Resource-Aware Asynchronous Online Federated Learning for Nonlinear Regression

Francois Gauthier*, Vinay Chakravarthi Gogineni*, Stefan Werner*, Yih-Fang Huang†, Anthony Kuh‡
*Dept. of Electronic Systems, Norwegian University of Science and Technology, Norway
†Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, IN, USA
‡Dept. of Electrical and Computer Engineering, University of Hawaii, Hawaii, USA
E-mails: {francois.gauthier, vinay.gogineni, stefan.werner}@ntnu.no, huang@nd.edu, kuh@hawaii.edu

Abstract—**Many assumptions in the federated learning literature present a best-case scenario that can not be satisfied in most real-world applications. An asynchronous setting reflects the realistic environment in which federated learning methods must be able to operate reliably. Besides varying amounts of non-IID data at participants, the asynchronous setting models heterogeneous client participation due to available computational power and battery constraints and also accounts for delayed communications between clients and the server. To reduce the communication overhead associated with asynchronous online federated learning (ASO-Fed), we use the principles of partial-sharing-based communication. In this manner, we reduce the communication load of the participants and, therefore, render participation in the learning task more accessible. We prove the convergence of the proposed ASO-Fed and provide simulations to analyze its behavior further. The simulations reveal that, in the asynchronous setting, it is possible to achieve the same convergence as the federated stochastic gradient (Online-FedSGD) while reducing the communication tenfold.**

I. Introduction

A vast amount of data is available on distributed devices, including edge devices. This motivates the development of distributed learning methods operating on those devices, and coping with edge devices. Federated learning (FL) is designed with this task in mind, as it provides an adaptive large-scale collaborative learning framework. In FL, a server aggregates information received from devices called clients to train a global model; the clients do not share any private data with the server, only their local model parameters [1], [2]. Amongst the features that make federated learning (FL) stand out from typical distributed learning are the assumptions of uneven data distribution and statistical heterogeneity [3].

There are, however, further complications when learning on edge devices that are not considered in most FL implementations [2], [4]–[9]. In a realistic setting, clients cannot be expected to have the same participation frequency, e.g., because of battery constraints, channel availability, or concurrent solicitations [10]–[12]. Furthermore, clients may become unavailable for a certain period during the learning process, i.e., some clients are temporarily out of order or not reachable by the server [10], [11]. Physical constraints may also introduce delays in the communication between clients and the server [11]–[13]. These constraints, frequently occurring in practice, impair the efficiency of existing FL methods and make the development of FL methods tailored for an asynchronous environment challenging [10]–[15].

Besides, the energy toll taken by FL methods on clients can sometimes be prohibitive, and methods are being developed to address this issue [7], [13]. Notably, the communication of model parameters to and from the server accounts for a substantial portion of the power consumption in FL. Hence, it is crucial to reduce this communication cost from the client's perspective [4], [7], [13], [14]. Furthermore, in asynchronous settings, efficiency in communication is essential for reducing energy and bandwidth thresholds at which devices choose to participate, hence increasing participation rates, and thus maximizing the benefits of the FL framework. Therefore, clients would benefit from an FL approach that ensures satisfactory learning in an asynchronous environment while imposing a minimal communication burden.

A considerable amount of research has been undertaken on both communication-efficient FL [4], [6], [8], [9], [16]–[18] and asynchronous FL [10]–[12], [15], [19], [20]; however, there is little work combining the two aspects into one. The works in [13], [17] reduced the communication overhead via compressed updates on the client-side, but they did not address communication needs on the server-side. Aside from the accuracy cost associated with this projection method, it also adds an extra computational burden, which is not appealing for low-battery clients. Moreover, the work in [17] did not consider asynchronous settings. Although the work in [14] reduced the communication load of the clients, it is specific to deep neural networks and does not provide a mathematical analysis of the presented results; in addition, the asynchronous setting considered do not include communication delays. The classical federated averaging [4] reduced the communication in FL by selecting a subset of the clients to participate at each iteration. However, because some clients may participate sporadically in the asynchronous setting, we do not intend to discard any participation by sub-sampling the available clients. Another option, explored until recently only in distributed learning, is the partial-sharing of the model parameters [21]. The partial-sharing-based online FL (PSO-Fed) [16] introduces partial-sharing in FL, but only in an ideal setting.

This work was supported by the Research Council of Norway.

This paper extends the work and analysis on partial-sharing-based communication to the asynchronous online FL. The developed method presents the advantages of being feasible in a realistic environment and reducing the computational load of participants. The proposed partial-sharing asynchronous online federated learning (PAO-Fed) algorithm oversees the collaborative estimation of a continuous nonlinear model represented on a random Fourier feature (RFF) space [22], [23], where only subsets of the RFF representation of the model are exchanged between the server and the clients. We prove that PAO-Fed converges even in a setting where client participation is random, and communication links suffer from delays. Lastly, we provide numerical results to compare PAO-Fed with existing methods in various asynchronous settings.

