Asynchronous Federated Learning with non-convex client objective functions and heterogeneous dataset

Ali Forootani, Senior Member, IEEE, Raffaele Iervolino, Senior Member, IEEE.

Abstract-Federated Learning is a distributed machine learning paradigm that enables model training across decentralized devices holding local data, thereby preserving data privacy and reducing the need for centralization. Despite its advantages, traditional FL faces challenges such as communication overhead, system heterogeneity, and straggler effects. Asynchronous Federated Learning has emerged as a promising solution, allowing clients to send updates independently, which mitigates synchronization issues and enhances scalability. This paper extends the Asynchronous Federated Learning framework to scenarios involving clients with non-convex objective functions and heterogeneous dataset, which are prevalent in modern machine learning models like deep neural networks. We provide a rigorous convergence analysis for this setting, deriving bounds on the expected gradient norm and examining the impacts of staleness, variance, and heterogeneity. To address the challenges posed by asynchronous updates, we introduce a staleness-aware aggregation mechanism that penalizes outdated updates, ensuring fresher data has a more significant influence on the global model. Additionally, we propose a dynamic learning rate schedule that adapts to client staleness and heterogeneity, improving stability and convergence.

Our approach effectively manages heterogeneous environments, accommodating differences in client computational capabilities, data distributions, and communication delays, making it suitable for real-world Federated Learning applications. We also analyze the effects of client selection methods—specifically, choosing clients with or without replacement—on variance and model convergence, providing insights for more effective sampling strategies. The practical implementation of our methods using PyTorch and Python's asyncio library demonstrates their applicability in real-world asynchronous and heterogeneous FL scenarios. Empirical experiments validate the proposed methods, showing improved performance and scalability in handling asynchronous updates, and non-convex client's objective function with associated heterogeneous dataset.

Impact Statement—Asynchronous Federated Learning addresses key challenges in distributed learning, yet its application to non-convex optimization remains underexplored. This work extends Asynchronous Federated Learning to non-convex client objectives and heterogeneous datasets, providing a rigorous convergence analysis and quantifying the effects of staleness and variance. We introduce a staleness-aware aggregation mechanism and a dynamic learning rate schedule, improving stability and convergence. Empirical results on MNIST and CIFAR-10 demonstrate superior scalability and robustness over synchronous Federated Learning. Implemented using PyTorch and "asyncio", our approach enhances real-world applicability in privacy-sensitive domains like healthcare and finance. This work significantly advances Asynchronous Federated Learning's theoretical foundations and practical deployment, enabling more efficient, decentralized learning.

Ali Forootani is with Helmholtz Center for Environmental Research-UFZ, Permoserstrasse 15, 04318 Leipzig, Germany (aliforootani@ieee.org/ali.forootani@ufz.de).

Raffaele Iervolino is with Department of Electrical Engineering and Information Technology, University of Naples, 80125 Napoli, Italy (rafierv@unina.it).

Index Terms—Federated Learning, Stochastic Gradient Descent, Client Drifts, Asynchronous Federated Learning.

1

I. Introduction

Paradigm that enables the training of models across multiple decentralized devices or servers holding local data samples, without requiring the exchange of the data itself [1], [2]. This approach addresses privacy concerns and reduces the need for extensive data centralization [3]. FL has become increasingly relevant with the growing ubiquity of edge devices and the rising importance of data privacy regulations [4], [5]. Its applications span various domains, including healthcare [6], finance [7], renewable energies [8], and mobile computing [9].

Despite its potential, traditional FL faces significant challenges. These include communication overhead, system heterogeneity, and straggler effects [10], [11]. Communication overhead arises due to frequent exchanges of model parameters between clients and the central server. System heterogeneity reflects the diversity in computation power, network connectivity, and local data distribution across devices [12], [13]. The straggler effect occurs when slower clients delay the global model aggregation process, impeding efficiency [14], [15], [16].

Asynchronous Federated Learning (AFL) has emerged as a promising solution to mitigate these challenges [17]. Unlike synchronous FL, where all participating devices must complete their local training before model aggregation, AFL allows clients to send updates to the central server independently and at different times [18]. This flexibility reduces waiting times, improves scalability, and accommodates devices with varying computational capabilities. Furthermore, AFL inherently addresses the straggler problem by not requiring synchronization across all clients, leading to faster convergence and more efficient resource utilization [19].

A. Literature Review

Federated Learning has garnered significant attention since its inception. Federated Averaging (FedAvg) algorithm is introduced in [3], which became a cornerstone in FL research. They demonstrated the feasibility of FL in training deep learning models while preserving user privacy. However, subsequent studies highlighted its limitations. For example in [4] a secure aggregation techniques to enhance privacy in FL has been explored, while the work reported in [20] proposed adaptive optimizers to tackle the challenges posed by non-IID (non-Independent and Identically Distributed) data distributions across clients.

Addressing communication bottlenecks has been another focus of research. For example in [12] the sparse updates and compression techniques has been introduced to reduce the volume of data exchanged between clients and the server.

In another line of research the use of personalization layers in FL to accommodate diverse client data distributions without excessive communication was discussed [10].

The concept of asynchronous FL was first explored in [18], who proposed FedAsync, an AFL approach that allows updates from clients to be aggregated as they arrive. This method showed promise in mitigating the straggler problem and improving convergence speed. Subsequent works, such as [19], investigated the robustness of AFL under varying client participation rates and proposed strategies to balance stale updates.

While AFL addresses many challenges, it introduces new complexities, such as handling stale updates and ensuring fairness in model aggregation. Several works have advanced the field of asynchronous decentralized and parallel stochastic gradient descent (SGD), addressing challenges such as gradient staleness, convergence, and scalability. For example AD-PSGD was introduced in [21], [22], which removes the central server bottleneck while maintaining optimal convergence rates despite staleness. Adaptive frameworks like MindTheStep-AsyncPSGD adjusts step sizes to mitigate the impact of stale gradients [23], while instance-based Adaptiveness to Staleness in Asynchronous SGD has been considered to improve performance [24].

Theoretical analyses, such as those reported in [25], high-light error-runtime trade-offs and scenarios where stale gradients can be advantageous [25]. Techniques like delay compensation effectively address delays in asynchronous SGD [26], while the parallelization limits under gradient staleness has been explored in [27]. Acceleration methods have also been integrated into asynchronous SGD to enhance convergence rates [28]. Studies reported in [29], investigated the tolerable limits of unbounded delays in distributed optimization, while in [30], the properties of asynchronous SGD in nonconvex settings has been investigated. In [31], an asynchronous communication algorithm to address stale gradients in parallel updates was proposed [31].

To address the challenges posed by stragglers in synchronous federated learning (SFL) and improve its efficiency, several AFL schemes have been proposed [17], [32], [33], [34]. [35] introduced the Semi-Asynchronous Federated Learning Algorithm (SAFA), which classifies clients into three categories—sustainable, moderate, and unsustainable—allowing only sustainable clients to operate asynchronously, thereby reducing delays and enhancing system efficiency. In [36] a buffering mechanism has been proposed where the server aggregates local updates once a predefined buffer size is reached, balancing asynchronous updates with consistency in global model performance. Meanwhile, [37] developed a twostage training process that assigns each client a staleness parameter during the convergence phase, ensuring updates arrive synchronously despite varying processing speeds, thereby maintaining robust convergence in asynchronous settings. These strategies collectively mitigate the straggler problem while preserving the effectiveness and scalability of federated learning.

Our previous work [8], focused on AFL with convex assumption on local cost functions. In this work, we extend this approach to clients with non-convex objective functions. This extension is significant since non-convex optimization problems are prevalent in machine learning, especially in training deep neural networks. Recent literature has also explored AFL with non-convex objectives. For instance, in [38] an asynchronous federated optimization algorithm was proposed and its convergence for both strongly convex and a restricted family of non-convex problems have been proved. Another relevant work has studied improved convergence rates for asynchronous stochastic gradient descent in non-convex settings [39].

This body of literature underscores the evolution of federated learning from its synchronous roots to asynchronous implementations. AFL represents a crucial step forward in addressing the inherent limitations of traditional FL, particularly in scenarios involving diverse and large-scale distributed systems. However, further research is needed to refine its algorithms, enhance robustness, and validate its efficacy across diverse real-world applications.

B. Contributions

- This paper extends the theoretical and practical framework of asynchronous federated learning to non-convex objective functions, addressing the challenges of multiple local minima and saddle points common in modern machine learning models.
- We provide a rigorous convergence analysis for nonconvex asynchronous federated learning, deriving bounds on the expected gradient norm and analyzing the impact of key factors such as staleness, variance, and heterogeneity.
- We introduce and formalize the concept of drift due to staleness in asynchronous updates, quantifying its impact on the global model and providing mathematical bounds to mitigate its effects.
- We propose staleness-aware aggregation mechanism, penalizing updates based on their staleness to ensure that fresher updates contribute more significantly to the global model.
- We introduce a dynamic learning rate schedule that adapts to client staleness and heterogeneity, improving stability and convergence in asynchronous federated learning.
- Our approach effectively handles heterogeneous environments, addressing differences in client computation capabilities, data distributions, and communication delays, making it suitable for real-world federated learning applications.
- We analyze the effect of choosing clients with replacement vs. choosing clients without replacement on variance and model convergence, providing insights for more effective sampling in federated learning.

