Machine Learning at the Wireless Edge: Distributed Stochastic Gradient Descent Over-the-Air

Mohammad Mohammadi Amiri and Deniz Gündüz
Electrical and Electronic Engineering Department, Imperial College London, London SW7 2BT, U.K.
Email: {m.mohammadi-amiri15, d.gunduz}@imperial.ac.uk

***Abstract*—We study collaborative machine learning at the wireless edge, where power and bandwidth-limited devices (*workers*), with limited local datasets, implement distributed stochastic gradient descent (DSGD) over-the-air with the help of a remote parameter server (PS). We consider a wireless multiple access channel (MAC) from the workers to the PS for communicating the local gradient estimates. We first introduce a digital DSGD (D-DSGD) scheme, assuming that the workers operate on the boundary of the MAC capacity region at each iteration of the DSGD algorithm, and digitize their estimates within the bit budget allowed by the employed power allocation. We then introduce an *analog* scheme, called A-DSGD, motivated by the additive nature of the wireless MAC, where the workers send their gradient estimates over the MAC through the available channel bandwidth without employing any digital code. Numerical results show that A-DSGD converges much faster than D-DSGD. The improvement is particularly compelling at low power and low bandwidth regimes. We also observe that the performance of A-DSGD improves with the number of workers, while D-DSGD deteriorates, limiting the ability of the latter in harnessing the computation power of many edge devices.**

I. Introduction

Many emerging technologies involve massive amount of data collection, and collaborative intelligence that can process this data. The current trend of many machine learning algorithms focuses on centralized algorithms, where a powerful learning technique, often a neural network, is trained on a massive dataset. In the case of wireless edge devices, sending the collected data to a central processor in a reliable manner may be too costly in terms of energy and bandwidth. This might also be undesirable due to privacy concerns. Also, communication is typically more costly than processing; thus, a much more desirable alternative is to develop distributed machine learning techniques that can exploit the local processing capabilities of edge nodes, requiring limited communications. In this paper, we consider machine learning at the wireless network edge, where distributed nodes with local data samples and connected to a central processing unit through a shared wireless medium, jointly train a learning model.

Machine learning problems often require the minimization of the empirical loss function $F(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} f(\boldsymbol{\theta}, \boldsymbol{u}_n)$, where $\boldsymbol{\theta} \in \mathbb{R}^d$ denotes the model parameters to be optimized, $\boldsymbol{u}_n$ is the n-th training data sample, $n \in \{1, \dots, N\} \triangleq [N]$, and $f(\cdot)$ is the loss function defined by the learning model. The minimization of $F(\boldsymbol{\theta})$ is typically carried out through iterative stochastic gradient descent (SGD), in which the model parameter at iteration t, $\boldsymbol{\theta}_t$, is updated with a stochastic gradient $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \cdot \boldsymbol{g}(\boldsymbol{\theta}_t)$, which satisfies $\mathbb{E}[\boldsymbol{g}(\boldsymbol{\theta}_t)] = \nabla F(\boldsymbol{\theta}_t)$, where η_t is the learning rate. SGD also allows parallelization when the dataset is distributed across multiple computation servers, called the *workers*. In distributed SGD (DSGD), at each iteration, worker m computes a gradient vector based on the global parameter vector with respect to its local dataset, denoted by $\mathcal{B}_m$, and sends the result to the *parameter server* (PS), which updates the global parameter vector according to

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \frac{1}{M} \sum\nolimits_{m=1}^{M} \boldsymbol{g}_m(\boldsymbol{\theta}_t), \tag{1}$$

where M denotes the number of workers, and $\boldsymbol{g}_m(\boldsymbol{\theta}_t) \triangleq \frac{1}{|\mathcal{B}_m|}\sum_{\boldsymbol{u}_n \in \mathcal{B}_m} \nabla f(\boldsymbol{\theta}_t, \boldsymbol{u}_n)$, $m \in [M]$. Ideally, the data parallelism with DSGD should speed up the process M times providing linear scalability. But, in practice, it suffers from extensive communications from the workers to the PS [1]–[4], and this will be an even bigger hurdle in wireless edge learning due to stringent bandwidth and energy constraints.