II. Preliminaries

A. Federated learning

We consider a global server linked to K geographically distributed devices, referred to as clients. The set of clients is denoted $\mathcal{K}$. The data of the network is distributed over those client, the local data of client $k \in \mathcal{K}$ is denoted $\mathbf{X}_k$. We consider that the entire data is not available at the beginning of the learning process but instead becomes available progressively. We denote the data available at client k at global iteration n by $\mathbf{x}_{k,n}$ and the corresponding desired output by $\mathbf{y}_{k,n}$; their relation is described as

$$y_{k,n} = f(\mathbf{x}_{k,n}) + \eta_{k,n}, \tag{1}$$

where $f(\cdot)$ is a continuous nonlinear model that we want to estimate, and $\eta_{k,n}$ is the observation noise. Here, we aim to estimate the global model $\mathbf{w}$, which is a linear representation of $f(\cdot)$ in the D-dimensional random Fourier feature (RFF) space. The global model $\mathbf{w}$ can be estimated by minimizing the following objective functions at the server and client k, given, as in [22], [23], by:

$$\mathcal{J}(\mathbf{w}) = \frac{1}{K}\sum_{k\in\mathcal{K}} \mathcal{J}_k(\mathbf{w}), \tag{2}$$
$$\mathcal{J}_k(\mathbf{w}) = \mathbb{E}[|y_{k,n} - \hat{y}_{k,n}|^2],$$

with $\hat{y}_{k,n} = \mathbf{w}^{\mathsf{T}}\mathbf{z}_{k,n}$, where $\mathbf{z}_{k,n}$ is the mapping of $\mathbf{x}_{k,n}$ into the RFF space.

B. Online-Fed

At each global iteration n, the server selects a subset of clients $\mathcal{K}_n \in \mathcal{K}$ to participate in the learning. It shares the global model $\mathbf{w}_n$ with the clients in $\mathcal{K}_n$ for them to perform the following local update step:

$$\mathbf{w}_{k,n+1} = \mathbf{w}_n + \mu\epsilon_{k,n}\mathbf{z}_{k,n}, \tag{3}$$

where μ is the learning rate and $\epsilon_{k,n} = y_{k,n} - \mathbf{w}_n^{\mathsf{T}}\mathbf{z}_{k,n}$ is the *a priori* error. The clients proceed to share their new models with the server who aggregates them as:

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{K}_n|}\sum_{k\in\mathcal{K}_n} \mathbf{w}_{k,n+1}. \tag{4}$$

In the particular case where, at each global iteration, $\mathcal{K}_n = \mathcal{K}$, that is, all the clients are always selected to participate, we denote the algorithm Online-FedSGD.

C. Partial Sharing

In partial sharing-based communications, as defined in [21], network clients only share a subset of the model parameters instead of the entire model parameter vector. To this aim, both the server and the clients will select a portion of their model to share prior to communication. This operation is computationally trivial and, therefore, does not induce a delay in the communication, unlike projection-based methods as used in [13], [17]. Further, the portion of the model shared will change at each iteration.

In practice, the selection is performed by multiplying the model parameter vector with a diagonal matrix with diagonal values of either 0 or 1, where the latter locations specify the parameters to share. The server will use matrix $\mathbf{M}_{k,n}$ to select the model parameters to share with client k at iteration n; similarly, client k will use matrix $\mathbf{S}_{k,n}$ to select which local model parameters to share with the server at iteration n.

D. PSO-Fed

The PSO-Fed algorithm in [16] uses partial sharing-based communication to reduce the communication overhead of the Online-Fed algorithm without compromising the accuracy. In contrast, projection-based models see, e.g., [13], [17], require increased computations while suffering some loss in accuracy. In addition, PSO-Fed allows clients not participating in the current global aggregation step to perform local updates. Specifically, if client k receives new data at an iteration n when it is not assisting in the global update, it updates its model as:

$$\mathbf{w}_{k,n+1} = \mathbf{w}_{k,n} + \mu\epsilon_{k,n}\mathbf{z}_{k,n}, \tag{5}$$

where $\epsilon_{k,n} = y_{k,n} - \mathbf{w}_{k,n}^{\mathsf{T}}\mathbf{z}_{k,n}$.