We emphasize a practical implementation using PyTorch and Python's asyncio library, demonstrating real-world

applicability for asynchronous and heterogeneous federated learning scenarios. We validate the proposed methods through *empirical experiments*, demonstrating improved performance and scalability in handling asynchronous updates and nonconvex objectives.

This paper is organized as follows: in Section II preliminaries are discussed. Problem formulation and an overview of the proposed AFL Algorithm is given in Section III. In section IV we will provide the convergence analysis of the AFL Algorithm with non-convex client objective function and heterogeneous dataset. We evaluate the AFL Algorithm and compare its results with synchronous FL on well known dataset such as MNIST and CIFAR-10 in Section V. The paper ends with Conclusion in Section VI.

II. PRELIMINARIES

In this section, we establish the necessary notation, definitions, and assumptions required for analyzing the AFL framework. We begin by introducing the mathematical notation used throughout this paper, followed by a discussion of different sampling methods and their statistical properties. Next, we present key theoretical results related to variance bounds and martingale properties, which form the foundation for our analysis. Finally, we introduce key assumptions that characterize the behavior of local objective functions, stochastic gradients, and model drift in the presence of stale updates.

A. Notation

Let $\|\cdot\|$ denote the standard Euclidean norm, applicable to both vectors and matrices. The parameter σ^2 provides an upper bound on the variance of the stochastic gradients for individual clients participating in the AFL process. The total number of clients is represented by C, with the index c referring to a specific client, and J indicating the subset of clients actively engaged in training during a particular round. Training proceeds over $\mathcal J$ global communication rounds, indexed by j, and each client performs I local update steps per round, indexed by i. The learning rate or step size is denoted by λ , and the effective learning rate in the AFL framework is defined as $\tilde{\lambda} = \lambda CI$.

A permutation of the client indices is represented by ψ , defined as $\{\psi_1,\psi_2,\ldots,\psi_C\}$, which maps the set $\{1,2,\ldots,C\}$ to a reordered sequence. The global objective function is denoted by \mathcal{L} , while the local objective function for client c is expressed as \mathcal{L}_c . The global model parameters at the j-th communication round are denoted by $\theta^{(j)}$, and the local model parameters of client c after i local update steps within the j-th round are represented by $\theta^{(j)}_{c,i}$. The stochastic gradients corresponding to the local objective function \mathcal{L}_{ψ_c} of client ψ_c , evaluated at $\theta^{(j)}_{c,i}$, are denoted by $\mathbf{q}^{(j)}_{\psi_c,i}$, where $\mathbf{q}^{(j)}_{\psi_c,i} \coloneqq \nabla l_{\psi_c}(\theta^{(j)}_{c,i};\xi)$, with ξ representing a stochastic sampling variable.

B. Reviews on some probability and stochastic concepts

We analyze the properties of the sample mean $\overline{\theta}_{\psi}$ under two common sampling approaches: (i) sampling with replacement

and (ii) sampling without replacement. First, the population mean $\bar{\theta}$ and variance ν^2 are defined as:

$$\overline{\theta} \coloneqq \frac{1}{m} \sum_{k=1}^{m} \theta_k, \quad \nu^2 \coloneqq \frac{1}{m} \sum_{k=1}^{m} \left\| \theta_k - \overline{\theta} \right\|^2,$$

where $\theta_1, \theta_2, \ldots, \theta_m$ represent the fixed vectors in the population. Given a sample of s vectors, denoted as $\mathbf{x}_{\psi_1}, \mathbf{x}_{\psi_2}, \ldots, \mathbf{x}_{\psi_s}$, drawn from this population, the sample mean is expressed as:

$$\overline{\theta}_{\psi} = \frac{1}{s} \sum_{p=1}^{s} \theta_{\psi_p}.$$

Below, we evaluate the expected value and variance of $\overline{\theta}_{\psi}$ under the two sampling methods:

a) Sampling with Replacement: (i) Expected Value: The expected value of the sample mean equals the population mean [40]:

$$\mathbb{E}[\overline{\mathbf{x}}_{\psi}] = \overline{\theta}.$$

(ii) Variance: The variance of the sample mean is given by:

$$\mathbb{E}\left[\left\|\overline{\mathbf{x}}_{\psi} - \overline{\theta}\right\|^{2}\right] = \frac{\nu^{2}}{s}.$$

This result shows that the variance decreases inversely with the sample size s but is independent of the total population size m.

b) Sampling without Replacement: (i) Expected Value: The expected value of the sample mean remains equal to the population mean [40]: $\mathbb{E}[\overline{\theta}_{\psi}] = \overline{\theta}$. (ii) Variance: The variance in this case is: $\mathbb{E}\left[\left\|\overline{\mathbf{x}}_{\psi} - \overline{\theta}\right\|^2\right] = \frac{m-s}{s(m-1)}\nu^2$. Here, the variance depends on both the sample size s and the total population size s, and it is typically smaller than the variance in the case of sampling with replacement.

In summary, while the expected value of the sample mean is identical for both sampling methods, the variance differs, with sampling without replacement usually leading to a smaller variance.

Consider a sequence of random variables $\{\epsilon_i\}_{i=1}^m$ and associated random vectors $\{\theta_i\}_{i=1}^m$, where each $\theta_i \in \mathbb{R}^d$ depends on the history $\epsilon_1, \epsilon_2, \ldots, \epsilon_i$. Assume the conditional expectation satisfies:

$$\mathbb{E}_{\epsilon_i}[\theta_i \mid \epsilon_1, \dots, \epsilon_{i-1}] = \mathbf{e}_i,$$

which implies that $\{\theta_i - \mathbf{e}_i\}_{i=1}^m$ forms a martingale difference sequence with respect to the filtration generated by $\{\epsilon_i\}_{i=1}^m$. Additionally, assume that the conditional variance of each $\theta_i - \mathbf{e}_i$ is uniformly bounded:

$$\mathbb{E}_{\epsilon_i}[\|\theta_i - \mathbf{e}_i\|^2 \mid \epsilon_1, \dots, \epsilon_{i-1}] \le \delta^2,$$

for some constant $\delta > 0$. Then, the total variance of the sum of the martingale difference sequence satisfies:

$$\mathbb{E}\left[\left\|\sum_{k=1}^{m}(\theta_k - \mathbf{e}_k)\right\|^2\right] = \sum_{k=1}^{m}\mathbb{E}\left[\|\theta_k - \mathbf{e}_k\|^2\right] \le m\delta^2. \quad (1)$$

This inequality demonstrates that the variance of the sum is bounded by $m\delta^2$, where m is the number of terms in the sequence and δ^2 is the uniform variance bound for each term.

Using the approach of sampling clients without replacement, define [20], [40]:

$$p_{c,i}(k) = \begin{cases} I - 1, & \text{if } k \le c - 1, \\ i - 1, & \text{if } k = c, \end{cases}$$

where I is a fixed integer. For $J \leq C$ and $C \geq 2$, the following inequality holds [20], [40]:

$$\sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathbb{E} \left\| \sum_{k=1}^{c} \sum_{j=0}^{p_{c,i}(k)} (\theta_{\psi_k} - \overline{\theta}) \right\|^2 \le \frac{1}{2} J^2 I^3 \nu^2,$$

where $\overline{\theta}$ is the population mean, and ν^2 is the population variance.

C. Definitions and main assumptions

In this subsection we provide definitions, and main assumptions that are required for the convergence proof of the AFL algorithm.

Definition 1. In AFL, each client updates its local model based on a potentially stale global model. This means that client updates might not align with the current global model, leading to a drift between the models at each client and the global model. Let's define this drift mathematically:

- At time t, the global model is $\theta^{(t)}$, and client i uses a stale model $\theta^{(t-\tau_i)}$ (with τ_i being the staleness of client i's model). - The drift for client i at time t can be expressed as:

$$\Delta \theta_i^{drift} = \theta^{(t)} - \theta^{(t-\tau_i)}.$$

a) Impact of Drift on Updates: This drift can affect the quality of local updates. The local gradient computed by client i is based on a stale model $\theta^{(t-\tau_i)}$, and the direction of the gradient will be influenced by the mismatch between $\theta^{(t)}$ and $\theta^{(t-\tau_i)}$.

Assumption 1. Each local objective $\mathcal{L}_i(\theta)$ is L-smooth:

$$\mathcal{L}_i(\theta_2) \le \mathcal{L}_i(\theta_1) + \langle \nabla \mathcal{L}_i(\theta_1), \theta_2 - \theta_1 \rangle + \frac{L}{2} \|\theta_2 - \theta_1\|^2.$$

Assumption 2. The variance of the stochastic gradients is bounded:

$$\mathbb{E}\left[\|\nabla \mathcal{L}_i(\theta) - \nabla \mathcal{L}(\theta)\|^2\right] \le \sigma^2.$$

Assumption 3. The staleness τ_i is bounded by τ_{max} :

$$\tau_i \leq \tau_{\max}$$
.

Assumption 4. The drift between the local and global models is bounded:

$$\|\Delta \theta_i^{drift}\| \leq \Delta \theta_{max},$$

where $\Delta\theta_{max}$ represents the maximum allowable drift between the global model and the local model due to staleness.

Assumption 5. Local updates are unbiased estimates of the true gradient:

$$\mathbb{E}[\nabla \mathcal{L}_i(\theta)] = \nabla \mathcal{L}_i(\theta).$$

Assumption 6. Clients are randomly selected at each round, and each client computes the update using the local model $\theta^{(t-\tau_i)}$.