Numerous studies have been dedicated to reduce the communication load of DSGD, where three main approaches, namely *quantization*, *sparsification*, and *local updates*, and their various combinations have been considered in the literature. Quantization methods implement lossy compression of the gradients by quantizing each of their entries to a finite-bit low precision value [1], [3], [5]. Sparsification reduces the communication time by sending only some values of the gradients [2], [6]–[8]. Another approach is to reduce the frequency of communication from the workers by allowing local parameter updates [2], [9], [10]. However, these works ignore the communication channel, and simply focus on reducing the amount of data transmitted by each worker to the PS.

In this paper, we consider DSGD over-the-air; that is, we consider a wireless shared medium from the workers to the PS, and treat each iteration of the DSGD algorithm as a distributed over-the-air computation problem. We will provide two distinct approaches for this wireless DSGD problem, based on *digital* and *analog* computation approaches, referred to as the D-DSGD and A-DSGD, respectively. We will show that A-DSGD can significantly speed up wireless DSGD, particularly in bandwidth-limited and low-power settings, typically experienced by wireless edge devices.

A similar over-the-air computation approach is considered

This work was supported in part by the Marie Sklodowska-Curie Action SCAVENGE (grant agreement no. 675891) and by the European Research Council (ERC) Starting Grant BEACON (grant agreement no. 725731).

in the federated learning context in two parallel works [11], [12] focusing on aligning the vectors received from different workers to have the same power level by performing power control and worker selection.

Notations: $\mathbb{R}$ represents the set of real values. For positive integer i, we let $[i] \triangleq \{1, \dots, i\}$. $\mathcal{N}\left(0, \sigma^2\right)$ denotes a zero-mean normal distribution with variance σ^2. We denote the cardinality of set $\mathcal{A}$ by $|\mathcal{A}|$, and l_2 norm of vector $\boldsymbol{x}$ by $\|\boldsymbol{x}\|_2$.

II. System Model

We consider distributed edge learning, where M nodes, called the workers, employ SGD with the help of a remote PS, to which they are connected through a noisy wireless MAC. Let $\mathcal{B}_m$ denote the set of data samples available at worker m, $m \in [M]$, and $\boldsymbol{g}_m(\boldsymbol{\theta}_t) \in \mathbb{R}^d$ be the stochastic gradient computed by worker m using local data samples. At iteration t of the DSGD algorithm in (1), the local gradient estimates of the workers are sent to the PS over s uses of a Gaussian MAC, characterized by:

$$\boldsymbol{y}_t = \sum\nolimits_{m=1}^{M} \boldsymbol{x}_{m,t} + \boldsymbol{z}_t, \tag{2}$$

where $\boldsymbol{x}_{m,t} \in \mathbb{R}^s$ is the length-s channel input vector sent by worker m, $\boldsymbol{y}_t \in \mathbb{R}^s$ is the channel output received by the PS, and $\boldsymbol{z}_t \in \mathbb{R}^s$ is the additive white Gaussian noise vector with each entry independent and identically distributed (i.i.d.) according to $\mathcal{N}\left(0, \sigma^2\right)$. The channel input vector of worker m at iteration t is a function of the current parameter vector $\boldsymbol{\theta}_t$, the local dataset $\mathcal{B}_m$, and the current gradient estimate at worker m, $\boldsymbol{g}_m(\boldsymbol{\theta}_t)$, $m \in [M]$. For a total of T iterations, the following total average transmit power constraint is imposed:

$$\frac{1}{MT} \sum\nolimits_{t=1}^{T} \sum\nolimits_{m=1}^{M} ||\boldsymbol{x}_{m,t}||_2^2 \leq \bar{P}. \tag{3}$$

The goal is to recover the average of the locally computed gradients $\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{g}_m(\boldsymbol{\theta}_t)$ at the PS, which then updates the model parameter as in (1). However, due to the pre-processing performed at each worker and the noise added by the wireless channel, the PS uses a noisy estimate to update the model parameter vector; i.e., we have $\boldsymbol{\theta}_{t+1} = \phi(\boldsymbol{\theta}_t, \boldsymbol{y}_t)$ for some update function $\phi : \mathbb{R}^d \times \mathbb{R}^s \to \mathbb{R}^d$. The updated model parameter is then multicast to the workers by the PS through an error-free shared link. We assume that the PS is not limited in power or bandwidth, so the workers receive a consistent parameter vector for their computations in the next iteration.