By combining scheduling and partial-sharing, the PSO-Fed algorithm can significantly reduce the amount of communication. Furthermore, it converges rapidly thanks to the local update performed independently by the clients, allowing them to refine their model prior to sharing it when selected to participate. The aggregation step at the server is given by:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{1}{|\mathcal{K}_n|}\sum_{k\in\mathcal{K}_n} \mathbf{S}_{k,n}(\mathbf{w}_{k,n} - \mathbf{w}_n). \tag{6}$$

E. Limitations

The aforementioned algorithms offer a considerable cut in communication overhead; however, they do not incorporate knowledge of the network environment or client resources. For example, when performing federated learning in real-time, clients may be unavailable for various reasons, e.g., limited battery power and channel availability. In addition, poor connection or sub-optimal communication channels may delay exchanged messages. Delays can also arise from some clients with limited computational resources that struggle to perform the task in time (so-called straggler-clients). Therefore, for a successful operation in a realistic environment, an online federated learning method needs to handle time-varying client participation and weigh the importance of delayed measurements. To that end, in this work, we modify the PSO-Fed algorithm to handle such an asynchronous setting. As we will see in the following, many choices made for the PSO-Fed algorithm in a perfect FL setting are not the best in the asynchronous setting.

III. Proposed method

A. Asynchronous setting

We consider an asynchronous setting where clients have access to uneven amounts of non-IID data and have various availability, meaning some will participate more often than others in the learning. In addition, clients may become unavailable for several iterations during the learning process. Furthermore, we expect communication from the clients to the server to be subject to delays. The consequence of the introduced delays is that not all updates from clients will arrive at the server simultaneously. An update sent by a client at time $n-l$ and delayed for l iterations will be received by the server at time n. We denote by $\mathcal{K}_{n,l}$ the set of all the clients who sent an update at iteration $n-l$ that reached the server at iteration n. Further, we define the set $\mathcal{K}_n = \sum_{l=0}^{\infty} \mathcal{K}_{n,l}$ of all the clients who sent an update that arrived at the server at iteration n. When receiving a delayed update from a client, a decision must be made whether to use it, as it may be outdated. To improve the learning accuracy of the algorithm, we propose an aggregation mechanism, where weights are given to received local models according to how recent they are.

B. PAO-Fed

In this setting, we chose not to sub-sample the available clients, some of whom might be rarely available. Instead, we will rely on partial-sharing to reduce the communication overhead in the asynchronous FL setting. At iteration n, the server will share a subset of its model with all the available clients. Further, the clients will share a subset of their models to the server, but although this reply is sent at iteration n, it may arrive to the server later. Clients receiving new data perform the local update step (5) if they are not available, allowing them to communicate a refined model later on. Performing this local update is a trivial computation for most devices and does not require communication with the server.

At global iteration n, the server shares a subset of its model, $\mathbf{w}_n$, to all the available clients, the selection matrices $\mathbf{M}_{k,n}$ dictates which subset goes to which client. Each available client k receives $\mathbf{M}_{k,n}\mathbf{w}_n$ and updates its local model as

$$\mathbf{w}_{k,n+1} = \mathbf{M}_{k,n}\mathbf{w}_n + (\mathbf{I} - \mathbf{M}_{k,n})\mathbf{w}_{k,n} + \mu e_{k,n}\mathbf{z}_{k,n}, \quad (7)$$

where the error $e_{k,n}$ is given by

$$e_{k,n} = y_{k,n} - (\mathbf{M}_{k,n}\mathbf{w}_n + (\mathbf{I} - \mathbf{M}_{k,n})\mathbf{w}_{k,n})^{\mathrm{T}}\mathbf{z}_{k,n}. \quad (8)$$

Then, the available clients communicate a portion of their updated local model, i.e., $\mathbf{S}_{k,n+1}\mathbf{w}_{k,n+1}$, to the server; this communication may be delayed. At the server, we consider the set of clients $\mathcal{K}_n$ whose updates arrived at the server at time n. We decompose this set according to the number of iterations during which the updates were delayed and update the server model as:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \sum_{l=0}^{\infty} \frac{\alpha_l}{|\mathcal{K}_{n,l}|} \sum_{k\in\mathcal{K}_{n,l}} \mathbf{S}_{k,n-l}(\mathbf{w}_{k,n-l} - \mathbf{w}_n), \quad (9)$$

while omitting any empty set $\mathcal{K}_{n,l}$, where $\alpha_l \in [0,1]$ is the weight given to the updates delayed by l iterations. We denote by l_{max} the maximum delay after which $\alpha_l = 0$. The weight α_0 given to the updates that are not delayed is 1, we will see in the simulation section the importance having $\alpha_l < 1$ for $l > 1$. The resulting algorithm is presented in Algorithm 1.

We note that in the eventuality where several clients in $\mathcal{K}_n$ update the same model parameter, only the most recent updates are considered, the selection matrices of the remaining updates are adjusted accordingly prior to computing (9).

