F(\mathbf{w}(0)) - F^*\right] + \frac{1-q^K}{1-q}\left(\frac{\beta\eta^2(1-q^\tau)}{2D^2(1-q)}\sum_{i\in\mathcal{N}}\frac{M_i D_i^2}{s_i} + \rho h(\tau)^2\right), \quad (11)$$

where $q = 1-\eta c\mu$, $h(\tau) = \frac{\delta}{\beta}\left((\eta\beta+1)^\tau - 1\right) - \eta\delta\tau$, and $\delta = \sum_{i\in\mathcal{N}}\frac{D_i\delta_i}{D}$. Especially, when $\tau = 1$, the above theorem is consistent with the DML convergence rate in prior works [43].

We provide the full proof in Appendix A.

Our bound (11) has a richer structure than those in [5], [8], [22] to show the effects of s_i, τ, and the data distributions. The first term is determined by the initial global loss, which continuously decreases during the training process. The term associated with M_i can be interpreted as the "gradient variance loss" resulting from the error of using a randomly selected batch to estimate the loss gradient under the entire local dataset. The last term $\rho h(\tau)^2$ can be regarded as the "local bias" which monotonically increases with τ, since a larger τ means less frequent communications between the clients and server and thus a larger gap between the global and local models.

5 An Offline Algorithm and Theories

In this section, we provide optimal solutions for co-optimized stationary batch sizes and the number of local updates in two cases. The total number of aggregation rounds K is pre-determined in our problem [15], [25], [41]. We assume they are all *offline* settings, where the parameters related to the model (in (11)) and the system (in the optimization constraints) can be obtained, e.g., through pre-run tests [6], [13]. We will design an adaptive control algorithm with parameters estimated online in Section 6.

5.1 Case 1: Co-optimizing uniform s and τ

We first consider the most common FL scenario in practice [1], [2] where every device has the same batch size s and number of local updates per round τ. Based on our bound (11), we derive closed-form solutions of s and τ in Theorem 2, by solving (7)–(10) with $s_i = s_{i'},\ \forall i \neq i'$.

Theorem 2 (Interplay of uniform s and τ)**.** *Given the number of aggregation rounds K and a feasible deadline ($\theta > Kt_{ui}$) and cost budget ($R > Kb$), the optimal uniform batch size s^* and the number of local updates τ^* satisfy:*

$$s^*(\tau) = \mathbf{min}\left\{\frac{R-Kb}{a\tau n}, \min_{i\in\mathcal{N}}\left\{\frac{p_i(\theta - Kt_{ui})}{K\tau}\right\}\right\}, \quad (12)$$

$$\tau_1 = \lfloor\hat{\tau}\rfloor,\ \tau_2 = \lceil\hat{\tau}\rceil,\ \frac{\partial f(\hat{\tau})}{\partial\tau} = 0, \quad (13)$$

$$\tau^* = \mathop{\arg\min}_{\tau\in\{\tau_1,\tau_2\}} f(\tau),\ s^* = \lfloor s^*(\tau^*)\rfloor, \quad (14)$$

where $h(\tau) = \frac{\delta}{\beta}\left((\eta\beta+1)^\tau - 1\right) - \eta\delta\tau$ and $f(\tau) = q^{K\tau}G(0) + \frac{1-q^K}{1-q}\left(\frac{\beta\eta^2(1-q^\tau)}{2D^2(1-q)}\sum_{i\in\mathcal{N}}\frac{M_iD_i^2}{s^*(\tau)} + \rho h(\tau)^2\right)$, $G(0) = [F(\mathbf{w}(0)) - F^*]$ and $q = 1-\eta c\mu$ as defined in Theorem 1.

We provide the full proof in Appendix B.

Our result quantitatively verifies an intuitive common practice that communicating with the PS every iteration ($\tau = 1$) is the optimum, if the number of aggregation rounds K and thus the total number of training iterations are sufficiently large (shown in Remark 1).

Remark 1. As the number of aggregation rounds K increases, the optimal solution τ^*, which is expressed in (14), will decrease to 1, i.e., $\lim_{K\to\infty}\tau^* = 1$.

Proof. When K is small, the first term $q^{K\tau}G(0)$ dominates $f(\tau)$, and it monotonically decreases with τ. As K grows larger, the second term dominates $f(\tau)$ and monotonically increases with τ. Thus the optimal local update steps τ^* decreases with the increase of K. □

Remark 1 can be intuitively explained as follows. When the number of communication round K is small, e.g., due to the high communication cost or limited bandwidth, a bigger τ leads to a larger total number of model updates and thus higher accuracy while incurs limited communication cost. In contrast, if K is sufficiently large, especially in the later stage of the training process, one should reduce τ and increase the batch size s, since a larger τ may increase the gap between the global and local models and thus incurs a larger final error. This implication is consistent with the intuition in [8].

5.2 Case 2: Co-optimizing τ and heterogeneous s_i

In this case, we generalize Case 1 by enabling different batch sizes assigned for different clients. Since edge devices can have different and potentially limited computation and communication capacities, increasing the batch size at different clients lengthens the total computation time by different amounts. Following [5], [12], the computation time of each step of local update can be modeled by $t_{ci} = s_i/p_i$. For the clarity of the following analysis, we first fix the value of τ, then our optimization problem becomes:

$$\mathop{\mathbf{Minimize}}_{\substack{\mathbf{s}=[s_1,s_2,\ldots s_N]\\ \tau\in[\tau_{max}]}} \quad \sum_{i\in\mathcal{N}}\frac{M_iD_i^2}{s_i} \quad (15)$$

$$\mathbf{S.t.} \quad s_i \leq p_i\left(\frac{\theta}{K\tau} - \frac{t_{ui}}{\tau}\right), s_i \in [D_i],\ \forall i \quad (16)$$

$$s_{tot} = \sum_{i\in\mathcal{N}} s_i \leq (R-Kb)/(a\tau) \quad (17)$$

Directly applying an integer programming optimizer such as Gurobi [45] to solve (15)–(17) or using brute force algorithm may incur a high time complexity with at least $O(\kappa^N\tau_{max})$, where $\kappa = \frac{s_{tot}}{N} >> 1$. Instead, we design a more efficient exact algorithm, as we state in the following theorem.

Theorem 3. *Given the number of aggregation rounds K and the maximum number of local updates per round τ_{max}, Algorithm 1 outputs the optimal batch sizes $\mathbf{s}^*$ and τ^* for FL training with at most $O(N^2\tau_{max})$ time complexity.*

Detailed proofs are all deferred to Appendix C.

Intuition of Algorithm 1. Since the objective function (15) decreases with s_i, the time constraint is transformed to (16), which defines the largest batch size allowed for any device i under deadline θ, i.e., $s_i(\theta) = p_i\left(\frac{\theta}{K\tau} - \frac{t_{ui}}{\tau}\right)$. Similarly, the cost constraint (17) is equivalent to defining a total batch size under the cost budget R, i.e., $\sum_{i\in\mathcal{N}} s_i = s_{tot}(R) =$

$\frac{R-Kb}{a\tau}$. If neglecting $s_i(\theta)$ firstly, the Cauchy–Schwarz inequality yields:

$$\sum_{i\in\mathcal{N}} \frac{M_i D_i^2}{s_i} \cdot \sum_{i\in\mathcal{N}} s_i \geq \sum_{i\in\mathcal{N}} \left(\sqrt{M_i} D_i\right)^2. \tag{18}$$

Since $\sum_{i\in\mathcal{N}} \left(\sqrt{M_i} D_i\right)^2$ and $s_{tot}(R)$ are both constants, we can minimize the objective function $\sum_{i\in\mathcal{N}} \frac{M_i D_i^2}{s_i}$ when the equality holds with $\frac{\sqrt{M_1}D_1}{s_1} = \frac{\sqrt{M_2}D_2}{s_2} = \cdots = \frac{\sqrt{M_N}D_N}{s_N}$, i.e., $s_i \propto \sqrt{M_i} D_i$. Then, considering $s_i(\theta)$, we need to reduce s_i to $s_i(\theta)$ for *time-constrained devices* which have $s_i > s_i(\theta)$ (Lines 9-10). The s_i of those devices will not be revised (Line 11) since they reach the maximum allowed batch size. In addition, we will re-assign (increase) the s_i of other devices while keeping $s_i \leq s_i(\theta)$ satisfied to make the best use of the extra data samples due to the reduced s_i of those *time-constrained devices*, which is in fact a sub-problem of our original optimization problem. We can get the final solution by repeating the previous procedure recursively (Lines 6-11), which can be proved optimal by using the Cauchy inequality again for $\sum_{i\in\mathcal{C}} s_i = s_{tot}(R) - \sum_{i\in\mathcal{N}\setminus\mathcal{C}} s_i(\theta)$, where $\mathcal{N}\setminus\mathcal{C}$ denotes the set of clients whose s_i have been regulated to be equal to $s_i(\theta)$. Since we always round down s_i (line 8), we may still have some remaining resource budget. due to the round down operation in the previous steps. We then increase the batch size of the device in the decreasing order of $\frac{M_i D_i^2}{s_i(s_i+1)} = \frac{M_i D_i^2}{s_i} - \frac{M_i D_i^2}{s_i+1}$ one at a time until the total batch size of all devices equals $s_{tot}(R)$ or $C = \emptyset$ (Lines 14-17). Finally, we can find τ^* that yields the smallest error bound according to (11), by enumerating each feasible τ under which s^* is optimized using the above method (Line 18).

In addition, we further consider a practical scenario where the FL training is performed with powerful GPUs, and the batch size should have little effect on the computation time t_{ci} [46], which can be simplified to be a constant rather than s_i/p_i in the time constraint (8). In this case, algorithm 1 can be simplified by using the following corollary to find the optimal s_i^* and τ^*.

Corollary 1 (Optimal τ^* and s_i^* with powerful GPUs)**.** *Suppose that each client i incurs a constant t_{ci}, e.g., running on powerful GPUs, the optimal s_i^* and τ^* satisfy:*

$$s_i(\tau) = \left\lfloor \frac{s_{tot}(\tau)\sqrt{M_i}D_i}{\sum_{i\in\mathcal{N}} \sqrt{M_i} D_i} \right\rfloor,\ s_{tot}(\tau) = \frac{R-Kb}{a\tau}, \tag{19}$$

$$\tau_1 = \lfloor \hat{\tau} \rfloor,\ \tau_2 = \lceil \hat{\tau} \rceil,\ \frac{\partial f(\hat{\tau})}{\partial \tau} = 0, \tag{20}$$

$$\tau^* = \min\left\{ \arg\min_{\tau\in\{\tau_1,\tau_2\}} f(\tau), \min_{i\in\mathcal{N}} \left\{ \frac{\theta - K t_{ui}}{K t_{ci}} \right\} \right\}, \tag{21}$$

$$s_i^* = s_i(\tau^*), \tag{22}$$

where $h(\tau) = \frac{\delta}{\beta}\left((\eta\beta+1)^\tau - 1\right) - \eta\delta\tau$, $f(\tau) = q^{K\tau}G(0) + \frac{1-q^K}{1-q}\left(\frac{\beta\eta^2(1-q^\tau)}{2D^2(1-q)} \sum_{i\in\mathcal{N}} \frac{M_i D_i^2}{s_i(\tau)} + \rho h(\tau)\right)$, $G(0) = [F(\mathbf{w}(0)) - F^*]$ *and* $q = 1 - \eta c \mu$ *as defined in Theorem 1.*

Detailed proof of Corollary 1 is provided in Appendix B.

Implication I. If the time constraint is not the bottleneck (Line 8 of Algorithm 1), s_i^* is proportional to $\sqrt{M_i}D_i$. It is intuitive as devices with larger data sizes (D_i) have the potential to contribute more samples in each training iteration while more various data (with a larger M_i) needs a larger batch size to reduce the local variance of its computed gradients. **This result reveals that either using full-batch ($s_i = D_i$) training [3], [25] or a uniform mini-batch size as FL practitioners usually adopt can be ineffective under non-i.i.d. clients' data.**

Implication II. Other batch size assignment schemes (e.g., [5]), on the other hand, focus on eliminating straggler effects brought by the system heterogeneity. They choose clients' batch sizes according to their computational capacity in order to minimize the average waiting time. **However, this "no-straggler" strategy is sub-optimal when cost constraints are present, which are quite common in edge systems [37]. Using their strategy [5], devices with higher computation capacities but possibly a smaller $D_i\sqrt{M_i}$ of data always have bigger batch sizes, which could significantly undermine the model accuracy.** Our Algorithm 1 instead captures both the data heterogeneity $(D_i\sqrt{M_i})$ and system heterogeneity (mitigating the straggler effects), as well as navigating the trade-off between the completion time and resource consumption.