Note that due to the channel noise and finite power constraint at the workers, they cannot transmit their local gradient estimates to the PS in a lossless fashion. The transmission of the local gradient computations to the PS with the goal of PS reconstructing their average can be considered as a distributed function computation problem over a MAC [13]. We will consider both a digital approach treating computation and communication separately, and an analog approach that does not use any coding, and instead applies gradient sparsification followed by a linear transformation to compress the gradients, which are then sent simultaneously in an uncoded fashion.

III. Digital DSGD (D-DSGD)

In this section, we present DSGD at the wireless network edge utilizing digital compression and transmission over the MAC, referred to as the D-DSGD algorithm. Since the variances of the gradient estimates at different workers are not known, the power is allocated equally among the workers, so that worker m sends $\boldsymbol{x}_{m,t}$ with power P_t, i.e., $||\boldsymbol{x}_{m,t}||_2^2 = P_t$, where, for a total of T iterations, P_t is chosen to satisfy $\sum_{t=1}^{T} P_t \leq T\bar{P}$. Due to the intrinsic symmetry of the model, we assume that the workers transmit at the same rate at each iteration (while the rate may change across iterations depending on the allocated power, P_t). Accordingly, the total number of bits that can be transmitted by each worker over s uses of the Gaussian MAC in (2), is upper bounded by

$$R_t = \frac{s}{2M} \log_2\left(1 + \frac{MP_t}{s\sigma^2}\right), \tag{4}$$

where MP_t/s is the sum-power per channel use. Note that this is an upper bound since it is the Shannon capacity of the underlying Gaussian MAC, and we further assumed that the capacity can be achieved over a finite blocklength of s.

We will adopt the scheme proposed in [8] for gradient compression at each iteration of the DSGD algorithm, as it provides the state-of-the-art in convergence speed with the minimum number of bits sent by each worker at each iteration. However, we modify this scheme by allowing different numbers of bits to be sent by the workers at each iteration.

At each iteration the workers sparsify their gradient estimates as described below. In order to retain the accuracy of their local gradient estimates, workers employ *error accumulation* [3], where the accumulated error vector at worker m until iteration t is denoted by $\boldsymbol{\Delta}_{m,t-1} \in \mathbb{R}^d$, where we set $\boldsymbol{\Delta}_{m,0} = \boldsymbol{0}$, $\forall m \in [M]$. Hence, after computing $\boldsymbol{g}_m(\boldsymbol{\theta}_t)$, worker m updates its estimate with the accumulated error as $\boldsymbol{g}_m(\boldsymbol{\theta}_t) + \boldsymbol{\Delta}_{m,t-1}$, $m \in [M]$. At iteration t, worker m sets all but the highest q_t and the smallest q_t of the entries of its gradient estimate vector $\boldsymbol{g}_m(\boldsymbol{\theta}_t) + \boldsymbol{\Delta}_{m,t-1}$ to zero, where $q_t \leq d/2$ (to have a communication-efficient scheme, in practice, the goal is to have $q_t \ll d$). Then, it computes the mean values of all the remaining positive entries and all the remaining negative entries, denoted by $\mu_{m,t}^{+}$ and $\mu_{m,t}^{-}$, respectively. If $\mu_{m,t}^{+} > \left|\mu_{m,t}^{-}\right|$, then it sets all the entries with negative values to zero and all the entries with positive values to $\mu_{m,t}^{+}$, and vice versa. We denote the resulting sparse vector at worker m by $\hat{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)$, and worker m updates the local accumulated error vector as $\boldsymbol{\Delta}_{m,t} = \boldsymbol{g}_m(\boldsymbol{\theta}_t) + \boldsymbol{\Delta}_{m,t-1} - \hat{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)$, $m \in [M]$. It then aims to send $\hat{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)$ over the channel by transmitting its mean value and the positions of its non-zero entries. For this purpose, we use a 32-bit representation of the absolute value of the mean (either $\mu_{m,t}^{+}$ or $\left|\mu_{m,t}^{-}\right|$) along with 1 bit indicating its sign. To send the positions of the non-zero entries $\log_2\binom{d}{q_t}$ bits are sufficient at each worker. Thus, with the D-DSGD scheme, the total number of bits sent by each worker at iteration t is given by $r_t = \log_2\binom{d}{q_t} + 33$, where q_t is chosen as the highest integer satisfying $r_t \leq R_t$.