Assumption 7. We assume that, at the global minimizer θ^* , the gradients of the individual local loss functions are closely aligned. Specifically, there exists a constant β^2_* such that:

$$\frac{1}{C} \sum_{c=1}^{C} \left\| \nabla \mathcal{L}_c(\theta^*) \right\|^2 = \beta_*^2,$$

where β_* measures the degree of similarity among the gradients at θ^* , the solution to the global loss $\mathcal{L}(\theta)$.

III. PROBLEM FORMULATION

The objective of the FL framework is to optimize a global loss function, defined as:

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathcal{L}(\theta) \coloneqq \frac{1}{C} \sum_{c=1}^{C} \mathcal{L}_c(\theta) \right\},\,$$

where $\mathcal{L}_c(\theta) \coloneqq \mathbb{E}_{\xi \sim \mathcal{D}_c}[\ell_c(\theta; \xi)]$ represents the local objective function for client c. Here, $\ell_c(\theta; \xi)$ denotes the client-specific loss function evaluated at model parameters θ for data point ξ , \mathcal{D}_c is the data distribution of client c, and C is the total number of clients.

If \mathcal{D}_c consists of a finite dataset, $\mathcal{D}_c = \{\xi_c^j \mid j = 1, 2, \dots, |\mathcal{D}_c|\}$, the local objective function can be expressed equivalently as:

$$\mathcal{L}_c(\theta) = \frac{1}{|\mathcal{D}_c|} \sum_{j=1}^{|\mathcal{D}_c|} \ell_c(\theta; \xi_c^j).$$

In federated learning, clients operate independently and often experience asynchronous delays in their updates. The delay for client c is denoted as τ_c , representing the time lag between the computation of the local gradient by the client and the incorporation of the gradient into the global model.

During each training round t, a subset of clients $J^{(t)} = \{c_1, c_2, \dots, c_k\} \subseteq \{1, 2, \dots, C\}$ is selected at random without replacement to participate in training. For each client $c \in J^{(t)}$, the process is as follows:

- 1) Model Initialization: The client initializes its local model parameters using the global parameters available at the time, $\theta^{(t-\tau_c)}$, where τ_c accounts for the update delay.
- 2) Local Training: The client performs I iterations of stochastic gradient descent (SGD) on its local dataset, producing intermediate parameter states $\theta_{c,i}^{(t)}$ for $i=0,1,\ldots,I$.
- 3) Parameter Transmission: After completing local updates, the client sends its final parameters, $\theta_{c,I}^{(t)}$, to the central server.

The local parameter updates at each iteration i are computed as:

$$\theta_{c,i+1}^{(t)} = \theta_{c,i}^{(t)} - \eta \mathbf{q}_{c,i}^{(j)},$$

where $\eta > 0$ is the learning rate, and $\mathbf{q}_{c,i}^{(j)} = \nabla \ell_c(\theta_{c,i}^{(t)}; \xi)$ is the stochastic gradient of the local loss with respect to $\theta_{c,i}^{(t)}$, computed using a randomly sampled data point $\xi \sim \mathcal{D}_c$. The initialization for local training is given by:

$$\theta_{c,0}^{(t)} = \theta^{(t-\tau_c)}.$$

Once all participating clients $c \in J^{(t)}$ complete their local training and transmit their updated parameters to the central server, the global model is updated using an aggregation function:

$$\boldsymbol{\theta}^{(t+1)} = \operatorname{Aggregate}\left(\{\boldsymbol{\theta}_{c,I}^{(t)} \mid c \in J^{(t)}\}\right),$$

where the aggregation function may involve techniques such as weighted averaging based on the size of local datasets or other strategies to mitigate the effects of heterogeneity across clients.

This iterative process is repeated over multiple rounds, progressively refining the global model parameters θ based on asynchronous updates from the distributed clients. The federated learning framework thus enables collaborative training while preserving data privacy and accommodating communication and computation constraints at the client level. Algorithm 1 provides an overview of the proposed AFL.

Algorithm 1 Federated Learning with Asynchronous Client Updates

```
1: Input: Learning rate \eta, number of local iterations I, number of communication rounds \mathcal{J}, convergence threshold \kappa.
```

2: **Initialize:** Global model $\theta^{(0)}$.

3: **for** $t = 0, 1, ..., \mathcal{J} - 1$ **do**

4: Randomly select a subset of clients $J^{(t)}\subseteq\{1,2,\ldots,C\}$.

5: **for** each client $c \in J^{(t)}$ **in parallel do**

6: Initialize local model using the delayed global model:

$$\theta_{c,0}^{(t)} = \theta^{(t-\tau_c)},$$

where τ_c is the staleness of client c.

7: **for**
$$i = 0, 1, ..., I - 1$$
 do

8: Perform local update using stochastic gradient

descent:

$$\theta_{c,i+1}^{(t)} = \theta_{c,i}^{(t)} - \eta \nabla \ell_c(\theta_{c,i}^{(t)}; \xi),$$

where $\xi \sim \mathcal{D}_c$ is a randomly sampled data point.

end for

10: Send updated parameters $\theta_{c,I}^{(t)}$ and staleness τ_c to the server.

11: end for

9:

12: Aggregate updates to compute the new global model:

$$\theta^{(t+1)} = \operatorname{Aggregate}\left(\{\theta_{c,I}^{(t)} \mid c \in J^{(t)}\}\right).$$

13: **if** $\|\theta^{(t+1)} - \theta^{(t)}\| \le \kappa$ **then**

14: Break.

15: **end if**

16: end for

17: **Output:** Final global model $\theta^{(\mathcal{J})}$.

IV. CONVERGENCE ANALYSIS

In this section, we establish theoretical guarantees on the expected improvement in the loss function across training rounds. Specifically, we derive a recursive bound that characterizes the evolution of the model parameters and provides

insights into how step size, variance, and system heterogeneity impact convergence. We first present a key recursion bound in Lemma 1, which quantifies the expected descent in the loss function over a single training round. This result is further complemented by Lemma 2, where we analyze the drift in model updates due to the decentralized nature of the optimization process. These results collectively provide a rigorous foundation for understanding the convergence properties of our algorithm in the presence of non-convex objectives.

Lemma 1 (Non-Convex Recursion Bound). *Under Assumptions 1, 2, and 7, if the step size* γ *satisfies*

$$\gamma \le \frac{1}{6LJI(1+\beta^2/J)},$$

then the following holds for the expected improvement in the loss function \mathcal{L} after one training round:

$$\begin{split} \mathbb{E}\left[\mathcal{L}(\theta^{(j+1)}) - \mathcal{L}(\theta^{(j)})\right] &\leq -\frac{1}{6}JI\gamma\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(j)})\|^2\right] \\ &+ 2L\gamma^2JI\sigma^2 \\ &+ 2L\gamma^2J^2I^2\frac{C-J}{J(C-1)}\nu^2 \\ &+ \frac{5}{6}L^2\gamma\sum_{c=1}^J\sum_{i=0}^{I-1}\mathbb{E}\left[\|\theta_{c,i}^{(j)} - \theta^{(j)}\|^2\right]. \end{split}$$

Proof:

We focus on a single training round and omit the superscript j for simplicity. All expectations are conditioned on $\theta^{(j)}$, unless otherwise stated. Since $\mathcal L$ is L-smooth (Assumption 1), we begin with:

$$\mathbb{E}\left[\mathcal{L}(\theta + \Delta\theta) - \mathcal{L}(\theta)\right] \leq \mathbb{E}\left[\left\langle \nabla \mathcal{L}(\theta), \Delta\theta \right\rangle\right] + \frac{L}{2} \mathbb{E}\|\Delta\theta\|^2.$$

We first bound $\mathbb{E}\left[\langle \nabla \mathcal{L}(\theta), \Delta \theta \rangle\right]$. Substitute the overall update $\Delta \theta$ into the inner product term:

$$\mathbb{E}\left[\left\langle \nabla \mathcal{L}(\theta), \Delta \theta \right\rangle\right] = -\gamma J I \mathbb{E}\left[\left\langle \nabla \mathcal{L}(\theta) \right\rangle\right], \frac{1}{J} \sum_{c=1}^{J} \frac{1}{I} \sum_{i=0}^{I-1} \left(\nabla \mathcal{L}_{\pi_c}(\theta_{c,i}) - \nabla \mathcal{L}_{\pi_c}(\theta) + \nabla \mathcal{L}_{\pi_c}(\theta)\right)\right]. \quad (2)$$

Simplifying the expression:

$$\mathbb{E}\left[\left\langle \nabla \mathcal{L}(\theta), \Delta \theta \right\rangle\right] = -\gamma J I \|\nabla \mathcal{L}(\theta)\|^{2}$$
$$-\gamma J I \mathbb{E}\left[\left\langle \nabla \mathcal{L}(\theta), \frac{1}{J} \sum_{c=1}^{J} \frac{1}{I} \sum_{i=0}^{I-1} \left(\nabla \mathcal{L}_{\pi_{c}}(\theta_{c,i}) - \nabla \mathcal{L}_{\pi_{c}}(\theta)\right)\right\rangle\right].$$