Algorithm 1 PAO-Fed

```
1: Initialization: $\mathbf{w}_0$ and $\mathbf{w}_{k,0}, k \in \mathcal{K}$ set to $\mathbf{0}$
2: Procedure at Local client $k$
3: for global iteration $n = 1, 2, \ldots, N$ do
4:     if Client $k$ receives new data at time $n$ then
5:         if $k$ is available then
6:             Receive $\mathbf{M}_{k,n}\mathbf{w}_n$ from the server.
7:             Compute $\mathbf{w}_{k,n+1}$ as in (7) .
8:             Share $\mathbf{S}_{k,n+1}\mathbf{w}_{k,n+1}$ to the server.
9:         else
10:            Update $\mathbf{w}_k$ as in (5).
11:        end if
12:    end if
13: end for
14: Procedure at Central Server
15: for global iteration $n = 1, 2, \ldots, N$ do
16:     Receive client updates from subset $\mathcal{K}_n \subset \mathcal{K}$.
17:     Share $\mathbf{M}_{k,n}\mathbf{w}_n$ with the available clients.
18:     Compute $\mathbf{w}_{n+1}$ as in (9).
19: end for
```

IV. Convergence analysis

First, we present the global update expression of the algorithm in matrix form. Similar to [24], before proceeding to the analysis, we define the extended model vector $\mathbf{w}_{e,n}$, $\mathbf{A}_{e,n}$ and $\mathbf{Z}_{e,n}$ as

$$\begin{aligned}
\mathbf{w}_{e,n} &= \mathrm{col}\{\mathbf{w}_n, \mathbf{w}_{1,n}, \ldots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n} \ldots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n-1}, \\
&\qquad \ldots, \mathbf{w}_{K,n-1}, \ldots, \mathbf{w}_{1,n-l_{\text{max}}}, \ldots, \mathbf{w}_{K,n-l_{\text{max}}}\}, \\
\mathbf{A}_{e,n} &= \mathrm{blockdiag}\{\mathbf{A}_n, \mathbf{I}_{DK}, \ldots, \mathbf{I}_{DK}\}, \\
\mathbf{Z}_{e,n} &= \mathrm{blockdiag}\{\mathbf{Z}_n, \mathbf{0}_{DK\times K}, \ldots, \mathbf{0}_{DK\times K}\}, \quad (10)
\end{aligned}$$

with

$$\mathbf{A}_n = \begin{bmatrix} \mathbf{I} & \mathbf{0}_{D\times 1} & \cdots & \mathbf{0}_{D\times 1} \\ a_{1,n}\mathbf{M}_{1,n} & \mathbf{I} - a_{1,n}\mathbf{M}_{1,n} & & \vdots \\ \vdots & \mathbf{0}_{D\times 1} & \ddots & \mathbf{0}_{D\times 1} \\ a_{K,n}\mathbf{M}_{K,n} & \vdots & & \mathbf{I} - a_{K,n}\mathbf{M}_{K,n} \end{bmatrix},$$

$$\mathbf{Z}_n = \text{blockdiag}\{\mathbf{0}_{D\times 1}, \mathbf{z}_{1,n}, \ldots, \mathbf{z}_{K,n}\}, \tag{11}$$

where $a_{k,n} = 1$ if $k \in \mathcal{K}_n$ and 0 otherwise, col$\{\cdot\}$ and blockdiag$\{\cdot\}$ represent column-wise stacking operator and block diagonalization operator, respectively. We can now, assuming that the representation in the RFF space is optimal, express the extended observation vector col$\{0, y_{1,n}, y_{2,n}, \ldots, y_{K,n}, \mathbf{0}_{K\times 1}, \ldots, \mathbf{0}_{K\times 1}\}$ as

$$\mathbf{y}_{e,n} = \mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{w}_e^* + \boldsymbol{\eta}_{e,n}, \tag{12}$$

where $\mathbf{w}_e^* = \mathbf{1}_{(K+1)l_{\text{max}}+1} \otimes \mathbf{w}^*$ and $\boldsymbol{\eta}_{e,n} = \text{col}\{0, \eta_{1,n}, \eta_{2,n}, \ldots, \eta_{K,n}, \mathbf{0}_{K\times 1}, \ldots, \mathbf{0}_{K\times 1}\}$. Further, we can express the extended estimation error vector as

$$\mathbf{e}_{e,n} = \mathbf{y}_{e,n} - \mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{A}_{e,n}\mathbf{w}_{e,n}. \tag{13}$$