Algorithm 1 A-DSGD

Initialize $\boldsymbol{\theta}_1 = 0$ and $\boldsymbol{\Delta}_{1,0} = \cdots = \boldsymbol{\Delta}_{M,0} = 0$
for $t = 1, \ldots, T$ **do**
- **Workers do:**
  for $m = 1, \ldots, M$ in parallel **do**
    Compute $\boldsymbol{g}_m(\boldsymbol{\theta}_t)$ with respect to $\boldsymbol{u}_i \in \mathcal{B}_m$
    $\boldsymbol{g}_m^{ec}(\boldsymbol{\theta}_t) = \boldsymbol{g}_m(\boldsymbol{\theta}_t) + \boldsymbol{\Delta}_{m,t-1}$
    $\boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t) = \text{sparse}_k(\boldsymbol{g}_m^{ec}(\boldsymbol{\theta}_t))$
    $\boldsymbol{\Delta}_{m,t} = \boldsymbol{g}_m^{ec}(\boldsymbol{\theta}_t) - \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t)$
    EPA:
      $\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t) = A_s \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t)$
      $\boldsymbol{x}_{m,t}(\boldsymbol{\theta}_t) = \sqrt{\alpha_t}\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)$
    UPA:
      $\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t) = A_{s-1} \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t)$
      $\boldsymbol{x}_{m,t}(\boldsymbol{\theta}_t) = \left[\sqrt{\alpha_{m,t}}\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)^T \quad \sqrt{\alpha_{m,t}}\right]^T$
  end for
- **PS does:**
  EPA:
    $\hat{\boldsymbol{g}}_{\text{EPA}}(\boldsymbol{\theta}_t) = \text{AMP}_{A_s}\left(\frac{1}{M\sqrt{\alpha_t}}\boldsymbol{y}(\boldsymbol{\theta}_t)\right)$
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \cdot \hat{\boldsymbol{g}}_{\text{EPA}}(\boldsymbol{\theta}_t)$
  UPA:
    $\hat{\boldsymbol{g}}_{\text{UPA}}(\boldsymbol{\theta}_t) = \text{AMP}_{A_{s-1}}\left(\frac{1}{y_s(\boldsymbol{\theta}_t)}\boldsymbol{y}^{s-1}(\boldsymbol{\theta}_t)\right)$
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \cdot \hat{\boldsymbol{g}}_{\text{UPA}}(\boldsymbol{\theta}_t)$
end for

IV. Analog DSGD (A-DSGD)

Next, we propose an analog DSGD (A-DSGD) algorithm, in which all the workers transmit their gradient estimates simultaneously without employing any digital coding. This is motivated by the fact that the PS is only interested in the average of gradient vectors, and the underlying MAC naturally provides the sum of the gradients. See Algorithm 1 for a description of the A-DSGD scheme.

