This article has been accepted for publication in IEEE Journal on Selected Areas in Information Theory. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAIT.2022.3205475

Straggler-Resilient Federated Learning: Leveraging the Interplay Between Statistical Accuracy and System Heterogeneity

Amirhossein Reisizadeh, Isidoros Tziotis, Hamed Hassani, Aryan Mokhtari, Ramtin Pedarsani

***Abstract*—Federated learning is a novel paradigm that involves learning from data samples distributed across a large network of clients while the data remains local. It is, however, known that federated learning is prone to multiple system challenges including system heterogeneity where clients have different computation and communication capabilities. Such heterogeneity in clients' computation speed has a negative effect on the scalability of federated learning algorithms and causes significant slow-down in their runtime due to slow devices (stragglers). In this paper, we propose `FLANP`, a novel straggler-resilient federated learning meta-algorithm that incorporates statistical characteristics of the clients' data to *adaptively* select the clients in order to speed up the learning procedure. The key idea of `FLANP` is to start the training procedure with faster nodes and gradually involve the slower ones in the model training once the statistical accuracy of the current participating nodes' data is reached, while the final model for each stage is used as a warm-start model for the next stage. Our theoretical results characterize the speedup provided by the meta-algorithm `FLANP` in comparison to standard federated benchmarks for strongly convex losses and i.i.d. samples. For particular instances, `FLANP` slashes the overall expected runtime by a factor of $\mathcal{O}(\ln(Ns))$, where N and s denote the total number of nodes and the number of samples per node, respectively. In experiments, `FLANP` demonstrates significant speedups in wall-clock time –up to $6\times$– compared to standard federated learning benchmarks.**

I. Introduction

Federated learning is a distributed framework whose objective is to train a model using the data of many clients (nodes), while keeping each node's data local. In contrast with centralized learning, the federated learning architecture allows for preserving the clients' privacy as well as reducing the communication burden caused by transmitting data to a cloud. Nevertheless, as we move towards deploying federated learning in practice, it is becoming apparent that several major challenges still remain and the existing frameworks need to be rethought to address them. Important among these challenges is system (device) heterogeneity due to existence of *straggling nodes* – slow nodes with low computational capability – that significantly slow down the model training [2], [3].

In this paper, we focus on system heterogeneity in federated learning and we leverage the interplay between statistical accuracy and system heterogeneity to design a straggler-resilient federated learning method that carefully and adaptively selects a subset of available nodes in each round of training. Federated networks consist of thousands of devices with a wide range of computational, communication, battery power, and storage characteristics. Hence, deploying traditional federated learning algorithms such as `FedAvg` [4] on such a highly heterogeneous cluster of devices results in significant and unexpected delays due to existence of slow clients or stragglers. In most of such algorithms, *all* the *available* clients participate in the model training –regardless of their computational capabilities. Consequently, in each communication round of such methods, the server has to wait for the slowest node to complete its local updates and upload its local model which significantly slows down the training process.

In this paper, we aim to mitigate the effect of stragglers in federated learning based on an adaptive node participation approach in which clients are selected to participate in different stages of training according to their computation speed. We call our straggler-resilient scheme a **F**ederated **L**earning method with **A**daptive **N**ode **P**articipation or `FLANP`. The key idea of this scheme is to start the model training procedure with only a few clients which are the fastest among all the nodes. These participating clients continue to train their shared models while interacting with the parameter server. Note that since the server waits only for the participating nodes, it takes a short time for the participating (and fast) clients to promptly train a shared model. This model is, however, not accurate as it is trained over only a fraction of samples. We next increase the number of participating clients and include the next fastest subset of nonparticipating nodes in the training. Note that the model trained from the previous stage can be a warm-start initialization for the current stage.

To discuss our main idea more precisely, consider a federated

This work was supported by NSF grants 190932, 2003035 and IIS-2112471.

Amirhossein Reisizadeh is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (email: amirr@mit.edu).

Isidoros Tziotis is with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA (email: isidoros_13@hotmail.com).

Hamed Hassani is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (email: hassani@seas.upenn.edu).

Aryan Mokhtari is with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA (email: mokhtari@austin.utexas.edu).

Ramtin Pedarsani is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, Santa Barbara, CA 93106 USA (email: ramtin@ece.ucsb.edu).

A preliminary and short version of this work is published at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2022 [1]. This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material includes the proofs of the paper's theoretical statements and related extensions. Also, the paper's code is accessible at the GitHub repository https://github.com/IsidorosTziotis/Straggler_Resilient

network of N available nodes each storing s data samples and suppose that we start the learning procedure with only m clients. Once we solve the empirical risk minimization (ERM) problem corresponding to $m \times s$ samples of these nodes up to its statistical accuracy, we geometrically increase the number of participating nodes to $n = \alpha m$ where $\alpha > 1$, by adding the next $n - m$ fastest clients in the network. By doing so, the new ERM problem that we aim to solve contains the samples from the previous stage as well as the samples of the newly participating nodes. Moreover, the solution for the ERM problem at the previous stage (with m clients) could be used as a warm-start for the ERM problem at the current stage with $n = \alpha m$ nodes. This is due to the fact that all samples are drawn from a common distribution, and as a result, the optimal solution of the ERM problem with fewer samples is not far from the optimal solution of the ERM problems with more samples, as long as the larger set contains the smaller set.

In the proposed `FLANP` algorithm, as time progresses, we gradually increase the number of participating clients until we reach the full training set and all clients are involved. Note that in this procedure, the slower clients are only used towards the end of the learning process, where the model is already close to the optimal model of the aggregate loss. Another essential observation is that since the model trained in previous rounds already has a reasonable statistical accuracy and this model serves as the initial point of the next round of the iterative algorithm, the slower nodes are only needed to contribute in the final rounds of training, leading to a smaller wall-clock time. This is in contrast with having all nodes participate in training from the beginning, which leads to computation time of each round being determined by the slowest node. In this paper, we formally characterize the gain obtained by using the proposed adaptive node participation scheme compared to the case that all available nodes contribute to training at each round. Next, we state a summary of our main contributions:

• We present a straggler-resilient federated learning meta-algorithm that leverages the interplay between statistical accuracy and device heterogeneity by adaptively activating heterogeneous clients.

• We specify the proposed meta-algorithm with a federated learning subroutine and present its optimization guarantees for strongly convex risks. Further, we characterize the wall-clock time of the proposed straggler-resilient scheme and demonstrate analytically that it achieves up to $\mathcal{O}(\ln(Ns))$ speedup gain compared to standard benchmarks.

• Our numerical results show that our framework significantly improves the wall-clock time compared to federated learning benchmarks –with either full or partial node participation– for both convex and non-convex risks.

