FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization

Amirhossein Reisizadeh
UC Santa Barbara

Aryan Mokhtari
UT Austin

Hamed Hassani
UPenn

Ali Jadbabaie
MIT

Ramtin Pedarsani
UC Santa Barbara

Abstract

Federated learning is a distributed framework according to which a model is trained over a set of devices, while keeping data localized. This framework faces several systems-oriented challenges which include (i) *communication bottleneck* since a large number of devices upload their local updates to a parameter server, and (ii) *scalability* as the federated network consists of millions of devices. Due to these systems challenges as well as issues related to statistical heterogeneity of data and privacy concerns, designing a provably efficient federated learning method is of significant importance yet it remains challenging. In this paper, we present FedPAQ, a communication-efficient **Fed**erated Learning method with **P**eriodic **A**veraging and **Q**uantization. FedPAQ relies on three key features: (1) periodic averaging where models are updated locally at devices and only periodically averaged at the server; (2) partial device participation where only a fraction of devices participate in each round of the training; and (3) quantized message-passing where the edge nodes quantize their updates before uploading to the parameter server. These features address the communications and scalability challenges in federated learning. We also show that FedPAQ achieves near-optimal theoretical guarantees for strongly convex and non-convex loss functions and empirically demonstrate the communication-computation tradeoff provided by our method.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

1 Introduction

In many large-scale machine learning applications, data is acquired and processed at the edge nodes of the network such as mobile devices, users' devices, and IoT sensors. *Federated Learning* is a novel paradigm that aims to train a statistical model at the "edge" nodes as opposed to the traditional distributed computing systems such as data centers [Konečnỳ et al., 2016, Li et al., 2019a]. The main objective of federated learning is to fit a model to data generated from network devices without continuous transfer of the massive amount of collected data from edge of the network to back-end servers for processing.

Federated learning has been deployed by major technology companies with the goal of providing privacy-preserving services using users' data [Bonawitz et al., 2019]. Examples of such applications are learning from wearable devices [Huang et al., 2018], learning sentiment [Smith et al., 2017], and location-based services [Samarakoon et al., 2018]. While federated learning is a promising paradigm for such applications, there are several challenges that remain to be resolved. In this paper, we focus on two significant challenges of federated learning, and propose a novel federated learning algorithm that addresses the following two challenges:

(1) **Communication bottleneck.** Communication bandwidth is a major bottleneck in federated learning as a large number of devices attempt to communicate their local updates to a central parameter server. Thus, for a communication-efficient federated learning algorithm, it is crucial that such updates are sent in a compressed manner and infrequently.

(2) **Scale.** A federated network typically consists of thousands to millions of devices that may be active, slow, or completely inactive during the training procedure. Thus, a proposed federated learning algorithm should be able to operate efficiently with partial device participation or random sampling of devices.

The goal of this paper is to develop a provably efficient federated learning algorithm that addresses the above-mentioned systems challenges. More precisely, we consider the task of training a model in a federated learning setup where we aim to find an *accurate* model over a collection of n distributed nodes. In this setting, each node contains m independent and identically distributed samples from an unknown probability distribution and a parameter server helps coordination between the nodes. We focus on solving the empirical risk minimization problem for a federated architecture while addressing the challenges mentioned above. In particular, we consider both strongly convex and non-convex settings and provide sharp guarantees on the performance of our proposed algorithm.

Contributions. In this work, we propose FedPAQ, a communication-efficient **Fed**erated learning algorithm with **P**eriodic **A**veraging and **Q**uantization, which addresses federated learning systems' bottlenecks. In particular, FedPAQ has three key features that enable efficient federated learning implementation:

(1) FedPAQ allows the nodes (users) of the network to run local training before synchronizing with the parameter server. In particular, each node iteratively updates its local model for a period of iterations using the stochastic gradient descent (SGD) method and then uploads its model to the parameter server where all the received models are averaged periodically. By tuning the parameter which corresponds to the number of local iterations before communicating to the server, periodic averaging results in slashing the number of communication rounds and hence the total communication cost of the training process.

(2) FedPAQ captures the constraint on availability of active edge nodes by allowing a partial node participation. That is, in each round of the method, only a fraction of the total devices–which are the active ones–contribute to train the model. This procedure not only addresses the scalability challenge, but also leads to smaller communication load compared to the case that all nodes participate in training the learning model.

(3) In FedPAQ, nodes only send a quantized version of their local information to the server at each round of communication. As the training models are of large sizes, quantization significantly helps reducing the communication overhead on the network.

While these features have been proposed in the literature, to the best of our knowledge, FedPAQ is the first federated learning algorithm that simultaneously incorporates these features and provides near-optimal theoretical guarantees on its statistical accuracy, while being communication-efficient via periodic averaging, partial node participation and quantization.

In particular, we analyze our proposed FedPAQ method for two general class of loss functions: strongly-convex and non-convex. For the strongly-convex setting, we show that after T iterations the squared norm of the distance between the solution of our method and the optimal solution is of $\mathcal{O}(1/T)$ in expectation. We also show that FedPAQ approaches a first-order stationary point for non-convex losses at a rate of $\mathcal{O}(1/\sqrt{T})$. This demonstrates that our method significantly improves the communication-efficiency of federated learning while preserving the optimality and convergence guarantees of the baseline methods. In addition, we would like to highlight that our theoretical analysis is based on few relaxed and customary assumptions which yield more technical challenges compared to the existing works with stronger assumptions and hence acquires novel analytical techniques. More explanations will be provided in Section 4.