Similarly to D-DSGD, workers employ error accumulation. Hence, after computing $\boldsymbol{g}_m(\boldsymbol{\theta}_t)$, worker m updates its estimate as $\boldsymbol{g}_m^{ec}(\boldsymbol{\theta}_t) \triangleq \boldsymbol{g}_m(\boldsymbol{\theta}_t) + \boldsymbol{\Delta}_{m,t-1}$, $m \in [M]$. In order to reduce the dimension of the gradient vector to that of the channel, the workers apply gradient sparsification. In particular, worker m sets all but k elements with the highest magnitudes of vector $\boldsymbol{g}_m^{ec}(\boldsymbol{\theta}_t)$ to zero, and obtain a sparse vector $\boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t)$, $m \in [M]$. This k-level sparsification is represented by function sparse_k in Algorithm 1, i.e., $\boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t) = \text{sparse}_k(\boldsymbol{g}_m^{ec}(\boldsymbol{\theta}_t))$. Worker m, $m \in [M]$, then updates $\boldsymbol{\Delta}_{m,t} = \boldsymbol{g}_m^{ec}(\boldsymbol{\theta}_t) - \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t)$. To transmit the sparse vectors, workers will employ a random projection matrix, similarly to compressive sensing.

Assuming datasets with identical distributions across workers, the local gradient estimates computed by different workers also follow identical distributions; hence, they are expected to have a similar sparsity pattern. A pseudo-random matrix $A_{\tilde{s}} \in \mathbb{R}^{\tilde{s} \times d}$, for some $\tilde{s} \leq s$, with each entry i.i.d. according to $\mathcal{N}(0, 1/\tilde{s})$, is generated and shared between the PS and the workers before starting the computations. At each iteration t, worker m computes $\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t) \triangleq A_{\tilde{s}} \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t) \in \mathbb{R}^{\tilde{s}}$, and transmits $\boldsymbol{x}_{m,t}(\boldsymbol{\theta}_t) \triangleq \left[\sqrt{\alpha_{m,t}}\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)^T \quad \boldsymbol{a}_{m,t}{}^T\right]^T$, where $\boldsymbol{a}_{m,t} \in \mathbb{R}^{s-\tilde{s}}$, over the MAC, $m \in [M]$, while satisfying the average power constraint (3). The PS receives

$$\boldsymbol{y}(\boldsymbol{\theta}_t) = \begin{bmatrix} A_{\tilde{s}} \sum_{m=1}^{M} \sqrt{\alpha_{m,t}} \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t) \\ \sum_{m=1}^{M} \boldsymbol{a}_{m,t} \end{bmatrix} + \boldsymbol{z}_t. \tag{5}$$

Next, we propose two schemes for this analog transmission approach employing different scaling coefficients, or equivalently, different power allocation schemes across workers.

A. Equal Power Allocation (EPA)

In the EPA scheme, we set $\tilde{s} = s$, and at iteration t, worker m computes $\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t) = A_s \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t)$, $m \in [M]$, and scales it by factor $\sqrt{\alpha_t}$, which is known by the workers and the PS, and sends $\boldsymbol{x}_{m,t}(\boldsymbol{\theta}_t) = \sqrt{\alpha_t}\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)$, i.e., $\boldsymbol{a}_{m,t} = \emptyset$. The scaling factor $\sqrt{\alpha_t}$ is chosen to satisfy the following average power constraint over T iterations of A-DSGD algorithm

$$\frac{1}{MT} \sum_{t=1}^{T} \sum_{m=1}^{M} \alpha_t \|\tilde{\boldsymbol{g}}_m(\theta_t)\|_2^2 \leq \bar{P}. \tag{6}$$

Thus, the received vector at the PS is given by

$$\boldsymbol{y}(\boldsymbol{\theta}_t) = \sqrt{\alpha_t} \sum_{m=1}^{M} \tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t) + \boldsymbol{z}_t. \tag{7}$$

Since α_t is known also at the PS, it performs:

$$\frac{1}{M\sqrt{\alpha_t}} \boldsymbol{y}(\boldsymbol{\theta}_t) = A_s \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t) + \frac{1}{M\sqrt{\alpha_t}} \boldsymbol{z}_t. \tag{8}$$

The PS employs the approximate message passing (AMP) algorithm [14] to recover $\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t)$ from its noisy observation above. The AMP algorithm is denoted by the AMP_{A_s} in Algorithm 1. The estimate $\hat{\boldsymbol{g}}_{\text{EPA}}(\boldsymbol{\theta}_t)$ is used to update the model parameters as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \cdot \hat{\boldsymbol{g}}_{\text{EPA}}(\boldsymbol{\theta}_t)$.

