Pattern Recognition 148 (2024) 110122

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Reducing communication in federated learning via efficient client sampling

Mónica Ribero, Haris Vikalo *

Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712, United States

ARTICLE INFO

Keywords:
Federated learning
Machine learning
Distributed optimization

ABSTRACT

Federated learning (FL) ameliorates privacy concerns in settings where a central server coordinates learning from data distributed across many clients; rather than sharing the data, the clients train locally and report the models they learn to the server. Aggregation of local models requires communicating massive amounts of information between the clients and the server, consuming network bandwidth. We propose a novel framework for updating the global model in communication-constrained FL systems by requesting input only from the clients with informative updates, and estimating the local updates that are not communicated. Specifically, describing the progression of the model's weights by an Ornstein–Uhlenbeck process allows us to develop sampling strategy for selecting a subset of clients with significant weight updates; model updates of the clients not selected for communication are replaced by their estimates. We test this policy on realistic federated benchmark datasets and show that the proposed framework provides up to 50% reduction in communication while maintaining competitive or achieving superior performance compared to baselines. The proposed method represents a new line of strategies for communication-efficient FL that is orthogonal to the existing user-driven techniques, such as compression, thus complementing rather than aiming to replace those existing methods.

1. Introduction

Federated learning is a framework for training machine learning models in Such settings are common in applications that involve mobile devices, automated vehicles, and Internet-of-Things (IoT) systems [1], as well as in cross-silo applications including healthcare and banking [2]. In FedAvg, the baseline federated learning procedure proposed by [1], a server distributes an initial model to clients who independently update the model on local training data. These updates are aggregated by the server which broadcasts a new global model to the clients and selects a subset of them to start a new round of local training; the procedure is repeated until convergence. Since clients communicate only their models to the server, federated learning provides data security that can be further strengthened and formalized via, e.g., differential privacy mechanisms [3–5].

The number of clients in federated learning systems may be on the order of millions, and the models they locally train and share with the server can be rather large; for example, VGG-16, the widely known neural network for image recognition, has 138M parameters [6], weighing 526MB when represented by 32 bits. Moreover, federated learning systems are often highly dynamic (e.g., mobile devices, IoT), with new users joining and old users continuing to generate new data; such settings may require a large number of training rounds and clients' model uploads. Since transmitting large models requires considerable communication resources, it is desirable to reduce the amount of information that has to be collected by the server [7,8]. This has been explored in the line of research focused on reducing clients' required communication budget by compressing the model [9–14].

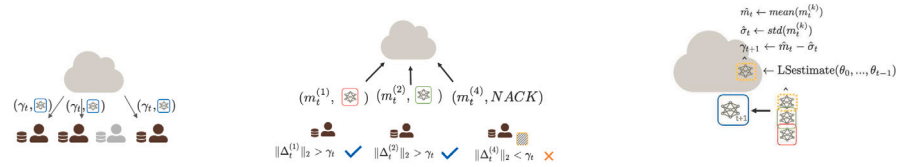
In existing federated learning systems, the number of clients participating in each round of updates (and, therefore, the required communication budget) is typically fixed. Meanwhile, the contributions of many clients in any given round may have limited impact on the global model, especially near convergence. Following this intuition, we propose a novel approach to reducing communication in federated learning by identifying and transmitting only the client updates that are deemed informative, and optimally estimating the absent ones. The contributions of the paper and the proposed methods are summarized below.[1]

Efficient communication with stable convergence. We present a federated learning algorithm that reduces the number of communicating clients without destabilizing the optimization by introducing (1) a novel client selection policy, and (2) a procedure for estimating updates of the clients that are not selected. To this end, we rely on techniques for optimal sampling of stochastic processes and adapt them to the problem of client selection in FL systems. Specifically, we interpret

* Corresponding author.
E-mail addresses: mribero@utexas.edu (M. Ribero), hvikalo@ece.utexas.edu (H. Vikalo).

[1] A preliminary version of this work is posted to arXiv under the title "Communication-efficient federated learning via optimal client sampling", arXiv:2007.15197.

https://doi.org/10.1016/j.patcog.2023.110122
Received 13 June 2023; Received in revised form 29 September 2023; Accepted 14 November 2023
Available online 20 November 2023



(a) Server selects N clients (in the illustration, clients 1, 2 and 4), and broadcasts the current global model and the threshold.

(b) Client k locally updates her model and computes $\Delta_t^{(k)}$. A client that meets the threshold send the update and its norm (in the illustration, clients 1 and 2). The other clients send only the norm of their update (client 4).

(c) In round t, server: (i) computes the mean and standard deviation of the received norms of the updates; (ii) estimates missing clients updates via (3); (iii) updates the global model and computes the threshold for round $t+1$.

Fig. 1. One round (t) of the proposed communication-efficient federated learning system deploying client sampling and estimation.

progression of the global model (i.e., the vector of model parameters during Stochastic Gradient Descent (SGD)) as a discretized sample path of an Ornstein–Uhlenbeck (OU) process. Exploiting the above connection, the server requests a number of clients to locally train and decide whether or not to communicate an update based on its informativeness (e.g., how far is the local path from the steady state). The optimal strategy that minimizes the mean-squared error (MSE) of predicting the sample paths of the process turns out to be a simple threshold on the update's norm and can thus be efficiently implemented at the client's side. We develop a dynamic strategy for selecting the threshold, varying it adaptively during the training. The server utilizes previous global models to efficiently compute unbiased least-squares estimates of the underlying process parameters, and deploys those parameters to predict updates of the clients who choose not to communicate in a given round of training.

Reduced bias and variance gradient estimation. To develop an intuition on why the proposed approach works we present a bound on the convergence of the proposed scheme for strongly convex smooth objective functions. Alternative state-of-the-art (SOTA) methods and proposed heuristics for distributed learning systems [15–18] either replace the missing models with historic values or simply ignore them, adding respectively bias and variance to the solutions, as demonstrated by our experiments. In the remainder of the paper we refer to these two strategies as the Zero and Ignore strategies, respectively. We demonstrate that by estimating the missing client updates via the OU model, one obtains unbiased update estimates and achieves variance reduction that boosts the performance of federated learning.

Extensive experimental verification in realistic settings and comparison with competitive baselines. Efficacy of the proposed methodology is demonstrated in experiments on realistic federated datasets, with highly non-convex objectives, showing that: (1) client sampling schemes that utilize side information (e.g., magnitude of local model updates) generally outperform random uniform sampling; (2) the proposed methods achieve the same rate of convergence and a comparable (or better) model accuracy as the baseline while reducing the communication up to 50%; (3) combining communicated and estimated updates has a major beneficial impact on the convergence speed and stability of training, as reflected by the reduction of bias and variance compared to existing techniques. In particular, we validate the proposed methods on a logistic regression task with synthetic data, and on three benchmarking tests involving real data: a character classification task on a real federated dataset, EMNIST, using a convolutional neural network architecture, a next-character prediction task with an RNN on Shakespeare dataset, and a classification task with ResNet-18 on CIFAR-100. Finally, we demonstrate that the proposed framework is complementary to and can readily be combined with compression-based techniques for reducing communications in federated learning systems.

We illustrate our proposed scheme in Fig. 1. Our approach requires two major modifications of the original Federated Averaging algorithm (FedAvg) [1]. As in the original FedAvg, the server broadcasts the current model and selects N clients to perform an update; unlike FedAvg, the server also communicates threshold γ_t to the selected clients (please see Fig. 1(a)). The selected clients train for E epochs and decide whether to communicate based on how much their update differs from the original model. In particular, if the norm of the difference between the updated and original model exceeds threshold τ_t, the update is transmitted; otherwise, only the norm of the update is sent to the server (see Fig. 1(b)). The server relies on the model history to estimate parameters of the underlying OU process, and uses them to predict missing clients updates (rather than zeroing or ignoring these clients updates). This is discussed in details in Section 3. By combining received and estimated updates, the server generates a new model. Finally, the server uses the collected local model norms to produce a new threshold for the next round; the described procedure repeats until a stopping criterion is met.

Organization. The remainder of the paper is organized as follows. Section 2 introduces key concepts from federated learning (FL), reviews previous efforts on improving communication efficiency in FL systems, and provides background on the Ornstein–Uhlenbeck process. Section 3 develops connection between training with SGD in the federated learning settings and the OU process, and introduces the proposed algorithm. In Section 4 we present numerical results that compare the developed algorithm to related methods, while Section 5 summarizes the paper and suggests future research directions.

2. Background and related work

2.1. Federated learning

Let $\ell(\cdot,\cdot)$ be a loss function. Given a set of K clients, each one with n_k data samples, FedAvg aims to solve the minimization problem

$$\min_{\theta\in\mathbb{R}^d} \sum_{k=1}^{K} p_k F_k(\theta), \tag{1}$$

where $F_k(\theta) = \frac{1}{n_k}\sum_{i=1}^{n_k} \ell(\theta, x_i)$ denotes the loss function at client k and $p_k = \frac{n_k}{\sum_j n_j}$ weighs each client's loss by its dataset size. For simplicity of the analysis we assume balanced datasets, i.e., $p_k = 1/K$. (Note that one can use a simple transformation to extend the results to the unbalanced setting — see, e.g., Section 3.3. in [19]). The FedAvg algorithm requires

a random subset of clients to send their updates to the server after having trained locally for E epochs on mini-batches of size B (see [20] for a comprehensive overview).