We can then express the global recursion for $\mathbf{w}_{e,n+1}$ as

$$\mathbf{w}_{e,n+1} = \mathbf{B}_{e,n}(\mathbf{A}_{e,n}\mathbf{w}_{e,n} + \mu\mathbf{Z}_{e,n}\mathbf{e}_{e,n}), \tag{14}$$

$$\mathbf{B}_{e,n} = \begin{bmatrix} \mathbf{B}_n & \mathbf{B}_{0,n} & \mathbf{0}_{D\times DK} & \mathbf{B}_{1,n} & \cdots & \mathbf{B}_{l_{\text{max}},n} \\ \mathbf{0}_{D\times 1} & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \cdots & \mathbf{0}_{DK} \\ \vdots & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \cdots & \mathbf{0}_{DK} \\ \vdots & \mathbf{0}_{DK} & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \mathbf{0}_{DK} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \mathbf{0}_{DK} \\ \mathbf{0}_{D\times 1} & \mathbf{0}_{DK} & \cdots & \mathbf{0}_{DK} & \mathbf{I}_{DK} & \mathbf{0}_{DK} \end{bmatrix}$$

$$\mathbf{B}_n = \mathbf{I} - \sum_{l=0}^{l_{\text{max}}} \alpha_l \sum_{k\in\mathcal{K}_{n,l}} \frac{a_{k,n,l}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{k,n+1-l}$$

$$\mathbf{B}_{l,n} = [\frac{\alpha_l a_{1,n,l}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{1,n+1-l}, \cdots, \frac{\alpha_l a_{K,n,1}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{K,n+1-l}].$$

where $\mathbf{0}_{DK} = \mathbf{0}_{DK\times DK}$, and $a_{k,n,l} = 1$ if $k \in \mathcal{K}_{n,l}$ and 0 otherwise. We will now prove the mean convergence of the PAO-Fed algorithm under the following assumptions:

Assumption 1: The mapping of the data vectors $\mathbf{z}_{k,n}, k \in \mathcal{K}$ are drawn at each time step from a WSS multivariate random sequence with correlation matrix $\mathbf{R}_k = \mathbb{E}[\mathbf{z}_{k,n}\mathbf{z}_{k,n}^{\mathsf{T}}]$.

Assumption 2: The observation noise $\eta_{k,n}$ is assumed to be white, and independent of all input and output data.

Assumption 3: At each client, the model parameter vector is assumed to be independent of the input data.

Note: All of those assumptions are commonly used in FL literature and are required to prove the following.

Theorem I: Under Assumptions 1–3, $\mathbb{E}[\tilde{\mathbf{w}}_{e,n}]$, with $\tilde{\mathbf{w}}_{e,n} = \mathbf{w}_e^* - \mathbf{w}_{e,n}$, converges to zero if and only if

$$0 < \mu < \frac{2}{\max\limits_{\forall k,i} \lambda_i(\mathbf{R}_k)}. \tag{15}$$

Proof. First, we note that by construction we have $\mathbf{w}_e^* = \mathbf{B}_{e,n}\mathbf{A}_{e,n}\mathbf{w}_e^*$ (as all rows in $\mathbf{B}_{e,n}$ and $\mathbf{A}_{e,n}$ sum to 1). Then, using (14), we can recursively express $\tilde{\mathbf{w}}_{e,n}$:

$$\begin{aligned} \tilde{\mathbf{w}}_{e,n+1} &= \mathbf{w}_e^* - \mathbf{w}_{e,n+1} \\ &= \mathbf{w}_e^* - \mathbf{B}_{e,n}\mathbf{A}_{e,n}\mathbf{w}_{e,n} - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\mathbf{e}_{e,n} \\ &= \mathbf{B}_{e,n}\mathbf{A}_{e,n}\tilde{\mathbf{w}}_{e,n} - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n} \\ &\quad - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^{\mathsf{T}}(\mathbf{w}_e^* - \mathbf{A}_{e,n}\mathbf{w}_{e,n}) \\ &= \mathbf{B}_{e,n}(\mathbf{I} - \mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^{\mathsf{T}})\mathbf{A}_{e,n}\tilde{\mathbf{w}}_{e,n} \\ &\quad - \mu\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n}. \end{aligned} \tag{16}$$

Its expectation $\mathbb{E}[\cdot]$ can be simplified as:

$$\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}] = \mathbb{E}[\mathbf{B}_{e,n}]\mathbb{E}[\mathbf{I} - \mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^{\mathsf{T}}]\mathbb{E}[\mathbf{A}_{e,n}]\mathbb{E}[\tilde{\mathbf{w}}_{e,n}]$$
$$\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}] = \mathbb{E}[\mathbf{B}_{e,n}](\mathbf{I} - \mu\mathbf{R}_e)\mathbb{E}[\mathbf{A}_{e,n}]\mathbb{E}[\tilde{\mathbf{w}}_{e,n}],$$

where $\mathbf{R}_e = \text{blockdiag}\{\mathbf{0}_D, \mathbf{R}_1, \mathbf{R}_1, \cdots, \mathbf{R}_K, \mathbf{0}_{DKl_{\text{max}}}\}$.