Algorithm 1: An exact offline algorithm to Co-Optimize batch sizes and the number of local updates for FL training (**CoOptFL**)

Input : $\mathbf{G}, M_i, D_i, K, \tau_{max}, a, b, R, \theta, p_i, t_{ui}, \forall i$
Output: $\tau^*, \mathbf{s}^* = [s_1, s_2, ..., s_N]$

1 **foreach** $\tau \in [1, \tau_{max}]$ **do**
2 Set $C = \mathcal{N}$, $s_{tot} = \frac{R-Kb}{a\tau}$, $s_r = s_{tot}$;
3 **foreach** *node* $i \in \mathcal{N}$ **do**
4 $s_i(\theta) = \lfloor p_i \left(\frac{\theta}{K\tau} - \frac{t_{ui}}{\tau}\right) \rfloor$
5 **repeat**
6 $flag = 0$;
7 **foreach** *node* $i \in C$ **do**
8 $s_i = \lfloor \frac{s_r\sqrt{M_i}D_i}{\sum_{i\in C}\sqrt{M_i}D_i} \rfloor$;
9 **if** $s_i \geq s_i(\theta)$ **then**
10 $s_i = s_i(\theta), s_r = s_r - s_i(\theta)$;
11 Remove node i from set C, $flag = 1$;
12 **until** $flag = 0$ *or* $C = \emptyset$;
13 **repeat**
14 Find $i' = \text{argmax}_{i\in C} \frac{D_i^2}{s_i(s_i+1)}$, $s_{i'} = s_{i'} + 1$;
15 **if** $s_{i'} = s_{i'}(\theta)$ **then**
16 Remove node i from set C;
17 **until** $\sum_{i\in\mathcal{N}} s_i = s_{tot}$ *or* $C = \emptyset$;
18 Find the optimum $(\tau^*, \mathbf{s}^*) = \text{argmin}_{(\tau,\mathbf{s})}\mathbf{G}$;
/* Offline: $\mathbf{G} \triangleq$ (11) Online: $\mathbf{G} \triangleq$ (30) */

6 Online Adaptive Control Algorithm

Section 5 provides optimal solutions of batch sizes and the number of local updates, but it does not consider how to adapt them online with potentially unknown parameters, such as the computation speed c_i, communication time t_{ui}

(*system dynamics*), and those associated with the model. Further, for the emerging applications of FL training under real-time data, e.g., video analytics [47], we identify that limited on-device storage and online data streams (*data dynamics*) need to be incorporated, especially for those performed on edge devices. For instance, the storage of smartphones can range from 128GB to 1TB, but the bit rates of data streams collected for running today's FL-supported video analytics tasks, can be as large as 3 gigabytes per minutes (1080p, 30fps, w/o compression). If the online training lasts longer, the edge storage may be used up [48] before the training ends. Moreover, it is not appropriate to use a static aggregation frequency and batch sizes solved from the offline optimization problem (see Section 5), given that the parameters related to the data and network dynamics are time-varying.

Therefore, in this section, we propose an adaptive algorithm to adjust $\mathbf{s}$ and τ at the beginning of each aggregation round, based on our online parameter estimation. It realizes a more practical edge FL training, supporting both conventional static heterogeneous local datasets and dynamic local data streams by considering fluctuating network characteristics (Section 6.2). We also integrate a simple but efficient data sampling method to address the potential data insufficiency due to the limited storage of edge devices (Section 6.3).

6.1 FL with Streaming data and Limited Storage

To adapt to the paradigm of federated learning on streaming data and limited on-device storage, we first slightly modify the definitions in traditional FL settings (see Section 3).

Similarly, we consider a parameter-server architecture, which consists of a set (defined as $\mathcal{N}$) of clients with N distributed edge devices and a centralized PS for global aggregation. Each device $i \in \mathcal{N}$ has a local data stream $\mathcal{D}_i^k$ representing all D_i^k data samples $\mathbf{x}_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ..., \mathbf{x}_{i,D_i^k}]$ that client i received from round 1 to round k, i.e. $\mathcal{D}_i^1 \subseteq \mathcal{D}_i^2 \subseteq ... \subseteq \mathcal{D}_i^k$, and the local loss function of device i and the global loss function at round k can be defined as:

$$F_i^k(\mathbf{w}) = \frac{1}{D_i^k} \sum_{j \in \mathcal{D}_i^k} f(\mathbf{w}, \mathbf{x}_{i,j}). \tag{23}$$

$$F^k(\mathbf{w}) = \sum_{i \in \mathcal{N}} \frac{D_i^k}{D_k} F_i^k(\mathbf{w}), \tag{24}$$

where D_k is defined as $D_k = \sum_{i \in \mathcal{N}} D_i^k$. Besides, each client can select training samples from her local data stream $\mathcal{D}_i^k$ and store them into a buffer $\mathcal{B}_i$ with a limited size. Then we define the batch loss function $F_{i,\mathcal{S}_i^k}^k(\mathbf{w})$ under a mini-batch for each end device i:

$$F_{i,\mathcal{S}_i^k}^k(\mathbf{w}) = \frac{1}{s_i^k} \sum_{j \in \mathcal{S}_i^k} f(\mathbf{w}, \mathbf{x}_{i,j}). \tag{25}$$

Unlike $\mathcal{S}_i$ in Eq.(3), $\mathcal{S}_i^k$ denotes a mini-batch randomly selected from the buffer $\mathcal{B}_i$ with dynamic and limited size of data samples of the client i rather than its local data stream $\mathcal{D}_i^k$ due to the limited on-device storage; B_i represents the maximum buffer size of $\mathcal{B}_i$ ($|\mathcal{B}_i| \leq B_i$) and s_i^k denotes the size of the mini-batch $\mathcal{S}_i^k$. The buffer $\mathcal{B}_i$ of each client i will be updated by sampling data samples from their local data stream in every communication round. With a learning rate $\eta > 0$, we stick with the conventional rules of local update and global aggregation at round k, assuming D_i^k is known to the corresponding client i:

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) - \eta g_i^k(\mathbf{w}_i(t-1)), \tag{26}$$

$$\mathbf{w}(t) = \frac{\sum_{i=1}^N D_i^k \mathbf{w}_i(t)}{D_k},\ t = \sum_{k'=1}^{k} \tau_{k'}, \forall k = \{1, ..., K\}, \tag{27}$$

where $g_i^k(\mathbf{w}_i(t-1)) \triangleq \nabla F_{i,\mathcal{S}_i^k}^k(\mathbf{w}_i(t-1))$, and τ_k represents the number of local updates in communication round k.

The ultimate goal is to train a global model $\mathbf{w}$ that minimizes the global loss function $F^K(\mathbf{w})$ at the final round K using data sampled from buffers $\mathcal{B}_i$ that have limited storage sizes and selectively store streaming data samples that incrementally arrive at the devices.

6.2 Marginal Error bound and Problem Formulation

Revisiting our offline optimization problem (7)–(10), the objective function derived in (11) with static parameters and decision variables is no longer suitable for our online setting. Therefore, we use a marginal upper bound, to quantify the gap between the optimum F^* and the expected global loss that will be improved **in aggregation round** k, formalized as $\mathbb{E}[F^k(\mathbf{w}^{(k)})] - F^*$. This performance metric is adopted to reflect the goal of making the best use of a limited buffer and adjusting our control variables $(\tau_k, \mathbf{s}_k)$ in order to be comparable with the optimal performance if having an unlimited size of buffer to store the entire updated dataset $\mathcal{D}_i^k, \forall i$. We derive the upper-bound of this gap in Lemma 1.

Lemma 1 (Marginal bound with heterogeneous batch size s_i on streaming data)**.** *Suppose the loss function satisfies Assumptions 1-5 and $F^* \geq 0$. For a fixed learning rate $0 \leq \eta \leq \frac{\mu}{\beta \mu_G^2}$, the expected error of the empirical loss after k global communication rounds with the number of updates τ_k and batch sizes s_i^k for round k, defined as $\mathbb{E}[F^k(\mathbf{w}^{(k)})] - F^*$, is at most*

$$\begin{aligned}\mathbb{E}[F^k(\mathbf{w}^{(k)})] - F^* \leq\ & q^{\tau_k} \mathbb{E}[F^{k-1}(\mathbf{w}^{(k-1)}) - F^* + \psi^k] \\ & + \frac{\beta \eta^2 (1 - q^{\tau_k})}{2 D_k^2 (1-q)} \sum_{i \in \mathcal{N}} \frac{M_i D_i^{k^2}}{s_i^k} + \rho h(\tau_k)^2,\end{aligned} \tag{28}$$

where $q = 1 - \eta c \mu$, $h(\tau_k) = \frac{\delta}{\beta}((\eta\beta + 1)^{\tau_k} - 1) - \eta\delta\tau_k$, $F(\mathbf{w}^{(k-1)}) \triangleq F(\mathbf{w}(\sum_{i=1}^{k-1} \tau_i))$, and $\psi^k = \mathbb{E}[F^k(\mathbf{w}^{(k-1)}) - F^{k-1}(\mathbf{w}^{(k-1)})]$.

Detailed proof is deferred to Appendix A.

Intuition of Lemma 1. Here, ψ^k in (28) quantifies the impact of the latest data samples in round k on the global model, which can statistically describe the freshness and heterogeneity of these latest receiving data. We point out that Lemma 1 can easily adapt to static data set by setting $\psi^k = 0$ and remove all the superscripts k and $k-1$ from $F(\cdot)$, D, and s. Compared to Theorem 1, Lemma 1 is defined for the setting, where the training data and network characteristics are time-varying. The lemma can well leverage the latest parameters collected from participated FL clients,

Algorithm 2: DYNAMITE (Procedure at the PS)

Input : $\theta, R, K, \tau_{max}, \eta$
Output: $\mathbf{w}(t)$
Initialize: $\theta_c \leftarrow \theta, R_c \leftarrow R, t \leftarrow 0, k \leftarrow 0$
$\tau_1 \leftarrow 1, \mathbf{w}(0), \mathbf{s_1} = [s_1^1, s_2^1, ..., s_N^1]$
Receive D_i^0, M_i from each device $i \in \mathcal{N}$;
$D_0 = \sum_{i=1}^N D_i^0$;
repeat
  $k_0 \leftarrow k, k \leftarrow k+1$;
  Send $\mathbf{w}(t), \tau_k, s_i^k, k$ to each node i;
  $t_0 \leftarrow t, t \leftarrow t + \tau_k$;
  Receive $\mathbf{w}_i(t), p_i, D_i^k$ from each device $i \in \mathcal{N}$;
  $D_k = \sum_{i=1}^N D_i^k$;
  Execute global update according to (27) ;
  if $t_0 > 0$ ***and*** $k < K$ **then**
    `// Parameters estimation`
    Receive from each device i:
    $\rho_i, \beta_i, c_i, M_i, F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t_0)), g_i^{k_0}(\mathbf{w}(t_0))$;
    Calculate $g(\mathbf{w}(t_0)) \leftarrow \frac{\sum_{i=1}^N D_i^{k_0} g_i^{k_0}(\mathbf{w}_i(t_0))}{D_{k_0}}$
    $\delta_i \leftarrow \left\| g_i^{k_0}(\mathbf{w}_i(t_0)) - g(\mathbf{w}(t_0)) \right\|$;
    Estimate $\rho \leftarrow \frac{\sum_{i=1}^N D_i^{k_0}\rho_i}{D_{k_0}}, \beta \leftarrow \frac{\sum_{i=1}^N D_i^{k_0}\beta_i}{D_{k_0}}$;
    Estimate $c \leftarrow \frac{\sum_{i=1}^N D_i^{k_0} c_i}{D_{k_0}}, \delta \leftarrow \frac{\sum_{i=1}^N D_i^{k_0}\delta_i}{D_{k_0}}$;
    Estimate remaining resources θ_c, R_c and communication time of each device t_{ui} ;
    Define function $\mathbf{G}$ to be (30);
    $\tau_{k+1}, \mathbf{s}_{k+1} =$ **CoOptFL**$(\mathbf{G}, M_i, D_i^k, K, \tau_{max}, a, b, R_c, \theta_c, p_i, t_{ui})$
  until $k < K$ ***or*** $\theta_c < 0$ ***or*** $R_c < 0$;
  Send $STOP$ flag to all devices;

Algorithm 3: DYNAMITE (Procedure at client i)

Initialize: $\mathcal{B}_i = \mathcal{D}_i^0$ $(B_i \geq D_i^0), t \leftarrow 0$
Estimate M_i based on $\mathcal{D}_i^0$;
Send the stream size D_i^0 and M_i to the server;
repeat
  Receive $\mathbf{w}(t), \tau_k, s_i^k, k$ from the server;
  $t_0 \leftarrow t$, $k_0 \leftarrow k-1$;
  if $t_0 > 0$ **then**
    $c_i \leftarrow \left\| \nabla F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t)) \right\|^2 / 2F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t))$;
    $\rho_i \leftarrow \left\| F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}_i(t)) - F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t)) \right\| / \left\| \mathbf{w}_i(t) - \mathbf{w}(t) \right\|^2$;
    $\beta_i \leftarrow \left\| \nabla F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}_i(t)) - \nabla F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t)) \right\| / \left\| \mathbf{w}_i(t) - \mathbf{w}(t) \right\|$
  $\mathbf{w}_i(t) \leftarrow \mathbf{w}(t)$;
  $[\mathcal{B}_i, D_i^k] = \mathbf{RS}(\mathcal{B}_i, B_i, D_i^{k_0})$;
  for $r = 1, 2, ..., \tau_k$ **do**
    $t \leftarrow t+1$;
    Execute local update according to (26);
  Record the average computation time t_{ci} and estimate the computing capacity by $p_i = s_i/t_{ci}$;
  Send $\mathbf{w}_i(t), p_i, D_i^k$ to the parameter server;
  if $k_0 > 0$ ***and*** $F_{i,\mathcal{S}_i^k}^k(\mathbf{w}_i(t_0)) - F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t_0)) > \epsilon$ **then**
    Re-evaluate M_i based on the current buffer $\mathcal{B}_i$;
  if $t_0 > 0$ **then**
    Send $\rho_i, \beta_i, c_i, M_i, F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t_0))$, and $g_i^{k_0}(\mathbf{w}(t_0))$ to the PS ;
until *$STOP$ flag is received*;

and thus obtain better estimates of the unknown model and system parameters in each new aggregation round.