• We extend our adaptive node participation approach to data heterogeneous setting where each node's data is drawn from one of C different distributions. We demonstrate theoretical speedups of the adaptive scheme in such clustered scenarios and theoretically examine its performance. We defer this section to Appendix.

Related Work. System (device) heterogeneity challenge, which refers to the case that clients have different computational, communication and storage characteristics, has been studied in the literature. Asynchronous methods have demonstrated improvements in distributed data centers. However, such methods are less desirable in federated settings as they rely on bounded staleness of slow clients [5], [6]. The active sampling approach is another direction in which the server aims for aggregating as many local updates as possible within a predefined time span [7]. More recently, [8] proposed a normalized averaging method to mitigate stragglers in federated systems and the objective inconsistency due to mismatch in clients' local updates. Deadline-based computation has also been studied to mitigate stragglers in decentralized settings [9]. In a different yet related direction, various federated algorithms have been studied to address the heterogeneity in clients' *data* distributions [10]–[15].

The idea of adaptive sample size training in which we solve a sequence of geometrically increasing ERM problems has been used previously for solving large-scale ERM problems. In particular, it has been shown that this scheme improves the overall computational cost of both first-order [16], [17] and second-order [18]–[20] methods for achieving the statistical accuracy of the full training set. In this paper, we exploit this idea to develop `FLANP` for a different setting to address the issue of device heterogeneity in federated learning. In a different setting , i.e. distributed machine learning, [21] proposes to exponentially increase the number of workers where the (stochastic) gradient computation task is distributed among the clients without local training.

As mentioned, `FLANP` is a general meta-algorithm that can be employed with any federated learning subroutine studied in the literature [4], [22]–[34]. In this paper, we showcase the gain obtained by combining `FLANP` with the `FedGATE` method [11].

II. Federated Learning Setup

In this section, we state our setup. Consider a federated architecture where N nodes interact with a central server, and each node $i \in [N] = \{1, \cdots, N\}$ has access to s data samples denoted by $\{z_1^i, \cdots, z_s^i\}$. These samples are drawn at the beginning of the training process, and nodes cannot draw new samples during training. Further, define $\ell(\cdot, \cdot) : \mathbb{R}^d \times \mathcal{Z} \to \mathbb{R}$ as a loss function where $\ell(\mathbf{w}, z_j^i)$ indicates how well the model $\mathbf{w}$ performs with respect to the sample z_j^i. Also, define the empirical loss of node i as $L^i(\mathbf{w}) \coloneqq \frac{1}{s}\sum_{j=1}^{s} \ell(\mathbf{w}, z_j^i)$. For any $1 \leq n \leq N$, we denote by $L_n(\mathbf{w})$ the collective empirical risk corresponding to samples of all nodes $\{1, \cdots, n\}$, which is defined as

$$L_n(\mathbf{w}) \coloneqq \frac{1}{n}\sum_{i=1}^{n} L^i(\mathbf{w}). \tag{1}$$

Here, $L_n(\mathbf{w})$ represents the average loss over the $n \times s$ samples stored at nodes $\{1, \cdots, n\}$. We let $\mathbf{w}_n^*$ denote the optimal minimizer of the loss $L_n(\mathbf{w})$, i.e., $\mathbf{w}_n^* = \arg\min_{\mathbf{w}} L_n(\mathbf{w})$. We assume that the samples z_j^i are i.i.d. realizations of a random variable Z with probability distribution $\mathcal{P}$. The problem of finding a global model for the aggregate loss of all available N nodes, which can be considered as the empirical risk

This article has been accepted for publication in IEEE Journal on Selected Areas in Information Theory. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAIT.2022.3205475

minimization (ERM) in (1) for $n = N$, i.e.,

$$\min_{\mathbf{w}} L_N(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L^i(\mathbf{w}) = \frac{1}{Ns}\sum_{i=1}^{N}\sum_{j=1}^{s} \ell(\mathbf{w}, z_j^i) \quad (2)$$

is a surrogate for the expected risk minimization problem $\min_{\mathbf{w}} L(\mathbf{w}) := \mathbb{E}_{Z\sim\mathcal{P}}[\ell(\mathbf{w}, Z)]$. Our ultimate goal is to find the optimal solution of the expected risk $\mathbf{w}^* = \arg\min_{\mathbf{w}} L(\mathbf{w})$; however, since distribution $\mathcal{P}$ is unknown and we only have access to a finite number of realizations of the random variable Z, i.e., $\{z_1^i, \cdots, z_s^i\}_{i=1}^N$, we settle for solving problem (2).

Statistical Accuracy. The difference of expected and empirical risks $L_n(\mathbf{w}) - L(\mathbf{w})$ is referred to as the estimation error and can be bounded by a function of the sample size. In particular, since $L_n(\mathbf{w})$ captures ns samples, we assume that there exists a constant V_{ns} bounding the estimation error with high probability, $\sup_{\mathbf{w}} |L_n(\mathbf{w}) - L(\mathbf{w})| \leq V_{ns}$.

The estimation error V_{ns} has been deeply studied in the statistical learning literature [35], [36]. In particular, it has been shown that for strongly convex functions the estimation error is proportional to the inverse of sample size [37], [38]. In this work, we also assume that $V_{ns} = \frac{c}{ns}$ for a constant c. Note that for the loss function L_n once we find a point $\tilde{\mathbf{w}}$ that has an optimization error of V_{ns}, i.e., $L_n(\tilde{\mathbf{w}}) - L_n(\mathbf{w}_n^*) \leq V_{ns}$, there is no gain in improving the optimization error as the overall error with respect to the expected risk L would not improve. Hence, when we find a point $\tilde{\mathbf{w}}$ such that $L_n(\tilde{\mathbf{w}}) - L_n(\mathbf{w}_n^*) \leq V_{ns}$, we state that it has reached the statistical accuracy of L_n. Our goal is to find a solution $\mathbf{w}_N$ that is within the statistical accuracy of the ERM problem of the full training set defined in (2).

System Heterogeneity Model. As mentioned earlier, federated clients attribute a wide range of computational powers leading to significantly different processing times for a fixed computing task such as gradient computation and local model update. To be more specific, for each node $i \in [N]$, we let T_i denote the (expected) time to compute one local model update. The time for such update is mostly determined by the computation time of a fixed batch-size stochastic gradient of the local empirical risk $L_i(\mathbf{w})$. Clearly, larger T_i corresponds to slower clients or stragglers. Without loss of generality and for the sake of simplicity in explanation, we assume that the nodes are sorted from faster to slower, that is, $T_1 \leq \cdots \leq T_N$ with node 1 and N respectively identifying the fastest and slowest nodes in the network. For practical purposes where the computing speeds are unknown, we will provide procedures to determine the (relative) speeds in Section III.