Further, we consider the restriction of the following vectors and matrices between the index $D+1$ and $D(K+1)$. We denote the restriction of $\mathbf{x}$ by $\mathbf{x}|_{\text{sel}}$. The block $\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}$ is defined as a linear sequence of order 1 in a normed algebra: $\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}|_{\text{sel}}] = (\mathbf{I} - \mu\mathbf{R}_e|_{\text{sel}})\mathbf{A}_{e,n}|_{\text{sel}}\mathbb{E}[\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}]$. To prove the convergence of $\mathbb{E}[\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}]$ to zero, we use the infinity norm. From the definition of $\mathbf{A}_n$, We have $||\mathbf{A}_n|_{\text{sel}}||_\infty = 1$. Then the convergence condition reduces to $||\mathbf{I} - \mu\mathbf{R}_e|_{\text{sel}}||_\infty < 1$, equivalently, $|1 - \mu\lambda_i(\mathbf{R}_k)| < 1$, $\forall k, i$, where $\lambda_i(\cdot)$ is the i^{th} eigenvalue of the argument matrix. This leads to the convergence condition given by (15) as, if $\mathbb{E}[\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}]$ converges to zero, then, by construction, $\mathbb{E}[\tilde{\mathbf{w}}_{e,n}]$ converges to zero. □

V. Numerical Simulations

A. Simulation Settings

We model the uneven client participation by giving each client $k \in \mathcal{K}$ a probability $p_{k,n}$ to participate in the learning at global iteration n, note that a client can only participate if it received new data at that iteration. The probabilistic nature of the selection enables us to model both the varied availability and potential downtime of the clients. The Bernouilli trial on $p_{k,n}$ dictates if a client is available or not at a given iteration. Further, each communication to the server has a probability $\delta^l, l \in \mathbb{N}$ of being delayed by l global iterations or more; this probability is assumed to be the same for all the clients.

The number of model parameters shared at each step, m, corresponds to the number of nonzero elements in the selection matrices $\mathbf{M}_{k,n}$ and $\mathbf{S}_{k,n}$. If $\forall k, n$, $\mathbf{M}_{k,n} = \mathbf{S}_{k,n}$, the local updates (5) are not being used. In contrast, we make use of the local updates if we set $\forall k, n$, $\mathbf{S}_{k,n} = \text{circshift}(\mathbf{M}_{k,n}, m)$, that is, the subset of model parameters shared is the one that has been through the most local updates since last updation. The operator circshift denotes a circular shift. In addition, we call coordinated partial-sharing the specific case where $\forall n, k \neq k'$, $\mathbf{M}_{k,n} = \mathbf{M}_{k',n}, \mathbf{S}_{k,n} = \mathbf{S}_{k',n}$; otherwise the partial-sharing is called uncoordinated. In the simulations, we implement uncoordinated partial-sharing with $\mathbf{M}_{k,n} = \text{circshift}(\mathbf{M}_{1,n}, mk)$ and $\mathbf{M}_{1,n} = \text{circshift}(\mathbf{M}_{1,0}, mn)$. It is shown in [16] that coordinated outperforms uncoordinated in a perfect FL setting.

We consider an RFF space of dimension $D = 200$ and $K = 256$ clients separated into 4 data groups for which the progressively available training set is of size 500, 1000, 1500, and 2000, respectively. The clients of each data group are further separated into 4 availability groups, dictating their probability to participate at each iteration. We consider 2 different asynchronous settings. In Setting I, the participation probabilities for the availability groups are 0.25, 0.1, 0.025, and 0.005; and each communication to the server will be delayed by more than l global iterations with probability $\delta^l, 0 < l < l_{\text{max}}$, with $\delta = 0.2$ and $l_{\text{max}} = 10$. Setting II models a harsher environment, availability groups are given the probabilities 0.025, 0.01, 0.0025, and 0.0005; and communications to the server have a probability $\delta = 0.4$ to be delayed. Further, delays last for more than l global iterations, l taking the values $10i, 0 \leqslant i \leqslant 6$, with probability $\delta^{\frac{l}{10}}$; l_{max} is set to 60. This notably implies that, in Setting II, delayed updates have a greater probability to arrive after a non-delayed update coming from the same client.