Lemma 1 provides the upper-bound of the error $\mathbb{E}[F^k(\mathbf{w}^{(k)})] - F^*$ incurred until the round k. Based on this, our optimization problem (7)–(10) can then be adapted to the following to solve for τ_k and $\mathbf{s}_k = [s_1^k, s_2^k, ..., s_N^k]$ used for each aggregation round $k \in [K]$.

$$\begin{aligned} \underset{\mathbf{s}_k, \tau_k}{\textbf{Minimize}} \quad & \mathbb{E}[F^k(\mathbf{w}^{(k)})] - F^* \quad \text{(Approximated by (28))} \\ \textbf{S.t.} \quad & \max_{i \in \mathcal{N}} \sum_{k=1}^{K} (\tau_k s_i^k / p_i + t_{ui}) \leq \theta, \ \forall i \\ & \sum_{k=1}^{K} (a\tau_k \sum_{i \in \mathcal{N}} s_i^k + b) \leq R, s_i^k \leq B_i, \ \forall i \\ & \tau_k \in [\tau_{max}], \ \forall k \end{aligned} \tag{29}$$

To solve (29), the remaining work is to estimate the unknown parameters c_i, t_{ui}, and those in (28) on the fly, as elaborated in Section 6.3.

6.3 Online Parameter Estimation and Data Sampling

To simplify the problem (29), we first set $F^* = 0$ as it is impossible to accurately evaluate it for model training. We then approximate the first term in (28), i.e. $F^{k-1}(\mathbf{w}^{(k-1)}) - F^* + \psi^k = F^k(\mathbf{w}^{(k-1)}) = \frac{\sum_{i=1}^N D_i^k F_i^k(\mathbf{w}^{(k-1)})}{D_k} \approx \sum_{i=1}^N D_i^k F_{i,\mathcal{S}_i^{k-1}}^{k-1}(\mathbf{w}^{(k-1)})/D_k \triangleq \hat{F}^k(\mathbf{w}^{(k-1)})$, by replacing the local loss $F_i^k(\cdot)$ with the batch loss $F_{i,\mathcal{S}_i^{k-1}}^{k-1}(\cdot)$, since it is impossible to calculate the local loss F_i^k with $\mathcal{D}_i^k$ which have not been received at the end of the round $k-1$. Thus, we approximate $F_i^k(\cdot)$ by $F_i^{k-1}(\cdot)$, which uses $\mathcal{D}_i^{k-1}$ instead of $\mathcal{D}_i^k$, and we use $F_{i,\mathcal{S}_i^{k-1}}^{k-1}(\cdot)$ rather than $F_i^{k-1}(\cdot)$, since it can be quite time-consuming to calculate the exact value of $F_i^{k-1}(\cdot)$, especially when client i has a large number of data samples. In this way, we use $\hat{F}^k(\mathbf{w}^{(k-1)})$ to capture the model drift due to the dynamic data, as it represents how good the old parameters $\mathbf{w}^{(k-1)}$ perform at the new data, which in turn reflects how well the model can generalize.

The estimation of ρ, β, c and δ takes two steps. First, each client estimates these parameters ρ_i, β_i, c_i, and $g_i^k(\mathbf{w}(t))$ using the global model $\mathbf{w}(t)$ just received at the beginning of every round k before synchronizing their local model $\mathbf{w}_i(t)$ with the global model. Consider that the network characteristics such as t_{ui} and t_{ci} are random variables. Since the training can take a large number of iterations, e.g., 10^5, the estimates based on taking average of empirical measurements will be accurate, at least in probability con-

verging to their true expectations, according to the law of large numbers. One can also pick a good online estimation approach, such as OMD, FTRL, and bandits methods [49], which is not the focus of this work can thus omitted. Then the clients send these results back to the PS to calculate ρ, β, c and δ as a weighted average of ρ_i, β_i, c_i and δ_i (see lines 13–14 in Algorithm 2). Note that these parameter estimates do not expose extra information of clients' raw data beyond that exposed by sending the computed gradients. Finally our objective function (28), can be approximated by the following error bound:

$$q^{\tau_k}\hat{F}^k(\mathbf{w}^{(k-1)}) + \frac{\beta\eta^2(1-q^{\tau_k})}{2D_k^2(1-q)}\sum_{i\in\mathcal{N}}\frac{M_i {D_i^k}^2}{s_i^k} + \rho h(\tau_k)^2 \tag{30}$$

where $q = 1-\eta c\mu$, $h(\tau_k) = \frac{\delta}{\beta}\left((\eta\beta+1)^{\tau_k}-1\right) - \eta\delta\tau_k$, $\hat{F}^k(\mathbf{w}^{(k-1)}) = \sum_{i=1}^N D_i^k F_{i,\mathcal{S}_i^{k-1}}^{k-1}(\mathbf{w}(\sum_{i=1}^{k-1}\tau_i))/D_k$.

For μ and μ_G, when $g(\mathbf{w},\xi_t)$ is an unbiased estimate of $\nabla F(\mathbf{w})$, or $\nabla F_{i,\mathcal{S}_i^k}^k(\mathbf{w},\xi_t)$ is an unbiased estimate of $\nabla F_i^k(\mathbf{w})$, we have $\mu = \mu_G = 1$, which are easily satisfied in the static data set case by randomly selecting each sample over the complete data set D_i of each client i. However, this property can hardly hold in the streaming data case where each client i can only select data samples from its limited buffer $\mathcal{B}_i$, after selecting data from their local data stream $\mathcal{D}_i^k$ and storing them at every FL round k. Regarding selecting data from $\mathcal{D}_i^k$ to $\mathcal{B}_i$, there are some straightforward data sampling methods, e.g., random sampling, which uniformly at random discards data stored in the buffer and replaces it with the latest-coming data, and FIFO sampling, which tends to preserve the data coming later while discarding those coming earlier. Such strategies will, however, inevitably lead to a biased global model. Hence, we adopt reservoir sampling [50] in our online algorithm to ensure that every data can have the same possibility of being stored in the buffer and thus selected into the batch during the whole training process.

Algorithm 4: Reservoir Sampling (RS)

Input : $\mathcal{B}_i, B_i, D_i^{k_0}$
Output: $\mathcal{B}_i, D_i^k$
Initialize: $D_i^k = D_i^{k_0}$
repeat
  for *every new data* $\mathbf{x}$ *received at round* k **do**
    $D_i^k = D_i^k + 1$;
    if $D_i^k < B_i$ **then**
      Add $\mathbf{x}$ to the buffer $\mathcal{B}_i$;
    else
      Uniformly sample an integer i in $[1, D_i^k]$;
      if $i \leq B_i$ **then**
        Replace the i_{th} data in buffer $\mathcal{B}_i$ with $\mathbf{x}$;
      else
        Discard data $\mathbf{x}$;
until *the client performs local update steps.*;

We specifically note that these online parameter estimation based on the latest data stored in clients' buffers, especially for δ_i and δ, can also help deal with the potentially changing gradient divergence or non-i.i.d. degree (Assumption 5) across clients, showing the capability of our **DYNAMITE** of adapting with data distribution shift.

Many advanced data selection methods have also been proposed in prior works, such as loss-based sampling [51], [52], importance-based sampling [53], gradient-norm sampling Mercury [54], FedBalancer [55], and the latest online streaming data selection method ODE [48]. However, these methods either need to evaluate all the data samples or incur a high time complexity during the selection process, which are not suitable for streaming data. Although data selection is not the focus of this work, our integrated sampling method (shown in Algorithm 4) is easy to implement and nicely preserves the good property (unbiased estimate of $\nabla F_i^k(\mathbf{w})$) that our online algorithm requires. We show the adaptiveness of our online control algorithm combined with different data sampling methods in experiments (Section 7).

6.4 The workflow of our adaptive control algorithm

In this subsection, we present our Online Co-Optimization based FL algorithm, named **DYNAMITE**, for the PS (Algorithm 2) and clients (Algorithm 3) to solve our refined batch size and aggregation frequency co-optimization problem shown in (29).

Algorithm 2. When the FL training starts, the PS initializes the remaining allowed completion time θ_c to be the deadline θ, the remaining cost budget R_c to be the total budget R, the current time t to be zero, and the number of local updates per round τ to be one; the model weights are $\mathbf{w}(\mathbf{0})$ and batch sizes of all the clients ($\mathbf{s_1}$) are initialized to be the same. In each aggregation round k, the server sends the global model $\mathbf{w}(t)$, number of local updates τ_k, batch size s_i^k, and current round index k to the corresponding clients. Besides, it estimates the unknown parameters of the FL model (e.g., ρ, β, and c), network characteristics (e.g., θ_c, R_c, and t_{ui}), and data distribution (non-i.i.d. degree δ), shown in lines 12–15 of Algorithm 2). After collecting related information from all the clients, the server updates τ and $\mathbf{s}$ using **CoOptFL** (line 17 in Algorithm 2).

Algorithm 3. On the client side, each device i first estimates M_i through pre-run tests over its initial buffer $\mathcal{B}_i = \mathcal{D}_i^0$. Then it updates the buffer $\mathcal{B}_i$ by replacing the stored data with the new training data sampled from the local stream using reservoir sampling (Algorithm 4). Then, each client i performs local updates and uploads the local model $\mathbf{w}_i(t)$ along with the estimated $c_i, \rho_i, \beta_i, M_i, F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t_0))$, and $g_i^{k_0}(\mathbf{w}(t_0))$, i.e. $\nabla F_{i,\mathcal{S}_i^{k_0}}^{k_0}(\mathbf{w}(t_0))$, to the PS (lines 7–20 of Algorithm 3). We note that the client will re-evaluate the value of M_i when the increase of the batch loss exceeds a threshold ϵ (Line 18 of Algorithm 3), since it indicates that the local data distribution has changed significantly.

Finally, the server will perform aggregation step to update the global model and adopt our **CoOptFL** (Algorithm 1) with the estimated parameters to compute τ_{k+1} and $\mathbf{s}_{k+1} = [s_1^{k+1}, s_2^{k+1}, ..., s_N^{k+1}]$ for all clients in the next round using the remaining budget R_c and θ_c (line 17 in Algorithm 3). The key is to utilize the marginal error bound (30) instead of the cumulative error bound (11) when using our subroutine algorithm **CoOptFL**. It finally outputs the optimal

solution (τ^* and $\mathbf{s}^*$), which is the combination of ($\tau, \mathbf{s}$) that minimizes the value of (30) for the next aggregation round. Based on the adapted error bound shown in Lemma 1 , this online adaptive control algorithm **DYNAMITE** (Algorithm 2 and 3) also adapt to classic FL where clients have heterogeneous but static local data set $\mathcal{D}_i$ by removing the reservoir sampling process.

7 Experimental Validation

In this section, we validate our theories and proposed algorithms in three parts: 1) Offline optimal local update step τ and uniform batch size s; 2) Optimal batch size assignment in CoOptFL (Algorithm 1); 3) Online adaptive control algorithm DYNAMITE (Algorithms 2 and 3) presented in Section 6.4. For the online adaptive control algorithm, we conduct experiments on both static datasets and dynamic data streams to demonstrate the superiority of our proposed algorithm **DYNAMITE**.

7.1 Experiment setup

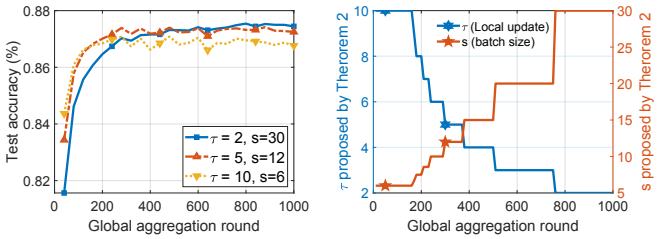
7.1.1 Testbed

To simulate the system heterogeneity, we first conduct our experiments in a small-scale testbed with various types of edge devices, including 1 laptop PC (CPU: Intel i5-7300HQ 4-core @2.50GHz), 1 desktop PC (CPU: Intel i5-1135G7 8-core @2.40GHz), and 3 docker containers [56] launched from a workstation. We manually assign different numbers of CPU cores (3, 6, 12) to each container. The PS instance is deployed on the container with the most CPU cores, while the rest of the containers and devices are used as clients.