2.2. Reducing communication in distributed learning systems

Existing schemes for reducing communication overhead in federated learning systems typically perform compression on the client side and thus require additional computation for encoding and decoding. Deterministic approaches such as low rank approximation, sparsification, subsampling, and quantization [10,13,14,21], as well as randomized approaches including random rotations and stochastic rounding [11], randomized approximation [12], or more recently communication-efficient surrogate likelihood framework [21] can be used to reduce the communication while maintaining high accuracy. Note that these methods may also be leveraged at the server [22].

Orthogonal to compression-based methods, approaches that dismiss updates of some workers have been proposed in the distributed learning literature [15–18]. The authors of [15] propose that only those updates whose magnitude exceeds a certain threshold should be considered significant and therefore communicated. A major drawback of this method is in ignoring updates near the convergence, thus causing stagnation in training. It was recently shown that the model aggregation step is crucial for ensuring rapid convergence of a training process in a federated learning system [23]. Indeed, our experiments demonstrate that ignoring updates leads to unstable behavior, particularly in the case of heterogeneous data. In [16], the authors introduce a thresholding method that relies on a central server for coordination, and propose to replace the update of a client that did not communicate by the client's previous update. This approach requires the server to store for all clients their updates from the previous round. [17] considers a similar approach for the fully decentralized scenario, with a notable difference of setting the threshold according to a schedule $c_t = o(t)$ to ensure convergence. The last two approaches are challenging to implement in federated settings because the number of clients can be on the order of millions, implying it is very likely that in each training round new clients are sampled. Therefore, updates of the clients that did not communicate are replaced by the latest model they received, likely slowing down the training process. Nevertheless, the idea of selecting only the clients with informative updates is worth pursuing since it may lead to significant savings in communications, as demonstrated by the heuristics which impose limits on the upload times of the updates [24]. In [18], the authors analyze the effect of biased user sampling on the convergence of federated learning; we show in our experiments that their loss-based selection with fixed top-k strategies unfortunately saturates relatively fast. Alternatively, recent work proposed feature selection as a communication reduction strategy on vertically partitioned data, i.e., where each client holds certain features about a set of records [25]. Our work focus on the horizontal setting described in Eq. (1).

2.3. Ornstein–Uhlenbeck process

In this section we provide a brief background on the Ornstein–Uhlenbeck process (OU), a stationary Gauss–Markov process which we utilize to statistically model training in a federated learning system.

Definition 1. The OU process $\{\theta_t\}_t$ is described by the stochastic differential equation

$$d\theta_t = \lambda(\mu - \theta_t)dt + \sigma dW_t, \tag{2}$$

where W_t denotes the standard Wiener process. Eq. (2) specifies the process that is drifting towards μ with velocity λ, and has volatility driven by a Brownian motion with variance σ.

Table 1
The notation used in the paper.

Variable	Definition
N	The number of clients to sample in each round
T	The total number of rounds
γ_t	Threshold sent by the server to the users at the beginning of round t
θ_t	Global model at round t
$\theta_{t+1}^{(k)}$	Local model at the end of round t at client k
$\Delta_t^{(k)} := \theta_{t+1}^{(k)} - \theta_t$	Update for client k at round t
$m_t^{(k)} := \|\Delta_t^{(k)}\|_2$	ℓ_2- norm of client k's update at t
m_t	Mean of the update norms at the end of round t
s_t	Standard deviation of the update norms at the end of round t
C_t	Set of clients initially selected to participate in round t

Remark 1. Rate-constrained sampling of stochastic processes has been widely studied in literature, primarily in the context of communications and control [26–33]. In the setting where samples are observed locally (by nodes/clients) but used for estimation only if communicated to the central processor, thresholding the signal magnitude increment is an optimal sampling policy for estimating parameters of an OU process [31–33].

3. Training via client sampling and model update estimation

Here we introduce Adaptive-OU, a federated learning framework formalized as Algorithm 1, inspired by techniques from optimal stochastic process sampling and developed with the goal of reducing communication and improving accuracy. Specifically, we present a strategy where a client transmits its model update to the server only if the norm of the update exceeds a time-varying threshold set by the server, and propose a non-trivial unbiased estimator of the model updates that did not meet the communication threshold. In Theorem 3 we provide a convergence bound for Algorithm 1.

3.1. Notation

For clarity, the notation used in the upcoming sections is summarized in Table 1.

3.2. SGD as an OU process

Recently, studies of SGD via stochastic differential equation models have gained significant attention [34–37]. Consider the loss $\mathcal{L}(\theta; X) = \sum_{i=1}^{N} \ell_i(\theta)$, where X is a dataset with N samples and $\ell_i(\theta)$ denotes the loss function evaluated at point $x_i \in X$, $i = 1, \dots, N$. In gradient descent, $\mathcal{L}(\theta; X)$ is typically minimized by finding in each iteration an approximation of the gradient using a mini-batch $\mathcal{B} \subseteq X$ of the data. In particular, letting $g_i(\theta) = \frac{d}{d\theta}\ell_i(\theta)$ denote the gradient of ℓ_i at θ,

$$\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} g_i(\theta_t).$$

The following observations and assumptions are commonly encountered in literature (see, e.g., [37]).

Observation 1. Note that the gradient at time t is formed as the empirical mean of per-sample gradients of the points in $\mathcal{B}$ that are drawn independently and uniformly at random. Consequently, as noted in [37], the central limit theorem implies that g can be approximated by a normal distribution as $\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} g_i(\theta) \to \mathcal{N}(g(\theta), B(\theta)B(\theta)^T)$, where $g(\theta)$ denotes the full gradient and $B(\theta)B(\theta)^T$ is the corresponding covariance matrix.

Assumption 1. When θ approaches a stationary value, $B(\theta) = B$ is constant [37].

Assumption 2. The iterates θ_t lie in a region where the loss can be approximated by a quadratic form $\mathcal{L}(\theta) = \frac{1}{2}\theta^T A\theta$ (readily justified in the case of smooth loss functions), and the process reaches a quasi-stationary distribution around a local minimum.

Let us consider the discrete process defined as $\Delta\theta_t := \theta_{t+1} - \theta_t = -\frac{\eta}{|B|}\sum_{i\in B} g_i(\theta_t)$. Predicated on the above,

$$\Delta\theta_t =\approx -\eta g(\theta_t) - \sqrt{\frac{\eta}{N}} B\mathcal{N}(0, \eta I).$$

This is precisely the discretized version of the OU process

$$d\theta = -g(\theta)d\tau + \sqrt{\frac{\eta}{N}} BdW = -A\theta d\tau + \sqrt{\frac{\eta}{N}} BdW$$

in the relative proximity of the steady state.

3.3. Optimal sampling and estimation of OU processes in FL settings

When a sampling frequency constraint (i.e., uplink bandwidth) limits the number of updates that could be collected by the server, clients should locally decide when to send an update. The optimal strategy is discussed below.

Lemma 1 (*Proposition 2, [33]*)**.** *Given the observation of an OU process at time t_0, θ_{t_0}, the optimal sampling strategy minimizing the mean-square estimation error is to collect the next sample at time $\tau^* = \inf\{\tau \geq t_0 : |\theta_\tau - \mathbb{E}[\theta_\tau|\theta_{t_0}]| > \gamma_{t_0}\}$, and the optimal estimate of θ_τ, $\tau \in [t_0, \tau^*)$, is $\hat{\theta}_\tau = \mathbb{E}[\theta_\tau|\theta_{t_0}]$. In particular, for the OU process in Eq. (2),*

$$\hat{\theta}_\tau = e^{-\lambda(\tau-t_0)}\theta_{t_0} + (1 - e^{-\lambda(\tau-t_0)})\mu, \tag{3}$$

where λ and μ denote parameters of the OU process in Eq. (2).

To establish the connection to federated learning, we recall the interpretation of SGD as an OU process and note that in each round t of a federated learning procedure, client k "observes" a partial sample path of an OU process that terminates in $\theta_{t+1}^{(k)}$ (i.e., the client records progression of its weights during local training); the sample path starts from point θ_t (i.e., the global model weights at round t) broadcasted by the server at the beginning of the current training round. Let $\Delta_t^{(k)} := \theta_{t+1}^{(k)} - \theta_t$ be the difference between a locally updated (by client k) and the previously broadcasted global model. Invoking the above sampling optimality results, we propose to schedule transmission of updates if $\|\Delta_t^{(k)}\|_2$ exceeds a judiciously selected threshold (see Algorithm 2). If client k decides not to communicate with the server due to the result of a thresholding test, the server may estimate the client's update according to Eq. (3).

3.4. A stable communication-efficient algorithm

We formalize the overall proposed communication-efficient federated learning framework as Algorithm 1. In brief, N clients are selected at the beginning of round t. The server broadcasts the model parameters θ_t, and the selected clients locally performs SGD with mini-batches of size B for E epochs. After comparing the norm of the local model update to a threshold, each client *locally* decides whether to communicate its updates or not, and transmits either the model updates $\theta_{t+1}^{(k)}$ or a negative-acknowledgment message (NACK), respectively. Finally, the server incorporates either the update received from client k or its estimate formed using Eq. (3). In practice, the process parameters λ and μ are unknown and would need to be estimated themselves; to this end, we replace Eq. (3) by

$$\hat{\theta}_{t+1}^{(k)} = \begin{cases} \theta_{t+1}^{(k)}, & \text{if an update was sent,} \\ \hat{a}_t\theta_t + \hat{b}_t, & \text{otherwise,} \end{cases} \tag{4}$$

where $\hat{a}_t$ and $\hat{b}_t$ are inferred from $\theta_0, \ldots, \theta_t$, previously aggregated at the server, by relying on a least-squares estimation procedure described in Section 3.6, producing unbiased estimates of these parameters. Rather than storing full past models, the server can update $\hat{a}_t$ and $\hat{b}_t$ via recursively computed rolling sums involving previous model parameters (see Section 4.5 for details), thus enabling efficient evaluation of $\hat{\theta}_{t+1}$. We formalize this procedure as Algorithm 3.