Using the inequality $|\langle a,b\rangle| \leq \frac{1}{2}||a||^2 + \frac{1}{2}||b||^2$, we obtain:

$$\mathbb{E}\left[\langle \nabla \mathcal{L}(\theta), \Delta \theta \rangle\right] \leq -\gamma J I \|\nabla \mathcal{L}(\theta)\|^2 + \frac{1}{2} \gamma J I \|\nabla \mathcal{L}(\theta)\|^2 + \frac{1}{2} \gamma \sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathbb{E} \|\nabla \mathcal{L}_{\pi_c}(\theta_{c,i}) - \nabla \mathcal{L}_{\pi_c}(\theta)\|^2.$$

Using the smoothness assumption (Assumption (1)):

$$\mathbb{E}\left[\langle \nabla \mathcal{L}(\theta), \Delta \theta \rangle\right] \leq -\frac{1}{2} \gamma J I \|\nabla \mathcal{L}(\theta)\|^{2} + \frac{1}{2} L^{2} \gamma \sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathbb{E} \|\theta_{c,i} - \theta\|^{2}.$$
 (3)

Next, we bound $\mathbb{E}\|\Delta\theta\|^2$ as follows:

$$\frac{1}{2}L\mathbb{E}\|\Delta\theta\|^{2} \leq 2L\gamma^{2}\mathbb{E}\left\|\sum_{c=1}^{J}\sum_{i=0}^{I-1}\left(\mathbf{g}_{\pi_{c},i} - \nabla\mathcal{L}_{\pi_{c}}(\theta_{c,i})\right)\right\|^{2} + 2L\gamma^{2}\mathbb{E}\left\|\sum_{c=1}^{J}\sum_{i=0}^{I-1}\left(\nabla\mathcal{L}_{\pi_{c}}(\theta_{c,i}) - \nabla\mathcal{L}_{\pi_{c}}(\theta)\right)\right\|^{2} + 2L\gamma^{2}\mathbb{E}\left\|\sum_{c=1}^{J}\sum_{i=0}^{I-1}\left(\nabla\mathcal{L}_{\pi_{c}}(\theta) - \nabla\mathcal{L}(\theta)\right)\right\|^{2} + 2L\gamma^{2}\mathbb{E}\left\|\sum_{c=1}^{J}\sum_{i=0}^{I-1}\nabla\mathcal{L}(\theta)\right\|^{2}.$$
(4)

We now bound each term on the right-hand side (4), for the first term we have

$$2L\gamma^2 \mathbb{E} \left\| \sum_{c=1}^{J} \sum_{i=0}^{I-1} \left(\mathbf{g}_{\pi_c,i} - \nabla \mathcal{L}_{\pi_c}(\theta_{c,i}) \right) \right\|^2 \le 2L\gamma^2 J I \sigma^2.$$
 (see (1), Assumption 2)

For the second term we write from Assumption 1:

$$2L\gamma^{2}\mathbb{E} \left\| \sum_{c=1}^{J} \sum_{i=0}^{I-1} \left(\nabla \mathcal{L}_{\pi_{c}}(\theta_{c,i}) - \nabla \mathcal{L}_{\pi_{c}}(\theta) \right) \right\|^{2}$$

$$\leq 2L^{3}\gamma^{2}JI \sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathbb{E} \|\theta_{c,i} - \theta\|^{2}.$$

For the third and forth term we get:

$$2L\gamma^{2}\mathbb{E}\left\|\sum_{c=1}^{J}\sum_{i=0}^{I-1}(\nabla\mathcal{L}_{\pi_{c}}(\theta) - \nabla\mathcal{L}(\theta))\right\|^{2} + 2L\gamma^{2}\mathbb{E}\left\|\sum_{c=1}^{J}\sum_{i=0}^{I-1}\nabla\mathcal{L}(\theta)\right\|^{2} \leq 2L\gamma^{2}J^{2}I^{2}\frac{C-J}{J(C-1)}\nu^{2} + 2L\gamma^{2}J^{2}I^{2}\left(1 + \frac{C-J}{J(C-1)}\right)\|\nabla\mathcal{L}(\theta)\|^{2}.$$
 (5)

Substituting the above bounds and applying $\gamma \leq \frac{1}{6LJI(1+\beta^2/J)}$, we recover the superscripts and obtain:

$$\mathbb{E}\left[\mathcal{L}(\theta^{(j+1)}) - \mathcal{L}(\theta^{(j)})\right] \leq -\frac{1}{6}JI\gamma\mathbb{E}\|\nabla\mathcal{L}(\theta^{(j)})\|^{2}
+ 2L\gamma^{2}JI\sigma^{2} + 2L\gamma^{2}J^{2}I^{2}\frac{C-J}{J(C-1)}\nu^{2}
+ \frac{5}{6}L^{2}\gamma\sum_{j=1}^{J}\sum_{i=0}^{I-1}\mathbb{E}\|\theta_{c,i}^{(j)} - \theta^{(j)}\|^{2}.$$
(6)

Lemma 2. Suppose Assumptions 1, 2, 7 are satisfied. If $\gamma \leq \frac{1}{6LJJ(1+\beta^2/J)}$, then

$$E_{j} \leq \frac{9}{4} J^{2} I^{2} \gamma^{2} \nu^{2} + \frac{9}{4} J^{2} I^{3} \gamma^{2} \mu^{2}$$

$$+ \frac{9}{4} \left(\frac{\beta^{2}}{J} + 1 \right) J^{3} I^{3} \gamma^{2} \mathbb{E} \left[\left\| \nabla \mathcal{L}(\theta^{(j)}) \right\|^{2} \right].$$

Proof: According to Algorithm 1, the overall updates of AFL from $\theta^{(j)}$ to $\theta^{(j)}_{c,i}$ are

$$\theta_{c,i}^{(j)} - \theta^{(j)} = -\gamma \sum_{k=1}^{c} \sum_{f=0}^{p_{c,i}(k)} \mathbf{q}_{\psi_k,f}^{(j)},$$

$$p_{c,i}(k) = \begin{cases} I - 1, & \text{if } k \le c - 1, \\ i - 1, & \text{if } k = c. \end{cases}$$

For the sake of simplicity we consider only one training round and we drop our dependency to superscripts j. Unless otherwise stated, the expectation is conditioned on $\theta^{(j)}$. We aim to bound $\mathbb{E}\left[\|\theta_{c,i}-\theta\|^2\right]$, which is given by

$$\mathbb{E}\left[\left\|\theta_{c,i} - \theta\right\|^{2}\right] \\
\leq 4\gamma^{2} \mathbb{E}\left[\left\|\sum_{k=1}^{c} \sum_{f=0}^{p_{c,i}(k)} \left(\mathbf{q}_{\psi_{k},f} - \nabla \mathcal{L}_{\psi_{k}}(\theta_{k,f})\right)\right\|^{2}\right] \\
+ 4\gamma^{2} \mathbb{E}\left[\left\|\sum_{k=1}^{c} \sum_{f=0}^{p_{c,i}(k)} \left(\nabla \mathcal{L}_{\psi_{k}}(\theta_{k,f}) - \nabla \mathcal{L}_{\psi_{k}}(\theta)\right)\right\|^{2}\right] \\
+ 4\gamma^{2} \mathbb{E}\left[\left\|\sum_{k=1}^{c} \sum_{f=0}^{p_{c,i}(k)} \left(\nabla \mathcal{L}_{\psi_{k}}(\theta) - \nabla \mathcal{L}(\theta)\right)\right\|^{2}\right] \\
+ 4\gamma^{2} \mathbb{E}\left[\left\|\sum_{k=1}^{c} \sum_{f=0}^{p_{c,i}(k)} \nabla \mathcal{L}(\theta)\right\|^{2}\right]. \quad (7)$$

Next, we bound each term in Eq. (7), for the first term we have:

$$4\gamma^{2} \mathbb{E} \left[\left\| \sum_{k=1}^{c} \sum_{f=0}^{p_{c,i}(k)} \left(\mathbf{q}_{\psi_{k},f} - \nabla \mathcal{L}_{\psi_{k}}(\theta_{k,f}) \right) \right\|^{2} \right]$$

$$\leq 4\gamma^{2} \sum_{k=1}^{c} \sum_{f=0}^{p_{c,i}(k)} \mathbb{E} \left[\| \mathbf{q}_{\psi_{k},f} - \nabla \mathcal{L}_{\psi_{k}}(\theta_{k,f}) \|^{2} \right] \leq 4\gamma^{2} \mathcal{Y}_{c,i} \nu^{2},$$
(8)

For the second term in Eq. (7) we have:

$$4\gamma^{2}\mathbb{E}\left[\left\|\sum_{k=1}^{c}\sum_{f=0}^{p_{c,i}(k)}\left(\nabla\mathcal{L}_{\psi_{k}}(\theta_{k,f})-\nabla\mathcal{L}_{\psi_{k}}(\theta)\right)\right\|^{2}\right]$$

$$\leq 4\gamma^{2}\mathcal{Y}_{c,i}\sum_{k=1}^{c}\sum_{f=0}^{p_{c,i}(k)}\mathbb{E}\left[\left\|\nabla\mathcal{L}_{\psi_{k}}(\theta_{k,f})-\nabla\mathcal{L}_{\psi_{k}}(\theta)\right\|^{2}\right]$$

$$\leq 4L^{2}\gamma^{2}\mathcal{Y}_{c,i}\sum_{k=1}^{c}\sum_{f=0}^{p_{c,i}(k)}\mathbb{E}\left[\left\|\theta_{k,f}-\theta\right\|^{2}\right], \quad (9)$$