III. Adaptive Node Participation Approach

Several federated learning algorithms have been proposed to solve the ERM problem in (2) such as `FedAvg` [4], `FedProx` [12], `SCAFFOLD` [10], `DIANA` [39], and `FedGATE` [11]. These methods consist of several rounds of local computations by the clients and communication with the server. Alas, in all such approaches, *all the available nodes* in the network –regardless of their computational capabilities– contribute to model learning throughout the entire procedure. As explained earlier, federated clients operate in a wide range of computational characteristics, and therefore, the server has to wait for the slowest node in each communication round to complete its local computation task. All in all, the slowest nodes determine the overall runtime of such federated algorithms which causes significant slow-down.

Algorithm 1: `FLANP`

Initialize fast-to-slow nodes $\{1, \cdots, N\}$, $n = n_0$
participating nodes with initial global model $\mathbf{w}_{n_0}$
while $n \leq N$ **do**
  while $L_n(\mathbf{w}_n) - L_n(\mathbf{w}_n^*) > V_{ns}$ **do**
    nodes $\{1, \cdots, n\}$ are participating and update local models via `Federated_Solver`
    server aggregates local models from nodes $\{1, \cdots, n\}$ and updates global model $\mathbf{w}_n$
  end
  $n \leftarrow \min\{2n, N\}$ `% doubling the participants`
end

In this section, we describe our proposed approach to mitigate stragglers in federated learning, and lay out the intuition behind that. Our proposal, `FLANP`, is essentially a meta-algorithm that can be specified with the choice of any particular federated learning subroutine. The rest of the section focuses on a particular case of `FLANP` where the federated learning subroutine is picked to be `FedGATE` proposed by [11].

A. *FLANP: A Straggler-Resilient FL Meta-Algorithm*

Our proposal to address the device heterogeneity and mitigate the stragglers is as follows.The server first solves the ERM problem corresponding to n_0 fastest nodes, where n_0 is much smaller than the total number of available nodes N. To identify the n_0 fastest nodes, the server first broadcasts a short hand-shake message to all nodes and waits for the first n_0 nodes that respond. These n_0 nodes will participate in the training process in the first stage. Using `Federated_Solver` which is a federated learning subroutine of choice, e.g., `FedAvg` or `FedGATE`, the n_0 participating nodes proceed to minimize the empirical risk corresponding to their data points, which we denote by $L_{n_0}(\mathbf{w})$ as defined in (1). This continues until the n_0 nodes reach their corresponding statistical accuracy, that is, they reach a global model $\mathbf{w}_{n_0}$ such that $L_{n_0}(\mathbf{w}_{n_0}) - L_{n_0}(\mathbf{w}_{n_0}^*) \leq V_{n_0 s}$. Note that at this stage the server has to wait only for the slowest client among the participating ones, i.e., node n_0, which is potentially much faster than the network's slowest node N.

Per our discussion in Section II, a more accurate solution than $\mathbf{w}_{n_0}$ would not help improving the optimality gap. Therefore, once statistical accuracy is achieved, the procedure is terminated and we increase the number of participating nodes from n_0 to $2n_0$. To select the $2n_0$ fastest nodes, we repeat the hand-shaking communication protocol that we discussed. Then, the selected nodes use `Federated_Solver` to find the minimizer of the loss corresponding to $2n_0$ participating nodes, while using the solution of the previous stage $\mathbf{w}_{n_0}$ as their starting point. Note that since the samples of nodes come from the same distribution, we can show that the solutions of two successive stages with n_0 and $2n_0$ participants are close to each other, as we discuss in Section IV.

Again, in this stage, the training process terminates when we find a point $\mathbf{w}_{2n_0}$ within the statistical accuracy of the loss corresponding to $2n_0$ participating nodes, i.e., $L_{2n_0}(\mathbf{w}_{2n_0}) - L_{2n_0}(\mathbf{w}^*_{2n_0}) \leq V_{2n_0 s}$. In the stage with $2n_0$ participating nodes, the computation delay is determined by the slowest participating node among $2n_0$ nodes, which is slower than the previous stage with n_0 participating nodes, but still faster than the slowest node of the network. The procedure of geometrically increasing the number of participating nodes continues till the set of participating nodes contains all the available N nodes, and these nodes find the final global model $\mathbf{w}_N$ within the statistical accuracy of the global loss function $L_N(\mathbf{w})$. Algorithm 1 summarizes the straggler-resilient meta-algorithm.

Remark 1. *In our proposed scheme, clients' computation speeds are not needed, and the parameter server figures out the fastest n nodes only via several approaches including the following:*

- *Handshaking protocol: In the beginning of each stage, the parameter server identifies the fastest n nodes by first broadcasting a short hand-shake message to all nodes and then waiting for the first n nodes that respond. Alternatively, the server may wait for* all *the nodes to respond which enables it to have the speed ranking of all the devices up front. By doing so, only one execution of handshaking is sufficient.*
- *The parameter server may assign a unique (and small) computing task to all the devices before the actual training starts. Each node report its computing time to the server which enables the server to determine the ranking of the computing speed s of the devices.*

From a high-level perspective, Algorithm 1 exploits faster nodes in the beginning of the learning procedure to promptly reach a global model withing their statistical accuracy. By doing so, the server avoids waiting for slower nodes to complete their local updates; however, the optimality gap of such models are relatively large since only a fraction of data samples have contributed in the global model. By gradually increasing the number of participating nodes and activating slower nodes, the quality of the global model improves while the synchronous computation slows down due to slower nodes. The key point is that slower nodes join the learning process towards the end.

The criterion in Algorithm 1, that is $L_n(\mathbf{w}_n) - L_n(\mathbf{w}^*_n) > V_{ns}$, verifies that the current global model does not satisfy the statistical accuracy corresponding to n participating nodes $\{1, \cdots, n\}$. This condition, however, is not easy to check since the optimal solution $\mathbf{w}^*_n$ is unknown. A sufficient and computationally feasible criterion is to check if $\|\nabla L_n(\mathbf{w}_n)\|^2 \leq 2\mu V_{ns}$, when ℓ is μ-strongly convex.

B. FLANP via FedGATE

As FLANP in Algorithm 1 is a general mechanism to mitigate stragglers in federated settings, one needs to specify the inner optimization subroutine Federated_Solver to quantify the speedup of the proposed approach. This subroutine could be any federated learning algorithm, but here we focus on FedGATE [11], a federated learning algorithm that employs gradient tracking variables to provide tight convergence guarantees for nodes with heterogeneous data distributions.