Note that the server's computationally cheap alternative to estimating a missing update is to reuse the client's model from the previous round, i.e., to set $\hat{\theta}_{t+1}^{(k)} = \theta_t$ (which we refer to as ZERO strategy), or to simply ignore the client (IGNORE strategy); as reported in Section 4, these alternatives consistently underperform our proposed policy. Finally, the server computes a new model according to $\theta_{t+1} = \frac{1}{N}\sum_{k=1}^N \hat{\theta}_{t+1}^{(k)}$ (line 13 of the pseudo-code), updates the threshold and the rolling sums used for parameter estimation, and proceeds to the next round of training.

Algorithm 1 Communication-Efficient FEDAVG (ADAPTIVE-OU)

1: **Input:** K clients, number of gradient steps E, learning rate η, number of training rounds T.
2: Initialize θ_0, prediction $\hat{\theta}_1$ and rolling sums $S = [S_x, S_{xx}, S_y, S_{yy}, S_{xy}]$ at the server
3: **for** $t = 1, \ldots, T$ **do**
4: $C_t \leftarrow$ random set of N clients.
5: **for** $k \in C_t$ clients **do**
6: $M_t^{(k)} \leftarrow$ CLIENTUPDATE$(k, \theta_t, E, \gamma_t)$
7: **end for**
8: **for** $k \in C_t$ Server **do**
9: $\hat{\theta}_{t+1}^{(k)} \leftarrow$ SERVERESTIMATE$(M_t^{(k)}, \hat{\theta}_{t+1})$ (Equation (5))
10: **end for**
11: Server update:
12: $\gamma_{t+1} = \text{mean}(m_t^{(k)} - \text{std}(m_t^{(k)}))$
13: $\theta_{t+1} \leftarrow \frac{1}{N}\sum_{k=1}^N \hat{\theta}_{t+1}^{(k)}$
14: Update S with θ_{t+1} (Equation (8))
15: Predict $\hat{\theta}_{t+2}$ (Equation (10))
16: **end for**
17: **Return** Global model θ_T

Algorithm 2 CLIENTUPDATE at client k

Input: Initial global model θ_t, threshold γ_t, number of gradient steps E.
initialize $\theta_{t+1}^{(k)} \leftarrow \theta_t$
for $i = 1, \ldots, E$ **do**
$\theta_{t+1}^{(k)} \leftarrow \theta_{t+1}^{(k)} - \eta_t \nabla\ell(\theta_{t+1}^{(k)}, x_i)$
end for
$\Delta_t^{(k)} = \theta_{t+1}^{(k)} - \theta_{t+1}^{(k)}$
$m_t^k = \|\Delta_t^{(k)}\|_2$
if $m_t > \gamma_t$ **then**
$M_t^{(k)} \leftarrow (\Delta_t^{(k)}, m_t^{(k)})$
else
$M_t^{(k)} \leftarrow (NACK, m_t^{(k)})$
end if
Return $M_t^{(k)}$ to server

Algorithm 3 SERVERESTIMATE for client k

Input: The message from client k at time t, $M_t^{(k)}$, estimate $\hat{\theta}_{t+1}$

$$\hat{\theta}_{t+1}^{(k)} = \begin{cases} \theta_{t+1}^{(k)}, & \text{if received updates} \\ \hat{\theta}_{t+1}, & \text{otherwise.} \end{cases} \tag{5}$$

Return: The update $\hat{\theta}_{t+1}^{(k)}$ of client k to use for the next global model.

3.5. Adaptive threshold selection

The policy of deciding whether or not to communicate based on comparing $\|\Delta_t^{(k)}\|_2$ to a threshold γ_t aims to reduce communication without incurring significant accuracy loss compared to a baseline. Since the norm of the gradients is expected to decrease as the training progresses, a fixed threshold may have detrimental effect on the learning process as it starts converging and approaches the minimum. We empirically explore this point in Section 4.3.7; in particular, we propose a strategy for adaptive modification of γ_t based on the magnitudes of the updates of participating clients. At $t = 0$, clients receive an initial threshold $\gamma_0 = 0$, implying that everyone transmits in the first round. In the following rounds, clients transmit either their model updates and the corresponding update norms $m_t^{(k)}$, or a NACK message along with the norm of their update. At the end of round t, the server estimates the mean of the update norms, m_t, their standard deviation, s_t, and sets threshold for the next round as $\gamma_{t+1} = m_t - s_t$. As stated in Lemma 2, the threshold should be set to a small value in order to reduce communication yet remain sufficiently large to allow collection of model updates and thus reduce the variance of the updates.

3.6. Estimating parameters of the OU process

Various techniques for estimating parameters of the OU process from the observations of its sample path have been proposed in literature, including least-squares, maximum likelihood [38] and Jackknife method [39]. For convenience, we here summarize the computationally efficient least-squares solution. First, note that by discretizing the continuous OU process we obtain

$$\theta_{t+1} = e^{-\lambda\Delta t}\theta_t + (1 - e^{-\lambda\Delta t})\mu + \sigma\sqrt{\frac{1 - e^{-2\lambda\Delta t}}{2\lambda}}\Delta W_t, \tag{6}$$

where Δt denotes the discretization (sampling) period and ΔW_t are i.i.d. increments of the Wiener process. This leads to a linear measurement model

$$\theta_{t+1} = a\theta_t + b + \epsilon_t, \tag{7}$$

where ϵ_t denotes i.i.d. noise and where

$$a = e^{-\lambda\Delta t}, \quad b = \mu(1 - e^{\lambda\Delta t}), \quad \text{and} \quad std(\epsilon_t) = \sigma\sqrt{\frac{(1 - e^{-2\lambda\Delta})}{2\lambda}}.$$

To enable efficient online (i.e., recursive) estimation of the relevant process parameters, let us define

$$\begin{aligned} S_{x,t} &= \sum_{i=1}^{t}\theta_{i-1}, \quad S_{y,t} = \sum_{i=1}^{t}\theta_i, \\ S_{xx,t} &= \sum_{i=1}^{t}\theta_{i-1}^2, \quad S_{yy,t} = \sum_{i=1}^{t}\theta_i^2, \\ S_{xy,t} &= \sum_{i=1}^{t}\theta_{i-1}\theta_i, \end{aligned} \tag{8}$$

where $\theta_1, \theta_2, \ldots, \theta_t$ denote samples of the OU process. It is straightforward to show that the least-square estimates of a_t, b_t and $std(\epsilon_t)$ given $\theta_1, \theta_2, \ldots \theta_t$ can be found as

$$\hat{a}_t = \frac{tS_{xy,t} - S_{x,t}S_{y,t}}{tS_{xx,t}S_{xx,t}}, \quad \hat{b}_t = \frac{S_{y,t} - \hat{a}_t S_{x,t}}{t}, \tag{9}$$

and

$$\widehat{std(\epsilon_t)} = \sqrt{\frac{tS_{yy,t} - S_{y,t}^2 - \hat{a}_t(tS_{xy,t} - S_{x,t}S_{y,t})}{t(t-1)}}.$$

Finally, the next value of the sample path, θ_{t+1}, is predicted as

$$\hat{\theta}_{t+1} = \hat{a}_t\theta_t + \hat{b}_t. \tag{10}$$

3.7. Convergence of Algorithm 1

Below we state Lemma 2 which analytically justifies the proposed policy and discuss convergence of the algorithm's output to the minimizer of Eq. (1) in Theorem 3. We defer a proof sketch to Section 3.9. For simplicity, we assume balanced datasets and that N clients are sampled (with replacement) from the set of K clients, each with probability $p_k = 1/K$. We compare the convergence of three estimation strategies which differ according to how they handle missing updates: (1) ignoring the missing updates (IGNORE); (2) replacing the missing updates $\Delta\theta_{t+1}^{(k)} = \theta_{t+1}^{(k)} - \theta_t$ by zero, i.e., assuming that the local model is equal to the previous global model (ZERO); and (3) our proposed ADAPTIVE-OU estimation strategy that controls which updates are missing by deploying the proposed threshold strategy and estimating them at the server via Eq. (5).

We start by analyzing the variance induced by our scheme and then state its convergence rate.

Lemma 2. *Assume client gradients in Eq. (1) are uniformly bounded so it holds that $\|\nabla F_k(\theta, x)\|^2 \leq G^2$, and execute* CLIENTOPT *in Algorithm 1 with E steps of SGD. Then, the variance of estimating a model at time t via the* OU *strategy that selects N and receives a fraction of $H(\gamma_t)$ clients (parametrized by the threshold γ_t) satisfies*

$$C_{\text{OU}}^t \leq H(\gamma_t)\frac{E^2G^2}{N} + (1 - H(\gamma_t))\frac{\gamma_t^2}{N}. \tag{11}$$

The first term in Eq. (11) captures the variance over clients which communicated their updates while the second term represents the variance over the remaining clients, i.e., those that did not meet the communication threshold. Consequently, compared to [19], the proposed scheme achieves smaller variance for the same communication budget; this follows from the fact that the variance of randomly sampling N clients and averaging models received from $H(\gamma_t)\cdot N$ of them is given by $C_{\text{IGNORE}}^t = \frac{E^2G^2}{H(\gamma_t)\cdot N}$. Since $H(\gamma_t) < 1$, it holds that $C_{\text{OU}}^t < C_{\text{IGNORE}}^t$.