B. Unequal Power Allocation (UPA)

With the UPA scheme, we set $\tilde{s} = s - 1$, which requires $s \geq 2$. At iteration t, we set $\boldsymbol{a}_{m,t} = \sqrt{\alpha_{m,t}}$, and worker m computes $\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t) = A_{s-1} \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t)$, and sends $\boldsymbol{x}_{m,t}(\boldsymbol{\theta}_t) = \left[\sqrt{\alpha_{m,t}}\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)^T \quad \sqrt{\alpha_{m,t}}\right]^T$ with transmit power $P_t = ||\boldsymbol{x}_{m,t}(\boldsymbol{\theta}_t)||_2^2$ satisfying the average power constraint $\sum_{t=1}^{T} P_t \leq T\bar{P}$, for $m \in [M]$. Accordingly, scaling factor $\sqrt{\alpha_{m,t}}$ is given by

$$\alpha_{m,t} = \frac{P_t}{\|\tilde{\boldsymbol{g}}_m(\theta_t)\|_2^2 + 1}, \quad \text{for } m \in [M]. \tag{9}$$

Since $\|\tilde{\boldsymbol{g}}_m(\theta_t)\|_2^2$ may vary across workers, so can the scaling factor $\sqrt{\alpha_{m,t}}$. Accordingly, at iteration t, worker m dedicates one channel use to provide the value of $\sqrt{\alpha_{m,t}}$ to the PS along with its scaled low-dimensional gradient vector $\tilde{\boldsymbol{g}}_m(\boldsymbol{\theta}_t)$, $m \in [M]$. The received vector at the PS is given by

$$\boldsymbol{y}(\boldsymbol{\theta}_t) = \begin{bmatrix} A_{s-1} \sum_{m=1}^{M} \sqrt{\alpha_{m,t}} \boldsymbol{g}_m^{sp}(\boldsymbol{\theta}_t) \\ \sum_{m=1}^{M} \alpha_{m,t} \end{bmatrix} + \boldsymbol{z}_t. \tag{10}$$

We define, for $i \in [s]$,

$$\boldsymbol{y}^i\left(\boldsymbol{\theta}_t\right) \triangleq \left[y_1\left(\boldsymbol{\theta}_t\right) \cdots y_i\left(\boldsymbol{\theta}_t\right)\right]^T, \tag{11a}$$

$$\boldsymbol{z}_t^i \triangleq \left[z_{t,1} \cdots z_{t,i}\right]^T, \tag{11b}$$

where $y_j\left(\boldsymbol{\theta}_t\right)$ and $z_{t,j}$ denote the j-th element of $\boldsymbol{y}\left(\boldsymbol{\theta}_t\right)$ and $\boldsymbol{z}_t$, respectively. Thus, we have

$$\boldsymbol{y}^{s-1}\left(\boldsymbol{\theta}_t\right) = A_{s-1} \sum\nolimits_{m=1}^{M} \sqrt{\alpha_{m,t}} \boldsymbol{g}_m^{sp}\left(\boldsymbol{\theta}_t\right) + \boldsymbol{z}_t^{s-1}, \tag{12a}$$

$$y_s\left(\boldsymbol{\theta}_t\right) = \sum\nolimits_{m=1}^{M} \alpha_{m,t} + z_{t,s}. \tag{12b}$$