For the third term in Eq. (7) we can write:

$$4\gamma^{2}\mathbb{E}\left[\left\|\sum_{k=1}^{c}\sum_{f=0}^{p_{c,i}(k)}\left(\nabla\mathcal{L}_{\psi_{k}}(\theta)-\nabla\mathcal{L}(\theta)\right)\right\|^{2}\right] \\ \leq 4\gamma^{2}\mathcal{Y}_{c,i}\mathbb{E}\left[\left\|\nabla\mathcal{L}_{\psi_{k}}(\theta)-\nabla\mathcal{L}(\theta)\right\|^{2}\right], \quad (10)$$

and finally for the fourth term in Eq. (7) we have:

$$4\gamma^2 \mathbb{E} \left[\left\| \sum_{k=1}^c \sum_{f=0}^{p_{c,i}(k)} \nabla \mathcal{L}(\theta) \right\|^2 \right] \le 4\gamma^2 \mathcal{Y}_{c,i}^2 \|\nabla \mathcal{L}(\theta)\|^2. \tag{11}$$

Suppose we denote $\mathcal{Y}_{c,i} = (c-1)I + i$, then by replacing into the Eq. (7), we have:

$$\begin{split} E_{j} &\leq 4\gamma^{2}\nu^{2} \sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathcal{Y}_{c,i} \\ &+ 4L^{2}\gamma^{2} \sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathcal{Y}_{c,i} \sum_{k=1}^{c} \sum_{f=0}^{p_{c,i}(k)} \mathbb{E}\left[\|\theta_{k,f} - \theta\|^{2}\right] \\ &+ 4\gamma^{2} \sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathcal{Y}_{c,i} + 4\gamma^{2} \sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathcal{Y}_{c,i}^{2} \|\nabla \mathcal{L}(\theta)\|^{2}. \end{split}$$

Applying the bounds $\sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathcal{Y}_{c,i} \leq \frac{1}{2} J^2 I^2$ and $\sum_{c=1}^{J} \sum_{i=0}^{I-1} \mathcal{Y}_{c,i}^2 \leq \frac{1}{3} J^3 I^3$, we have:

$$\begin{split} E_{j} & \leq 2J^{2}I^{2}\gamma^{2}\sigma^{2} + 2L^{2}J^{2}I^{2}\gamma^{2}E_{j} + 2J^{2}I^{3}\gamma^{2}\nu^{2} \\ & + 2\left(\frac{\beta^{2}}{J} + 1\right)J^{3}I^{3}\gamma^{2}\|\nabla\mathcal{L}(\theta)\|^{2}. \end{split}$$

Manipulating and forcing the relation $\gamma \leq \frac{1}{6LIJ(1+\beta^2/J)}$, we reach the result:

$$E_{j} \leq \frac{9}{4}J^{2}I^{2}\gamma^{2}\sigma^{2} + \frac{9}{4}J^{2}I^{3}\gamma^{2}\nu^{2} + \frac{9}{4}\left(\frac{\beta^{2}}{J} + 1\right)J^{3}I^{3}\gamma^{2}\|\nabla\mathcal{L}(\theta)\|^{2}. \quad (12)$$

The claim follows after recovering the superscripts and taking unconditional expectations.

A. Tuning the learning rate

In the next Lemma we drive a bound on the delay aware learning rate for the AFL algorithm.

Lemma 3. We consider two non-negative sequences $\{d_t\}_{t\geq 0}$, $\{g_t\}_{t\geq 0}$, which satisfy the relation

$$d_{t+1} \le d_t - b\zeta_t g_t + a_1 \zeta_t^2 + a_2 \zeta_t^3, \tag{13}$$

for all $t \ge 0$, where b > 0, $a_1, a_2 \ge 0$, and ζ_t is the learning rate defined as

$$\zeta_t = \frac{\zeta_0}{\sqrt{t+1} \cdot (1+\alpha \cdot \tau_t)}.$$

Then, using $\pi_t = 1$ and $\Pi_T := \sum_{t=0}^T \pi_t = T+1$, we have:

$$\Psi_T := \frac{1}{\Pi_T} \sum_{t=0}^T g_t \pi_t \le \frac{d_0}{b\zeta_0(T+1)} \left(\sqrt{T+1} + \alpha \sum_{t=0}^T \tau_t \right) + \frac{a_1\zeta_0}{b} \cdot \frac{\log(T+1)}{\sqrt{T+1}} + \frac{a_2\zeta_0^2}{b} \cdot \frac{\log(T+1)}{T}.$$

Proof: Rearranging Eq. (13) and multiplying both sides by $\pi_t = 1$, we get:

$$bg_t \le \frac{d_t}{\zeta_t} - \frac{d_{t+1}}{\zeta_t} + a_1 \zeta_t + a_2 \zeta_t^2.$$

Summing over t = 0 to T results in:

$$b\sum_{t=0}^{T} g_t \le \sum_{t=0}^{T} \frac{d_t - d_{t+1}}{\zeta_t} + a_1 \sum_{t=0}^{T} \zeta_t + a_2 \sum_{t=0}^{T} \zeta_t^2.$$

The first term forms a telescoping sum:

$$\sum_{t=0}^{T} \frac{d_t - d_{t+1}}{\zeta_t} = \frac{d_0}{\zeta_0} \left(\sqrt{T+1} + \alpha \sum_{t=0}^{T} \tau_t \right).$$

For the second term, substituting $\zeta_t = \frac{\zeta_0}{\sqrt{t+1}\cdot(1+\alpha\cdot\tau_t)}$, we use bounds:

$$\sum_{t=0}^{T} \zeta_t \le \zeta_0 \sum_{t=0}^{T} \frac{1}{\sqrt{t+1}} \le \zeta_0 (2\sqrt{T+1} - 1).$$

Similarly, for ζ_t^2 :

$$\sum_{t=0}^{T} \zeta_t^2 \le \zeta_0^2 \sum_{t=0}^{T} \frac{1}{t+1} \le \zeta_0^2 \log(T+1).$$

Dividing by $\Pi_T = T + 1$, we have:

$$\Psi_T \le \frac{d_0}{b\zeta_0(T+1)} \left(\sqrt{T+1} + \alpha \sum_{t=0}^T \tau_t \right) + \frac{a_1\zeta_0}{b} \cdot \frac{\log(T+1)}{\sqrt{T+1}} + \frac{a_2\zeta_0^2}{b} \cdot \frac{\log(T+1)}{T}.$$

The first term scales as $O\left(\frac{1}{\sqrt{T}}\right)$ when $\tau_t=0$ (homogeneous case). The second and third terms decay at $O\left(\frac{\log(T)}{\sqrt{T}}\right)$ and $O\left(\frac{\log(T)}{T}\right)$, respectively, showing that higher-order terms become negligible for large T. Thus, the result reveals that the average gradient norm Ψ_T decreases asymptotically as $O(1/\sqrt{T})$, assuming a decaying learning rate schedule.

Theorem 1. Assume that all the local objective functions are L-smooth (Assumption 1) and non-convex. For the AFL algorithm (Algorithm 1), there exists a constant effective learning rate $\tilde{\gamma} = \gamma CI$ and weights $\{\pi_j\}_{j\geq 0}$ such that the weighted average of the global parameters

$$\bar{\theta}^{(\mathcal{J})} = \frac{\sum_{j=0}^{\mathcal{J}} \pi_j \theta^{(j)}}{\sum_{j=0}^{\mathcal{J}} \pi_j},$$

Under Assumptions 2 (bounded variance) and 7 (bounded heterogeneity), if $\tilde{\gamma} \leq \frac{1}{6L(1+\beta^2/C)}$ and $\pi_j = 1$, then satisfies the following bound:

$$\mathbb{E}\left[\|\nabla \mathcal{L}(\theta^{(j)})\|^2\right] \leq \frac{6(\mathcal{L}(\theta^{(0)}) - \mathcal{L}^*)}{\tilde{\eta}\mathcal{J}} + \frac{12L\tilde{\gamma}\sigma^2}{CI} + \frac{45L^2\tilde{\gamma}^2\sigma^2}{4CI} + \frac{45L^2\tilde{\gamma}^2\nu^2}{4C}$$

Proof: From Lemmas 1 and 2, and assuming $\gamma \leq \frac{1}{6LIJ(1+\beta^2/J)}$, the recursion simplifies to (see the Appendix VII for details):

$$\mathbb{E}\left[\mathcal{L}(\theta^{(j+1)}) - \mathcal{L}(\theta^{(j)})\right] \le -\frac{1}{6}IJ\gamma\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(j)})\|^2\right]$$
(14)
+ $2L\gamma^2IJ\sigma^2 + 2L\gamma^2J^2I^2\frac{C-J}{J(C-1)}\nu^2$
(15)
+ $\frac{15}{8}L^2\gamma^3J^2I^2\sigma^2 + \frac{15}{8}L^2\gamma^3J^2I^3\nu^2$.
(16)