Theorem 3. *Assuming (i) L-smooth and μ−strongly convex local loss functions, (ii) bounded variance σ_L of local stochastic gradients, $\|\nabla\ell(\theta, x) - F_k(\theta)\|^2 \leq \sigma_L^2$ for $k = 1, \ldots, K$, and (iii) uniformly bounded client gradients, $\|\nabla F_k(\theta, x)\|^2 \leq G^2$, then after T iterations the model produced by Algorithm 1 satisfies*

$$\mathbb{E}(F_T) - F^* = O\left(\frac{\sigma_L^2/K + \Gamma + E^2G^2 + C_{\text{OU}}}{T} + B\right), \tag{12}$$

where σ_L^2/K captures variability of mini-batch gradients, $\Gamma = F^* - \sum_k p_k F_k^*$ reflects data heterogeneity (F_k^* denotes the minimum value of F_k), E^2G^2 represents divergence of the clients' models from the average, $C_{\text{OU}} = \frac{1}{T}\sum_{t=1}^{T} C_{\text{OU}}^t = O(E^2G^2)$ is the average variance of the client sampling procedure defined in Lemma 2, and B is a bias term capturing how far F is from its quadratic approximation.

Remark 2. Unlike the result above, the bound in [19] does not have a bias term. However, client sampling variance of the approach in [19] can be relatively much larger; as noted in Lemma 1, $C_{OU}^t < C_{\text{IGNORE}}^t$. Our experiments demonstrate that the increased variance can in practice be more detrimental to accuracy than the bias term.

3.8. Computational complexity

Limited power and memory of users' devices in federated learning systems require rational use of computational resources. The time complexity of our proposed framework is essentially as same as the complexity of the traditional federated averaging algorithm. In particular, the proposed scheme does not make significant contribution to the clients' computational burden since the only additional operations include: (i) computing the norm of the update, and (ii) comparing the

computed norm to a threshold. The former is often already computed in federated learning systems in order to enable clipping large updates or to bound gradient sensitivity and guarantee differential privacy; the latter is negligible. On the server side, our framework increases memory consumption by a constant factor proportional to the number of parameters d in order to store the five arrays in S (see Algorithm 1), maintaining $O(d)$ memory consumption. Moreover, an additional computational overhead is needed at the server to predict missing model updates via recursive least squares. This step is performed independently for each weight, and only requires element-wise sums and multiplications.

3.9. Sketch of the proof of Theorem 3

Proof. Let θ^* be the true minimizer of Eq. (1), and $\overline{\mathbf{v}}_{t+1}$ be the model at round t assuming all clients participate in training, i.e.,

$$\overline{\mathbf{v}}_{t+1} = \sum_{k=1}^{K} \theta_{t+1}^{(k)}.$$

Below we utilize standard distributed optimization techniques [40] and follow a line of arguments similar to the convergence proof for FedAvg in Theorem 2 of [19]. First, we note that

$$\|\theta_{t+1} - \theta^*\|^2 = \underbrace{\|\theta_{t+1} - \overline{\mathbf{v}}_{t+1}\|^2}_{A_1} + \underbrace{\|\overline{\mathbf{v}}_{t+1} - \theta^*\|^2}_{A_2} + \underbrace{2\langle \theta_{t+1} - \overline{\mathbf{v}}_{t+1}, \theta_{t+1} - \theta^* \rangle}_{b_{t+1}}. \tag{13}$$

In Theorem 2 of [19], b_{t+1} vanishes due to unbiasedness of the estimator. Here, let b_t denote the bias introduced at round t. Then

$$\mathbb{E}\|\theta_{t+1} - \theta^*\|^2 \leq (1 - \eta_t \mu)\mathbb{E}\|\theta_t - \theta^*\|^2 + \eta_t^2 (C_{OU}^t + \mathrm{Var}) + b_t, \tag{14}$$

where

$$C_{OU}^t = \mathbb{E}\left[\|\overline{\mathbf{v}}_{t+1} - \theta_{t+1}\|^2\right], \quad \mathrm{Var} = \frac{\sigma_L^2}{K} + 6\Gamma + 8(E-1)^2 G^2,$$

and we use Lemma 2 and Lemma 1 in [19] to upper bound A_1 and A_2 in (13), respectively. Letting $\eta_t = \beta/(t+\gamma)$ for some $\beta > 1/\mu$ and $\gamma > 0$, it can be shown by induction that

$$\mathbb{E}[\|\theta_{t+1} - \theta^*\|^2] \leq \frac{\upsilon}{\gamma + t} + \sum_{i=1}^{t-1} \left(1 - \frac{\beta\mu}{t+\gamma}\right) b_i + b_t, \tag{15}$$

where $\upsilon = \max\left\{\frac{\beta^2(\mathrm{Var}+C_{OU})}{\beta\mu - 1}, (\gamma+1)\|\theta_1 - \theta^*\|\right\}$. In particular, for $t = 1$ this inequality holds due to the definition of υ. Now, assume Eq. (15) holds for t. We know from Eq. (14) that

$$\mathbb{E}\|\theta_{t+1} - \theta^*\|^2 \leq (1 - \eta_t \mu)\mathbb{E}\|\theta_t - \theta^*\|^2 + \eta_t^2 (C_{OU}^t + \mathrm{Var}) + b_t.$$

The inductive assumption $\mathbb{E}[\|\theta_t - \theta^*\|^2] \leq \frac{\upsilon}{\gamma+t} + \sum_{i=1}^{t-1}\left(1 - \frac{\beta\mu}{t+\gamma}\right) b_i$ implies

$$\mathbb{E}\|\theta_{t+1} - \theta^*\|^2 \leq (1 - \eta_t\mu)\left(\frac{\upsilon}{\gamma+t} + \sum_{i=1}^{t-1}\left(1 - \frac{\beta\mu}{t+\gamma}\right) b_i\right) + \eta_t^2 (C_{OU}^t + \mathrm{Var}) + b_t.$$

Now, since $\eta_t = \frac{\beta}{t+\gamma}$, it must be that

$$\mathbb{E}\|\theta_{t+1} - \theta^*\|^2 \leq \left(1 - \frac{\mu\beta}{t+\gamma}\right)\left(\frac{\upsilon}{\gamma+t} + \sum_{i=1}^{t-1}\left(1 - \frac{\beta\mu}{t+\gamma}\right) b_i\right) + \frac{\beta^2}{(t+\gamma)^2}(C_{OU}^t + \mathrm{Var}) + b_t. \tag{16}$$

Multiplying the second to last term by $\frac{\beta\mu - 1}{\beta\mu - 1}$ yields

$$\frac{\beta^2}{(t+\gamma)^2}(C_{OU}^t + \mathrm{Var}) = \frac{\beta^2(\beta\mu - 1)}{(t+\gamma)^2(\beta\mu - 1)}(C_{OU}^t + \mathrm{Var}) = \frac{\upsilon(\beta\mu - 1)}{(t+\gamma)^2}.$$

By substituting this term in Eq. (16) we obtain

$$\begin{aligned}\mathbb{E}\|\theta_{t+1} - \theta^*\|^2 &\leq \left(1 - \frac{\mu\beta}{t+\gamma}\right)\left(\frac{\upsilon}{\gamma+t} + \sum_{i=1}^{t-1}\left(1 - \frac{\beta\mu}{t+\gamma}\right) b_i\right) + \frac{\upsilon(\beta\mu - 1)}{(t+\gamma)^2} + b_t \\ &= \frac{t+\gamma-\mu\beta}{(t+\gamma)^2}\upsilon + \left(1 - \frac{\mu\beta}{t+\gamma}\right)\sum_{i=1}^{t-1}\left(1 - \frac{\beta\mu}{t+\gamma}\right) b_i + \frac{\upsilon(\beta\mu - 1)}{(t+\gamma)^2} + b_t.\end{aligned}$$

Reorganizing the terms on the right-hand side leads to

$$\mathbb{E}\|\theta_{t+1} - \theta^*\|^2 \leq \frac{t+\gamma-1}{(t+\gamma)^2}\upsilon + \left(1 - \frac{\mu\beta}{t+\gamma}\right)\sum_{i=1}^{t-1}\left(1 - \frac{\beta\mu}{t+\gamma}\right) b_i + b_t. \tag{17}$$

Now, note that $1 - \frac{\beta\mu}{t+\gamma} \leq 1$ implies

$$\sum_{i=1}^{t-1}\left(1 - \frac{\beta\mu}{t+\gamma}\right) b_i \leq \sum_{i=1}^{t-1} b_i.$$

Observing that $\frac{t+\gamma-1}{t+\gamma} \leq 1$ leads to the desired result,

$$\mathbb{E}\|\theta_{t+1} - \theta^*\|^2 \leq \frac{\upsilon}{t+\gamma} + \left(1 - \frac{\mu\beta}{t+\gamma}\right)\sum_{i=1}^{t-1} b_i + b_t. \tag{18}$$

Now, it follows from [19] that the first term on the right-hand side (RHS) of Eq. (17) bounds the first two terms on the RHS of (14); the bound on the remaining term in (17) is due to unfolding the recurrence on b_t. Moreover, due to strong convexity

$$\mathbb{E}[F(\theta_t)] - F(\theta^*) \leq \frac{L}{2}\mathbb{E}[\|\theta_t - \theta^*\|^2].$$