Related Work. The main premise of federated learning has been collective learning using a network of common devices such as phones and tablets. This framework potentially allows for smarter models, lower latency, and less power consumption, all while ensuring privacy. Successfully achieving these goals in practice requires addressing key challenges of federated learning such as communication complexity, systems heterogeneity, privacy, robustness, and heterogeneity of the users. Recently, many federated methods have been considered in the literature which mostly aim at reducing the communication cost. McMahan et al. [2016] proposed the FedAvg algorithm, where the global model is updated by averaging local SGD updates. Guha et al. [2019] proposed one-shot federated learning in which the master node learns the model after a single round of communication.

Optimization methods for federated learning are naturally tied with tools from stochastic and distributed optimization. Minibatch stochastic gradient descent distributed optimization methods have been largely studied in the literature without considering the communication bottleneck. Addressing the communication bottleneck via quantization and compression in distributed learning has recently gained considerable attention for both master-worker [Alistarh et al., 2017, Bernstein et al., 2018, Seide et al., 2014, Smith et al., 2016] and masterless topologies [Koloskova et al., 2019, Reisizadeh et al., 2019a, Wang et al., 2019, Zhang et al., 2018]. Moreover, Wang et al. [2019] reduces the communication delay by decomposing the graph.

Local updates, as another approach to reduce the communication load in distributed learning has been stud-

ied in the literature, where each learning node carries out multiple local updates before sharing with the master or its neighboring nodes. Stich [2018] considered a master-worker topology and provides theoretical analysis for the convergence of local-SGD method. Lin et al. [2018] introduced a variant of local-SGD namely post-local-SGD which demonstrates empirical improvements over local-SGD. Wang and Joshi [2018] provided a general analysis of such cooperative method for decentralized settings as well.

Statistical heterogeneity of users' data points is another major challenge in federated learning. To address this heterogeneity, other methods such as multitask learning and meta learning have been proposed to train multiple local models [Li et al., 2019b, Nichol et al., 2018, Smith et al., 2017]. Many methods have been proposed to address systems heterogeneity and in particular stragglers in distributed learning using coding theory, e.g., [Dutta et al., 2016, Lee et al., 2018, Reisizadeh et al., 2019b, Tandon et al., 2016, Yu et al., 2017]. Another important challenge in federated learning is to preserve privacy in learning [Duchi et al., 2014]. Agarwal et al. [2018], McMahan et al. [2017] proposed privacy-preserving methods for distributed and federated learning using differential privacy techniques. Federated heavy hitters discovery with differential privacy was proposed in [Zhu et al., 2019].

Robustness against adversarial devices is another challenge in federated learning and distributed learning that has been studied in [Chen et al., 2017, Ghosh et al., 2019, Yin et al., 2018]. Finally, several works have considered communication-efficient collaborative learning where there is no master node, and the computing nodes learn a model collaboratively in a decentralized manner [Doan et al., 2018, Koloskova et al., 2019, Lalitha et al., 2019, Reisizadeh et al., 2019a, Zhang et al., 2018]. While such techniques are related to federated learning, the network topology in masterless collaborative learning is fundamentally different.

2 Federated Learning Setup

In this paper, we focus on a federated architecture where a parameter server (or server) aims at finding a model that performs well with respect to the data points that are available at different nodes (users) of the network, while nodes exchange their local information with the server. We further assume that the data points for all nodes in the network are generated from a common probability distribution. In particular, we consider the following stochastic learning problem

$$\min_{\mathbf{x}} f(\mathbf{x}) \coloneqq \min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}), \tag{1}$$

where the local objective function of each node i is defined as the expected loss of its local sample distributions

$$f_i(\mathbf{x}) \coloneqq \mathbb{E}_{\xi \sim \mathcal{P}^i} [\ell(\mathbf{x}, \xi)]. \tag{2}$$

Here $\ell : \mathbb{R}^p \times \mathbb{R}^u \to \mathbb{R}$ is a stochastic loss function, $\mathbf{x} \in \mathbb{R}^p$ is the model vector, and $\xi \in \mathbb{R}^u$ is a random variable with unknown probability distribution $\mathcal{P}^i$. Moreover, $f : \mathbb{R}^p \to \mathbb{R}$ denotes the expected loss function also called population risk. In our considered federated setting, each of the n distributed nodes generates a local loss function according to a distribution $\mathcal{P}^i$ resulting in a local stochastic function $f_i(\mathbf{x}) \coloneqq \mathbb{E}_{\xi \sim \mathcal{P}^i} [\ell(\mathbf{x}, \xi)]$. A special case of this formulation is when each node i maintains a collection of m samples from distribution $\mathcal{P}^i$ which we denote by $\mathcal{D}^i = \{\xi_1^i, \cdots, \xi_m^i\}$ for $i \in [n]$. This results in the following empirical risk minimization problem over the collection of nm samples in $\mathcal{D} \coloneqq \mathcal{D}^1 \cup \cdots \cup \mathcal{D}^n$:

$$\min_{\mathbf{x}} L(\mathbf{x}) = \min_{\mathbf{x}} \frac{1}{nm} \sum_{\xi \in \mathcal{D}} \ell(\mathbf{x}, \xi), \tag{3}$$

We denote the optimal model $\mathbf{x}^*$ as the solution to the expected risk minimization problem in (1) and denote the minimum loss $f^* \coloneqq \min_{\mathbf{x}} f(\mathbf{x}) = f(\mathbf{x}^*)$ as the optimal objective function value of the expected risk minimization problem in (1). In this work, we focus on the case that the data over the n nodes is independent and identically distributed (i.i.d.), which implies the local distributions are common.