Algorithm 2: FLANP via FedGATE

Initialize $n = n_0$ participating nodes, initial model $\mathbf{w}_{n_0}$, initial gradient tracking $\delta_i^{(0)} = 0$ for participating nodes $i \in \{1, \cdots, n_0\}$
while $n \leq N$ **do**
  $r = 0$ % reset round counter for each stage
  for participating nodes $i \in \{1, \cdots, n\}$ **do**
    $\delta_i^{(0)} = 0$ % reset gradient tracking
  end
  while $\|\nabla L_n(\mathbf{w}_n)\|^2 > 2\mu V_{ns}$ **do**
    for participating nodes $i \in \{1, \cdots, n\}$ **do**
      $\mathbf{w}_i^{(0,r)} = \mathbf{w}_n$
      for $c = 0, \cdots, \tau_n - 1$ **do**
        set $d_i^{(c,r)} = \widetilde{\nabla} L^i(\mathbf{w}_i^{(c,r)}) - \delta_i^{(r)}$
        update $\mathbf{w}_i^{(c+1,r)} = \mathbf{w}_i^{(c,r)} - \eta_n d_i^{(c,r)}$
      end
      send $\Delta_i^{(r)} = (\mathbf{w}_n - \mathbf{w}_i^{(\tau_n, r)})/\eta_n$ to server
      update $\delta_i^{(r+1)} = \delta_i^{(r)} + \frac{1}{\tau_n}(\Delta_i^{(r)} - \Delta^{(r)})$
    end
    server broadcasts $\Delta^{(r)} = \frac{1}{n}\sum_{i=1}^n \Delta_i^{(r)}$
    server broadcasts $\mathbf{w}_n \leftarrow \mathbf{w}_n - \eta_n \gamma_n \Delta^{(r)}$
    participating nodes $i \in \{1, \cdots, n\}$ upload gradients $\nabla L^i(\mathbf{w}_n)$ to server
    $r \leftarrow r + 1$
  end
  $n \leftarrow \min\{2n, N\}$ % doubling the participants
end

Why FedGATE**?** We would like to reiterate that FLANP is a meta-procedure that can be used for *any* federated learning solver including FedGATE to make it resilient against straggling nodes. Nevertheless, we use FedGATE as the subroutine since it provides convergence bounds that scale with the number of participating nodes (which translates to the number of incorporated data samples), which is desired for the purpose of our adaptive node method. Algorithm 2 demonstrates how adaptive node participation in FLANP is adopted to mitigate straggler delays in FedGATE.

We begin the first stage of Algorithm 2 with activating the $n = n_0$ fastest nodes $\{1, \cdots, n_0\}$ and initialize them with global model $\mathbf{w}_{n_0}$. We also reset the gradient tracking variables $\delta_i^{(0)}$ to be zero for all participating nodes at the beginning of each stage. Variables δ_i aim to correct the directions of local updates at node i by tracking the difference of local gradients $\widetilde{\nabla} L^i$ and global gradients ∇L_n such that directions d_i closely follow the correct global gradient direction. After τ_n iterations of local updates at any participating node in round r, accumulations of local gradients $\Delta_i^{(r)}$ are uploaded to the server where it updates the global model $\mathbf{w}_n$ using two stepsizes η_n, γ_n. Note that the stepsizes η_n, γ_n are fixed throughout each stage with n participating nodes but vary for different stages as n increases. After updating the global model $\mathbf{w}_n$ at the end of each round, participating nodes upload their local gradients $\nabla L^i(\mathbf{w}_n)$ such that the server aggregates and computes the global gradient $\nabla L_n(\mathbf{w}_n)$ and checks whether

$\|\nabla L_n(\mathbf{w}_n)\|^2 \leq 2\mu V_{ns}$. After R_n rounds of communications, this condition is satisfied and the set of n participating nodes reach the model $\mathbf{w}_n$ within their statistical accuracy. Therefore, we augment the set of participating nodes (from faster to slower) from $\{1, \cdots, n\}$ to $\{1, \cdots, 2n\}$ leading to a new stage. The above procedure continues until the set of participating nodes contains all N available nodes.

IV. Theoretical Results

In this section, we provide rigorous analysis for `FLANP` outlined in Algorithm 2, which employs `FedGATE` as its subroutine. We first characterize optimization guarantees of Algorithm 2. Using such results, we derive the expected runtime of our proposed algorithm and the speedup gain it provides compared to naive methods.

A. Optimization Guarantees

Next, we analyze `FLANP` outlined in Algorithm 2, which employs `FedGATE` as its subroutine. We first characterize optimization guarantees of Algorithm 2. Using such results, we derive the expected runtime of our proposed algorithm and the speedup it provides compared to naive methods.

Connection between two successive stages. As we discussed in Section III-A, we expect the solution of each stage with m participating nodes to be close to the solution of the next stage with n nodes, where $n > m$, if the larger set of nodes contain the smaller set. This is due to the fact that within each cluster, samples are drawn from the same distribution. To formalize this claim, consider a subset of m participating nodes and a model $\mathbf{w}_m^*$ within their statistical accuracy, i.e., $L_m(\mathbf{w}_m) - L_m(\mathbf{w}_m^*) \leq V_{ms}$. Next, we show that the suboptimality error of $\mathbf{w}_m$ for the next loss with n nodes is small, when the set of n nodes contains m nodes.

Proposition 1. *Consider two subsets of nodes $\mathcal{N}_m \subseteq \mathcal{N}_n$ and assume that model $\mathbf{w}_m$ attains the statistical accuracy for the empirical risk associated with nodes in $\mathcal{N}_m$, i.e., $\|\nabla L_m(\mathbf{w}_m)\|^2 \leq 2\mu V_{ms}$ where the loss function ℓ is μ-strongly convex. Then the suboptimality of $\mathbf{w}_m$ for risk L_n is w.h.p. bounded above by*

$$L_n(\mathbf{w}_m) - L_n(\mathbf{w}_n^*) \leq \frac{2(n-m)}{n}(V_{(n-m)s} + V_{ms}) + V_{ms}.$$

Proof: We defer the proof to the Appendix. ∎

Proposition 1 demonstrates that a model attaining the statistical accuracy for m nodes can be used as an initial model for the ERM corresponding to a larger set with n nodes. In particular, when the number of participating nodes is doubled, i.e., $n = 2m$, then the initial sub-optimality error is bounded above by $L_n(\mathbf{w}_m) - L_n(\mathbf{w}_n^*) \leq 3V_{ms}$.

Next, we characterize the required communication and computation for solving each subproblem. Specifically, consider the case that we are given a model $\mathbf{w}_m$ which is within the statistical accuracy of L_m corresponding to m fastest nodes in each cluster, and the goal is to find a new model $\mathbf{w}_n$ that is within the statistical accuracy of L_n corresponding to n fastest nodes of each cluster, where $n = 2m$. To analyze this procedure, we must specify three parameters: the choice of stepsizes η_n, γ_n, the number of local updates τ_n at each participating node, and the number of communication rounds with the server R_n. For these parameters we use index n, as they refer to the case that n nodes participate in the training. Next, we state our main assumptions.