Using this in Eq. (17),

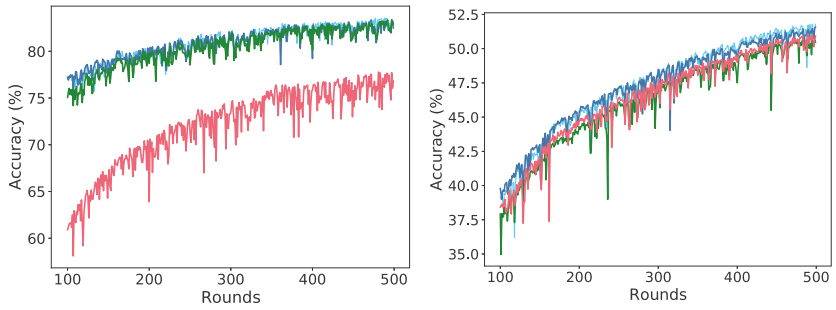
$$\mathbb{E}[F(\theta_T)] - F(\theta^*) = O\left(\frac{L}{\mu(\gamma + T - 1)}\left(\frac{\mathrm{Var} + C_{OU}}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}\|\theta_1 - \theta^*\| + \sum_{t=1}^{T}\left(1 - \frac{\beta\mu}{T+\gamma}\right) b_t + b_T\right)\right).$$

Finally, Theorem 3 follows by letting $B = \frac{1}{T}\left(\sum_{t=1}^{T}\left(1 - \frac{\beta\mu}{t+\gamma}\right) b_t + b_T\right)$. For arbitrary functions, B could be unbounded and grow asymptotically. However, the error of approximating a smooth, strongly convex function in a bounded domain is at worst constant because such functions are readily upper and lower bounded by quadratic functions; thus, $b_t = O(1)$ and $B = O(1)$. In practice, the quadratic approximation improves closer to the optimum, meaning that $\|b_t\|$ decreases over time; e.g., if $b_t = O(\frac{1}{T})$, it holds that $B = O\left(\frac{\log T}{T}\right)$, and the bias vanishes. □

4. Experiments

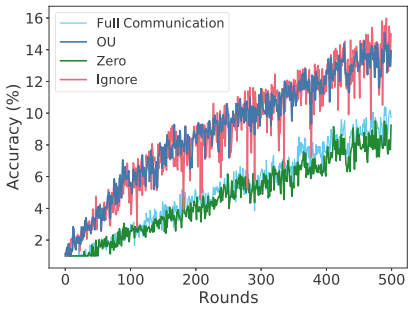
We present extensive experiments that demonstrate the performance of our proposed algorithm on several datasets and for various realistic settings and models. In particular, we benchmark the proposed client selection strategy on four different datasets – a synthetic dataset, EMNIST with 62 categories, Shakespeare, and CIFAR100 – with respective models: (1) logistic regression; (2) a convolutional neural network; (3) a recurrent neural network for a next character prediction task; and (4) ResNet-18. For each task we use the optimal learning rate found in [41] and the same batch size (specific values are provided in the appendix). Our codes are publicly available[2] and were implemented using the Tensorflow-Federated API [42].

[2] https://github.com/mriberodiaz/selection_ou_weights.git.



(a) Test accuracy of EMNIST CNN.

(b) Test accuracy of Shakespeare RNN.



(c) Test accuracy of CIFAR100 Resnet-18.

Fig. 2. The Adaptive-OU strategy is consistently more accurate than the competing methods.

Table 2
Datasets.

Dataset	Users	Samples
EMNIST	3.4 K	60 K
CIFAR100	500	50 K
Shakespeare	715	16 K

4.1. Federated datasets

For the federated learning experiments with homogeneous data and convex learning objective, we use a synthetic dataset and consider a logistic regression task. We synthesize this data by generating 10^4 samples $X_i \in \mathbb{R}^{100} \sim \mathcal{N}(0, I_{100})$. Moreover, we generate $\beta \sim \mathcal{N}(0, I_{100})$ and, finally, set labels $y_i = round(X_i^T \beta)$. The samples are split evenly among 100 clients.

For a more realistic task, we rely on the EMNIST dataset [43], a reprocessed version of the original MNIST dataset with 62 categories: each image is a character linked to its original writer, providing a natural non-i.i.d. distribution and thus allowing emulation of a federated learning setting. This dataset consists of images attributed to 3843 users. For the task of interest, character recognition, we train and test a convolutional neural network (CNN).

To investigate a language modeling task under data heterogeneity, we use the Shakespeare dataset [1], a language modeling dataset with 725 clients, each one a different speaking role in each play from the collective works of William Shakespeare. Each client's dataset is split into training and validation sets. We train a recurrent neural network (RNN) with just under 1M parameters for the next character prediction task.

Finally, for CIFAR-100 we use the partition introduced by [41] that applies Latent Dirichlet Allocation to produce a realistic heterogeneous distribution. We train ResNet-18, replacing batch with group normalization, a modification that has shown improvements in federated settings [44].

Size of the datasets is summarized in the Table 2.

4.2. Benchmarks

We test our adaptive thresholding for communication reduction and compare the results to those of the following baselines: (i) a non-restricted communication scheme where all clients communicate their model updates to the server; (ii) Power-of-Choice (PoC), the concurrent approach proposed by [18]: there, clients are sorted in decreasing order according to their loss at the beginning of a round, and only the top-k are used for training. We present results for different values of the hyperparameter k.

Next, we turn our attention to the strategies for dealing with missing updates and consider two previously mentioned alternatives found in literature: (i) zero strategy [16,17], where the server replaces missing updates with zeros; and (ii) ignore strategy [15], where the server averages only the updates it received.

4.3. Results

We investigate the following aspects of the client sampling problem: (i) communication efficiency vs. accuracy achieved by a client selection strategy; and (ii) the effects of different approaches to dealing with the clients that did not communicate their model updates to the server.

In the experiments with EMNIST, Shakespeare, and CIFAR-100 datasets we fix hyperparameters as in [41], selecting $N = 10$ clients at random in each round. Models are trained for 500 rounds.

4.3.1. Accuracy

In Fig. 2, we plot the test accuracy progression during training for all combinations of adaptive selection and the three estimation strategies (OU, zero, and ignore) on EMNIST, Shakespeare, and CIFAR100 datasets. We further report the accuracy averaged over the last 100 rounds and its standard deviation in Table 3. In the experiments on EMNIST and Shakespeare, the Adaptive-OU strategy achieves the best accuracy; for the task involving Shakespeare dataset, Adaptive-OU even achieves better accuracy than the full-communication baseline scheme.

The results reveal that thresholding by magnitude reduces communication without sacrificing performance only if paired with a suitable

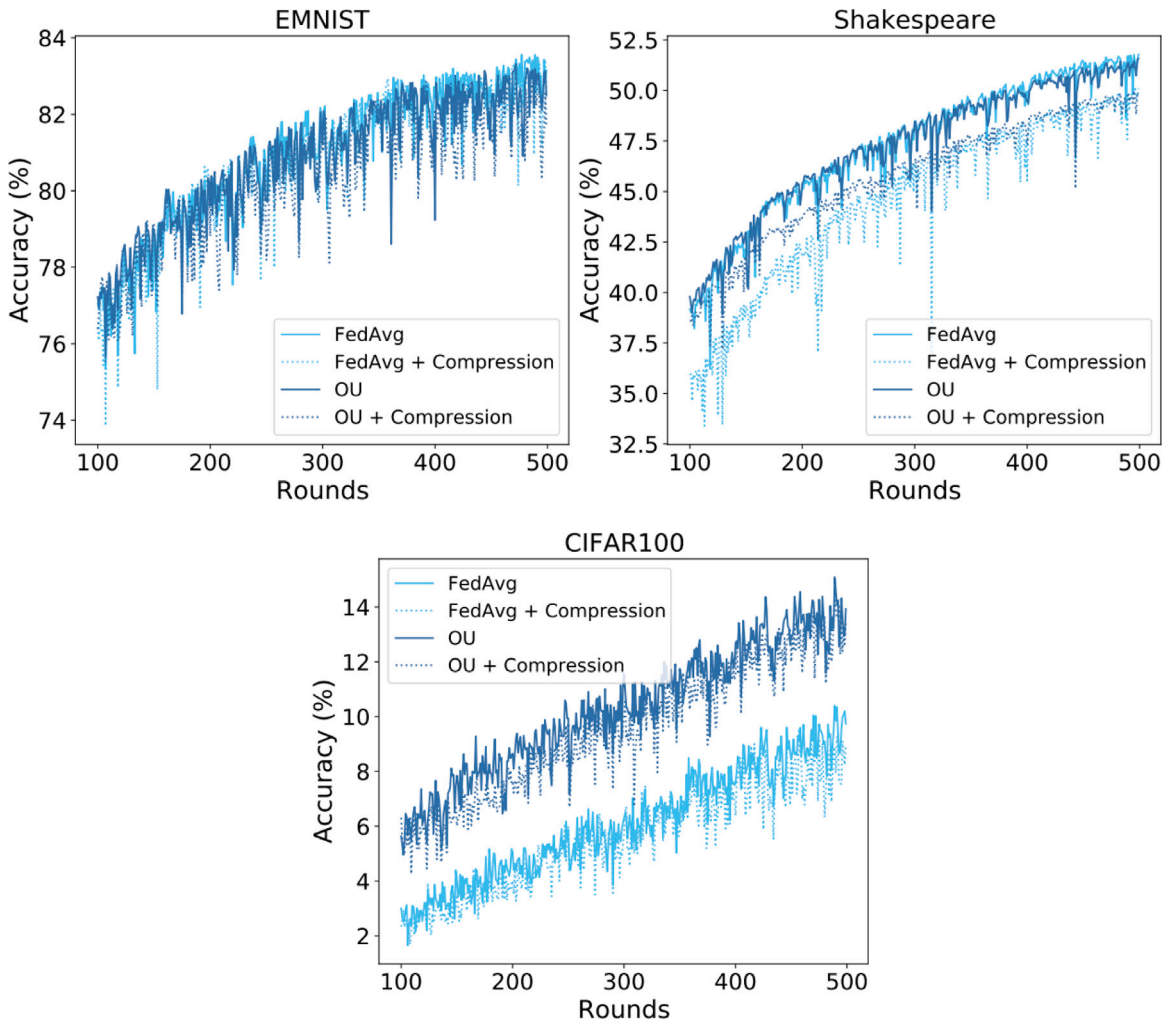


Fig. 3. Combining Adaptive-OU with compression. Model compression has no significant impact on the performance of our selection-estimation strategy.