As stated above, our goal is to minimize the expected loss $f(\mathbf{x})$. However, due to the fact that we do not have access to the underlying distribution $\mathcal{P}$, there have been prior works that focus on minimizing the empirical risk $L(\mathbf{x})$ which can be viewed as an approximation of the expected loss $f(\mathbf{x})$. The accuracy of this approximation is determined by the number of samples $N = nm$. It has been shown that for convex losses ℓ, the population risk f is at most $\mathcal{O}(1/\sqrt{nm})$ distant from the empirical risk L, uniformly and with high probability [Bottou and Bousquet, 2008]. That is, $\sup_{\mathbf{x}} |f(\mathbf{x}) - L(\mathbf{x})| \leq \mathcal{O}(1/\sqrt{nm})$ with high probability. This result implies that if each of the n nodes separately minimizes its local empirical loss function, the expected deviation from the local solution and the solution to the population risk minimization problem is of $\mathcal{O}(1/\sqrt{m})$ (note that each node has access to m data samples). However, if the nodes manage to somehow share or synchronize their solutions, then a more accurate solution can be achieved, that is a solution with accuracy of order $\mathcal{O}(1/\sqrt{nm})$. Therefore, when all the mn available samples are leveraged, one can obtain a solution $\hat{\mathbf{x}}$ that satisfies $\mathbb{E}[L(\hat{\mathbf{x}}) - L(\mathbf{x}^*)] \leq \mathcal{O}(1/\sqrt{nm})$. This also implies that $\mathbb{E}[f(\hat{\mathbf{x}}) - \min_{\mathbf{x}} f(\mathbf{x})] \leq \mathcal{O}(1/\sqrt{nm})$.

For the case of non-convex loss function ℓ, however, finding the solution to the expected risk minimization problem in (1) is hard. Even further, finding (or testing) a local optimum is NP-hard in many cases [Murty and Kabadi, 1987]. Therefore, for non-convex losses we relax our main goal and instead look for first-order optimal solutions (or stationary points) for (1). That is, we aim to find a model $\hat{\mathbf{x}}$ that satisfies $\left\|\nabla f(\hat{\mathbf{x}})\right\| \leq \epsilon$ for an arbitrarily small approximation error ϵ. Mei et al. [2018] characterized the gap for the gradients of the two expected risk and empirical risk functions. That is, if the gradient of loss is sub-Gaussian, then with high probability $\sup_{\mathbf{x}}\left\|\nabla L(\mathbf{x}) - \nabla f(\mathbf{x})\right\| \leq \mathcal{O}(1/\sqrt{nm})$. This result further implies that having all the nodes contribute in minimizing the empirical risk results in better approximation for a first-order stationary point of the expected risk L. In summary, our goal in non-convex setting is to find $\hat{\mathbf{x}}$ that satisfies $\left\|\nabla f(\mathbf{x})\right\| \leq \mathcal{O}(1/\sqrt{nm})$ which also implies $\left\|\nabla L(\mathbf{x})\right\| \leq \mathcal{O}(1/\sqrt{nm})$.

3 Proposed FedPAQ Method

In this section, we present our proposed communication-efficient federated learning method called FedPAQ, which consists of three main modules: (1) periodic averaging, (2) partial node participation, and (3) quantized message passing.

3.1 Periodic averaging

As explained in Section 2, to leverage from all the available data samples on the nodes, any training method should incorporate synchronizing the intermediate models obtained at local devices. One approach is to let the participating nodes synchronize their models through the parameter server in each iteration of the training. This, however, implies many rounds of communication between the federated nodes and the parameter server which results in communication contention over the network. Instead, we let the participating nodes conduct a number of local updates and synchronize through the parameter server periodically. To be more specific, once nodes pull an updated model from the server, they update the model locally by running τ iterations of the SGD method and then send proper information to the server for updating the aggregate model. Indeed, this periodic averaging scheme reduces the rounds of communication between server and the nodes and consequently the overall communication cost of training the model. In particular, for the case that we plan to run T iterations of SGD at each node, nodes need to communicate with the server $K = T/\tau$ rounds, hence reducing the total communication cost by a factor of $1/\tau$.

Choosing a larger value of τ indeed reduces the rounds of communication for a fixed number of iterations T. However, if our goal is to obtain a specific accuracy ε, choosing a very large value for τ is not necessarily optimal as by increasing τ the noise of the system increases and the local models approach the local optimal solutions instead of the global optimal solution. Hence, we might end up running more iterations T to achieve a specific accuracy ε comparing to a case that τ is small. Indeed, a crucial question that we need to address is finding the optimal choice of τ for minimizing the overall communication cost of the process.

3.2 Partial node participation

In a federated network, often there is a large number of devices such as smart phones communicating through a base station. On one hand, base stations have limited download bandwidth and hence only a few of devices are able to simultaneously upload their messages to the base station. Due to this limitation the messages sent from the devices will be pipelined at the base station which results in a dramatically slow training. On the other hand, having all of the devices participate through the whole training process induces a large communication overhead on the network which is often costly. Moreover, in practice not all the devices contribute in each round of the training. Indeed, there are multiple factors that determine whether a device can participate in the training [McMahan and Ramage, 2017]: a device should be available in the reachable range of the base station; a device should be idle, plugged in and connected to a free wireless network during the training; etc.

Our proposed FedPAQ method captures the restrictions mentioned above. In particular, we assume that among the total of n devices, only r nodes ($r \leq n$) are available in each round of the training. We can also assume that due to the availability criterion described before, such available devices are randomly and uniformly distributed over the network [Sahu et al., 2018]. In summary, in each period $k = 0, 1, \cdots, K-1$ of the training algorithm, the parameter server sends its current model $\mathbf{x}_k$ to all the r nodes in subset $\mathcal{S}_k$, which are distributed uniformly at random among the total n nodes, i.e., $\Pr[\mathcal{S}_k] = 1/\binom{n}{r}$.