Unless specified otherwise, every PAO-Fed-based algorithm is set to $m = 4$, and, therefore, reduce the communication load of the algorithm by 98%. We consider seven different versions of the PAO-Fed algorithm. In particular, methods whose names start with PAO-Fed-C use coordinated partial-sharing, while those with PAO-Fed-U use uncoordinated partial-sharing. Similarly, the methods whose names end with 0 do not use the local update steps, and those whose names end with 1 make full use of those. All the aforementioned methods are set up with $\alpha_l = 1, 0 \leqslant l \leqslant l_{\text{max}}$, the method titled PAO-Fed-C2 is identical to PAO-Fed-C1 except for the fact that it is set up with $\alpha_l = 0.2^l, 0 \leqslant l \leqslant l_{\text{max}}$.

We evaluate the performance of the algorithms on a test dataset with the mean squared error given at iteration n by:

$$\text{Testing MSE} = \frac{1}{T}||\mathbf{y}_{\text{test}} - \mathbf{Z}_{\text{test}}^{\mathsf{T}}\mathbf{w}_n||_2^2, \tag{17}$$

where $\mathbf{w}_n$ is the server's model parameter of the considered method and T is the size of the test dataset.

B. *Simulation Analysis*

Fig. 1 shows the impact of the choice of the selection matrices on the convergence of the algorithm. We observe that taking advantage of the local updates (5) greatly improve the convergence properties of the algorithm. Moreover, we see that it is best to use uncoordinated partial-sharing in the presence of delays. This contradicts the behavior of partial-sharing FL in the absence of delays, where coordinated partial-sharing outperforms uncoordinated [16]. The reason is that, in coordinated partial-sharing, all agents share the same subset of model parameters at a given global iteration. While in the absence of delays, this ensures that the aggregation (6) represents an average over a good number of clients, in the presence of delays, the latest updates overwrite the previous ones (9).

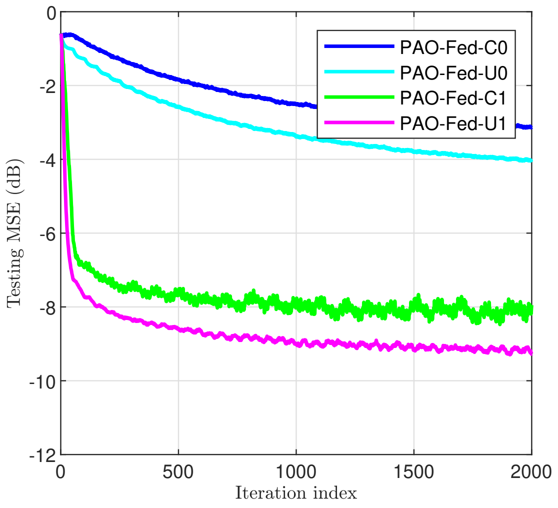

Fig. 1: PAO-Fed performance on Setting I.

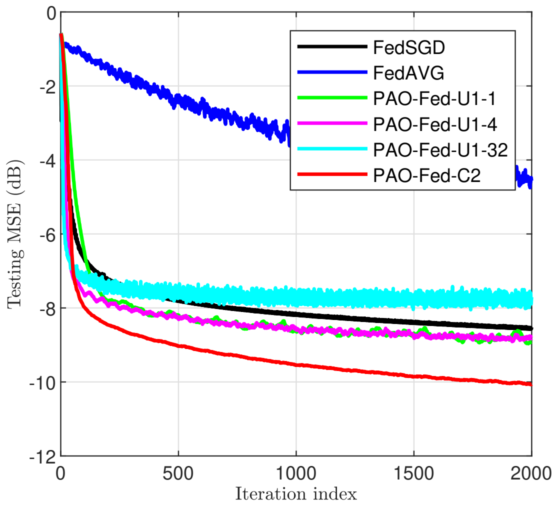

Fig. 2: Choice of m and α_l on Setting I.

Fig. 2 shows the learning curves for Online-FedSGD, Online-Fed, PAO-Fed-U1 for different values of m, and PAO-Fed-C2 that decreases the weight of delayed updates. First, we see that using local updates (5) allows the PAO-Fed algorithm to achieve higher accuracy than Online-FedSGD while using 98% less communication overhead. Second, Online-Fed achieves poor accuracy in this setting; in fact, selecting a subset of clients in the already reduced pool of available participants is not a viable solution to reduce communication overhead in the asynchronous setting. Third, by comparing the PAO-Fed methods using $m = 1$, 4, and 32, we can see that, in the presence of delays, sharing more model parameters at each update does not necessarily increase accuracy as it does in a normal setting [16]. In fact, because it increases the potential negative impact of one single delayed update, it decreases accuracy. Sharing a small number of model parameters ensures better fitting of the server's model parameters to the overall average. Last, we observe that decreasing the weight of the