Assumption 1. *The loss $\ell(\mathbf{w}, z)$ is μ-strongly convex with respect to $\mathbf{w}$, and the gradient $\nabla_{\mathbf{w}}\ell(\mathbf{w}, z)$ is L-Lipschitz continuous. The condition number is defined as $\kappa := L/\mu$.*

The conditions in Assumption 1 imply that the empirical risks $L_n(\mathbf{w})$ and local loss functions $L^i(\mathbf{w})$ are μ-strongly convex and have L-Lipschitz gradients. As we discussed in Section II, the gap between the expected and the empirical risks corresponding to ns data samples can be bounded as $|L_n(\mathbf{w}) - L(\mathbf{w})| \leq V_{ns}$, with high probability. Next, we formalize this assumption.

Assumption 2. *The approximation error for the expected loss $L(\mathbf{w})$ using ns samples of n nodes in the empirical risk $L_n(\mathbf{w})$ is w.h.p. upper-bounded as $\sup_{\mathbf{w}} |L_n(\mathbf{w}) - L(\mathbf{w})| \leq V_{ns}$, where $V_{ns} = \mathcal{O}(1/ns)$.*

Moreover, we assume that the approximation error for gradients is upper-bounded by $\sup_{\mathbf{w}} \left\|\nabla L_n(\mathbf{w}) - \nabla L(\mathbf{w})\right\| \leq \sqrt{V_{ns}}$, w.h.p.

We now turn our focus to the proposed Algorithm 2.

Theorem 1. *Consider the federated ERM problem in (2) and suppose Assumptions 1 and 2 hold. Let the proposed `FLANP` in Algorithm 2 be initialized with the fastest n_0 nodes in $\{1, \cdots, n_0\}$ and the model $\mathbf{w}_{n_0}$. Moreover, suppose the variance of stochastic local gradients is bounded above by σ^2, i.e., $\mathbb{E}[\|\widetilde{\nabla} L^i(\mathbf{w}) - \nabla L^i(\mathbf{w})\|^2] \leq \sigma^2$ for all nodes i. At any stage of Algorithm 2 with n participating nodes, if for sufficiently small α_n the stepsizes are $\eta_n = \frac{\alpha_n}{\tau_n \sqrt{n}}, \gamma_n = \frac{\sqrt{n}}{2\alpha_n L}$, and each node runs $\tau_n = 1.5s\sigma^2/c$ local updates, where c captures the constant term in the statistical accuracy $V_{ns} = \frac{c}{ns}$, then nodes reach the statistical accuracy of L_n after $R_n = 12\kappa \ln(6)$ rounds of communication.*

Proof: We defer the proof to the Appendix. ∎

The result in Theorem 1 guarantees that if we initialize Algorithm 2 with n_0 fastest nodes and in each stage the participating nodes update their local models according to Algorithm 2 for $\tau = \mathcal{O}(s)$ iterations and $R = \mathcal{O}(\kappa)$ rounds, before doubling the number of participating nodes, then at the end of the final stage in which all N nodes are participating, we reach a model $\mathbf{w}_N$ that attains the statistical accuracy of the empirical risk $L_N(\mathbf{w})$. Specifically, we have $\mathbb{E}[L_N(\mathbf{w}_N) - L_N(\mathbf{w}_N^*)] \leq V_{Ns}$. Note that to obtain the best guarantee, τ_n and R_n are independent of number of participating nodes n, while the stepsizes η_n and γ_n change as the number of participating nodes increases.

B. Wall-Clock Time Analysis

We have thus far established the convergence properties of Algorithm 2. It is, however, equally important to show that it provably mitigates stragglers in a federated learning framework and hence speeds up the overall wall-clock time.

This article has been accepted for publication in IEEE Journal on Selected Areas in Information Theory. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAIT.2022.3205475

In the following, we first characterize the running time of Algorithm 2 and then compare it with the one for straggler-prone FedGATE benchmarks.

Let $T_1 \leq \cdots \leq T_N$ denote the computation times of the N available nodes. We also denote by $\bar{T}_{\texttt{FLANP}}$ the average runtime of FLANP in Algorithm 2 to reach the overall statistical accuracy of the ERM problem corresponding to all nodes defined in (2). As discussed before, at the stage of FLANP with n participating nodes, the slowest node determines the computation time of that stage. More precisely, the computation time of each iteration of FLANP with n participating node per cluster is $T_n = \max\{T_1, \cdots, T_n\}$. Since each stage consists of R communication rounds each with τ local updates, the average run-time of each stage is $R\tau T_n$. Therefore, the overall wall-clock time of Algorithm 2 is on average $\bar{T}_{\texttt{FLANP}} = R\tau(T_{n_0} + T_{2n_0} + \cdots + T_N)$ with $R = 12\kappa\ln(6)$ and $\tau = 1.5s\sigma^2/c$ as characterized in Theorem 1.

This further demonstrates how the adaptive node participation approach incorporates faster nodes in order to save in the overall wall-clock time. As Theorem 1 shows, it suffices for each participating node in the straggler-resilient Algorithm 2 to run $R = \mathcal{O}(\kappa)$ rounds of local updates and $\tau = \mathcal{O}(s)$ iterations per round to reach the final statistical accuracy. Therefore, the overall wall-clock time of Algorithm 2 is order-wise $\bar{T}_{\texttt{FLANP}} = \mathcal{O}(\kappa s\sigma^2(T_{n_0} + T_{2n_0} + \cdots + T_N))$.

To quantify the speedup gain provided by our proposed method, we need to characterize the wall-clock time for the non-adaptive benchmark FedGATE. Note that in this benchmark, all the N available nodes participate in the training from the beginning.

Proposition 2. *The average runtime for the non-adaptive benchmark* FedGATE *to solve the federated ERM problem* (2) *and to reach the statistical accuracy of all the samples of the N nodes is $\bar{T}_{\texttt{FedGATE}} = \mathcal{O}(\kappa s\sigma^2\ln(Ns)T_N)$ where T_N is the unit computation time of the slowest node.*

Proof: We defer the proof to the Appendix. ■

As expected, the result in Proposition 2 indicates that as all N nodes participate in training since the beginning of the algorithm, the overall wall-clock time depends only on the slowest node with computation time $T_N = \max\{T_1, \cdots, T_N\}$.