To further evaluate our proposed algorithms **CoOptFL** and **DYNAMITE**, we conduct two larger scale experiments: 1) 100 clients simulated in a lab server cluster; and 2) a 20-client testbed deployed at 20 geo-distributed VM instances rented from Hetzner [57], including six 1-vCPU instance (2GB RAM, 20GB storage), seven 2-vCPU instances (4GB RAM, 40GB storage), and seven 4-vCPU instances (8GB RAM, 80GB storage) for reflecting computational heterogeneity among clients. We deploy our PS on one of the 4-core instances.

7.1.2 Models and datasets

We implement all the FL training models with Tensorflow [58]. We use MNIST [59], EMNIST [60] and CIFAR-10 datasets [61] to train a SVM model (loss function: $\frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{2}\max\{0; 1 - y_j \mathbf{w}^\top x_j\}^2, \lambda = 0.1$) and a 9-layer CNN model (two $5 \times 5 \times 32$ convolution layers, each followed by a 2×2 max pooling and a local response normalization layer, two fully connected layers ($z \times 256$, 256×10, where z = 1568 for MNIST and EMNIST and z = 2048 for CIFAR-10) and a softmax output layer with 10 units). To evaluate our online algorithm **DYNAMITE**, we first adopt non-i.i.d. data distribution settings proposed in [8] to simulate data heterogeneity among clients for static dataset case. We further conduct extensive experiments on streaming data by transforming the static local dataset into dynamic data stream, where we constantly distribute data samples to FL clients from complete dataset with different data stream configurations and data arrival patterns (Section 7.1.3) during the whole FL training process. To evaluate our (offline) algorithm **CoOptFL**, we initialize its input parameters (t_{ui}, M_i, ρ, c, β, δ, p_i, and t_{ci}) using the same estimation method as in our online algorithm **DYNAMITE**.



(a) Test Accuracy (b) τ, s proposed in Theorem 2

Fig. 2. Optimal τ and s (squared-SVM, MNIST)

7.1.3 Streaming Data configurations and arrival patterns

In this subsection, we introduce the design of our online streaming data configurations and data arrival patterns applied in our following experiments. We design two different data stream configurations (I.I.D. stream and Continuous stream) and three distinct data arrival pattern (Smooth, Burst, Random) to fully simulate the data dynamics in online FL training and demonstrate the adaptability of our online algorithm **DYNAMITE**.

Two streaming data configurations (in terms of feature class). **I.I.D. stream:** Clients will receive all classes of data samples in every interval (e.g. every 100 communication rounds; please see Table 2) and the number of each class of the data is the same. **Continuous stream:** Every client receives the same single class of data samples during the training process at any given time, and the chosen class will gradually change over time, i.e., once the training data of the currently chosen class is processed, a new class will be uniformly at random chosen from the set of classes that are not chosen before for the training task.

Three data arrival patterns (in terms of the number of data samples). **Smooth arrival:** Clients will receive the same number of data samples at each regular interval. **Burst arrival:** Each client will receive a massive amount of data samples in a specific round after the training starts, and few samples are received in other rounds. **Random-arrival:** The quantity and the arrival time of data samples of each client are uncertain; the arrival patterns are time-varying and heterogeneous across clients.

Parameters used in this section are clarified in Section 7.1.5 and shown in Table 2.

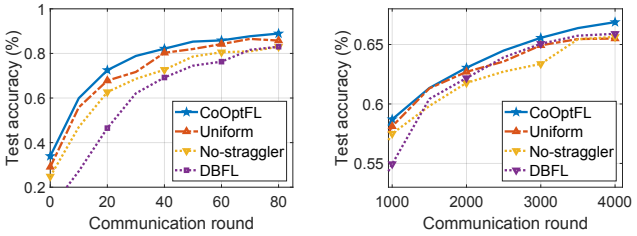
7.1.4 Baselines

To demonstrate the effectiveness of our carefully chosen batch size configurations for different clients using **CoOptFL**, we compare with **Uniform**, a widely-adopted method with uniform batch size [8] for all the clients, and with **No-straggler**, a time-efficient batch size strategy proposed by [5], and with **DBFL**, a dynamic batch size selection scheme proposed in [29].

To evaluate the performance of our online algorithm **DYNAMITE** under imperfect estimation of the model and

TABLE 2: Main parameter set-up. "Time (s)", "time-c", and "cost-c" represent time budget, time-constrained, and cost-constrained, respectively; Smooth-I (or Smooth-C) represents Smooth data arrival pattern in terms of the number of data samples per round and I.I.D. (or Continuous) configuration in terms of the arriving feature classes. Arrival rate: "5k 100r" denotes that 5000 samples will be received in every 100 rounds (Smooth) or in the 100th round (Burst).

Dataset		MNIST/EMNIST					CIFAR				
Configuration		Static	Smooth-I	Smooth-C	Burst	Random	Static	Smooth-I	Smooth-C	Burst	Random
Time (s)	time-c	1k	500	800	500	800	1k	800	1k	500	1k
	cost-c	5k	5k	5k	5k	5k	5k	5k	5k	5k	5k
Cost	time-c	80k	80k	80k	80k	80k	80k	80k	80k	80k	80k
	cost-c	40k	32k	32k	32k	32k	40k	48k	48k	48k	48k
Buffer Size	time-c	-	10k	10k	25k	10k	-	10k	10k	25k	10k
	cost-c	-	10k	10k	25k	10k	-	10k	10k	25k	10k
Step size	time-c	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3
	cost-c	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3
Arrival rate	time-c	-	5k 100r	5k 100r	50k 500r	-	-	3k 100r	3k 100r	25k 1000r	-
	cost-c	-	5k 100r	5k 100r	50k 500r	-	-	3k 100r	3k 100r	25k 1000r	-



(a) Accuracy (MNIST) 5-client (b) Accuracy (CIFAR) 20-client

Fig. 3. Our batch size assignment in offline algorithm **CoOptFL** achieves the highest accuracy for both datasets

system parameters, we compare with **FedAvg**, which maintains τ and batch size unchanged, an adaptive aggregation control algorithm **Dynamic-τ** proposed by [8], and the time-efficient **No-straggler** algorithm in [5].

To verify the impact of the data sampling methods in FL training with dynamic data stream, we also compare the performance of our adopted **Reservoir Sampling** (Algorithm 4) with other general sampling methods like **Random Sampling**, a straight-forward sampling strategy which uniformly at random discards a stored data sample and replace it with the latest one, and with **FIFO Sampling**, a classic data selection strategy to update clients' buffer simply following the "First-in First-out" principle.

7.1.5 *Parameters and run-time traces of the FL training.*

In our experiments, we initialize models with $\mathbf{w}(\mathbf{0})$ and set the default local update step $\tau = 2$, mini-batch size $s = 60$, and step size $\eta = 0.005$ unless otherwise specified. To evaluate the training cost and the completion time, we set the computation cost per sample $a = 0.0005$ and the communication cost per round $b = |\mathcal{N}|/10$. On each of our testbeds, the clients are training the same FL model, but they have different resource configurations and run-times. In particular, the run-time logs of the 5-client and 20-client experiments are real; but in the 100-client simulation, we sample the run-time of each client from the run-time trace collected at the 20 VM instances located in different edge clusters to simulate the real-world communication and computation overhead. Other important parameters used in the experiments are presented in Table 2.

We clarify that Smooth-I and Smooth-C denote I.I.D stream and Continuous stream configurations under smooth arrival pattern, respectively. The arrival rate denotes the number of samples that clients received in every 100 round. (Smooth arrival) or in a specific round (Burst arrival, EMNIST:Round 500, CIFAR: Round 1000).

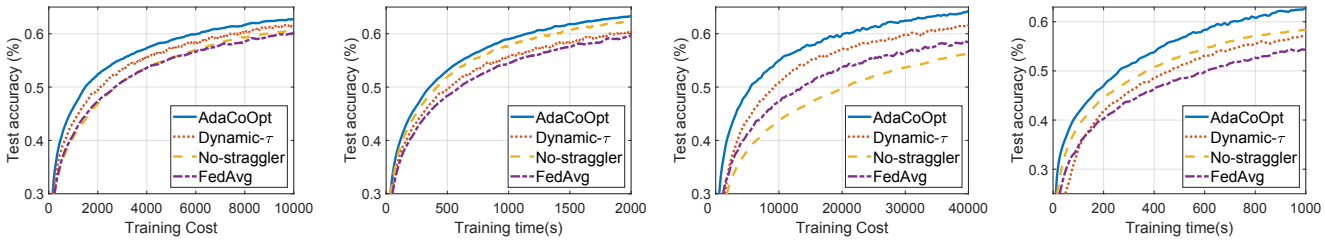
7.2 Experimental results and interpretation

7.2.1 Optimal number of local updates per round τ and uniform batch size s

We find the optimal combination τ^* and s^* using our Theorem 2 for a squared-SVM model training and testing under the MNIST dataset. We compare three different combinations: $\tau = 10, s = 6$; $\tau = 5, s = 12$; $\tau = 2, s = 30$. Fig. 2a shows the optimal τ and s combination varying K, e.g., $(\tau = 10, s = 6)$ for $K < 50$, $(\tau = 5, s = 12)$ for $K \approx 300$, and $(\tau = 2, s = 30)$ for $K > 800$ achieve the highest accuracy respectively. We also mark the optimal τ and batch size proposed in our Theorem 2 at the corresponding rounds in Fig. 2b for better visualization. Besides, Fig. 2b shows that the optimal τ decreases with the increase of K, supporting our theoretical result in Remark 1.

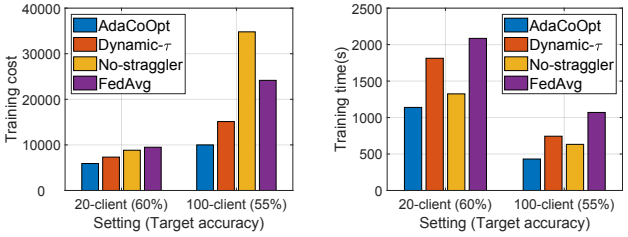
7.2.2 Optimal heterogeneous batch sizes across clients

We compare our offline algorithm **CoOptFL** to **No-straggler** [5], which configures batch sizes s_i according to clients' computing capacities ($s_i \propto p_i$) so as to eliminate the straggler effect across clients, **Uniform** ($s = 60$) and **DBFL** (Initial incremental factor $= 1.1$) . We set the communication round $K = 80$ and $K = 3000$ for the MNIST and CIFAR-10 datasets, respectively. For fairness, we set the total batch size $s_{tot} = \sum_{i \in \mathcal{N}} s_i$ as a constant to ensure that clients will process the same amount of data samples $K \cdot s_{tot}$ in total while using different batch size assignment strategies.



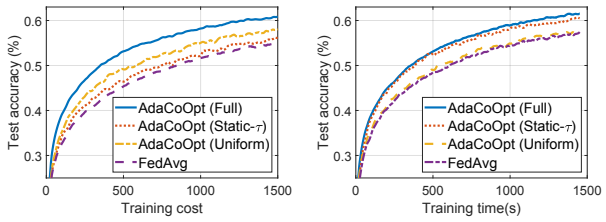
(a) Cost-dominant (20-client) $R = 10000, \theta = 5000s$ (b) Time-dominant (20-client) $R = 20000, \theta = 2000s$ (c) Cost-dominant (100-client) $R = 40000, \theta = 5000s$ (d) Time-dominant (100-client) $R = 80000, \theta = 1000s$

Fig. 4. Our online algorithm **DYNAMITE** achieves the highest accuracy under CIFAR-10 in both cost-sensitive and time-sensitive scenarios



(a) Cost-dominant scenario (b) Time-dominant scenario

Fig. 5. Our online algorithm **DYNAMITE** consumes minimal cost and time to achieve the target accuracy under CIFAR-10 in both cost-sensitive and time-sensitive scenarios



(a) Cost-dominant scenario (b) Time-dominant scenario

Fig. 6. Ablation Experiment under CIFAR-10

Fig. 3 shows that **CoOptFL** can converge faster and achieve better final testing accuracy compared to the baselines in both 5-client and 20-client settings. Note that **No-straggler** always tends to assign bigger batch sizes to devices with higher computing capacities regardless of their non-i.i.d. data properties, which leads to a lower model accuracy and resource utilization than **Uniform**, especially when devices with higher computing capacity have fewer and similar data samples (Theorem 3). Similar results can be found in Fig. 4c and Fig. 5a in the following online experiments as well. The slower convergence speed of **DBFL** can be attributed to the smaller initial batch size configurations.