3.3 Quantized message-passing

Another aspect of the communication bottleneck in federated learning is the limited uplink bandwidth at the devices which makes the communication from devices to the parameter server slow and expensive. Hence, it is critical to reduce the size of the uploaded messages from the federated devices [Li et al., 2019a]. Our proposal is to employ quantization operators on the transmitted massages. Depending on the accuracy

of the quantizer, the network communication overhead is reduced by exchanging the quantized updates.

In the proposed FedPAQ, each node $i \in \mathcal{S}_k$ obtains the model $\mathbf{x}_{k,\tau}^{(i)}$ after running τ local iterations of an optimization method (possibly SGD) on the most recent model $\mathbf{x}_k$ that it has received form the server. Then each node i applies a quantizer operator $Q(\cdot)$ on the difference between the received model and its updated model, i.e., $\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k$, and uploads the quantized vector $Q(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k)$ to the parameter server. Once these quantized vectors are sent to the server, it decodes the quantized signals and combines them to come up with a new model $\mathbf{x}_{k+1}$.

Next, we describe a widely-used random quantizer.

Example 1 (Low-precision quantizer [Alistarh et al., 2017]). *For any variable $\mathbf{x} \in \mathbb{R}^p$, the low precision quantizer $Q^{LP} : \mathbb{R}^p \to \mathbb{R}^p$ is defined as below*

$$Q_i^{LP}(\mathbf{x}) = \|\mathbf{x}\| \cdot \text{sign}(x_i) \cdot \xi_i(\mathbf{x}, s), \quad i \in [p], \tag{4}$$

where $\xi_i(\mathbf{x}, s)$ is a random variable taking on value $^{l+1}/_s$ with probability $\frac{|x_i|}{\|\mathbf{x}\|} s - l$ and $^l/_s$ otherwise. Here, the tuning parameter s corresponds to the number of quantization levels and $l \in [0, s)$ is an integer such that $^{|x_i|}/_{\|\mathbf{x}\|} \in [^l/_s, ^{l+1}/_s)$.

3.4 Algorithm update

Now we use the building blocks developed in Sections 3.1-3.3 to precisely present FedPAQ. Our proposed method consists of K periods, and during a period, each node performs τ local updates, which results in total number of $T = K\tau$ iterations. In each period $k = 0, \cdots, K-1$ of the algorithm, the parameter server picks $r \leq n$ nodes uniformly at random which we denote by $\mathcal{S}_k$. The parameter server then broadcasts its current model $\mathbf{x}_k$ to all the nodes in $\mathcal{S}_k$ and each node $i \in \mathcal{S}_k$ performs τ local SGD updates using its local dataset. To be more specific, let $\mathbf{x}_{k,t}^{(i)}$ denote the model at node i at t-th iteration of the k-th period. At each local iteration $t = 0, \cdots, \tau-1$, node i updates its local model according to the following rule:

$$\mathbf{x}_{k,t+1}^{(i)} = \mathbf{x}_{k,t}^{(i)} - \eta_{k,t} \widetilde{\nabla} f_i \left(\mathbf{x}_{k,t}^{(i)} \right), \tag{5}$$

where the stochastic gradient $\widetilde{\nabla} f_i$ is computed using a random sample[1] picked from the local dataset $\mathcal{D}^i$. Note that all the nodes begin with a common initialization $\mathbf{x}_{k,0}^{(i)} = \mathbf{x}_k$. After τ local updates, each node computes the overall update in that period, that is $\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k$, and uploads a quantized update $Q(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k)$ to the parameter server. The parameter server then aggregates the r received quantized local updates and computes the next model according to

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{r} \sum_{i \in \mathcal{S}_k} Q \left(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k \right), \tag{6}$$

and the procedure is repeated for K periods. The proposed method is formally summarized in Algorithm 1.

[1] The method can be easily made compatible with using a mini-batch during each iteration.

Algorithm 1 FedPAQ

for $k = 0, 1, \cdots, K-1$ **do**
  server picks r nodes $\mathcal{S}_k$ uniformly at random
  server sends $\mathbf{x}_k$ to nodes in $\mathcal{S}_k$
  for node $i \in \mathcal{S}_k$ **do**
    $\mathbf{x}_{k,0}^{(i)} \leftarrow \mathbf{x}_k$
    for $t = 0, 1, \cdots, \tau - 1$ **do**
      compute stochastic gradient
      $\widetilde{\nabla} f_i(\mathbf{x}) = \nabla \ell(\mathbf{x}, \xi)$ for a $\xi \in \mathcal{P}^i$
      set $\mathbf{x}_{k,t+1}^{(i)} \leftarrow \mathbf{x}_{k,t}^{(i)} - \eta_{k,t} \widetilde{\nabla} f_i(\mathbf{x}_{k,t}^{(i)})$
    end for
    send $Q(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k)$ to the server
  end for
  server finds $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \frac{1}{r} \sum_{i \in \mathcal{S}_k} Q(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k)$
end for

4 Convergence Analysis

In this section, we present our theoretical results on the guarantees of the FedPAQ method. We first consider the strongly convex setting and state the convergence guarantee of FedPAQ for such losses in Theorem 1. Then, in Theorem 2, we present the overall complexity of our method for finding a first-order stationary point of the aggregate objective function f, when the loss function ℓ is non-convex (All proofs are provided in the supplementary material). Before that, we first mention three customary assumptions required for both convex and non-convex settings.

Assumption 1. *The random quantizer $Q(\cdot)$ is unbiased and its variance grows with the squared of l_2-norm of its argument, i.e.,*

$$\mathbb{E}\left[Q(\mathbf{x})|\mathbf{x}\right] = \mathbf{x}, \quad \mathbb{E}\left[\left\|Q(\mathbf{x}) - \mathbf{x}\right\|^2 |\mathbf{x}\right] \leq q\|\mathbf{x}\|^2, \tag{7}$$

for some positive real constant q and any $\mathbf{x} \in \mathbb{R}^p$.