Thus far, we have characterized the order-wise expressions of the average wall-clock time for FLANP and FedGATE methods as follows

$$\bar{T}_{\texttt{FLANP}} = \mathcal{O}(\kappa s\sigma^2\,(T_{n_0} + T_{2n_0} + \cdots + T_N)),$$
$$\bar{T}_{\texttt{FedGATE}} = \mathcal{O}(\kappa s\sigma^2\ln(Ns)T_N). \tag{3}$$

To establish the speedup for the straggler-resilient method, we consider a random exponential time model for clients computation times, which has been widely used to capture the computation delay for distributed clusters [40], [41]. We assume that nodes computation time are independent realizations of an exponential random variable and characterize the speedup of the resilient Algorithm 2 compared to the benchmark FedGATE.

Theorem 2. *Suppose the clients' computation times are i.i.d. random variables drawn from an exponential distribution with parameter λ. That is, $T_1, \cdots, T_N \sim \exp(\lambda)$. Then, the speedup gain of the* FLANP *Algorithm 2 compared to the naive* FedGATE *method is*

$$\frac{\mathbb{E}[\bar{T}_{\texttt{FLANP}}]}{\mathbb{E}[\bar{T}_{\texttt{FedGATE}}]} \leq \mathcal{O}\left(\frac{1}{\ln(Ns)}\right).$$

Proof: We defer the proof to the Appendix. ■

Theorem 2 establishes a $\mathcal{O}(\ln(Ns))$ speedup gain for FLANP compared to its non-adaptive and straggler-prone benchmark FedGATE, when the clients' computation times are drawn from a random exponential time model.

We have so far considered device heterogeneous federated clients with potentially well-spread computation speeds and demonstrated the speedup gain obtained by adaptive node participation approach, particularly in Theorem 2. We would like to add that our method provides provable speedups even for device *homogeneous* clients with identical speeds, i.e., $T_1 = \cdots = T_N$. Comparing the expected runtimes in (3) yields that FLANP in Algorithm 2 slashes the expected wall-clock time of FedGATE by a factor $\ln(Ns)/\ln(N)$. This observation demonstrates that the adaptive node participation approach results in two different speedup gains: (i) leveraging faster nodes to speedup the learning and (ii) adaptively increase the effective sample size by participating more clients.

V. Numerical Experiments

We conduct various numerical experiments for convex and nonconvex risks and evaluate the performance of the proposed method versus other benchmarks.

Benchmarks. Bellow is a brief description for multiple federated learning benchmarks that we use to compare with the proposed FLANP in Algorithm 2. In these benchmarks, all the available N nodes participate in the training process.

- FedAvg [4]. Nodes update their local model using a simple SGD rule for τ local iterations in each round, before uploading to the server.
- FedGATE [11]. This is the subroutine used in Algorithm 2. Here we consider it as a benchmark running with all the available N nodes with model update rule similar to the subroutine in Algorithm 2.
- FedNova [8]. In each round, each node i updates its local model for τ_i iterations where τ_is vary across the nodes. To mitigate the heterogeneity in τ_is, the server aggregates normalized updates (w.r.t. τ_i) from the clients and updates the global model.

We compare the performance of FLANP with such benchmarks in terms of communication rounds and wall-clock time. We examine FLANP against the benchmarks under both full and partial node participation scenarios and highlight its practicality. We consider computation speeds that are uniformly distributed and exponentially distributed.

Uniform computation speeds.

Data and Network. We use three main datasets for different problems: MNIST ($60,000$ training, $10,000$ test samples), CIFAR10 ($50,000$ training, $10,000$ test samples) and synthetic ($10,000$ samples) datasets. To implement our algorithm, we employ a federated network of $N \in \{20, 50, 100\}$ heterogeneous clients and in order to model the device heterogeneity,

This article has been accepted for publication in IEEE Journal on Selected Areas in Information Theory. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAIT.2022.3205475

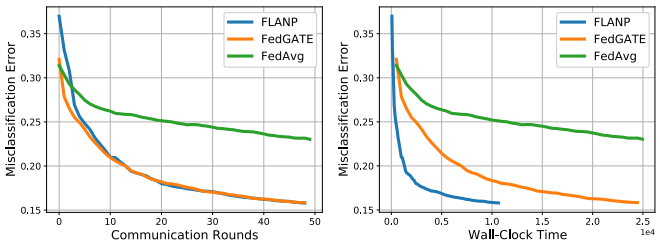


Fig. 1: Logistic Regression over MNIST

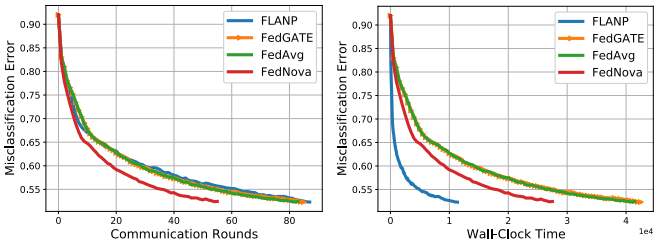


Fig. 2: NN on CIFAR10

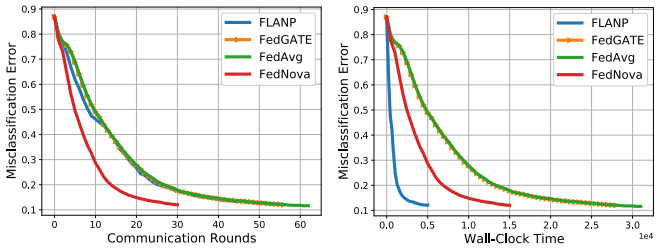


Fig. 3: NN on MNIST

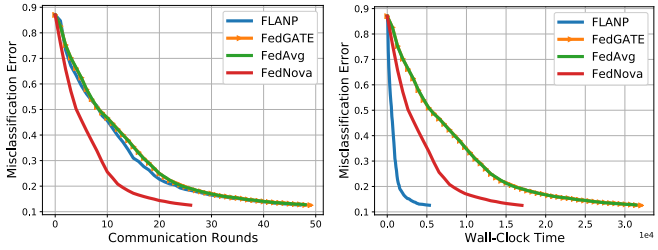
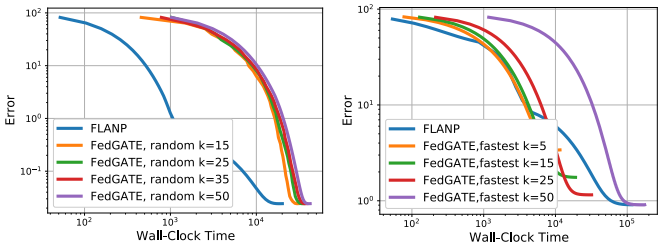
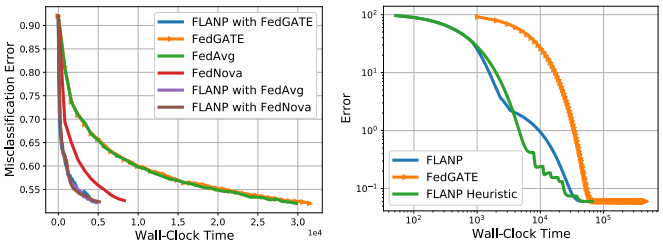


Fig. 4: NN on MNIST



(a) k randomly picked. (b) k fastest picked.