7.2.3 Adaptive control for co-optimized aggregation frequency and heterogeneous batch sizes (static datasets)

We first further compare our **DYNAMITE** with three benchmarks for CIFAR-10 FL training using static dataset: the first two are vanilla FedAvg [3] and the time-efficient **No-straggler** [5]; the third one is **Dynamic-τ** [8] which dynamically adjusts τ_k for each round k.

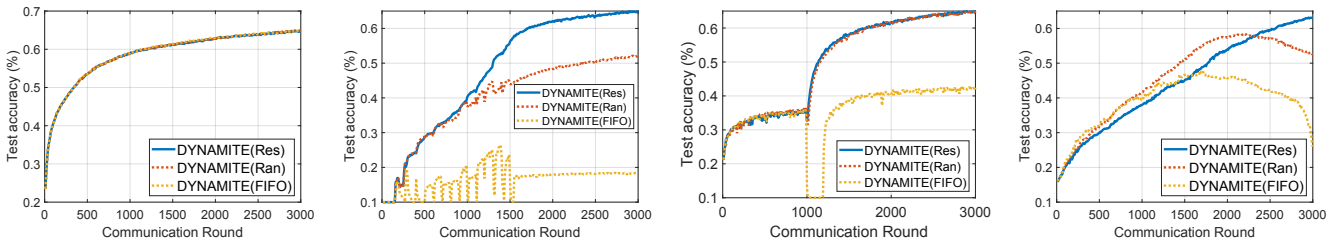
We compare the strategies in two different scenarios of our optimization problem, where the cost constraint and time constraint dominates, respectively. We set different values of R and θ to simulate these two different scenarios. Fig.4 and Fig.5 together show that **DYNAMITE** can outperform the baselines in both scenarios under different settings. For instance, in the cost-dominant scenario, **DYNAMITE** can achieve a 2.7%–7.9% higher final test accuracy than **FedAvg** and reduce the cost by 37.6%–58% when achieving the same accuracy. It achieves a 3.8%–8.4% higher final test accuracy and 45.4%–59.6% less completion time if achieving the same accuracy in the time-dominant scenario. These results indicate great adaptability of **DYNAMITE**.

Moreover, we conduct ablation experiments on both 20-client and 100-client settings to test the value of co-optimizing τ and s_i of our **DYNAMITE** in both cost-dominant and time-dominant scenarios. We compare our **DYNAMITE** with **DYNAMITE (Static-τ)**, which only optimizes the batch sizes using a fixed aggregation frequency and **DYNAMITE (Uniform)**, which uses a uniform batch size among clients, only adjusting the local update steps adaptively. Fig. 6a shows that a timely adjusted global aggregation frequency (**DYNAMITE (Uniform)**) can effectively reduce the training(communication) cost and thus is more critical in a cost-dominant training scenario. On the other hand, Fig. 6b shows that a careful batch size assignment (**DYNAMITE (Static-τ)**) can well capture the system and data heterogeneity so as to achieve a better model accuracy in a time-dominant scenario. These results also match our experiments in Fig. 4, where **Dynamic-τ** performs better than **No-straggler** in the cost-sensitive scenario, but worse than **No-straggler** in the time-sensitive scenario.

7.2.4 Adaptive control for co-optimized aggregation frequency and heterogeneous batch sizes (streaming data)

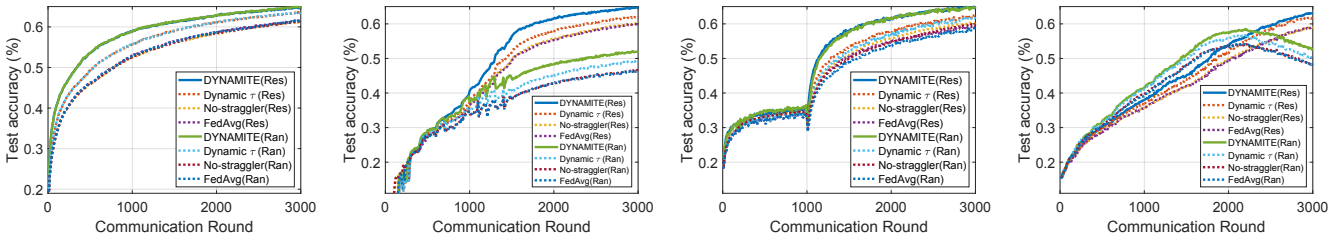
In this section, we further compare our **DYNAMITE** with three baselines using dynamic data streams with various data configurations presented in Section 7.1.3.

Comparison of sampling methods. Different from static datasets, data sampling strategies can be significant in FL training on dynamic data streams with limited on-device storage. Thus we first examine the impact of different data sampling strategies in different online settings. DYNAMITE(Res), DYNAMITE(Ran), and DYNAMITE(FIFO)



(a) Smooth-IID (b) Smooth-Continuous (c) Burst-Continuous (d) Random-Continuous

Fig. 7. Sampling methods under CIFAR-10 (DYNAMITE)

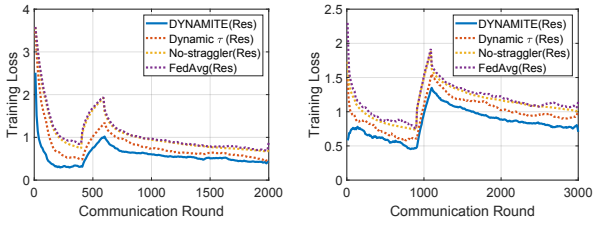


(a) Smooth-IID (b) Smooth-Continuous (c) Burst-Continuous (d) Random-Continuous

Fig. 8. Sampling methods under CIFAR-10 (Extensive)

TABLE 3: Test accuracy in online FL training (Reservoir Sampling)

Dataset		EMNIST				CIFAR			
Configuration		Smooth-I	Smooth-C	Burst	Random	Smooth-I	Smooth-C	Burst	Random
Cost Constrained	FedAvg	0.785	0.761	0.774	0.764	0.610	0.599	0.592	0.59
	No-straggler	0.785	0.763	0.780	0.768	0.618	0.601	0.603	0.593
	Dynamic τ	0.817	0.791	0.800	0.794	0.636	0.620	0.627	0.622
	DYNAMITE	**0.824**	**0.810**	**0.824**	**0.811**	**0.649**	**0.649**	**0.650**	**0.632**
Time Constrained	FedAvg	0.699	0.580	0.724	0.634	0.539	0.469	0.530	0.400
	No-strag	0.776	0.662	0.750	0.701	0.608	0.534	0.598	0.456
	Dynamic τ	0.734	0.556	0.735	0.641	0.557	0.444	0.552	0.410
	DYNAMITE	**0.787**	**0.793**	**0.817**	**0.798**	**0.618**	**0.629**	**0.631**	**0.572**

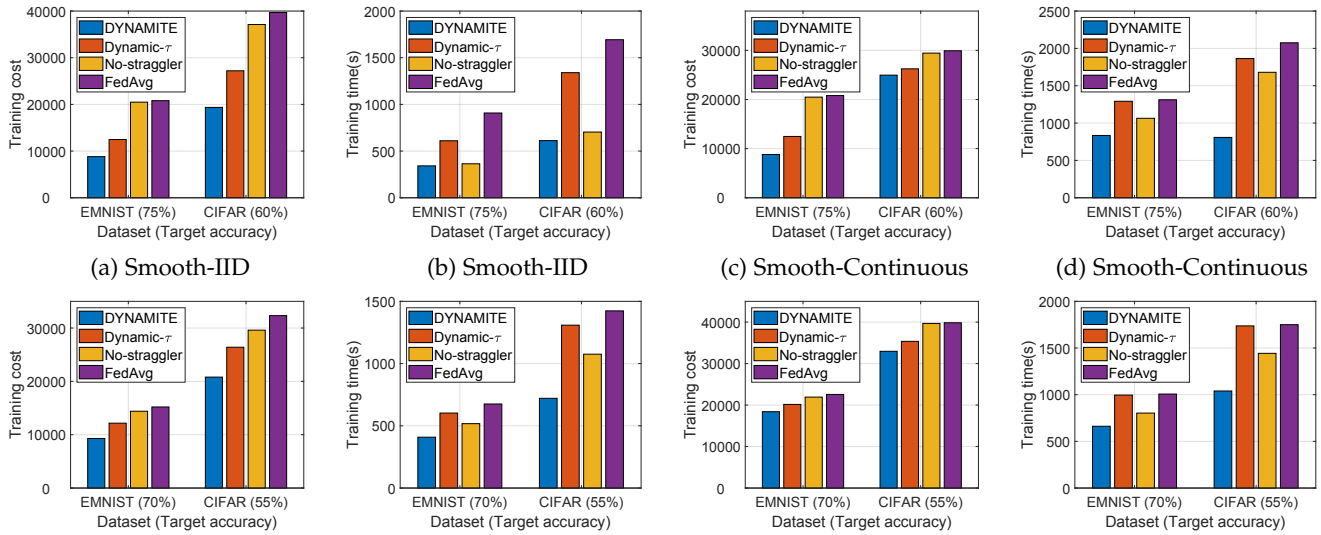


(a) EMNIST (b) CIFAR-10

Fig. 9. Training loss (Burst arrival)

are strategies of using DYNAMITE combined with sampling methods of Reservoir Sampling, Random Sampling, and FIFO, respectively. Fig.7 shows that DYNAMITE(Res) which uses **Reservoir Sampling** has significant advantages over other sampling strategies, especially in the **Continuous stream** settings, i.e., Smooth-Continuous, Burst-Continuous, Random-Continuous. It can be explained as follows. Both **Random Sampling** and **FIFO sampling** methods prefer to select data arrived later and discard data arrived earlier, which can easily lead to a biased model training and thus a biased global FL model. Fig. 7b and Fig. 7d together show that biased sampling can not only lead to poor model performance but also catastrophic forgetting when training on a continuous data stream. On the other hand, if the data stream is i.i.d. (Fig 7a), the performance difference among various sampling methods can be negligible, since the classes and the number of data samples that clients receive is always nearly the same, and thus how to select data is not of vital importance in this case.

We also notice that unlike other arrival patterns in continuous stream setting, DYNAMITE(Ran) and DYNAMITE(Res) can have similar performance in burst arrival setting (Fig. 7c). We here present a reasonable interpretation to explain this result. First, **Random Sampling** and **Reservoir Sampling** are two similar methods in general. The major difference is that reservoir sampling guarantees that every data sample can have the same probability to be stored in the buffer by uniformly sampling both an up-coming and a previously stored data point. **Random Sampling**, on the other hand, only discards data stored in the buffer uniformly at random, and it replaces a discarded data point with the latest-coming data. It then can inevitably result



(a) Smooth-IID (b) Smooth-IID (c) Smooth-Continuous (d) Smooth-Continuous

(e) Burst-Continuous (f) Burst-Continuous (g) Random-Continuous (h) Random-Continuous

Fig. 10. Training cost and time under EMNIST and CIFAR-10 (Reservoir Sampling)

in biases, since the distribution of the selected data batch does not represent the full data stream. Consequently, model performance after every buffer update step is impeded, as presented in Fig. 7b and 7d. However, clients receive data samples in a short period of time in **Burst arrival** setting. So the clients have less frequent buffer updates compared to **Smooth arrival** or **Random arrival**, which eventually close the gap of these two similar sampling methods (Fig. 7c).

Full comparison of FL baselines and sampling methods. In addition, we conduct extensive experiments on FL baselines (**Dynamic τ, No-straggler and FedAvg**) combined with **Reservoir Sampling** and **Random Sampling** in Fig. 8 under both EMNIST and CIFAR-10 datasets, revealing the importance and advantages of reservoir sampling.

Comparison in accuracy, cost, and run-time. Moreover, we evaluate these control algorithms under different online stream settings presented in Section 7.1.3. Similar to the static dataset case, we compare these algorithms in two different scenarios, where the cost constraint and time constraint dominates respectively. Table 3 and Figs. 8-10 show that **DYNAMITE** can still outperform the baselines in both scenarios under different online data stream settings. **DYNAMITE** can achieve a 3.9%–5.8% higher final accuracy than FedAvg while reducing 16.7%–51.2% training cost in cost-dominant scenario, and a 7.9%–21.3% higher final accuracy with 39.4%–63.8% less completion time to achieve the same accuracy in time-dominant scenarios. Specifically, Fig. 9 shows that **DYNAMITE** can have a smoother training process and faster reboot in the burst scenario where a large number of training samples are fed into the clients suddenly. Moreover, Fig. 8 also reveals that our **DYNAMITE** still outperforms the baselines in different online data stream setting, either using **Random Sampling** or **Reservoir Sampling**, showing great adaptability of our **DYNAMITE**.

8 Conclusion

This work proposes a novel framework to quantify and optimize the interplay of the number of local update steps and heterogeneous batch sizes across clients for federated learning performed at distributed edge devices. Technically, we derive a novel convergence bound with respect to those control variables, and analyze the performance metrics of cost and training time as well. We then provide closed-form solutions for our joint optimization in special cases and propose an efficient exact algorithm for the general case. Our strategies consider both heterogeneous system characteristics and non-i.i.d. data, which can improve the common strategies that FL practitioners adopt. Moreover, we adapt our offline strategy to dynamically adjust the decisions on the fly, with superiority of several performances demonstrated in extensive experiments.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of Artificial intelligence and statistics*, 2017.