Assumption 2. *The loss functions f_i are L-smooth with respect to $\mathbf{x}$, i.e., for any $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^p$, we have $\left\|\nabla f_i(\mathbf{x}) - \nabla f_i(\hat{\mathbf{x}})\right\| \leq L\|\mathbf{x} - \hat{\mathbf{x}}\|$.*

Assumption 3. *Stochastic gradients $\widetilde{\nabla} f_i(\mathbf{x})$ are unbiased and variance bounded, i.e., $\mathbb{E}_\xi[\widetilde{\nabla} f_i(\mathbf{x})] = \nabla f_i(\mathbf{x})$ and $\mathbb{E}_\xi[\|\widetilde{\nabla} f_i(\mathbf{x}) - \nabla f_i(\mathbf{x})\|^2] \leq \sigma^2$.*

The conditions in Assumption 1 ensure that output of quantization is an unbiased estimator of the input with a variance that is proportional to the norm-squared of

the input. This condition is satisfied with most common quantization schemes including the low-precision quantizer introduced in Example 1. Assumption 2 implies that the gradients of local functions ∇f_i and the aggregated objective function ∇f are also L-Lipschitz continuous. The conditions in Assumption 3 on the bias and variance of stochastic gradients are also customary. Note that this is a much weaker assumption compared to the one that uniformly bounds the expected norm of the stochastic gradient.

Challenges in analyzing the FedPAQ method. Here, we highlight the main theoretical challenges in proving our main results. As outlined in the description of the proposed method, in the k-th round of FedPAQ, each participating node i updates its local model for τ iterations via SGD method in (5). Let us focus on a case that we use a constant stepsize for the purpose of this discussion. First consider the naive parallel SGD case which corresponds to $\tau = 1$. The updated local model after $\tau = 1$ local update is

$$\mathbf{x}_{k,\tau}^{(i)} = \mathbf{x}_{k,0}^{(i)} - \eta \widetilde{\nabla} f_i \left(\mathbf{x}_{k,0}^{(i)} \right). \tag{8}$$

Note that $\mathbf{x}_{k,0}^{(i)} = \mathbf{x}_k$ is the parameter server's model sent to the nodes. Since we assume the stochastic gradients are unbiased estimators of the gradient, it yields that the local update $\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k$ is an unbiased estimator of $-\eta \nabla f(\mathbf{x}_k)$ for every participating node. Hence, the aggregated updates at the server and the updated model $\mathbf{x}_{k+1}$ can be simply related to the current model $\mathbf{x}_k$ as one step of parallel SGD. However, this is not the case when the period length τ is larger than 1. For instance, in the case that $\tau = 2$, the local updated model after $\tau = 2$ iterations is

$$\mathbf{x}_{k,\tau}^{(i)} = \mathbf{x}_k - \eta \widetilde{\nabla} f_i \left(\mathbf{x}_k \right) - \eta \widetilde{\nabla} f_i \left(\mathbf{x}_k - \eta \widetilde{\nabla} f_i \left(\mathbf{x}_k \right) \right). \tag{9}$$

Clearly, $\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k$ is not an unbiased estimator of $-\eta \nabla f(\mathbf{x}_k)$ or $-\eta \nabla f(\mathbf{x}_k - \eta \nabla f(\mathbf{x}_k))$. This demonstrates that the aggregated model at server cannot be treated as τ iterations of parallel SGD, since each local update contains a bias. Indeed, this bias gets propagated when τ gets larger. For our running example $\tau = 2$, the variance of the bias, i.e. $\mathbb{E}\|\eta \widetilde{\nabla} f_i(\mathbf{x}_k - \eta \widetilde{\nabla} f_i(\mathbf{x}_k))\|^2$ is not uniformly bounded either (Assumption 3), which makes the analysis even more challenging compared to the works with bounded gradient assumption (e.g. [Stich, 2018, Yu et al., 2019]).

4.1 Strongly convex setting

Now we proceed to establish the convergence rate of the proposed FedPAQ method for a federated setting with strongly convex and smooth loss function ℓ. We first formally state the strong convexity assumption.

Assumption 4. *The loss functions f_i are μ-strongly convex, i.e., for any $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^p$ we have that $\langle \nabla f_i(\mathbf{x}) - \nabla f_i(\hat{\mathbf{x}}), \mathbf{x} - \hat{\mathbf{x}} \rangle \geq \mu \|\mathbf{x} - \hat{\mathbf{x}}\|^2$.*

Theorem 1 (Strongly convex loss)**.** *Consider the sequence of iterates $\mathbf{x}_k$ at the parameter server generated according to the FedPAQ method outlined in Algorithm 1. Suppose the conditions in Assumptions 1–4 are satisfied. Further, let us define the constant B_1 as*

$$B_1 = 2L^2 \left(\frac{q}{n} + \frac{n-r}{r(n-1)} 4(1+q) \right), \tag{10}$$

where q is the quantization variance parameter defined in (7) and r is the number of active nodes at each round of communication. If we set the stepsize in FedPAQ as $\eta_{k,t} = \eta_k = 4\mu^{-1}/k\tau+1$, then for any $k \geq k_0$ where k_0 is the smallest integer satisfying

$$k_0 \geq 4 \max \left\{ \frac{L}{\mu}, 4 \left(\frac{B_1}{\mu^2} + 1 \right), \frac{1}{\tau}, \frac{4n}{\mu^2 \tau} \right\}, \tag{11}$$

the expected error $\mathbb{E}[\|\mathbf{x}_k - \mathbf{x}^*\|]^2$ is bounded above by

$$\mathbb{E}\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq \frac{(k_0\tau+1)^2}{(k\tau+1)^2} \|\mathbf{x}_{k_0} - \mathbf{x}^*\|^2 + C_1 \frac{\tau}{k\tau+1} + C_2 \frac{(\tau-1)^2}{k\tau+1} + C_3 \frac{\tau-1}{(k\tau+1)^2}, \tag{12}$$

where the constants in (12) are defined as

$$C_1 = \frac{16\sigma^2}{\mu^2 n} \left(1 + 2q + 8(1+q) \frac{n(n-r)}{r(n-1)} \right), \quad C_2 = \frac{16eL^2\sigma^2}{\mu^2 n}, \quad C_3 = \frac{256eL^2\sigma^2}{\mu^4 n} \left(n + 2q + 8(1+q) \frac{n(n-r)}{r(n-1)} \right). \tag{13}$$