Note that the goal is to recover $\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{g}_m^{sp}\left(\boldsymbol{\theta}_t\right)$ at the PS, while, from $\boldsymbol{y}^{s-1}\left(\boldsymbol{\theta}_t\right)$ the PS observes a noisy version of $\sum_{m=1}^{M} \sqrt{\alpha_{m,t}} \boldsymbol{g}_m^{sp}\left(\boldsymbol{\theta}_t\right)$ projected to a low-dimensional vector through A_{s-1}. According to (9), each value of $\left\|\tilde{\boldsymbol{g}}_m\left(\boldsymbol{\theta}_t\right)\right\|_2^2$ results in a distinct scaling factor $\alpha_{m,t}$. However, since the gradient estimates follow identical distributions, for large enough d and $|\mathcal{B}_m|$, the values of $\left\|\tilde{\boldsymbol{g}}_m\left(\boldsymbol{\theta}_t\right)\right\|_2^2, \forall m \in [M]$, are not going to be too different across workers. As a result, scaling factors $\sqrt{\alpha_{m,t}}, \forall m \in [M]$, are not going to be very different either. Accordingly, to diminish the effect of scaled gradient vectors, we choose to scale down the received vector $\boldsymbol{y}^{s-1}\left(\boldsymbol{\theta}_t\right)$ at the PS with the sum of the scaling factors, i.e., $\sum_{m=1}^{M} \sqrt{\alpha_{m,t}}$, whose noisy version is received by the PS as $y_s\left(\boldsymbol{\theta}_t\right)$. The resulting scaled vector at the PS is given by

$$\frac{1}{y_s\left(\boldsymbol{\theta}_t\right)} \boldsymbol{y}^{s-1}\left(\boldsymbol{\theta}_t\right) = A_{s-1} \sum\nolimits_{m=1}^{M} \frac{\sqrt{\alpha_{m,t}}}{\sum_{i=1}^{M} \sqrt{\alpha_{i,t}} + z_{t,s}} \boldsymbol{g}_m^{sp}\left(\boldsymbol{\theta}_t\right) + \frac{1}{\sum_{i=1}^{M} \sqrt{\alpha_{i,t}} + z_{t,s}} \boldsymbol{z}_t^{s-1}. \tag{13}$$

The PS then tries to recover $\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{g}_m^{sp}\left(\boldsymbol{\theta}_t\right)$ from $\boldsymbol{y}^{s-1}\left(\boldsymbol{\theta}_t\right)/y_s\left(\boldsymbol{\theta}_t\right)$, and it estimates $\hat{\boldsymbol{g}}_{\mathrm{UPA}}\left(\boldsymbol{\theta}_t\right)$ using the AMP algorithm. The estimate $\hat{\boldsymbol{g}}_{\mathrm{UPA}}\left(\boldsymbol{\theta}_t\right)$ is then used to update the model parameter as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \cdot \hat{\boldsymbol{g}}_{\mathrm{UPA}}\left(\boldsymbol{\theta}_t\right)$.

V. Experiments

Here we evaluate the performances of the A-DSGD and D-DSGD algorithms for image classification. We run experiments on MNIST dataset [15] with $N = 60000$ training and 10000 test samples, and train a single layer neural network with $d = 7850$ parameters utilizing ADAM optimizer [16]. A random set of $\lfloor N/M \rfloor$ data samples is assigned to each worker, ans we assume $\sigma^2 = 1$. The performance is measured as the accuracy with respect to the training dataset versus iteration count t, and the final accuracy with respect to the test samples, i.e., test accuracy, is also provided based on the parameter vector obtained after 50 training iterations.

In Fig. 1, we compare the performance of the A-DSGD algorithm with both EPA and UPA with that of D-DSGD algorithm for two different values of average transmit power $\bar{P}_1 = 127$ and $\bar{P}_2 = 422$. Since we need $r_t \leq R_t$ for the digital approach, we set number of channel uses s and $\bar{P}$ to relatively high values and number of workers M to a relatively small value to make sure that $q_t \geq 1$, $\forall t$. Accordingly, we consider $M = 25$ workers and $s = d/2$ channel uses.