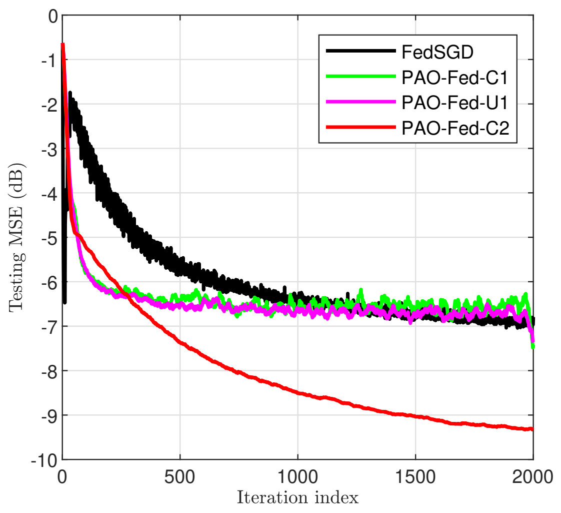

Fig. 3: Performance on Setting II.

received updates that have been delayed increases the accuracy of the algorithm. This is due to the fact that we prevent information coming from a small set of clients to overwrite the server's model parameters learned from a larger number of clients. Instead, we chose to consider the innovation coming from these delayed messages with reduced weight to take into account the lower relevance of these model parameters.

Fig. 3 shows the performance of the algorithms on Setting II, less favorable to learning. We observe that, in this setting, reducing the weight given to the delayed updates gains importance as the accuracy difference between PAO-Fed-C2 and PAO-Fed-U1 increases. This more significant difference is due to the fact that, in addition to overwriting the server's model with parameters from a smaller subset of clients, the delayed update provides some potentially outdated information.

VI. Conclusions

We designed an energy-efficient FL algorithm adapted to a realistic environment. The proposed federated learning algorithm operates with significantly reduced communication requirements and can cope with an unevenly distributed system with poor client availability, channel blockage, and delays. Furthermore, the proposed partial-sharing mechanism reduces the communication overhead and diminishes the negative impact of delayed updates on accuracy. We further propose a weight-decreasing system for delayed updates that improve the performance of the algorithm, especially in an environment with poor participation and long delays.

References

[1] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, Oct. 2016.

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

[3] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial intel. statis.*, pp. 1273–1282, Apr. 2017.

[5] E. Ozfatura, K. Ozfatura, and D. Gündüz, "FedADC: accelerated federated learning with drift control," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2021, pp. 467–472.

[6] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[7] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Nov. 2020.

[8] Z. Lian, W. Wang, and C. Su, "COFEL: Communication-efficient and optimized federated learning with local differential privacy," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[9] Y. Lu, Z. Liu, and Y. Huang, "Parameters compressed mechanism in federated learning for edge computing," in *Proc. IEEE Int. Conf. Cyber Secur. Cloud Comput.*, Jun. 2021, pp. 161–166.

[10] Y. Chen, Z. Chai, Y. Cheng, and H. Rangwala, "Asynchronous federated learning for sensor data with concept drift," *arXiv preprint arXiv:2109.00151*, Sep. 2021.

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

[12] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2020, pp. 15–24.

[13] 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," *arXiv preprint arXiv:2010.05958*, Oct. 2020.

[14] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Dec. 2019.

[15] Z. Wang, Z. Zhang, and J. Wang, "Asynchronous federated learning over wireless communication networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–7.

[16] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning framework for nonlinear regression," *IEEE Int. Conf. Acoust., Speech and Signal Process.*, May 2022.

[17] "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[18] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, Oct. 2016.

[19] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," Jun. 2020, pp. 125–136.

[20] X. Zhang, Y. Liu, J. Liu, A. Argyriou, and Y. Han, "D2D-Assisted Federated Learning in Mobile Edge Computing Networks," Mar. 2021, pp. 1–7.

[21] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Adaptive distributed estimation based on recursive least-squares and partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3510–3522, Jul. 2014.

[22] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, "Efficient KLMS and KRLS algorithms: a random Fourier feature perspective," in *Proc. IEEE Stat. Signal Process. Workshop*, Jun. 2016, pp. 1–5.

[23] A. Rahimi, B. Recht *et al.*, "Random features for large-scale kernel machines." in *Proc. Conf. on Neural Inf. Proc. Syst.*, vol. 3, no. 4, Dec. 2007, pp. 1–5.

[24] V. C. Gogineni, S. P. Talebi, and S. Werner, "Performance of clustered multitask diffusion lms suffering from inter-node communication delays," *IEEE Trans. on Circuits and Syst. II: Express Briefs*, vol. 68, no. 7, pp. 2695–2699, 2021.