Table 3
Accuracy and communication cost of communication-reduction strategies.

Dataset	Accuracy (%)		
	EMNIST	Shakespeare	CIFAR100
Baseline	82.7 (±0.57)	50.9 (±0.69)	8.7 (±0.76)
Adaptive-OU	**82.4** (±0.63)	**50.6** (±0.55)	13.1(±0.77)
Zero	82.3 (±0.61)	49.8 (±0.69)	7.8 (±0.71)
Ignore	75.9 (±1.16)	50.1 (±0.61)	13.3 (±1.3)
	Communication used (%)		
Baseline	100	100	100
Adaptive-OU	79	48	82
Zero	77	47	82
Ignore	66	49	77

strategy for estimating missing updates: over the three datasets, thresholding strategies learn a model with a performance comparable to the full communication baseline, even outperforming it on CIFAR100. However, thresholding is not sufficient — the way missing updates are handled has a significant impact. For instance, the ignore strategy achieves similar performance as Adaptive-OU on CIFAR100, yet considerably underperforms on EMNIST, and at a lower degree on Shakespeare datasets. Similarly, the zero strategy achieves solid results on EMNIST but falls behind on CIFAR100. Meanwhile, Adaptive-OU performs consistently well across different tasks.

We notice that the approaches which take into account missing updates and replace them with an estimate (OU and Zero) help smooth the training process. This can also be seen by considering the standard deviation over the last 100 rounds in Table 3, where we observe that both Zero and Adaptive-OU lead to smaller variance over the training rounds and a more stable convergence trajectory. However, by zeroing out clients' updates, the convergence of Zero is considerably slowed down on all datasets, confirming the effect of bias analyzed in Section 3.7. Indeed, as shown in Fig. 2, Adaptive-OU induces less variance over the optimization trajectory than Ignore (which tends to exhibit drastic drops in accuracy) while maintaining competitive accuracy. This can be theoretically explained by the larger variance of the latter, analyzed in Lemma 2, and also empirically observed in the standard deviation in Table 3.

4.3.2. *Communication savings*

Table 3 shows that our thresholding-based sampling combined with OU estimation, Adaptive-OU, requires smaller amount of communication to achieve accuracy comparable to the baseline (i.e., to the scheme using updates of all initially sampled clients). The zero estimation strategy has communication savings similar to OU but slower convergence rates and inferior final accuracy. Among all strategies, ignore achieves the highest communication savings but is ultimately not capable of matching the accuracy of the OU method in Shakespeare and EMNIST. This lower communication rate of the ignore strategy is due to the fact that the ignored clients will continue to have high norm value in the

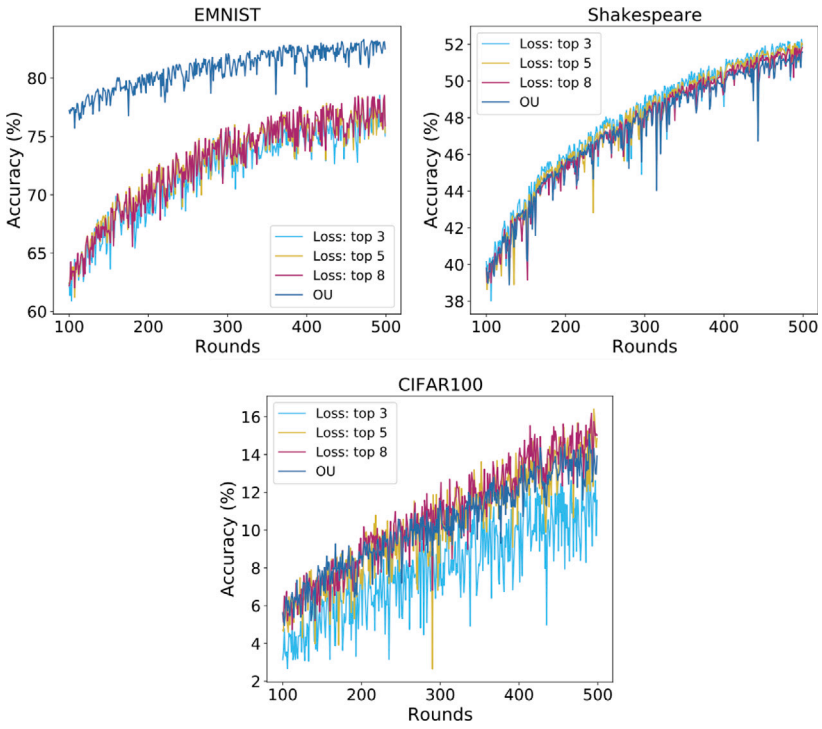


Fig. 4. Comparing Adaptive-OU with top-k selection and averaging based on loss values. Adaptive-OU delivers more consistent results across different datasets, without a need for setting additional hyperparameters (e.g., k).

subsequent iterations, which further inflates the threshold and leads to even fewer clients in the next round.

4.3.3. Combining client sampling with model compression

It was shown in [10] that FedAvg is compatible with model compression. Here we demonstrate that our method is also unaffected by and may readily be combined with compression strategies. Fig. 3 compares the performance of Adaptive-OU without compression to the setting where the model is compressed to 8 bits. We observe that the Adaptive-OU selection-estimation policy is robust to model compression. To compare, we also show the effect of compression on FedAvg with the same number of participating users. We observe that neither algorithm is significantly affected, confirming that our approach is not competing with but rather complementary to model compression.

4.3.4. Comparison with top-k loss selection

In [18], the authors proposed client selection based on averaging only the top-k (in terms of the loss) clients out of N that are sampled. The authors argue that client selection strategy can speed up convergence at the cost of inducing bias. The loss is an intuitive metric for selecting clients, and empirically shown to bring improvements compared to collecting updates from all the clients. However, the loss magnitudes are not necessarily comparable across clients: two clients may have different loss values yet the one with a smaller loss could be farther from the optimum. Alternatively, gradient magnitude may be a better indicator of proximity to the optimum, and is the default indicator of stability for smooth functions [45,46]. As shown in Section 3.7, our method is approximately unbiased under the same conditions introduced in [18] (smoothness and strong convexity).

In Fig. 4, we observe that in experiments on EMNIST, Adaptive-OU outperforms the loss-selection approach for all values of k; in experiments on Shakespeare, all methods perform similarly. Finally, in the experiments on CIFAR100 we observe that by setting $k = 5$ or $k = 8$, loss-based client selection performs similarly to Adaptive-OU; however, loss-based selection requires tuning the hyperparameter k. Thus, Adaptive-OU delivers more consistent results across datasets without needing to set additional hyperparameters.

4.3.5. Additional selection strategies

A trivial yet from the communication perspective effective alternative to judicious client selection is to randomly drop some of the clients. However, random client selection leads to deterioration in accuracy, especially in heterogeneous and non-convex settings. In our experiments (results omitted for brevity), random selection of clients significantly underperforms thresholding strategies on EMNIST and Shakespeare datasets, while it remains relatively competitive on CIFAR100. These fluctuations in performance are likely due to inherent heterogeneity of federated data in EMNIST and Shakespear experiments, which presents a major challenge to the random sampling policy; on the other hand, heterogeneity is not as pronounced in the artificially federated CIFAR100 data.

In particular, estimating the true gradient from a subset of clients is especially difficult in non-homogeneous and non-convex settings, and thus randomly dropping clients in such scenarios slows down the convergence. The convergence slowdown is also in part due to dropping too many clients near the optimum where the gradient norms become smaller. Our adaptive thresholding strategy overcomes the aforementioned problems by changing the threshold in each round based on the clients' update norms. We observe similar benefits when clients are selected using their loss function values, as proposed by [18]. However, the appropriate value of the hyperparameter k (the number of collected model updates) in such scenarios is unclear, which in practice leads to either excessive communication if the selected k is too large, or a failure to converge if k is too small (see Section 4.3.4).