Fig. 5: Partial node participation



(a) Other FL solvers. (b) Heuristics FLANP.

Fig. 6: Heuristics FLANP and other solvers

we realize and then fix the computation speed of each node i, i.e. T_i from the interval $[50, 500]$ uniformly at random.

Logistic Regression. We use the MNIST dataset to learn a multi-class logistic regression model. In a network of $N = 50$ nodes, each client stores $s = 1200$ samples from the MNIST dataset. As demonstrated in Figure 1 (left), FLANP is slightly outperformed by FedGATE at the initial rounds. This is however expected as FLANP starts with only a fraction of nodes participating which leads to less accurate models. With respect to wall-clock time however, FLANP outperforms both FedAvg and FedGATE benchmarks due to the fact that the initial participating nodes are indeed the fastest ones. As Figure 1 (right) shows, the adaptive node participation approach leads FLANP to speedup gains of up to $2.1\times$ compared to FedGATE.

Neural Network Training. We train a fully connected neural network with two hidden layers with 128 and 64 neurons and compare with three other benchmarks including FedNova which is stragglers-resilient. We conduct two sets of experiments over CIFAR10 and MNIST on a network of $N = 20$ clients, as demonstratd in Figures 2 and 3 where FLANP accelerates the training by up to $3\times$ compared to FedNova.

Random exponential computation speeds. We conduct another set of experiments using the same setup described earlier. However, we here pick the clients' computation speed to be i.i.d. random exponential variables, i.e. consistent with Theorem 2. We train a fully connected neural network with two hidden layers with 128 and 64 neurons on MNIST and compare with benchmarks FedAvg, FedGATE and FedNova as demonstrated in Figure 4.

Comparison with partial node participation methods. Thus far, we have compared the FLANP method with federated benchmarks in which *all* of the available nodes participate in training in every round. To demonstrate the resiliency of FLANP to partial node participation methods, we consider two different scenarios. First, we compare the wall-clock time of a neural network training of FLANP with partial node participation FedGATE in which only k out of $N = 50$ nodes are randomly picked and participate in each round. As demonstrated in Figure 5(a), FLANP is significantly faster than FedGATE with partial node participation. Second, we consider the case that the k participating nodes are not randomly picked, but are the fastest clients. As shown in Figure 5(b), although partial participation methods with k fastest nodes begin to outperform FLANP, towards the end of the training, they suffer from higher training error saturation as the data samples of *only* k nodes contribute in the learned model and hence the final model is significantly inaccurate.

FLANP with other federated solvers. To illustrate the compatibility of FLANP with solvers other than FedGATE, we train the neural network on MNIST and employ FedAvg

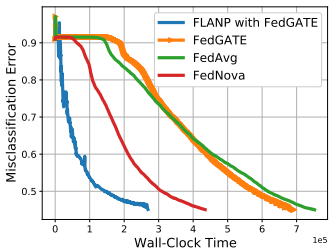


Fig. 7: NN on FEMNIST

and FedNova as solvers of FLANP. As shown in Figure 6(a), FLANP is able to significantly speedup all three solvers.

FLANP on other datasets (FEMNIST). Next, we demonstrate the performance of FLANP with another commonly used dataset that is FEMNIST which includes lower-case and upper-case letters and digits (Figure 7). In our implementation, we include $60,000$ samples from upper-case letters and digits and implement over a federated network of 60 clients.

Lastly, we note that from the practical point of view, there are several heuristic approaches to estimate the constant parameters μ, c, V_{ns} in Algorithm 2. We conducted an experiment to learn a linear regression model with Gaussian synthetic data in which none of the constants are assumed to be known. Rather, we heuristically tune the threshold for each stage transition (i.e. doubling the nodes) by monitoring the norm of the global gradient and successively halving the threshold. As shown in Figure 6(b), the performance of such heuristic methods is indeed close to FLANP which highlights its practicality.

VI. Concluding Remarks

In this paper, we targeted straggler and system heterogeneity challenge in federated learning frameworks and proposed an adaptive node participation scheme to mitigate slow nodes during the training, namely FLANP. In our proposal, the training begins with only a handful of devices which are the fastest among the total N available nodes in the network. After the trained model on such devices reaches their corresponding statistical accuracy, FLANP doubles the number of participating nodes. We rigorously discussed how such doubling procedure enables the trained model at the end of each stage to be a proper warm-up initial model for the next stage. Doubling the participants continues till all the N nodes are incorporated in the training. For strongly convex objectives, we characterized the convergence guarantees of FLANP when combined with FedGATE as the inner federated learning solver. We also established order-wise speedup gain of the proposed adaptive methods compared to its non-adaptive counter method. Our numerical experiments also demonstrate significant speedups in different convex and non-convex scenarios, where we highlighted the practicality of the proposed method as well.

References

[1] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani, "Adaptive node participation for straggler-resilient federated learning," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8762–8766.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *arXiv preprint arXiv:1908.07873*, 2019.

[3] 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," *arXiv preprint arXiv:1912.04977*, 2019.

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

[5] S. U. Stich, "Local sgd converges fast and communicates little," in *ICLR 2019 ICLR 2019 International Conference on Learning Representations*, no. CONF, 2019.

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

[7] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[8] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *arXiv preprint arXiv:2007.07481*, 2020.

[9] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, and R. Pedarsani, "Robust and communication-efficient collaborative learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 8388–8399.

[10] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for on-device federated learning," *arXiv preprint arXiv:1910.06378*, 2019.

[11] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," *arXiv preprint arXiv:2007.01154*, 2020.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[13] A. Reisizadeh, F. Farnia, R. Pedarsani, and A. Jadbabaie, "Robust federated learning: The case of affine distribution shifts," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[14] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*, 2019, pp. 4615–4625.

[15] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.

[16] A. Mokhtari and A. Ribeiro, "First-order adaptive sample size methods to reduce complexity of empirical risk minimization," in *NeurIPS*, 2017.

[17] A. Mokhtari, A. Ozdaglar, and A. Jadbabaie, "Efficient nonconvex empirical risk minimization via adaptive sample size methods," in *AISTATS*, 2019.

[18] A. Mokhtari, H. Daneshmand, A. Lucchi, T. Hofmann, and A. Ribeiro, "Adaptive Newton method for empirical risk minimization to statistical accuracy," in *NeurIPS*, 2016.