Introducing the substitution $\tilde{\gamma} := IJ\gamma$, we rewrite the recursion after subtracting \mathcal{L}^* from both sides:

$$\begin{split} \mathbb{E}\left[\mathcal{L}(\theta^{(j+1)}) - \mathcal{L}^*\right] &\leq \mathbb{E}\left[\mathcal{L}(\theta^{(j)}) - \mathcal{L}^*\right] \\ &- \frac{\tilde{\gamma}}{6} \mathbb{E}\left[\|\nabla \mathcal{L}(\theta^{(j)})\|^2\right] \\ &+ \frac{2L\tilde{\gamma}^2 \sigma^2}{IJ} + 2L\tilde{\gamma}^2 \frac{C - J}{J(C - 1)} \nu^2 \\ &+ \frac{15}{8} \frac{L^2\tilde{\gamma}^3 \sigma^2}{IJ} + \frac{15}{8} \frac{L^2\tilde{\gamma}^3 \nu^2}{J}. \end{split}$$

Now, applying Lemma 3, we define the following parameters: $t=j,\ T=\mathcal{J},\ \zeta=\tilde{\gamma},\ d_t=\mathbb{E}\left[\mathcal{L}(\theta^{(j)})-\mathcal{L}^*\right],\ b=\frac{1}{10},$ $g_t=\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(j)})\|^2\right],\ \pi_t=1,\ a_1=\frac{2L\sigma^2}{IJ}+2L\frac{C-J}{J(C-1)}\nu^2,$ and $a_2=\frac{15}{8}\frac{L^2\sigma^2}{IJ}+\frac{15}{8}\frac{L^2\nu^2}{J}$. Using these substitutions, we derive:

$$\mathbb{E}\left[\|\nabla \mathcal{L}(\theta^{(j)})\|^{2}\right] \leq \frac{6\left(\mathcal{L}(\theta^{0}) - \mathcal{L}^{*}\right)}{\tilde{\gamma}\mathcal{J}} + \frac{12L\tilde{\gamma}\sigma^{2}}{IJ} + \frac{12L\tilde{\gamma}\nu^{2}(C - J)}{J(C - 1)} + \frac{45L^{2}\tilde{\gamma}^{2}\sigma^{2}}{4IJ} + \frac{45L^{2}\tilde{\gamma}^{2}\nu^{2}}{4IJ}.$$
(17)

When J=C, the result in the theorem statement follows. Specifically: 1. The first term, $O(1/\mathcal{J})$, dominates the bound as the number of iterations increases. 2. The second term, $O(\gamma)$, reflects variance (σ^2) and diminishes as γ decreases. 3. The remaining terms, $O(\gamma^2)$, arise from both variance and heterogeneity (ν^2) , and scale quadratically with γ .

Choosing a learning rate of $\gamma \sim O(1/\sqrt{\mathcal{J}})$ ensures the expected gradient norm satisfies:

$$\mathbb{E}\left[\|\nabla \mathcal{L}(\theta^{(j)})\|^2\right] \sim O\left(\frac{1}{\sqrt{\mathcal{J}}}\right),\,$$

which is consistent with the established convergence rate for non-convex optimization.

V. SIMULATIONS

In this section we consider AFL algorithm to tackle challenges related to heterogeneous client environments and the optimization of non-convex objective functions. We evaluate the performance of our AFL algorithm using well-known benchmark datasets, namely MNIST and CIFAR, with non-convex optimization objectives. Additionally, we compare the results of the AFL algorithm with those of a synchronous federated learning (FL) approach.

A. Hardware Setup

Sion simplifies to (see the Appendix Our deep learning model training was conducted on high-performance hardware to meet the computational demands of wind power simulations. Specifically, we utilized an NVIDIA $+2L\gamma^2IJ\sigma^2+2L\gamma^2J^2I^2\frac{C-J}{J(C-1)}\nu^2$ (VRAM)¹. Additionally, each CPU core was allocated 256 GB of memory to facilitate large-scale data processing.

The training process was managed using the Simple Linux Utility for Resource Management (SLURM) job scheduler². Job-specific constraints were applied to ensure compatibility with the A100 GPU's extensive memory capacity, optimizing parallel processing and maintaining model training stability.

B. Code and data availability statement.

The AFL algorithm is implemented using the PyTorch framework, taking advantage of its advanced tools for constructing and training deep learning models. It is openly available to enable reproducibility and facilitate further research at this Github repository ³. The code for data extraction, interpolation, statistical analysis, visualization, DNN training and evaluation is provided in the associated repository. It includes detailed comments and instructions for reproducing the results. Additionally, a version will be archived on Zenodo ⁴ for reference.

C. Classification task on MNIST dataset

The dataset used in this experiment is the MNIST dataset, which contains 60,000 training images and 10,000 test images of handwritten digits. To simulate realistic federated learning conditions, the data was partitioned among clients in a non-IID manner using a Dirichlet distribution with a concentration parameter $\alpha=0.5$. A lower value of α creates more uneven distributions, meaning some clients may have a disproportionate number of samples for specific classes. Each image is normalized and transformed into a PyTorch tensor for model input.

a) Training setup: The model used for both clients and the server is a Convolutional Neural Network (CNN). The network architecture consists of three convolutional layers, each followed by a ReLU activation function. The convolutional layers use 32 filters with a kernel size of 3×3 , a stride of 1, and padding of 1, ensuring the spatial dimensions remain consistent across layers. The weights of the convolutional layers are initialized using Xavier uniform initialization to improve convergence. Following the convolutional layers is an adaptive average pooling layer, which reduces the spatial dimensions to 1×1 . A fully connected layer maps the resulting features

¹VRAM refers to the dedicated memory on a GPU, optimized for high-speed graphics processing, parallel computations, and deep learning tasks. Unlike standard RAM, which is used by the CPU, VRAM efficiently handles large-scale matrix operations, simulations, and neural network training.

²SLURM is a widely used open-source workload manager for highperformance computing (HPC) environments. Developed by the Lawrence Livermore National Laboratory, it efficiently allocates resources such as CPUs, GPUs, and memory across computing nodes, enabling optimal job scheduling and execution.

³https://github.com/Ali-Forootani/AFL non convex non iid

⁴https://zenodo.org/records/14962410

to 10 output classes, corresponding to the digits 0 through 9. The final output layer applies a log-softmax function, making the model suitable for multi-class classification.

The federated learning procedure follows the standard server-client communication protocol. At the start of each communication round, a subset of clients is selected randomly to participate. Each selected client trains its local model on its dataset for a specified number of epochs before sending the updated model weights back to the server. The server aggregates the client models using a weighted average, where the weight corresponds to the size of the client's local dataset. This aggregated model is then broadcast back to the clients for the next round of training.

The learning rate at each client is adjusted dynamically based on the delay using the formula $\gamma_t = \frac{\gamma_0}{\sqrt{t+1}\cdot(1+\alpha\cdot d_t)}$, where γ_0 is the initial learning rate, t is the epoch number, α is the delay scaling factor, and d_t is the delay time. This ensures that clients with longer delays make smaller updates, preventing them from destabilizing the global model. An early stopping mechanism is implemented to terminate training if the client's loss does not improve for 10 consecutive epochs, reducing unnecessary computation.

PyTorch is used for model definition and training, while the asyncio and nest_asyncio packages enabled asynchronous client updates. The experiment is configured with C=10 clients, each training for I=10 local epochs per communication round, with a batch size of 64. The total number of communication rounds is set to $\mathcal{J}=1000$, and in each round. The initial learning rate is set to $\gamma_0=10^{-3}$, and the delay is computed dynamically in the code.

The loss function used for training is negative log-likelihood loss (NLLLoss), appropriate for multi-class classification tasks. The Adam optimizer [41] is chosen for its adaptive learning rate capabilities, which complement the delay-aware adjustments.

b) Results: In the first experiment J=5 clients are randomly selected to participate in the training. The global server model demonstrated consistent improvement across the communication rounds as shown in figure 1. The server loss decreased steadily, with diminishing returns toward the later rounds, indicating convergence. This trend suggests that the server is able to generalize well despite the heterogeneous data distributions and asynchronous training times of the clients.

The non-IID data distribution can introduce additional challenges, as clients with highly imbalanced datasets required more rounds to contribute effectively to the global model. Despite this, the model aggregation strategy is able to handle the variability in client updates, resulting in a robust and generalized server model.

Figure 2 presents a histogram depicting the frequency of client selection in the federated learning process. The results indicate that some clients, such as client 2, have been selected more frequently than others. Despite imbalance participation of the clients in the training loop, the global convergence of the server is not affected significantly as shown in 1.

Figure 3, showing the server loss in the AFL algorithm for different MNIST experiments, illustrates the impact of varying client participation on the training process. Different

Server Loss Across Rounds in MNIST dataset with 50% clients participation

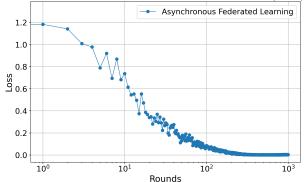


Fig. 1: Server loss across rounds in AFL algorithm for MNIST dataset with participation of 5 clients in the training loop.

curves correspond to experiments where the number of clients participating per round ranges from 3 to 8.

From the plot 3, it is evident that increasing the number of clients per round leads to a faster reduction in server loss, indicating a more rapid convergence of the global model. As the number of clients per round increases, the frequency of model updates also rises, allowing the server to aggregate more information from diverse clients. This leads to faster improvement of the global model, especially in the early rounds of training.

However, after a certain point, the improvement in the server loss starts to diminish, as shown by the flattening of the curves. This indicates the onset of convergence, where further model updates yield smaller improvements. The results also suggest that, despite the asynchronous nature of the client updates, the server can still benefit from increased client participation.