In summary, the reported results demonstrate that the level of accuracy in federated learning can be maintained or even exceeded at a reduced communication by judiciously subsampling the clients performing updates; however, the clients that communicate model updates to the server need to be carefully selected, and the missing updates judiciously accounted for in the new global model. The results in Table 3 suggest that while there is no single method which is uniformly superior in exploring accuracy-communication trade-offs, the thresholding strategies are an efficient way of sub-selecting clients without a significant deterioration of accuracy. As shown, among the

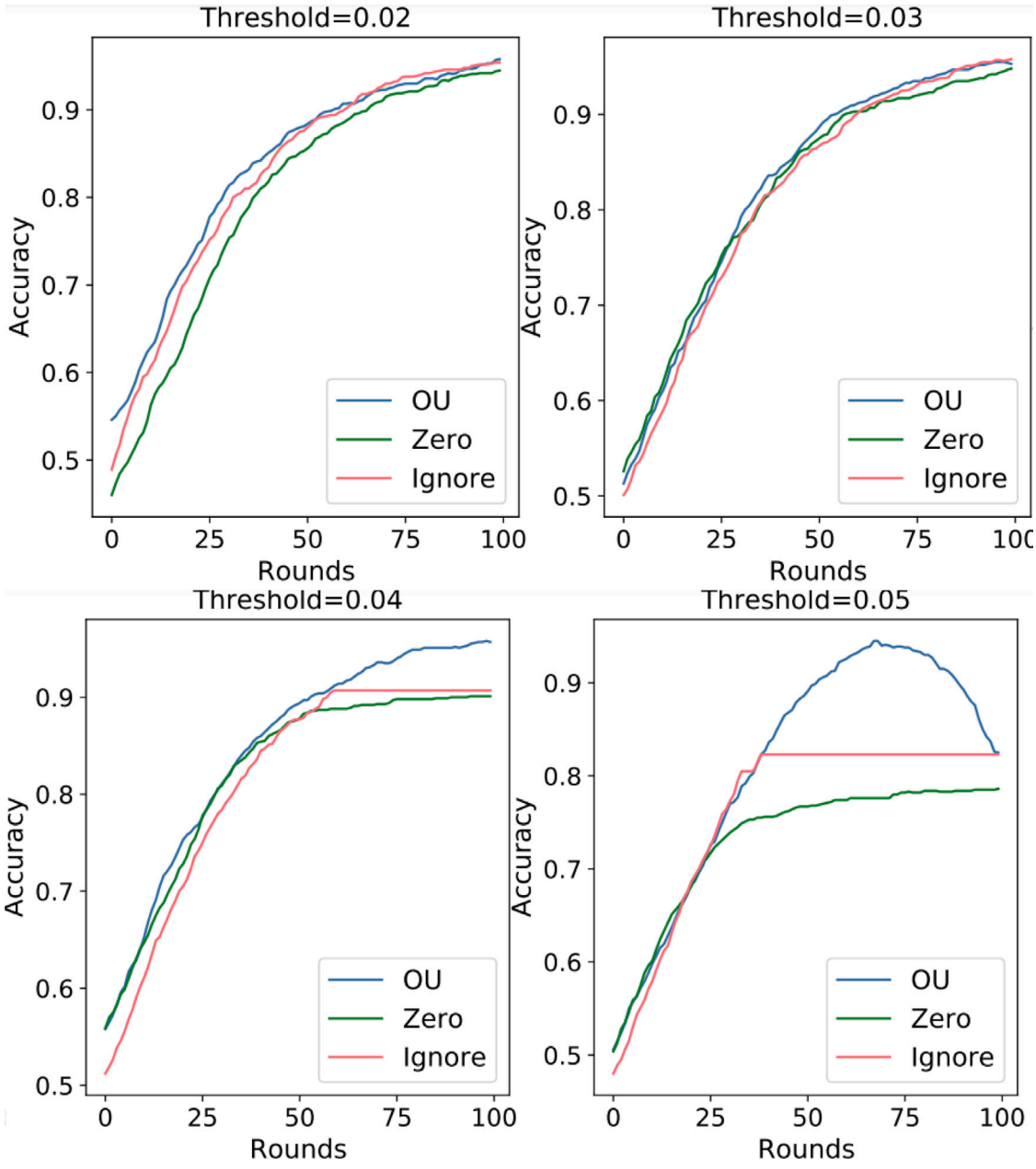


Fig. 5. Accuracy of FedAvg with a fixed threshold communication strategy for varied values of the threshold. High thresholds prohibit convergence to an accurate model since all clients stop transmitting once their updates become smaller than the threshold.

tested methods our is the only one capable of effectively reducing the number of users and maintaining high system performance without arduous hyperparameter tuning.

4.3.6. The initial number of sampled clients N

Recall that when a new round starts, the server samples a fixed predetermined number N of clients. As one would expect, increasing N leads to improvement of accuracy of all the methods considered. We test how the performance varies with N and show that the proposed method consistently outperforms other strategies (see Tables 4 and 5).

4.3.7. Fixed threshold

As described in Section 3, our proposed thresholding strategy relies on varying the value of the threshold according to the mean and standard deviation of the norms of the model updates. It is worth considering if a simpler scheme employing a fixed threshold might suffice. For this, we test on synthetic data a version of Algorithm

Table 4
Accuracy (%) on Shakespeare after 120 rounds as the number of clients N varies.

Method	Adaptive-OU	Ignore	Zero	Full comm.
N = 10	**23.3**	22.22	22.18	22.8
N = 20	**23.96**	23.37	23.06	23.55
N = 50	**25.17**	24.31	24.78	26.63

Table 5
Communication rate (%) on Shakespeare with varying number of initial clients N.

Method	Adaptive-OU	Ignore	Zero
N = 10	**46.6**	46.02	47.3
N = 20	47.43	**47.14**	47.98
N = 50	48.95	**48.93**	49.13

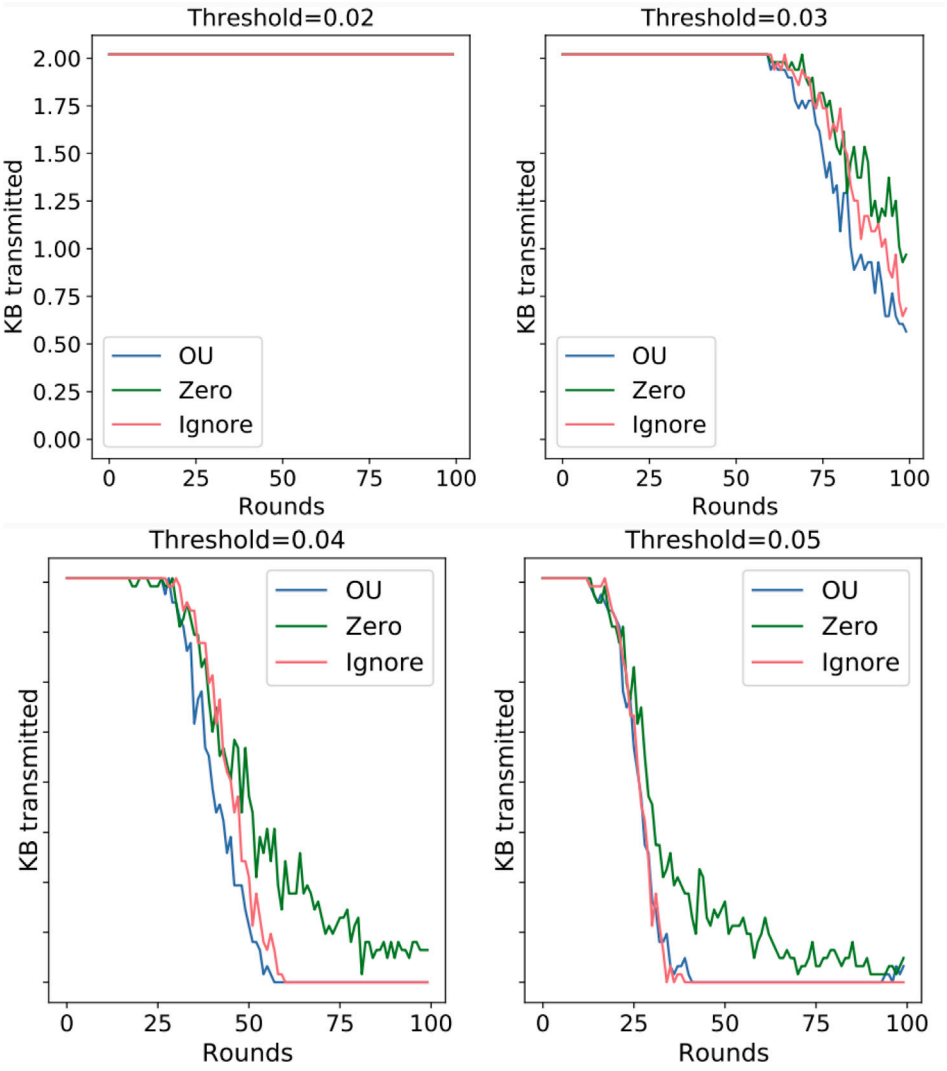


Fig. 6. Per round communication of FedAvg with a fixed threshold communication strategy as thresholds vary. Small thresholds fail to reduce communication, while high thresholds completely stop communication as training progresses thus leading to inaccurate models.

1 utilizing a fixed threshold and different estimation strategies. The results in Figs. 5 and 6 show that the OU client selection outperforms competing methods. Specifically, we observe in the two right-most plots in Fig. 5 that ZERO and IGNORE strategies tend to converge slower than the OU strategy. However, using a fixed threshold fails to provide accurate yet communication-efficient FL systems. First, treating threshold as a hyperparameter adds a layer of complexity to the system design problem since different values of the threshold may lead to very different results, as observed in Fig. 5. Tuning the threshold would go against the objective of reducing communication since a large number of rounds might be needed to facilitate such tuning. Finally, we observe that fixing the threshold to small values fails to reduce communication (the left-most plot in Fig. 6) while setting it to large values leads to inaccurate models (the right-most plot in Fig. 5).