[19] M. Eisen, A. Mokhtari, and A. Ribeiro, "Large scale empirical risk minimization via truncated adaptive Newton method," in *AISTATS*, 2018.

[20] M. Jahani, X. He, C. Ma, A. Mokhtari, D. Mudigere, A. Ribeiro, and M. Takác, "Efficient distributed hessian free algorithm for large-scale empirical risk minimization via accumulating sample strategy," in *AISTATS*, 2020.

[21] X. Zhang, J. Wang, L.-F. Lee, T. Yang, A. Kalra, G. Joshi, and C. Joe-Wong, "Machine learning on volatile instances: Convergence, runtime, and cost tradeoffs," *IEEE/ACM Transactions on Networking*, 2021.

[22] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

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

[24] Z. Huo, Q. Yang, B. Gu, L. C. Huang *et al.*, "Faster on-device training using new federated momentum algorithm," *arXiv preprint arXiv:2002.02090*, 2020.

[25] G. Malinovsky, D. Kovalev, E. Gasanov, L. Condat, and P. Richtarik, "From local sgd to local fixed point methods for federated learning," *arXiv preprint arXiv:2004.01442*, 2020.

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

[27] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," *arXiv preprint arXiv:1708.01012*, 2017.
[28] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," *arXiv preprint arXiv:1910.14425*, 2019.
[29] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Local sgd with periodic averaging: Tighter analysis and adaptive synchronization," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 082–11 094.
[30] A. K. R. Bayoumi, K. Mishchenko, and P. Richtarik, "Tighter theory for local sgd on identical and heterogeneous data," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 4519–4529.
[31] S. U. Stich and S. P. Karimireddy, "The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication," *arXiv preprint arXiv:1909.05350*, 2019.
[32] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd," *arXiv preprint arXiv:1810.08313*, 2018.
[33] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. U. Stich, "A unified theory of decentralized sgd with changing topology and local updates," *arXiv preprint arXiv:2003.10422*, 2020.
[34] X. Liang, S. Shen, J. Liu, Z. Pan, E. Chen, and Y. Cheng, "Variance reduced local sgd with lower communication complexity," *arXiv preprint arXiv:1912.12844*, 2019.
[35] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.
[36] O. Bousquet, "Concentration inequalities and empirical processes theory applied to the analysis of learning algorithms," Ph.D. dissertation, École Polytechnique: Department of Applied Mathematics Paris, France, 2002.
[37] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.
[38] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford, "Competing with the empirical risk minimizer in a single pass," in *Conference on learning theory*, 2015, pp. 728–763.
[39] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik, "Distributed learning with compressed gradient differences," *arXiv preprint arXiv:1901.09269*, 2019.
[40] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.
[41] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.

Amirhossein Reisizadeh (Member, IEEE) is currently a Postdoctoral Associate at the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology. He received his Ph.D. and master's degree both in Electrical and Computer Engineering from UC Santa Barbara and UCLA, in 2021 and 2016, respectively. He also received his bachelor's degree from Sharif University of Technology, Tehran, Iran, in 2014. He was a finalist in the Qualcomm Innovation Fellowship program in 2019. His primary research interests include optimization for machine learning, distributed and federated learning, and data-driven decision making.

Isidoros Tziotis is a Ph.D. student in the ECE Department (DICE track), at the University of Texas at Austin, advised by Prof. Aryan Mokhtari. He is also a part of the Wireless Networking and Communications Group (WNCG). His research interest include theory and applications of optimization in large-scale machine learning and data science problems, federated learning, decentralized optimization, and adaptive optimization. Before joining UT Austin, he completed his undergraduate studies at the Department of Informatics and Telecommunications and received his M.Sc. degree on Logic, Algorithms and Theory of Computation from the Department of Mathematics at the University of Athens.

amed Hassani is currently an Assistant Professor of Electrical and Systems Engineering department as well as the Computer and Information Systems department, and the Statistics department at the University of Pennsylvania. Prior to that, he was a research fellow at Simons Institute for the Theory of Computing (UC Berkeley) affiliated with the program of Foundations of Machine Learning, and a post-doctoral researcher in the Institute of Machine Learning at ETH Zurich. He received a Ph.D. degree in Computer and Communication Sciences from EPFL, Lausanne. He is the recipient of the 2014 IEEE Information Theory Society Thomas M. Cover Dissertation Award, 2015 IEEE International Symposium on Information Theory Student Paper Award, 2017 Simons-Berkeley Fellowship, 2018 NSFCRII Research Initiative Award, 2020 Air Force Office of Scientific Research (AFOSR) Young Investigator Award, 2020 National Science Foundation (NSF) CAREER Award, and 2020 Intel Rising Star award. He has recently been selected as the distinguished lecturer of the IEEE Information Theory Society in 2022-2023.

Aryan Mokhtari is an Assistant Professor in the Electrical and Computer Engineering Department of the University of Texas at Austin (UT Austin) where he holds the Fellow of Texas Instruments/Kilby. Before joining UT Austin, he was a Postdoctoral Associate in the Laboratory for Information and Decision Systems (LIDS) at the Massachusetts Institute of Technology (MIT), from January 2018 to July 2019. Before that, he was a Research Fellow at the Simons Institute for the Theory of Computing at the University of California, Berkeley, for the program on "Bridging Continuous and Discrete Optimization", from August to December 2017. Prior to that, he was a graduate student at the University of Pennsylvania (Penn) where he received his M.Sc. and Ph.D. degrees in electrical and systems engineering in 2014 and 2017, respectively, and his A.M. degree in statistics from the Wharton School in 2017. During his graduate study, he was a Research Intern with the Big-Data Machine Learning Group at Yahoo!, Sunnyvale, CA, USA, from June to August 2016. Dr. Mokhtari received his B.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2011. His research interests include the areas of optimization, machine learning, and artificial intelligence. His current research focuses on the theory and applications of convex and non-convex optimization in large-scale machine learning and data science problems. He has received several awards and fellowships, including the Army Research Office (ARO) Early Career Program Award, the Simons-Berkeley Research Fellowship, and Penn's Joseph and Rosaline Wolf Award for Best Doctoral Dissertation in electrical and systems engineering.

Ramtin Pedarsani (Senior Member, IEEE) is an Associate Professor in the ECE Department at the University of California, Santa Barbara. He received the B.Sc. degree in electrical engineering from the University of Tehran, Tehran, Iran, in 2009, the M.Sc. degree in communication systems from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 2011, and his Ph.D. from the University of California, Berkeley, in 2015. His research interests include machine learning, information and coding theory, networks, and transportation systems. Ramtin is a recipient of the Communications Society and Information Theory Society Joint Paper Award in 2020, the best paper award in the IEEE International Conference on Communications (ICC) in 2014, and the NSF CRII award in 2017.