Remark 1. *Under the same conditions as in Theorem 1 and for a total number of iterations $T = K\tau \geq k_0\tau$ we have the following convergence rate*

$$\mathbb{E}\|\mathbf{x}_K - \mathbf{x}^*\|^2 \leq \mathcal{O}\left(\frac{\tau}{T} \right) + \mathcal{O}\left(\frac{\tau^2}{T^2} \right) + \mathcal{O}\left(\frac{(\tau-1)^2}{T} \right) + \mathcal{O}\left(\frac{\tau-1}{T^2} \right). \tag{14}$$

As expected, the fastest convergence rate is attained when the contributing nodes synchronize with the parameter server in each iteration, i.e. when $\tau = 1$. Theorem 1 however characterizes how large the period length τ can be picked. In particular, any pick of $\tau = o(\sqrt{T})$ ensures the convergence of the FedPAQ to the global optimal for strongly convex losses.

Remark 2. *By setting $\tau = 1$, $q = 0$ and $r = n$, Theorem 1 recovers the convergence rate of vanilla parallel SGD, i.e., $\mathcal{O}(1/T)$ for strongly-convex losses. Our result is however more general since we remove the uniformly bounded assumption on the norm of stochastic gradient. For $\tau \geq 1$, Theorem 1 does not recover the result in [Stich, 2018] due to our weaker condition in Assumption 3. Nevertheless, the same rate $\mathcal{O}(1/T)$ is guaranteed by FedPAQ for constant values of τ.*

4.2 Non-convex setting

We now present the convergence result of FedPAQ for smooth non-convex loss functions.

Theorem 2 (Non-convex Losses). *Consider the sequence of iterates $\mathbf{x}_k$ at the parameter server generated according to the FedPAQ method outlined in Algorithm 1. Suppose the conditions in Assumptions 1–3 are satisfied. Further, let us define the constant B_2 as*

$$B_2 := \frac{q}{n} + \frac{4(n-r)}{r(n-1)}(1+q), \tag{15}$$

where q is the quantization variance parameter defined in (7) and r is the number of active nodes at each round. If the total number of iterations T and the period length τ satisfy the following conditions,

$$T \geq 2, \qquad \tau \leq \frac{\sqrt{B_2^2 + 0.8} - B_2}{8}\sqrt{T}, \tag{16}$$

and we set the stepsize as $\eta_{k,t} = 1/L\sqrt{T}$, then the following first-order stationary condition holds

$$\frac{1}{T}\sum_{k=0}^{K-1}\sum_{t=0}^{\tau-1}\mathbb{E}\left\|\nabla f(\overline{\mathbf{x}}_{k,t})\right\|^2 \leq \frac{2L(f(\mathbf{x}_0)-f^*)}{\sqrt{T}} + N_1\frac{1}{\sqrt{T}} + N_2\frac{\tau-1}{T}, \tag{17}$$

where the constants in (17) are defined as

$$N_1 := (1+q)\frac{\sigma^2}{n}\left(1 + \frac{n(n-r)}{r(n-1)}\right), \quad N_2 := \frac{\sigma^2}{n}(n+1).$$

Remark 3. *The result in Theorem 2 implies the following order-wise rate*

$$\frac{1}{T}\sum_{k=0}^{K-1}\sum_{t=0}^{\tau-1}\mathbb{E}\left\|\nabla f(\overline{\mathbf{x}}_{k,t})\right\|^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \mathcal{O}\left(\frac{\tau-1}{T}\right).$$

Clearly, the fastest convergence rate is achieved for the smallest possible period length, i.e., $\tau = 1$. This however implies that the edge nodes communicate with the parameter server in each iteration, i.e. T rounds of communications which is costly. On the other hand, the conditions (16) in Theorem 2 allow the period length τ to grow up to $\mathcal{O}(\sqrt{T})$ which results in an overall convergence rate of $\mathcal{O}(1/\sqrt{T})$ in reaching an stationary point. This result shows that with only $\mathcal{O}(\sqrt{T})$ rounds of communication FedPAQ can still ensure the convergence rate of $\mathcal{O}(1/\sqrt{T})$ for non-convex losses.

Remark 4. *Theorem 2 recovers the convergence rate of the vanilla parallel SGD [Yu et al., 2019] for non-convex losses as a special case of $\tau = 1$, $q = 0$ and $r = n$. Nevertheless, we remove the uniformly bounded assumption on the norm of the stochastic gradient in our theoretical analysis. We also recover the result in [Wang and Joshi, 2018] when there is no quatization $q = 0$ and we have a full device participation $r = n$.*

It is worth mentioning that for Theorems 1 and 2, one can use a batch of size m for each local SGD update and the same results hold by changing σ^2/n to σ^2/mn.