5. Conclusion

We proposed a novel approach to reducing communication rates in FL by judiciously subselecting clients – a method which may be used in conjunction with traditional model compression strategies. Utilizing an interpretation of SGD as a stochastic process leads to an efficient estimator of missing client model updates, helping maintain and even improve the accuracy of the baseline scheme while cutting communication by up to 50%. Experimental results demonstrate efficacy of the proposed methods in various settings. The proposed client selection protocol is theoretically justified by the existing results on optimal OU process sampling. It is worth pointing out that in practical systems some users may not be available at certain iterations, which may adversely affect the rate of convergence of the proposed methods. Further, federated learning is often combined with differential privacy tools to limit the exposure of users' sensitive data. These privacy techniques first clip clients' updates and then add Gaussian noise, which could adversely affect our magnitude-based client selection policy. Future research directions include incorporating into our work an availability model that captures fluctuation of users over time, and studying the interplay between privacy and client selection policies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data we used is publicly available.

Acknowledgements

This work was supported in part by the National Science Foundation under grant 2148224 and in part by OUSD R&E, NIST, and Industry Partners as specified in the Resilient & Intelligent NextG Systems (RINGS) Program.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.patcog.2023.110122.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: International Conference on Artificial Intelligence and Statistics, 2017.
[2] M. Ribero, J. Henderson, S. Williamson, H. Vikalo, Federating recommendations using differentially private prototypes, Pattern Recognit. (2022).
[3] M. Abadi, A. Chu, I. Goodfellow, H.B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: SIGSAC, ACM, 2016.
[4] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, J. Naughton, Bolt-on differential privacy for scalable stochastic gradient descent-based analytics, in: SIGMOD, 2017.
[5] H.B. McMahan, G. Andrew, U. Erlingsson, S. Chien, I. Mironov, N. Papernot, P. Kairouz, A general approach to adding differential privacy to iterative training procedures, 2018, arXiv preprint arXiv:1812.06210.
[6] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[7] O. Shahid, S. Pouriyeh, R.M. Parizi, Q.Z. Sheng, G. Srivastava, L. Zhao, Communication efficiency in federated learning: Achievements and challenges, 2021, arXiv preprint arXiv:2107.10996.
[8] S. Huang, W. Shi, Z. Xu, I.W. Tsang, J. Lv, Efficient federated multi-view learning, Pattern Recognit. 131 (2022) 108817.
[9] H. Tang, X. Lian, T. Zhang, J. Liu, Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression, 2019, arXiv preprint arXiv:1905.05957.
[10] J. Konečnỳ, H.B. McMahan, F.X. Yu, P. Richtárik, A.T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, 2016, arXiv preprint arXiv:1610.05492.
[11] A.T. Suresh, F.X. Yu, S. Kumar, H.B. McMahan, Distributed mean estimation with limited communication, in: International Conference on Machine Learning, 2017.
[12] J. Konečnỳ, P. Richtárik, Randomized distributed mean estimation: Accuracy vs. communication, Front. Appl. Math. Statist. (2018).
[13] D. Alistarh, D. Grubic, J. Li, R. Tomioka, M. Vojnovic, Qsgd: Communication-efficient sgd via gradient quantization and encoding, in: Advances in Neural Information Processing Systems, 2017.
[14] S. Horvath, C.-Y. Ho, L. Horvath, A.N. Sahu, M. Canini, P. Richtarik, Natural compression for distributed deep learning, 2019, arXiv preprint arXiv:1905.10988.
[15] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G.R. Ganger, P.B. Gibbons, O. Mutlu, Gaia: Geo-distributed machine learning approaching {lan} speeds, in: {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI}), 2017.
[16] T. Chen, G. Giannakis, T. Sun, W. Yin, Lag: Lazily aggregated gradient for communication-efficient distributed learning, in: Advances in Neural Information Processing Systems, 2018.
[17] N. Singh, D. Data, J. George, S. Diggavi, Sparq-sgd: Event-triggered and compressed communication in decentralized stochastic optimization, 2019, arXiv preprint arXiv:1910.14280.
[18] Y.J. Cho, J. Wang, G. Joshi, Client selection in federated learning: Convergence analysis and power-of-choice selection strategies, 2020, arXiv preprint arXiv:2010.01243.
[19] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of fedavg on non-iid data, 2019, arXiv preprint arXiv:1907.02189.
[20] 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, 2019, arXiv preprint arXiv:1912.04977.
[21] X. Zhou, L. Chang, P. Xu, S. Lv, Communication-efficient and byzantine-robust distributed learning with statistical guarantee, Pattern Recognit. (2023).
[22] S. Caldas, J. Konečny, H.B. McMahan, A. Talwalkar, Expanding the reach of federated learning by reducing client resource requirements, 2018, arXiv preprint arXiv:1812.07210.
[23] Z. Li, T. Lin, X. Shang, C. Wu, Revisiting weighted aggregation in federated learning with neural networks, in: International Conference on Machine Learning (ICML), 2023.
[24] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), 2019.
[25] T. Castiglia, Y. Zhou, S. Wang, S. Kadhe, N. Baracaldo, S. Patterson, LESS-VFL: Communication-efficient feature selection for vertical federated learning, in: International Conference on Machine Learning (ICML), 2023.
[26] O.C. Imer, T. Basar, Optimal estimation with limited measurements, in: IEEE Conference on Decision and Control, 2005.
[27] P.A. Bommannavar, T. Başar, Optimal estimation over channels with limits on usage, IFAC Proc. Vol. (2008).
[28] A. Nayyar, T. Başar, D. Teneketzis, V.V. Veeravalli, Optimal strategies for communication and remote estimation with an energy harvesting sensor, IEEE Trans. Automat. Control (2013).
[29] M. Rabi, G.V. Moustakides, J.S. Baras, Adaptive sampling for linear state estimation, SIAM J. Control Optim. (2012).
[30] K. Nar, T. Başar, Sampling multidimensional wiener processes, in: IEEE Conference on Decision and Control, 2014.
[31] Y. Sun, Y. Polyanskiy, E. Uysal-Biyikoglu, Remote estimation of the wiener process over a channel with random delay, in: IEEE International Symposium on Information Theory (ISIT), IEEE, 2017.
[32] T.Z. Ornee, Y. Sun, Sampling for remote estimation through queues: Age of information and beyond, 2019, arXiv preprint arXiv:1902.03552.
[33] N. Guo, V. Kostina, Optimal causal rate-constrained sampling for a class of continuous markov processes, 2020, arXiv preprint arXiv:2002.01581.
[34] G. Blanc, N. Gupta, G. Valiant, P. Valiant, Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process, 2019, arXiv preprint arXiv:1904.09080.
[35] Y. Wang, Asymptotic analysis via stochastic differential equations of gradient descent algorithms in statistical and computational paradigms, 2017, arXiv preprint arXiv:1711.09514.
[36] T. Li, L. Liu, A. Kyrillidis, C. Caramanis, Statistical inference using sgd, in: AAAI Conference on Artificial Intelligence, 2018.
[37] S. Mandt, M. Hoffman, D. Blei, A variational analysis of stochastic gradient algorithms, in: International Conference on Machine Learning, 2016.
[38] R.S. Liptser, A.N. Shiryaev, Statistics of Random Processes II: Applications, Springer Science & Business Media, 2013.
[39] J. Shao, D. Tu, The Jackknife and Bootstrap, Springer Science & Business Media, 2012.
[40] S.U. Stich, Local SGD converges fast and communicates little, 2018, arXiv preprint arXiv:1805.09767.
[41] S.J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, H.B. McMahan, Adaptive federated optimization, in: International Conference on Learning Representations (ICLR), 2020.
[42] The TFF Authors, Tensorflow federated, 2019, URL https://www.tensorflow.org/federated.
[43] G. Cohen, S. Afshar, J. Tapson, A. Van Schaik, Emnist: Extending mnist to handwritten letters, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017.
[44] K. Hsieh, A. Phanishayee, O. Mutlu, P. Gibbons, The non-iid data quagmire of decentralized machine learning, in: International Conference on Machine Learning, 2020.
[45] Y. Nesterov, Introductory lectures on convex programming volume i: Basic course, Lect. Not. (1998).
[46] Z. Allen-Zhu, Natasha 2: Faster non-convex optimization than sgd, in: Advances in Neural Information Processing Systems, 2018.

Mónica Ribero received the B.Sc. degree in mathematics from the Universidad de los Andes, Bogotá, Colombia, in 2015, and the Ph.D. degree in electrical and computer engineering from the University of Texas, Austin, TX, USA, in 2022. She is currently working on federated learning under privacy and communication constraints with the University of Texas. She joined Google Research New York, NY, USA, as a Research Scientist. She held Research Internship Positions, Bell Laboratories, Murray Hill, NJ, USA, in 2008, CognitiveScale, Austin, TX, in 2019, and Google Research in 2020.

Haris Vikalo received the B.S. degree in electrical engineering from the University of Zagreb, Zagreb, Croatia, in 1995, the M.S. degree in electrical engineering from Lehigh University, Bethlehem, PA, USA, in 1997, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2003. He has held a short-term appointment with Bell Laboratories, Murray Hill, NJ, USA, in the summer of 1999. From January 2003 to July 2003, he was a Postdoctoral Researcher, and from July 2003 to August 2007, he was an Associate Scientist with the California Institute of Technology, Pasadena, CA, USA. Since September 2007, he has been with the Department of Electrical and Computer Engineering, the University of Texas, Austin, TX, USA. His research interests include signal processing, machine learning, communications, and bioinformatics. Prof. Vikalo was the recipient of the 2009 National Science Foundation Career Award.