The variations in the curves highlight the balance between the speed of model convergence and the communication cost, as adding more clients per round can improve the server model more quickly but also increase the computational burden on the server to aggregate the larger number of updates. This tradeoff is an important consideration when designing federated learning systems, especially in scenarios with high client variability, like the one presented with the MNIST dataset in your experiment.

Figure 4 compares the training loss of asynchronous and synchronous federated learning (FL) on the MNIST dataset. Both approaches show a decreasing loss trend, confirming effective model training. However, synchronous FL exhibits a smoother loss reduction, while AFL experiences higher variance, particularly in early training rounds. This fluctuation is likely due to staggered client updates and varying staleness in AFL. Despite this variance, both methods eventually converge to similar loss levels, demonstrating that asynchronous FL remains a viable alternative to synchronous FL.

D. Classification task on CIFAR dataset

In the next example we employ the AFL algorithm on the CIFAR-10 dataset using a ResNet architecture. CIFAR-10 contains 50,000 training and 10,000 test images in 10 classes, each image being 32×32 pixels with three color channels

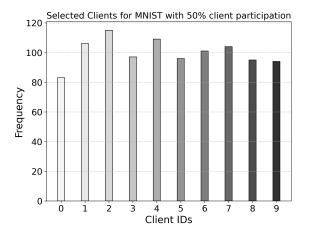


Fig. 2: Frequency distribution of different clients during the training loop in AFL algorithm with participation of 5 clients in the training loop for the MNIST dataset.

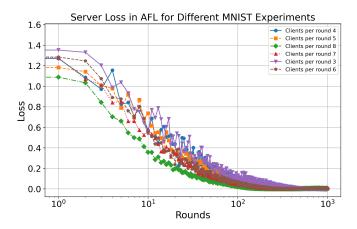


Fig. 3: Server loss across rounds based on the various number of client participation in the AFL algorithm for the classification task for the MNIST dataset.

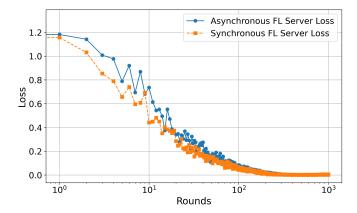


Fig. 4: Comparison of the AFL algorithm with Synchronous FL with participation of 5 clients in the training process.

(RGB). To reflect a realistic setup, the data is partitioned in a non-i.i.d. manner among 10 clients using a Dirichlet distribution with concentration parameter $\alpha=0.5$, so some clients may have data strongly skewed toward certain classes.

a) Training setup: Each client maintains a local ResNet model consisting of three layers of stacked blocks, where each block performs two 3×3 convolutions with batch normalization and adds a shortcut path to mitigate vanishing gradients. After these layers, global average pooling reduces the spatial dimension, and a final fully connected layer outputs 10 logits 5 followed by a log-softmax activation for classification. To mimic real-world conditions, client's delay are computed before training starts, and the code uses asyncio to handle these tasks concurrently.

A delay-aware learning rate is adopted, scaling the base learning rate $\gamma_0=10^{-3}$ by $\frac{1}{\sqrt{t+1}\left(1+\alpha\cdot d_t\right)}$, where $\alpha=0.01$ and d_t is the client's delay, so heavily delayed clients contribute smaller updates that reduce the risk of destabilizing the global model. In addition early stopping, with a patience of 10 epochs, halts local training if the loss fails to improve, saving computation on less productive updates. Once local training finishes, the server aggregates the client's parameters via a weighted average proportional to each client's dataset size, forming the updated global model.

b) Results: At the first experiment in each communication round, a random subset of J=5 out of the C=10 clients is chosen to train locally for I=10 epochs, after which their models are sent back to a global server. Total number of training rounds is considered $\mathcal{J}=200$.

The simulation result for the first experiment is shown in the figure 5. Over $\mathcal{J}=200$ communication rounds, the server's average loss converges despite heterogeneous client distributions and client's delays, demonstrating the resilience of asynchronous federated learning with ResNet on CIFAR-10. In the early rounds, the server loss decreases significantly, suggesting that even with a small number of updates, the global model rapidly adapts to the distributed data.

The loss curve exhibits fluctuations at certain points, particularly in the first few rounds. This behavior is characteristic of AFL, where updates arrive at different times, and clients contribute to the global model asynchronously.

The histogram in 6 illustrates the frequency of client participation during the AFL training process on the CIFAR-10 dataset, where 5 out of 10 clients are randomly selected per communication round. The variation in participation frequency among clients highlights the non-uniform selection process inherent in AFL. Some clients, such as Client 0 and Client 5, are chosen more frequently than others, while some, like Client 7, have fewer training opportunities. This imbalance in client participation can introduce variability in model updates, yet the global model still converges effectively, as observed in the loss curve. The randomness in client selection ensures diversity in training data aggregation, helping the model generalize better despite the non-IID data distribution. These results

⁵the term *logits*, computed as the weighted sum of input features with an added bias, represent the raw decision scores before classification; in this dataset, they are thresholded at zero to assign binary labels.

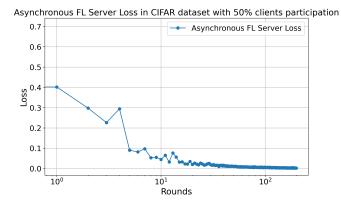


Fig. 5: Server loss across rounds in AFL algorithm for CIFAR dataset with participation of 5 clients in the training loop.

demonstrate the robustness of AFL in handling heterogeneous client availability while maintaining stable learning dynamics.

The comparison between AFL and Synchronous FL on the CIFAR-10 dataset is shown in the figure 7 reveals key trade-offs in their training dynamics. While SFL ensures more stable updates by aggregating all selected client models before updating the global model, it introduces delays due to waiting for all clients. In contrast, AFL enables faster updates by allowing clients to contribute asynchronously, resulting in more fluctuations in the early rounds but maintaining steady convergence over time.

Figure 8 presents the impact of varying the number of participating clients per round on server loss in AFL for the CIFAR-10 dataset. Each curve corresponds to a different number of participating clients per round, ranging from 3 to 8. From this figure, we observe that increasing the number of clients per round leads to a faster reduction in server loss, indicating improved convergence. When more clients participate, the global model receives more frequent and diverse updates, accelerating learning. However, the difference in convergence rates becomes less pronounced after a certain number of rounds, suggesting diminishing returns in increasing client participation. The results also highlight the robustness of AFL in handling asynchronous updates, as all configurations eventually converge despite fluctuations in early rounds.

This experiment underscores the trade-off between efficiency and convergence speed in federated learning. While selecting more clients per round enhances learning stability, it also increases communication overhead. The results validate that even with a lower number of participating clients, AFL remains effective, demonstrating its adaptability to real-world federated learning scenarios where client availability may be limited or unpredictable.

Despite initial variations, both AFL and Synchronous FL achieve similar final loss values, demonstrating that AFL can effectively learn from decentralized data without requiring strict synchronization. This makes AFL a more communication-efficient alternative for federated learning settings where client availability varies.

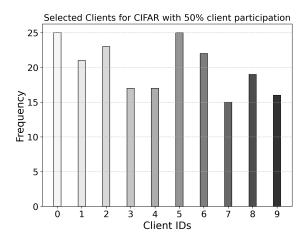


Fig. 6: Frequency distribution of different clients during the training loop in AFL algorithm with participation of 5 clients in the training loop for the CIFAR dataset.

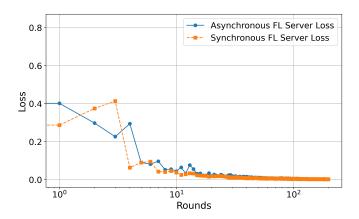


Fig. 7: Comparison of the AFL algorithm with Synchronous FL with participation 5 clients in the training loop for CIFAR dataset.

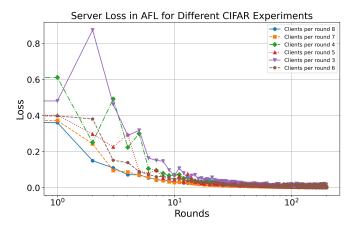


Fig. 8: Server loss across rounds based on the various number of client participation in the AFL algorithm for the classification task for the CIFAR dataset.

VI. CONCLUSION

In this paper, we extended asynchronous federated learning (AFL) to the optimization of non-convex objective functions, addressing key challenges such as client staleness, heterogeneity, and communication delays. Our approach introduced a staleness-aware aggregation mechanism and a dynamic learning rate schedule to adapt to the asynchronous nature of client updates, ensuring more stable convergence. The theoretical analysis provided mathematical bounds on the impact of stale gradients, demonstrating that our approach effectively mitigates their negative effects. Through extensive simulations on the MNIST and CIFAR-10 datasets, we validated the effectiveness of AFL in handling non-IID data distributions and heterogeneous client environments. The results demonstrated that AFL maintains robust convergence while reducing the impact of straggler clients, making it a scalable alternative to Synchronous FL. Specifically, our findings highlighted that increasing client participation per round accelerates convergence while preserving stability. The comparison between AFL and Synchronous FL showed that despite early-round fluctuations, AFL achieves similar final performance with lower synchronization overhead. This study contributes to the advancement of federated learning in practical, large-scale, and heterogeneous environments. Future work can focus on further refining staleness compensation mechanisms, incorporating adaptive sampling strategies, and exploring AFL 's applicability to additional real-world tasks such as healthcare, finance, and smart grid systems.