5 Numerical Results and Discussions

The proposed FedPAQ method reduces the communication load by employing three modules: periodic averaging, partial node participation, and quantization. This communication reduction however comes with a cost in reducing the convergence accuracy and hence requiring more iterations of the training, which we characterized in Theorems 1 and 2. In this section, we empirically study this communication-computation trade-off and evaluate FedPAQ in comparison to other benchmarks. To evaluate the total cost of a method, we first need to specifically model such cost. We consider the total training time as the cost objective which consists of communication and computation time [Berahas et al., 2018, Reisizadeh et al., 2019c]. Consider T iterations of training with FedPAQ that consists of $K = T/\tau$ rounds of communication. In each round, r workers compute τ iterations of SGD with batchsize B and send a quantized vector of size p to the server.

Communication time. We fix a bandwidth BW and define the communication time in each round as the total number of uploaded bits divided by BW. Total number of bits in each round is $r \cdot |Q(p,s)|$, where $|Q(p,s)|$ denotes the number of bits required to encode a quantized vector of dimension p according to a specific quantizer with s levels. In our simulations, we use the low-precision quantizer described in Example 1 and assume it takes pF bits to represent an unquantized vector of length p, where F is typically 32 bits.

Computation time. We consider the well-known shifted-exponential model for gradient computation time [Lee et al., 2017]. In particular, we assume that for any node, computing the gradients in a period with τ iterations and using batchsize B takes a deterministic shift $\tau \cdot B \cdot \texttt{shift}$ plus a random exponential time with mean value $\tau \cdot B \cdot \texttt{scale}^{-1}$, where shift and scale are respectively shift and scale parameters of the shifted-exponential distribution. Total computation time of each round is then the largest local computation time among the r contributing nodes. We also define a communication-computation ratio

$$\frac{C_{\texttt{comm}}}{C_{\texttt{comp}}} = \frac{pF/\texttt{BW}}{\texttt{shift} + 1/\texttt{scale}}$$

as the communication time for a length-p-vector over the average computation time for one gradient vector. This ratio captures the relative cost of communication and computation, and since communication is a major bottleneck, we have $C_{\texttt{comm}}/C_{\texttt{comp}} \gg 1$. In all of our

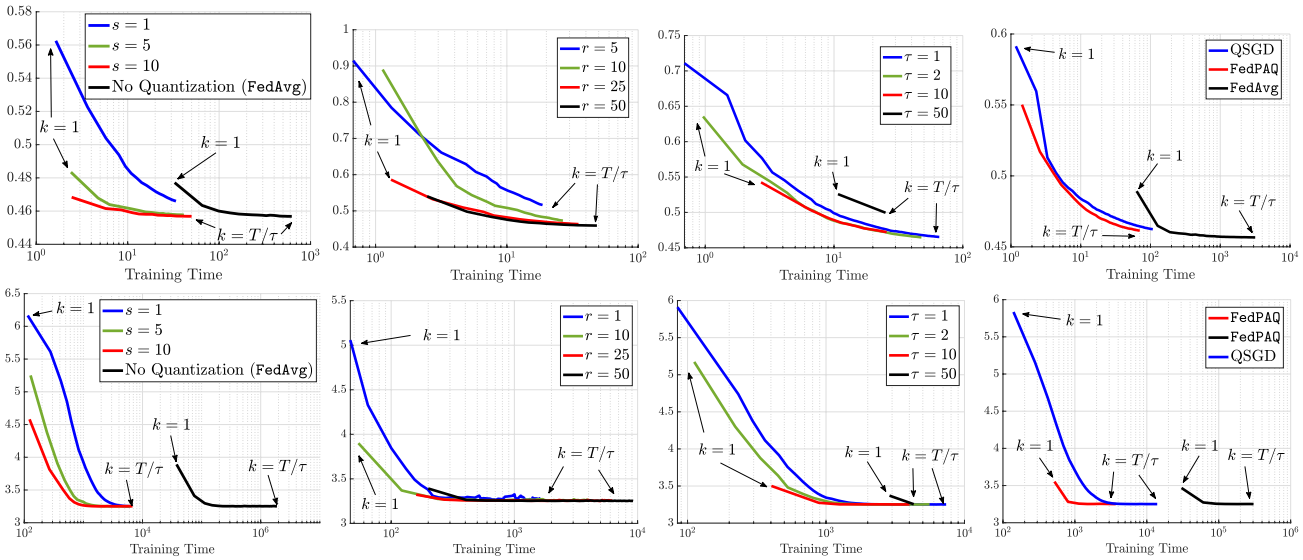

Figure 1: Training Loss vs. Training Time: Logistic Regression on MNIST (top). Neural Network on CIFAR-10 (bottom).

experiments, we use batchsize $B = 10$ and finely tune the stepsize's coefficient.

5.1 Logistic Regression on MNIST

In Figure 1, the top four plots demonstrate the training time for a regularized logistic regression problem over MNIST dataset ('0' and '8' digits) for $T = 100$ iterations. The network has $n = 50$ nodes each loaded with 200 samples. We set $C_{\texttt{comm}}/C_{\texttt{comp}} = 100/1$ to capture the communication bottleneck. Among the three parameters quantization levels s, number of active nodes in each round r, and period length τ, we fix two and vary the third one. First plot demonstrates the relative training loss for different quantization levels $s \in \{1, 5, 10\}$ and the case with no quantization which corresponds to the `FedAvg` method [McMahan et al., 2016]. The other two parameters are fixed to $(\tau, r) = (5, 25)$. Each curve shows the training time versus the achieved training loss for the aggregated model at the server for each round $k = 1, \cdots, T/\tau$. In the second plot, $(s, \tau) = (1, 5)$ are fixed. The third plot demonstrates the effect of period length τ in the communication-computation tradeoff. As demonstrated, after T/τ rounds, smaller choices for τ (e.g. $\tau = 1, 2$) result in slower convergence while the larger ones (e.g. $\tau = 50$) run faster though providing less accurate models. Here $\tau = 10$ is the optimal choice. The last plot compares the training time of `FedPAQ` with two other benchmarks `FedAvg` and QSGD. For both `FedPAQ` and `FedAvg`, we set $\tau = 2$ while `FedPAQ` and QSGD use quantization with $s = 1$ level. All three methods use $r = n = 50$ nodes in each round.