[2] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[3] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proc. of International Conference on Learning Representations*, 2019.

[4] J. Liu, H. Xu, L. Wang, Y. Xu, C. Qian, J. Huang, and H. Huang, "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, early access, 2021.

[5] Z. Ma, Y. Xu, H. Xu, Z. Meng, L. Huang, and Y. Xue, "Adaptive batch size for federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, early access, 2021.

[6] X. Zhang, J. Wang, G. Joshi, and C. Joe-Wong, "Machine learning on volatile instances," in *Proc. of IEEE INFOCOM*, 2020.

[7] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia:{Geo-Distributed} machine learning approaching {LAN} speeds," in *Proc. of USENIX NSDI*, 2017.

[8] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[9] X. Mo and J. Xu, "Energy-efficient federated edge learning with joint communication and computation design," *Journal of Communications and Information Networks*, vol. 6, no. 2, pp. 110–124, 2021.

[10] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient resource management for federated edge learning with cpu-gpu heterogeneous computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7947–7962, 2021.

[11] Y. Ruan, X. Zhang, and C. Joe-Wong, "How valuable is your data? optimizing client recruitment in federated learning," in *Proc. of WiOpt*, 2021.

[12] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. of USENIX OSDI*, 2021.

[13] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," *arXiv preprint arXiv:2112.11256*, 2021.

[14] S. Tyagi and P. Sharma, "Taming resource heterogeneity in distributed ml training with dynamic batching," in *Proc. of 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 2020.

[15] Z. Charles, Z. Garrett, Z. Huo, S. Shmulyian, and V. Smith, "On large-cohort training for federated learning," in *Proc. of NeurIPS*, 2021.

[16] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[17] L. Zhu, H. Lin, Y. Lu, Y. Lin, and S. Han, "Delayed gradient averaging: Tolerate the communication latency for federated learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. of AISTATS*, 2020.

[19] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[20] L. Wang, W. Wang, and B. LI, "Cmfl: Mitigating communication overhead for federated learning," in *Proc. of IEEE ICDCS*, 2019.

[21] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[22] S. Liu, G. Yu, R. Yin, J. Yuan, and F. Qu, "Adaptive batchsize selection and gradient compression for wireless federated learning," in *Proc. of IEEE GLOBECOM*, 2020.

[23] J. Zhang, S. Guo, Z. Qu, D. Zeng, Y. Zhan, Q. Liu, and R. A. Akerkar, "Adaptive federated learning on non-iid data with resource constraint," *IEEE Transactions on Computers*, 2021.

[24] W. Xia, T. Q. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7108–7123, 2020.

[25] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning," in *Proc. of AISTATS*, 2021.

[26] J. Cipar, Q. Ho, J. K. Kim, S. Lee, G. R. Ganger, G. Gibson, K. Keeton, and E. Xing, "Solving the straggler problem with bounded staleness," in *Proc. of 14th Workshop on Hot Topics in Operating Systems (HotOS XIV)*, 2013.

[27] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "Fedsa: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3654–3672, 2021.

[28] J. Park, D. Yoon, S. Yeo, and S. Oh, "Amble: Adjusting mini-batch and local epoch for federated learning with heterogeneous devices," *Journal of Parallel and Distributed Computing*, vol. 170, pp. 13–23, 2022.

[29] D. Shi, L. Li, M. Wu, M. Shu, R. Yu, M. Pan, and Z. Han, "To talk or to work: Dynamic batch sizes assisted time efficient federated learning over future mobile edge devices," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 11 038–11 050, 2022.

[30] H. Jiang, X. Zhang, and C. Joe-Wong, "Doll: Distributed online learning using preemptible cloud instances," *ACM SIGMETRICS Performance Evaluation Review*, vol. 50, no. 2, pp. 21–23, 2022.

[31] Z. Zhou, S. Yang, L. Pu, and S. Yu, "Cefl: Online admission control, data scheduling, and accuracy tuning for cost-efficient federated learning across edge nodes," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9341–9356, 2020.

[32] B. Veloso, J. Gama, and B. Malheiro, "Self hyper-parameter tuning for data streams," in *International Conference on Discovery Science*. Springer, 2018, pp. 241–255.

[33] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," *Advances in neural information processing systems*, vol. 27, 2014.

[34] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 491–506, 2019.

[35] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *Proc. of IEEE INFOCOM*, 2020.

[36] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *Proc. of IEEE ICC*, 2020.

[37] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *Proc. of IEEE INFOCOM*, 2021.

[38] J. Yuan, M. Xu, X. Ma, A. Zhou, X. Liu, and S. Wang, "Hierarchical federated learning through lan-wan orchestration," *arXiv preprint arXiv:2010.11612*, 2020.

[39] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[40] C. Li, D. Y. Li, G. Miklau, and D. Suciu, "A theory of pricing private data," *Communications of the ACM*, vol. 60, no. 12, 2017.

[41] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. of NeurIPS*, 2020.

[42] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[43] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[44] H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the polyak-lojasiewicz condition," 2020. [Online]. Available: https://arxiv.org/pdf/1608.04636.pdf

[45] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022. [Online]. Available: https://www.gurobi.com

[46] K. Cremanns and D. Roos, "Deep gaussian covariance network," *arXiv preprint arXiv:1710.06202*, 2017.

[47] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "Fedvision: An online visual object detection platform powered by federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 08, 2020, pp. 13 172–13 179.

[48] 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," *arXiv preprint arXiv:2209.00195*, 2022.

[49] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[50] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[51] I. Loshchilov and F. Hutter, "Online batch selection for faster training of neural networks," *arXiv preprint arXiv:1511.06343*, 2015.

[52] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.

[53] A. Li, L. Zhang, J. Tan, Y. Qin, J. Wang, and X.-Y. Li, "Sample-level data selection for federated learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[54] X. Zeng, M. Yan, and M. Zhang, "Mercury: Efficient on-device distributed dnn training via stochastic importance sampling," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, pp. 29–41.

[55] J. Shin, Y. Li, Y. Liu, and S.-J. Lee, "Fedbalancer: Data and pace control for efficient federated learning on heterogeneous clients," in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, 2022, p. 436–449.

[56] D. Merkel *et al.*, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.
[57] Hetzner Online GmbH. [Online]. Available: https://www.hetzner.com/cloud
[58] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: A system for {Large-Scale} machine learning," in *Proc. of USENIX OSDI 16*, 2016.
[59] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[60] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.
[61] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

Weijie Liu (Student Member, IEEE) received the B.E. degree in electronics and communication engineering from the Sun Yat-sen University in 2021. He is currently working toward the M.E. degree at the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. His research interests include federated learning and edge computing.

Xiaoxi Zhang (Member, IEEE) received the B.E. degree in electronics and information engineering from the Huazhong University of Science and Technology in 2013 and the Ph.D. degree in computer science from The University of Hong Kong in 2017. She is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-sen University. Before joining SYSU, she was a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, Carnegie Mellon University. She is broadly interested in optimization and algorithm design for networked systems, including cloud and edge computing networks, NFV systems, and distributed machine learning systems.

Jingpu Duan (Member, IEEE) received the B.E. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2013, and the Ph.D. degree from the University of Hong Kong, Hong Kong, China, in 2018. He is currently a Research Assistant Professor with the Institute of Future Networks, Southern University of Science and Technology, Shenzhen, China. He also works with the Department of Communications, Pengcheng Laboratory, Shenzhen, China. His research interest includes designing and implementing high-performance networking systems.

Carlee Joe-Wong (Senior Member, IEEE) received the A.B. degree (magna cum laude) in mathematics, and the M.A. and Ph.D. degrees in applied and computational mathematics from Princeton University in 2011, 2013, and 2016, respectively. From 2013 to 2014, she was the Director of Advanced Research at DataMi, a startup she co-founded from her research on mobile data pricing. She is currently the Robert E. Doherty Assistant Professor of electrical and computer engineering with Carnegie Mellon University. Her research interests lie in optimizing various types of networked systems, including applications of machine learning and pricing to cloud computing, mobile/wireless networks, and ridesharing networks. She received the NSF CAREER Award in 2018 and the ARO Young Investigator Award in 2019.

Zhi Zhou (Member, IEEE) received the B.S., M.E., and Ph.D. degrees from the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2012, 2014, and 2017, respectively. He is currently an associate professor with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. In 2016, he has been a visiting scholar with the University of Gottingen. His research interests include edge computing and cloud computing and distributed systems.

Xu Chen (Senior Member, IEEE) received the Ph.D. degree in information engineering from the Chinese University of Hong Kong, Hong Kong, in 2012. He is currently a Full Professor with Sun Yat-sen University, Guangzhou, China, the Director of Institute of Advanced Networking and Computing Systems, and Vice Director of National and Local Joint Engineering Laboratory. From 2012 to 2014, he was a Postdoctoral Research Associate with Arizona State University, Tempe, AZ, USA, and from 2014 to 2016 a Humboldt Scholar Fellow with the Institute of Computer Science of the University of Goettingen, Germany. He was the recipient of the prestigious Humboldt Research Fellowship Awarded by the Alexander von Humboldt Foundation of Germany, 2014 Hong Kong Young Scientist Runner-up Award, 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, 2017 IEEE ComSoc Young Professional Best Paper Award, Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics, Best Paper Runner-up Award of 2014 IEEE International Conference on Computer Communications, and Best Paper Award of 2017 IEEE International Conference on Communications. He is currently the Area Editor the IEEE OPEN JOURNAL OF THE Communications Society, he is an Associate Editor for the IEEE Transactions on Wireless Communications, IEEE Transactions on Vehicular Technology, IEEE Internet of Things Journal, and IEEE Journal on Selected Areas in Communications Series on Network Softwarization and Enablers.

APPENDIX A
PROOF OF THEOREM 1 AND LEMMA 1

Lemma 2. *For any interval $[k]$, and $t \in [(k-1)\tau, k\tau)$,we have*

$$\|\mathbf{w}_i(t) - \mathbf{v}_{[k]}(t)\| \leq h_i(t-(k-1)\tau),$$

where $h_i(x) = \frac{\delta_i}{\beta}\left((\eta\beta+1)^x - 1\right)$.

Proof. When $t = (k-1)\tau$, we have $\mathbf{w}_i(t) = \mathbf{v}_{[k]}(t)$ by the definition of $\mathbf{v}_{[k]}(t)$. Therefore we have $\|\mathbf{w}_i(t) - \mathbf{v}_{[k]}(t)\| = h_i(0) = 0$. For the induction, we assume that

$$\|\mathbf{w}_i(t-1) - \mathbf{v}_{[k]}(t-1)\| \leq h_i(t-1-(k-1)\tau)$$

We now show that $\|\mathbf{w}_i(t) - \mathbf{v}_{[k]}(t)\| \leq h_i(t-(k-1)\tau)$ holds for t. We have

$$\begin{aligned}
&\left\|\mathbf{w}_i(t) - \mathbf{v}_{[k]}(t)\right\| \\
=&\| \left(\mathbf{w}_i(t-1) - \eta g_i\left(\mathbf{w}_i(t-1)\right)\right) \\
&- \left(\mathbf{v}_{[k]}(t-1) - \eta g\left(\mathbf{v}_{[k]}(t-1)\right)\right) \| \\
=&\| \left(\mathbf{w}_i(t-1) - \mathbf{v}_{[k]}(t-1)\right) - \eta\left[g_i\left(\mathbf{w}_i(t-1)\right)\right. \\
&\left. - g_i\left(\mathbf{v}_{[k]}(t-1)\right) + g_i\left(\mathbf{v}_{[k]}(t-1)\right) - g\left(\mathbf{v}_{[k]}(t-1)\right)\right] \| \\
\leq& \left\|\mathbf{w}_i(t-1) - \mathbf{v}_{[k]}(t-1)\right\| \\
&+ \eta\left\|g_i\left(\mathbf{w}_i(t-1)\right) - g_i\left(\mathbf{v}_{[k]}(t-1)\right)\right\| \\
&+ \eta\left\|g_i\left(\mathbf{v}_{[k]}(t-1)\right) - g\left(\mathbf{v}_{[k]}(t-1)\right)\right\| \\
\leq&(\eta\beta+1)\left\|\mathbf{w}_i(t-1) - \mathbf{v}_{[k]}(t-1)\right\| + \eta\delta_i \\
\leq&(\eta\beta+1)g_i(t-1-(k-1)\tau) + \eta\delta_i \\
=&(\eta\beta+1)\left(\frac{\delta_i}{\beta}\left((\eta\beta+1)^{t-1-(k-1)\tau} - 1\right)\right) + \eta\delta_i \\
=&\frac{\delta_i}{\beta}(\eta\beta+1)^{t-(k-1)\tau} - \frac{\delta_i}{\beta}(\eta\beta+1) + \eta\delta_i \\
=&\frac{\delta_i}{\beta}(\eta\beta+1)^{t-(k-1)\tau} - \frac{\delta_i}{\beta} \\
=&\frac{\delta_i}{\beta}\left((\eta\beta+1)^{t-(k-1)\tau} - 1\right) \\
=&h_i(t-(k-1)\tau).
\end{aligned}$$

Lemma 3. *For any interval $[k]$, and $t \in [(k-1)\tau, k\tau)$,we have*

$$\|\mathbf{w}(t) - \mathbf{v}_{[k]}(t)\| \leq h(t-(k-1)\tau),$$

where $h(x) = \frac{\delta}{\beta}\left((\eta\beta+1)^x - 1\right) - \eta\delta x$.