VII. APPENDIX

This section provides more details related to the intermediate steps for derivation of Eq.(14) in Theorem 1. Starting from the inequality given in Lemma 1:

$$\mathbb{E}\left[\mathcal{L}(\theta^{(j+1)}) - \mathcal{L}(\theta^{(j)})\right] \le -\frac{1}{6}JI\gamma\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(j)})\|^2\right] + 2L\gamma^2JI\sigma^2 + 2L\gamma^2J^2I^2\frac{C-J}{J(C-1)}\nu^2 + \frac{5}{6}L^2\gamma E_j, \quad (18)$$

and substituting E_i from Lemma 2:

$$E_{j} \leq \frac{9}{4}J^{2}I^{2}\gamma^{2}\sigma^{2} + \frac{9}{4}J^{2}I^{3}\gamma^{2}\nu^{2} + \frac{9}{4}\left(\frac{\beta^{2}}{J} + 1\right)J^{3}I^{3}\gamma^{2}\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(j)})\|^{2}\right],$$

we substitute E_j into the recursion to get:

$$\begin{split} \mathbb{E}\left[\mathcal{L}(\theta^{(j+1)}) - \mathcal{L}(\theta^{(j)})\right] &\leq -\frac{1}{6}JI\gamma\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(j)})\|^2\right] \\ &\quad + 2L\gamma^2JI\sigma^2 \\ &\quad + 2L\gamma^2J^2I^2\frac{C-J}{J(C-1)}\nu^2 + \frac{15}{8}L^2\gamma^3J^2I^2\sigma^2 + \frac{15}{8}L^2\gamma^3J^2I^3\nu^2 \\ &\quad + \frac{15}{8}L^2\gamma^3\left(\frac{\beta^2}{J} + 1\right)J^3I^3\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(j)})\|^2\right]. \end{split}$$

Focusing on the gradient-related terms:

$$-\frac{1}{6}JI\gamma\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(r)})\|^{2}\right] + \frac{15}{8}L^{2}\gamma^{3}\left(\frac{\beta^{2}}{J}+1\right)J^{3}I^{3}\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^{(r)})\|^{2}\right].$$

Substituting $L=\frac{1}{6\gamma JI\left(1+\frac{\beta^2}{J}\right)}$, we rewrite L^2 as: $L^2=\frac{1}{36\gamma^2J^2I^2\left(1+\frac{\beta^2}{J}\right)^2}$. Substituting L^2 into the second term:

$$\begin{split} \frac{15}{8}L^2\gamma^3 \left(\frac{\beta^2}{J} + 1\right) J^3 I^3 \\ &= \frac{15}{8} \cdot \frac{\gamma^3}{36\gamma^2 J^2 I^2 \left(1 + \frac{\beta^2}{J}\right)^2} \cdot \left(\frac{\beta^2}{J} + 1\right) J^3 I^3 \\ &= \frac{15}{8} \cdot \frac{\gamma J I}{36 \left(1 + \frac{\beta^2}{J}\right)} \mathbb{E} \left[\|\nabla \mathcal{L}(\theta^{(r)})\|^2 \right] \end{split}$$

Combining with the first term:

$$JI\gamma \mathbb{E}\left[\|\nabla \mathcal{L}(\theta^{(r)})\|^2\right] \left(-\frac{1}{6} + \frac{5}{96\left(1 + \frac{\beta^2}{J}\right)}\right).$$

Simplify the coefficient:

$$-\frac{1}{6} + \frac{5}{96\left(1 + \frac{\beta^2}{J}\right)} = \frac{-16\left(1 + \frac{\beta^2}{J}\right) + 5}{96\left(1 + \frac{\beta^2}{J}\right)} = \frac{-11 - \frac{16\beta^2}{J}}{96\left(1 + \frac{\beta^2}{J}\right)}.$$

To provide a suitable upper bound for this coefficient, note that: $\frac{-11 - \frac{16\beta^2}{J}}{96\left(1 + \frac{\beta^2}{J}\right)} \le -\frac{1}{6}$. Thus, the final simplified recursion is:

$$\begin{split} \mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}^{(r+1)}) - \mathcal{L}(\boldsymbol{\theta}^{(r)})\right] &\leq -\frac{1}{6}JI\gamma\mathbb{E}\left[\|\nabla\mathcal{L}(\boldsymbol{\theta}^{(r)})\|^2\right] \\ &+ \text{higher-order terms.} \end{split}$$

REFERENCES

- H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 5693–5700, 2019.
- [2] 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.
- [3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al., "Communication-efficient learning of deep networks from decentralized data," in Artificial Intelligence and Statistics, pp. 1273–1282, PMLR, 2017
- [4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, et al., "Towards federated learning at scale: System design," Proceedings of the 2nd SysML Conference, 2019.
- [5] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021
- [6] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of healthcare informatics research*, vol. 5, pp. 1–19, 2021.
- [7] G. Long, Y. Tan, J. Jiang, and C. Zhang, "Federated learning for open banking," in *Federated learning: privacy and incentive*, pp. 240–254, Springer, 2020.

- [8] A. Forootani and R. Iervolino, "Asynchronous federated learning: A scalable approach for decentralized machine learning," arXiv preprint arXiv:2412.17723, 2024.
- [9] R. Yu and P. Li, "Toward resource-efficient federated learning in mobile edge computing," *IEEE Network*, vol. 35, no. 1, pp. 148–155, 2021.
- [10] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems* 2020, pp. 429–450, 2020.
- [11] 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.
- [12] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 174–223, 2020.
- [13] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, "Realizing the heterogeneity: A self-organized federated learning framework for iot," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3088–3098, 2020.
- [14] A. Harlap, H. Cui, W. Dai, J. Wei, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing, "Addressing the straggler problem for iterative convergent parallel ml," in *Proceedings of the seventh ACM symposium on cloud computing*, pp. 98–111, 2016.
- [15] 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 14th Workshop on Hot Topics in Operating Systems (HotOS XIV), 2013.
- [16] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data," *ArXivorg*, 2020.
- [17] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in 2020 IEEE International Conference on Big Data (Big Data), pp. 15–24, IEEE, 2020
- [18] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "Non-synchronous federated learning," arXiv preprint arXiv:2001.08288, 2020
- [19] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," arXiv preprint arXiv:2003.10972, 2020.
- [20] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*, pp. 5132–5143, PMLR, 2020.
- [21] X. Lian, C. Zhang, C.-J. Hsieh, W. Zhang, J. Liu, J. T. Kwok, and I. S. Dhillon, "Asynchronous decentralized parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [22] W. Wang, K. Zhou, T. Li, et al., "Adaptive federated learning in asynchronous networks," arXiv preprint arXiv:2106.00741, 2021.
- [23] K. Bäckström, A. Forslund, T. N. Vu, and A. Jakobsson, "Mindthestepasyncpsgd: Adaptive asynchronous parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, 2022.
- [24] K. Bäckström, A. Forslund, T. N. Vu, and A. Jakobsson, "Asap sgd: Instance-based adaptiveness to staleness in asynchronous sgd," *Proceedings of the 39th International Conference on Machine Learning*, 2022.
- [25] S. Dutta, G. Joshi, S. Ghosh, and S. Sra, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd," *Proceedings* of the 22nd International Conference on Artificial Intelligence and Statistics, 2018.
- [26] S. Zheng, H. Cui, Y. Li, L. Chen, A. Smola, and J. Liu, "Asynchronous stochastic gradient descent with delay compensation," *Proceedings of* the 34th International Conference on Machine Learning, 2017.
- [27] Y. Zhang, L. Xiao, T. Wang, and S. Wang, "On the parallelization upper bound for asynchronous stochastic gradient descent," *Journal of Optimization Theory and Applications*, vol. 192, pp. 185–203, 2022.
- [28] E. Author and Others, "Asynchronous accelerated stochastic gradient descent," *Journal of Machine Learning Research*, 2022.
- [29] X. Zhou, P. Mertikopoulos, and N. Bambos, "Distributed asynchronous optimization with unbounded delays: How slow can you go?," Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018.
- [30] X. Lian, C. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," Advances in Neural Information Processing Systems, 2015.
- [31] J. Keuper and F.-J. Pfreundt, "Asynchronous parallel stochastic gradient descent," Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops, 2015.

- [32] B. Gu, A. Xu, Z. Huo, C. Deng, and H. Huang, "Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning," *IEEE Transactions on Neural Networks and Learn*ing Systems, 2021.
- [33] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," arXiv preprint arXiv:1903.03934, 2019.
- [34] T. Chen, G. B. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in Advances in Neural Information Processing Systems, vol. 31, 2018.
- [35] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "Safa: a semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, 2020.
- [36] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," arXiv preprint arXiv:2106.06639, 2021.
- [37] M. Chen, B. Mao, and T. Ma, "Fedsa: A staleness-aware asynchronous federated learning algorithm with non-iid data," *Future Generation Computer Systems*, vol. 120, pp. 1–12, 2021.
- [38] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," arXiv preprint arXiv:1903.03934, 2019.
- [39] A. Koloskova, S. U. Stich, and M. Jaggi, "Sharper convergence guarantees for asynchronous sgd for distributed and federated learning," in Advances in Neural Information Processing Systems, 2022.
- [40] A. Forootani and R. Iervolino, "Asynchronous federated learning: A scalable approach for decentralized machine learning," arXiv preprint arXiv:2412.17723, 2024.
- [41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.