5.2 Neural Network training over CIFAR-10

We conduct another set of numerical experiments to evaluate the performance of `FedPAQ` on non-convex and smooth objectives. Here we train a neural network with four hidden layers consisting of $n = 50$ nodes and more thatn 92K parameters, where we use 10K samples from CIFAR-10 dataset with 10 labels. Since models are much larger than the previous setup, we increase the communication-computation ratio to $C_{\texttt{comm}}/C_{\texttt{comp}} = 1000/1$ to better capture the communication bottleneck for large models. The bottom four plots in Figure 1 demonstrate the training loss over time for $T = 100$ iterations. In the first plot, $(\tau, r) = (2, 25)$ are fixed and we vary the quantization levels. The second plot shows the effect of r while $(s, \tau) = (1, 2)$. The communication-computation tradeoff in terms of period length τ is demonstrated in the third plot, where picking $\tau = 10$ turns out to attain the fastest convergence. Lastly, we compare `FedPAQ` with other benchmarks in the forth plot. Here, we set $(s, r, \tau) = (1, 20, 10)$ in `FedPAQ`, $(r, \tau) = (20, 10)$ in `FedAvg` and $(s, r, \tau) = (1, 50, 1)$ for QSGD.

6 Conclusion

In this paper, we addressed some of the communication and scalability challenges of federated learning and proposed `FedPAQ`, a communication-efficient federated learning method with provable performance guarantees. `FedPAQ` is based on three modules: (1) periodic averaging in which each edge node performs local iterative updates; (2) partial node participation which captures the random availability of the edge nodes; and (3) quantization in which each model is quantized before being uploaded to the server. We provided rigorous analysis for our proposed method for two general classes of strongly-convex and non-convex losses. We further provided numerical results evaluating the performance of `FedPAQ`, and discussing the trade-off between communication and computation.

References

Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed sgd. In *Advances in Neural Information Processing Systems*, pages 7564–7575, 2018.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

Albert Berahas, Raghu Bollapragada, Nitish Shirish Keskar, and Ermin Wei. Balancing communication and computation in distributed optimization. *IEEE Transactions on Automatic Control*, 2018.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, 2017.

Thinh T Doan, Siva Theja Maguluri, and Justin Romberg. Accelerating the convergence rates of distributed subgradient methods with adaptive quantization. *arXiv preprint arXiv:1810.13245*, 2018.

John C Duchi, Michael I Jordan, and Martin J Wainwright. Privacy aware learning. *Journal of the ACM (JACM)*, 61(6):38, 2014.

Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover. Short-dot: Computing large linear transforms distributedly using coded short dot products. In *Advances In Neural Information Processing Systems*, pages 2092–2100, 2016.

Avishek Ghosh, Justin Hong, Dong Yin, and Kannan Ramchandran. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*, 2019.

Neel Guha, Ameet Talwlkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. Loadaboost: Loss-based adaboost federated machine learning on medical data. *arXiv preprint arXiv:1811.12629*, 2018.

Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173*, 2019.

Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, 2017.

Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, 2018.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019a.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019b.

Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018.

Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017. Accessed: 2019-09-13.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

Song Mei, Yu Bai, Andrea Montanari, et al. The landscape of empirical risk for nonconvex losses. *The Annals of Statistics*, 46(6A):2747–2774, 2018.

Katta G Murty and Santosh N Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. An exact quantized decentralized gradient descent algorithm. *IEEE Transactions on Signal Processing*, 67(19):4934–4947, 2019a.

Amirhossein Reisizadeh, Saurav Prakash, Ramtin Pedarsani, and Amir Salman Avestimehr. Coded-reduce: A fast and robust framework for gradient aggregation in distributed learning. *arXiv preprint arXiv:1902.01981*, 2019b.

Amirhossein Reisizadeh, Hossein Taheri, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. Robust and communication-efficient collaborative learning. *arXiv preprint arXiv:1907.10595*, 2019c.

Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Sumudu Samarakoon, Mehdi Bennis, Walid Saady, and Merouane Debbah. Distributed federated learning for ultra-reliable low-latency vehicular communications. *arXiv preprint arXiv:1807.08127*, 2018.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Virginia Smith, Simone Forte, Chenxin Ma, Martin Takac, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *arXiv preprint arXiv:1611.02189*, 2016.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Rashish Tandon, Qi Lei, Alexandros G Dimakis, and Nikos Karampatziakis. Gradient coding. *arXiv preprint arXiv:1612.03301*, 2016.

Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

Jianyu Wang, Anit Kumar Sahu, Zhouyi Yang, Gauri Joshi, and Soummya Kar. Matcha: Speeding up decentralized sgd via matching decomposition sampling. *arXiv preprint arXiv:1905.09435*, 2019.

Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700, 2019.

Qian Yu, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. Polynomial codes: an optimal design for high-dimensional coded matrix multiplication. *arXiv preprint arXiv:1705.10464*, 2017.

Xin Zhang, Jia Liu, Zhengyuan Zhu, and Elizabeth S Bentley. Compressed distributed gradient descent: Communication-efficient consensus over networks. *arXiv preprint arXiv:1812.04048*, 2018.

Wennan Zhu, Peter Kairouz, Haicheng Sun, Brendan McMahan, and Wei Li. Federated heavy hitters discovery with differential privacy. *arXiv preprint arXiv:1902.08534*, 2019.