Proof.

$$\begin{aligned}
&\left\|\mathbf{w}(t) - \mathbf{v}_{[k]}(t)\right\| \\
&= \|\mathbf{w}(t-1) - \eta\frac{\sum_i D_i g_i\left(\mathbf{w}_i(t-1)\right)}{D} - \mathbf{v}_{[k]}(t-1) \\
&+ \eta g\left(\mathbf{v}_{[k]}(t-1)\right) \| \\
&= \|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1) \\
&- \eta\left(\frac{\sum_i D_i g_i\left(\mathbf{w}_i(t-1)\right)}{D} - g\left(\mathbf{v}_{[k]}(t-1)\right)\right) \| \\
&= \|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1) \\
&- \eta\left(\frac{\sum_i D_i\left(g_i\left(\mathbf{w}_i(t-1)\right) - g_i\left(\mathbf{v}_{[k]}(t-1)\right)\right)}{D}\right) \| \\
&\leq \left\|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1)\right\| \\
&+ \eta\left(\frac{\sum_i D_i\left\|g_i\left(\mathbf{w}_i(t-1)\right) - g_i\left(\mathbf{v}_{[k]}(t-1)\right)\right\|}{D}\right) \\
&\leq \left\|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1)\right\| \\
&+ \eta\beta\left(\frac{\sum_i D_i\left\|\mathbf{w}_i(t-1) - \mathbf{v}_{[k]}(t-1)\right\|}{D}\right) \\
&\leq \left\|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1)\right\| \\
&+ \eta\beta\frac{\sum_i D_i h_i(t-1-(k-1)\tau)}{D} \\
&= \left\|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1)\right\| \\
&+ \eta\beta\left(\frac{\sum_i D_i\frac{\delta_i}{\beta}\left((\eta\beta+1)^{t-1-(k-1)\tau} - 1\right)}{D}\right) \\
&= \left\|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1)\right\| \\
&+ \eta\left(\frac{\sum_i D_i\delta_i}{D}\right)\left((\eta\beta+1)^{t-1-(k-1)\tau} - 1\right) \\
&= \left\|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1)\right\| + \eta\delta\left((\eta\beta+1)^{t-1-(k-1)\tau} - 1\right).
\end{aligned}$$

Equivalently,

$$\begin{aligned}
&\left\|\mathbf{w}(t) - \mathbf{v}_{[k]}(t)\right\| - \left\|\mathbf{w}(t-1) - \mathbf{v}_{[k]}(t-1)\right\| \\
&\leq \eta\delta\left((\eta\beta+1)^{t-1-(k-1)\tau} - 1\right)
\end{aligned}$$

Summing up the above equation over different values of t, we have

$$\begin{aligned}
&\left\|\mathbf{w}(t) - \mathbf{v}_{[k]}(t)\right\| \\
&= \sum_{y=(k-1)\tau+1}^{t}\left\|\mathbf{w}(y) - \mathbf{v}_{[k]}(y)\right\| - \left\|\mathbf{w}(y-1) - \mathbf{v}_{[k]}(y-1)\right\| \\
&\leq \eta\delta\sum_{y=(k-1)\tau+1}^{t}\left((\eta\beta+1)^{y-1-(k-1)\tau} - 1\right) \\
&= \eta\delta\sum_{z=1}^{t-(k-1)\tau}\left((\eta\beta+1)^{z-1} - 1\right) \\
&= \eta\delta\sum_{z=1}^{t-(k-1)\tau}(\eta\beta+1)^{z-1} - \eta\delta(t-(k-1)\tau) \\
&= \eta\delta\frac{\left(1-(\eta\beta+1)^{t-(k-1)\tau}\right)}{-\eta\beta} - \eta\delta(t-(k-1)\tau) \\
&= \eta\delta\frac{(\eta\beta+1)^{t-(k-1)\tau} - 1}{\eta\beta} - \eta\delta(t-(k-1)\tau) \\
&= \frac{\delta}{\beta}\left((\eta\beta+1)^{t-(k-1)\tau} - 1\right) - \eta\delta(t-(k-1)\tau) \\
&= h(t-(k-1)\tau).
\end{aligned}$$

Using the ρ-quadratic-Lipschitz property of $F_i(x)$, we have

$$F(\mathbf{w}(t)) - F(\mathbf{v}_{[k]}(t)) \leq \rho h(t-(k-1)\tau)^2.$$

Lemma 4.

$$\mathbb{E}\left[\left\|g(\mathbf{v}_{[k]}(t), \xi_t)\right\|_2^2\right] \leq \frac{1}{D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i} + \mu_G^2\left\|\nabla F_i(\mathbf{w})\right\|^2.$$

Proof.

$$\because g_i(\mathbf{v}_{[k]}(t), \xi_t) = \sum_{\mathbf{x}_{i,j}\in\mathcal{S}_i}\frac{\nabla f(\mathbf{w}, \mathbf{x}_{i,j})}{s_i}$$

$$\therefore \mathbb{V}\left[g_i(\mathbf{v}_{[k]}(t), \xi_t)\right] \leq \frac{M_i}{s_i}$$

$$\because g(\mathbf{v}_{[k]}(t),\xi_t)=\sum_{i=1}^{N}\frac{D_i g_i(\mathbf{v}_{[k]}(t),\xi_t)}{D}$$
$$\therefore \mathbb{V}\left[g(\mathbf{v}_{[k]}(t),\xi_t)\right]\le\sum_{i\in[N]}\frac{D_i^2 M_i}{D^2 s_i}$$

Given the above, we have:

$$\begin{aligned}&\mathbb{E}\left[\left\|g(\mathbf{v}_{[k]}(t),\xi_t)\right\|_2^2\right]\\&=\left\|\mathbb{E}_{\xi_t}[g(\mathbf{v}_{[k]}(t),\xi_t)\right\|_2^2+\mathbb{V}\left[g(\mathbf{v}_{[k]}(t),\xi_t)\right]\\&\le\mu_G^2\left\|\nabla F_i(\mathbf{w})\right\|^2+\frac{1}{D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}.\end{aligned}$$

Lemma 5. *For any interval $[k]$, and $t\in[(k-1)\tau,k\tau)$, when $\eta\le\frac{\mu}{\beta M_G}$, $M_G=\mu_G^2$, we have*

$$\begin{aligned}&\mathbb{E}[F(\mathbf{v}_{[k]}(t+1))]-F^*\\&\le(1-\eta c\mu)\left(\mathbb{E}[F(\mathbf{v}_{[k]}(t))]-F^*\right)+\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}.\end{aligned}$$

Proof.

$$\begin{aligned}&F(\mathbf{v}_{[k]}(t+1))-F(\mathbf{v}_{[k]}(t))\\&\le\nabla F(\mathbf{v}_{[k]}(t))^T(\mathbf{v}_{[k]}(t+1)-\mathbf{v}_{[k]}(t))\\&+\frac{\beta}{2}\left\|\mathbf{v}_{[k]}(t+1)-\mathbf{v}_{[k]}(t)\right\|^2\\&=-\eta\nabla F(\mathbf{v}_{[k]}(t))^T g(\mathbf{v}_{[k]}(t),\xi_t)+\frac{\beta\eta^2}{2}\left\|g(\mathbf{v}_{[k]}(t),\xi_t)\right\|^2.\end{aligned}$$

Taking the expectation on the t_{th} sample process ξ_t and using the PŁ-condition: $2c(F(\mathbf{w})-F^*)\le\|\nabla F(\mathbf{w})\|_2^2$.

$$\begin{aligned}&\mathbb{E}_{\xi_t}[F(\mathbf{v}_{[k]}(t+1))]-F(\mathbf{v}_{[k]}(t)\\&\le-\eta\nabla F(\mathbf{v}_{[k]}(t))^T\mathbb{E}_{\xi_t}[g(\mathbf{v}_{[k]}(t),\xi_t)]\\&\quad+\frac{\beta\eta^2}{2}\mathbb{E}_{\xi_t}[\left\|g(\mathbf{v}_{[k]}(t),\xi_t)\right\|^2]\\&\le-\eta(\mu-\frac{\beta\eta M_G}{2})\left\|F(\mathbf{v}_{[k]}(t))\right\|^2+\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}\\&\le-\eta c\mu(F(\mathbf{v}_{[k]}(t))-F^*)+\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}.\end{aligned}$$

Subtract F^* from both sides, then taking total expectation.

$$\begin{aligned}\mathbb{E}[F(\mathbf{v}_{[k]}(t+1))]-F^*&\le(1-\eta c\mu)\left(\mathbb{E}[F(\mathbf{v}_{[k]}(t))]-F^*\right)\\&+\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}.\end{aligned}$$

Let $G_{[k]}(t)=\mathbb{E}[F(\mathbf{v}_{[k]}(t))]-F^*$, $q=1-\eta c\mu$, we have

$$G_{[k]}(t+1)\le qG_{[k]}(t)+\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}.$$

Apply the above inequality recursively over τ local updates.

$$G_{[k]}(k\tau)\le q^\tau G_{[k]}((k-1)\tau)+\frac{1-q^\tau}{1-q}\cdot\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}.$$

By the definition of $G_{[k]}(k\tau)$ and $F(\mathbf{v}_{[k]}(t+1))$, we have

$$\begin{aligned}&G_{[k+1]}(k\tau)-G_{[k]}(k\tau)=F(\mathbf{v}_{[k+1]}(k\tau))-F(\mathbf{v}_{[k]}(k\tau))\\&=F(\mathbf{w}(k\tau))-F(\mathbf{v}_{[k]}(k\tau))\\&\le\rho h(\tau)^2.\end{aligned}$$

Combining the equations above, we can have

$$\begin{aligned}&G_{[k+1]}(k\tau)\le G_{[k]}(k\tau)+\rho h(\tau)^2 &(31)\\&\le q^\tau G_{[k]}((k-1)\tau)+\frac{1-q^\tau}{1-q}\cdot\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}+\rho h(\tau)^2. &(32)\end{aligned}$$

Apply the above inequality recursively over all K communication rounds.

$$\begin{aligned}&\mathbb{E}[F(\mathbf{w}(K\tau))]-F^*\le q^{K\tau}\left(F(\mathbf{w}(0)-F^*\right)\\&+\frac{1-q^{K\tau}}{1-q^\tau}\left(\frac{1-q^\tau}{1-q}\cdot\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}+\rho h(\tau)^2\right).\end{aligned}$$

Due to the following inequality:

$$\frac{1-q^{K\tau}}{1-q^\tau}\le\frac{1-q^K}{1-q},$$

Theorem 1 is as follows:

$$\begin{aligned}&\mathbb{E}[F(\mathbf{w}(K\tau))]-F^*\le q^{K\tau}\left(F(\mathbf{w}(0)-F^*\right)\\&+\frac{1-q^K}{1-q}\left(\frac{1-q^\tau}{1-q}\cdot\frac{\beta\eta^2}{2D^2}\sum_{i\in[N]}\frac{M_i D_i^2}{s_i}+\rho h(\tau)^2\right),\end{aligned}$$

where we have:

$$h(\tau)\triangleq\frac{\delta}{\beta}\left((\eta\beta+1)^\tau-1\right)-\eta\delta\tau.$$

The above inequality (32) can derive a marginal error bound which adapts to dynamic local update steps τ_k, mini-batch size s_i^k and streaming data set $\mathcal{D}_i^k$ in every FL round k by rewriting (32) for a single FL round k as follows:

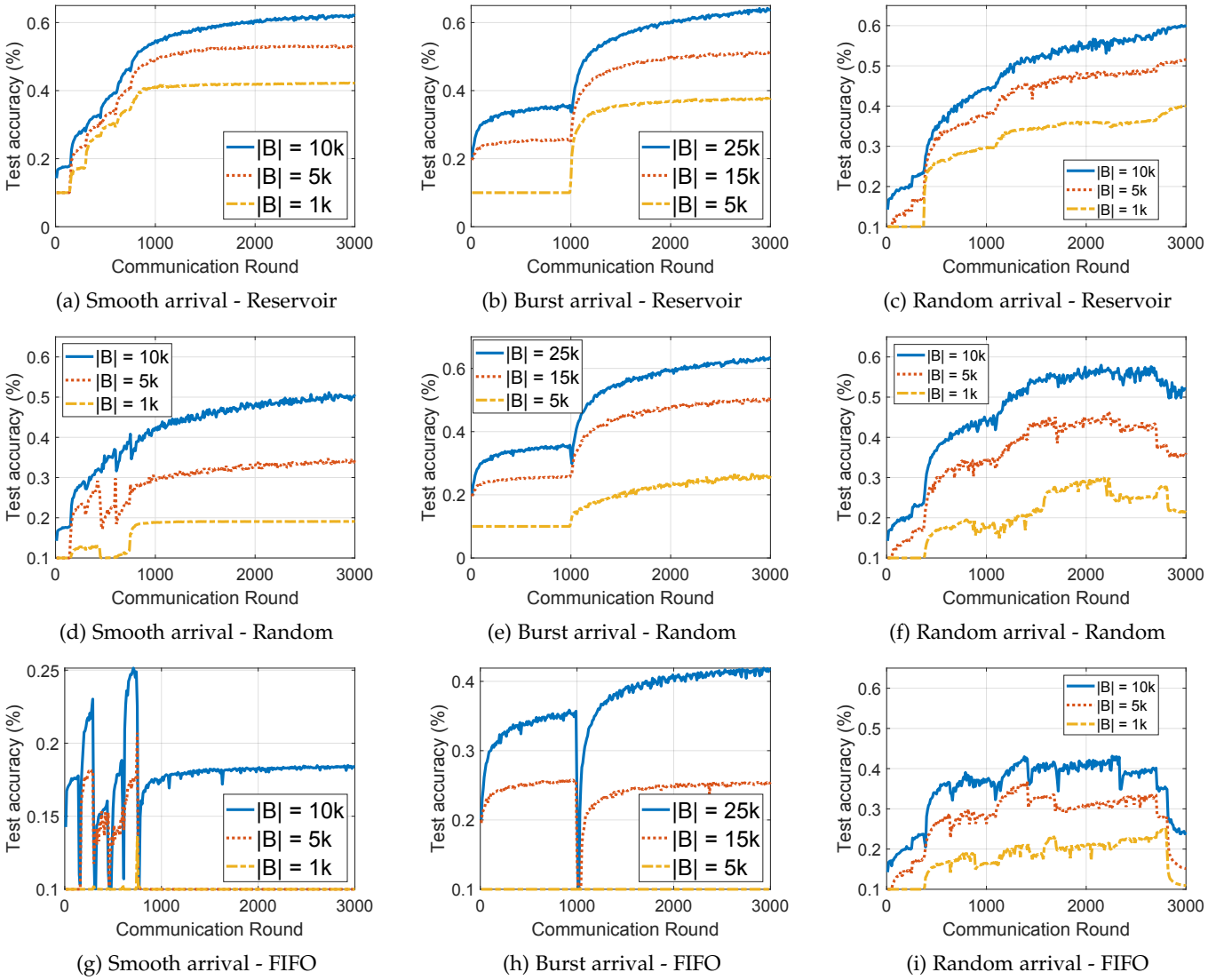
$$\begin{aligned}\mathbb{E}[F^k(\mathbf{w}^{(k)})-F^*]&\le q^{\tau_k}\left(\mathbb{E}[F^k(\mathbf{w}^{(k-1)})-F^*]\right)\\&+\frac{1-q^{\tau_k}}{1-q}\cdot\frac{\beta\eta^2}{2D_k^2}\sum_{i\in[N]}\frac{M_i^k D_i^{k^2}}{s_i^k}+\rho h(\tau_k)^2,\end{aligned}$$

where $\mathbf{w}^{(k)}\triangleq\mathbf{w}(\sum_{i=1}^k\tau_i)$. Then we define $\psi^k=\mathbb{E}[F^k(\mathbf{w}^{(k-1)})-F^{k-1}(\mathbf{w}^{(k-1)})]$ to quantify and describe the freshness and heterogeneity of the latest receiving data in round k and the Lemma 1 is as follows:

$$\begin{aligned}\mathbb{E}[F^k(\mathbf{w}^{(k)})]-F^*&\le q^{\tau_k}[\mathbb{E}[F^{k-1}(\mathbf{w}^{(k-1)})]-F^*+\psi^k]\\&+\frac{\beta\eta^2(1-q^{\tau_k})}{2D_k^2(1-q)}\sum_{i\in\mathcal{N}}\frac{M_i D_i^{k^2}}{s_i^k}+\rho h(\tau_k)^2.\end{aligned}\quad(33)$$

APPENDIX B
PROOF OF THEOREM 2 AND COROLLARY 1

We first present the proof of Theorem 2 with uniform batch-size. Given that the objective function is monotonically decreasing with the client batch size, we can obtain the optimal uniform batch size $s^*(\tau)$ by finding its maximum value under both completion time and training cost constraints. Then we can substitute $s^*(\tau)$ into the objective function, i.e.,



(a) Smooth arrival - Reservoir (b) Burst arrival - Reservoir (c) Random arrival - Reservoir

(d) Smooth arrival - Random (e) Burst arrival - Random (f) Random arrival - Random

(g) Smooth arrival - FIFO (h) Burst arrival - FIFO (i) Random arrival - FIFO

Fig. 11. The impact of buffer size $|B|$ on DYNAMITE under different sampling methods and arrival patterns

$f(\tau) = q^{K\tau}G(0) + \frac{1-q^K}{1-q}\left(\frac{\beta\eta^2(1-q^\tau)}{2D^2(1-q)}\sum_{i\in\mathcal{N}}\frac{M_iD_i^2}{s^*(\tau)} + \rho h(\tau)^2\right)$, which can be easily proved to be a convex function when $\tau < 2/log(1/q)$. Since the value of $q = 1-\eta c\mu$ is only slightly less than 1 , this interval of τ can be large enough. Thus, we can solve the $\hat{\tau}$ to minimize the expected error bound by letting the derivative of $f(\tau)$ to be zero. However, $\hat{\tau}$ can be fractional, so we need to compare the values of $f(\lfloor\hat{\tau}\rfloor)$ and $f(\lceil\hat{\tau}\rceil)$ to find the optimal τ^*. In Corollary 1, the optimal batch size s_i^* for each client i is determined by the Cauchy-Schwarz inequality and the total batch size $s_{tot}(\tau^*)$, where the optimal τ^* can be obtained following the same procedures as described in the proof of Theorem 2.

APPENDIX C
PROOF OF THEOREM 3 (ALGORITHM 1)

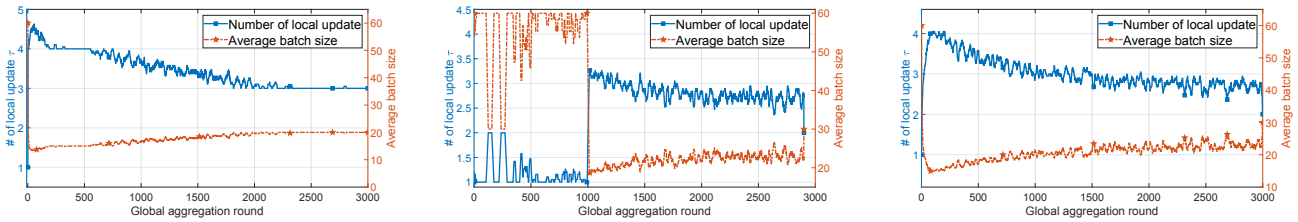
The optimality of the initial assignment based on Cauchy–Schwarz inequality has been proved in Section 5.2. Here we continue to prove that the batch size of the 'time-constrained' devices ($s_i \geq s_i(\theta)$) will always satisfy $s_i = s_i(\theta)$ in the optimal batch-size distribution.

Lemma 6. *Compared to a normal device j, a time-constrained device i satisfy:*

$$\frac{\sqrt{M_i}D_i}{s_i(\theta)} > \frac{\sqrt{M_j}D_j}{s_j} \tag{34}$$

$$\frac{M_iD_i^2}{s_i(\theta)} + \frac{M_jD_j^2}{s_j} < \frac{M_iD_i^2}{s_i(\theta)-1} + \frac{M_jD_j^2}{s_j+1} \tag{35}$$

Every device i will have the same value of $\frac{\sqrt{M_i}D_i}{s_i}$ after the initial batch-size distribution based on Cauchy inequality. But the 'time-constrained' devices have to reduce their batch-size s_i to $s_i(\theta)$ due to the limited time budget and yields Lemma 6. We can assume a batch-size distribution scheme $[s_1, ..., s_i(\theta), s_j, ..., s_N]$. According to (35), we can never find a better batch-size distribution scheme $[s_1, .., s_i(\theta)-1, s_j+1, ..., s_N]$ to obtain a smaller objective function, where device j can be any other normal devices. Therefore, the batch size of the 'time-constrained' devices ($s_i \geq s_i(\theta)$) will always satisfy $s_i = s_i(\theta)$ in the optimal batch-size distribution, and we can exclude these constrained devices from the following batch-size computa-



(a) Smooth arrival (b) Burst arrival (c) Random arrival

Fig. 12. The change in the number of local updates and average batch size under streaming datasets CIFAR-10

tion.

As for the final adjustment, the value of $\frac{M_i D_i^2}{s_i} - \frac{M_i D_i^2}{s_i+1}$ represents the reduction of the objective function if the batch-size of device i plus one, so we can still have the optimal solution if we increase the batch-size of the devices with the largest value of $\frac{M_i D_i^2}{s_i(s_i+1)}$ one at a time.

Appendix D
Error bound with client selection

Theorem 4 (Error bound with heterogeneous batch sizes s_i, local update τ and client selection)**.** *Suppose that the loss functions satisfy Assumptions 1-4 proposed in [3], and L, μ, σ_k, G defined therein. If N clients are selected randomly with replacement according to the sampling probabilities $p_1, ..., p_N$, given the initial global parameter $\mathbf{w}(0)$, the expected error after K aggregation rounds with τ local updates per round is:*

$$\mathbb{E}\left[F\left(\mathbf{w}(K\tau)\right)\right] - F^* \leq \frac{\kappa}{\gamma + K\tau - 1} \cdot \left(\frac{2(B+C)}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}\left\|\mathbf{w}(0) - \mathbf{w}^*\right\|^2\right),$$

where $\kappa = L/\mu, \gamma = \max\{8\kappa, \tau\}, \Gamma = F^* - \sum_{i=1}^N p_i F_i^*, B = \sum_{i=1}^N \frac{p_i^2 \sigma_i^2}{s_i} + 6L\Gamma + 8(\tau-1)^2 G^2, C = \frac{4}{K}\tau^2 G^2$.

Following the main proof of [3], this theorem can be easily derived by analyzing different variance reductions brought by the heterogeneous batch size (s_i in term B) across clients, using the same theoretical analysis as described in Lemma 4. This implies that we can incorporate the device selection mechanism into the DYNAMITE algorithm based on this theorem by adopting a similar workflow as presented in our DYNAMITE algorithm.

Appendix E
The impact of buffer size on DYNAMITE under different arrival patterns and sampling methods

Figure 11 shows that, under different arrival patterns and sampling methods, DYNAMITE always has a better model performance and faster convergence rate with a larger buffer. These experimental results are consistent with our theoretical expectations and intuitive understanding. The insight is that clients can store more diverse training samples with a larger buffer, leading to a more stable and better FL training process. Additionally, compared to smooth arrival and random arrival, DYNAMITE is more sensitive to the buffer size under burst arrival scenarios, where most data samples are received in a short period of time, which poses greater challenges to clients with limited buffers. We also observe that DYNAMITE often requires a larger buffer to reach the same test accuracy in burst arrival settings (Burst arrival: 25k, Smooth and Random arrival: 10k). Further, the reservoir sampling method adopted by DYNAMITE algorithm consistently yields superior model performance across all buffer size settings. In contrast, the other sampling methods (Random sampling and FIFO sampling) have demonstrated varying degrees of catastrophic forgetting, resulting in model degradation issues due to their biased selection principles.

Appendix F
The number of local updates and average batch sizes of DYNAMITE under streaming datasets

In Figure 12, we present the change in the number of local updates τ and average batch size $\sum_{i\in[N]} s_i/N$ during the FL training using **DYNAMITE** algorithm under streaming datasets. The new experiments are performed under three different data arrival patterns (explained in Section 7.1.3) to fully simulate the data dynamics in online training. In the case of smooth arrival and random arrival patterns of streaming datasets, the number of local updates demonstrates a general decrease while the average batch size displays a consistent increase during the training process. Compared to the smooth arrival pattern, both of these two variables exhibit more pronounced fluctuations under the random arrival pattern due to the higher degree of uncertainty in random arrival settings. On the other hand, we have observed that, under burst arrival settings, the number of local updates exhibits an abrupt increase in the 1000-th round, precisely aligning with the time when the participating clients receive an enormous amount of data and therefore have a significant increase in the model training loss. These results match our theoretical analysis in Theorem 1 and Remark 1, as well as the experimental results in static datasets shown in Figure 2b.