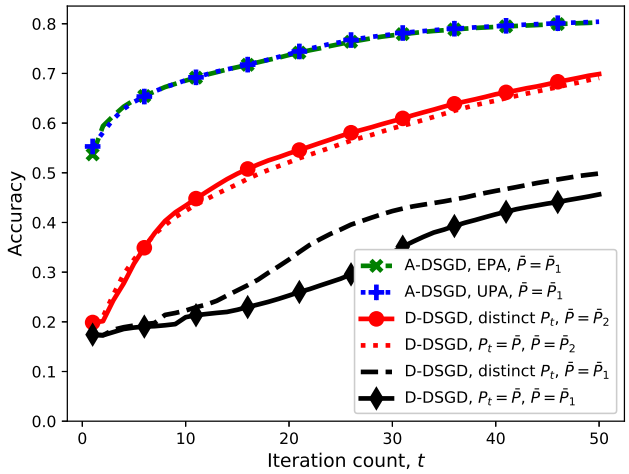

Fig. 1: Performance of the A-DSGD and D-DSGD algorithms for different $\bar{P}$ values.

We set a fixed ratio $k = \lfloor s/2 \rfloor$ for sparsification. The test accuracy of different DSGD algorithms is given in Table I. We observe that the analog approach significantly outperforms the standard digital approach of separating computation from communication. We did not include the performance of the A-DSGD algorithm for $\bar{P} = \bar{P}_2$ since it is very close to the one with $\bar{P} = \bar{P}_1$ for both power allocation schemes. Bearing this in mind, we observe that, unlike the A-DSGD scheme, the performance of D-DSGD significantly deteriorates by reducing $\bar{P}$ for both power allocation schemes under consideration. Thus, analog computation approach is particularly attractive for learning across low-power devices as it allows them to align their limited transmit powers to dominate the noise term. For the UPA, we set $P_t = \bar{P}$, $\forall t$, and for the EPA, we set $\alpha_t = 100 + 10t/3$ and $\alpha_t = 300 + 10t$ resulting in $\bar{P} = \bar{P}_1$ and $\bar{P} = \bar{P}_2$, respectively. We consider two different power allocation schemes with D-DSGD: in the first scheme, we set $P_t = \bar{P}$, $\forall t$, and in the second, we let P_t to be the same as the sum-power consumed by the workers at iteration t of the A-DSGD algorithm with EPA. Observe that, for the D-DSGD algorithm, letting P_t vary over time improves the performance.

In Fig. 2, we compare the performance of A-DSGD with UPA and D-DSGD, where, for both algorithms, we set $P_t = \bar{P} = 1100$, $\forall t$, for different M and s values. We consider two different values of $M \in \{20, 40\}$, and two different values of $s \in \{0.3d, 0.5d\}$, and a fixed ratio $k = \lfloor s/2 \rfloor$. We present the final test accuracy of different DSGD algorithms in Table II. As it can be seen, for $s = 0.3d$, increasing M by a factor of 2 deteriorates the performance of D-DSGD. Accordingly, the performance of D-DSGD algorithm is vulnerable to a relatively small increase in M, as well as a decrease in the average transmit power $\bar{P}$, whose effect was observed in Fig. 1. We can conclude that the digital scheme prefers to have a smaller number of workers, which means that it cannot harvest the computation power of many edge devices. On the other hand, we observe that the performance of A-DSGD improves with M, and is significantly superior compared to

TABLE I: Final test accuracy for various DSGD schemes considered in Fig. 1

D-DSGD $P_t = \bar{P}, \bar{P} = \bar{P}_1$	D-DSGD distinct P_t, $\bar{P} = \bar{P}_1$	D-DSGD $P_t = \bar{P}, \bar{P} = \bar{P}_2$	D-DSGD distinct P_t, $\bar{P} = \bar{P}_2$	A-DSGD UPA, $\bar{P} = \bar{P}_1$	A-DSGD EPA, $\bar{P} = \bar{P}_1$
0.459	0.501	0.698	0.705	0.811	0.812

TABLE II: Final test accuracy for various DSGD schemes considered in Fig. 2

D-DSGD $M = 40, s = 0.3d$	D-DSGD $M = 20, s = 0.3d$	D-DSGD $M = 20, s = 0.5d$	A-DSGD $M = 20, s = 0.3d$	A-DSGD $M = 40, s = 0.3d$	A-DSGD $M = 20, s = 0.5d$
0.704	0.729	0.76	0.811	0.816	0.828

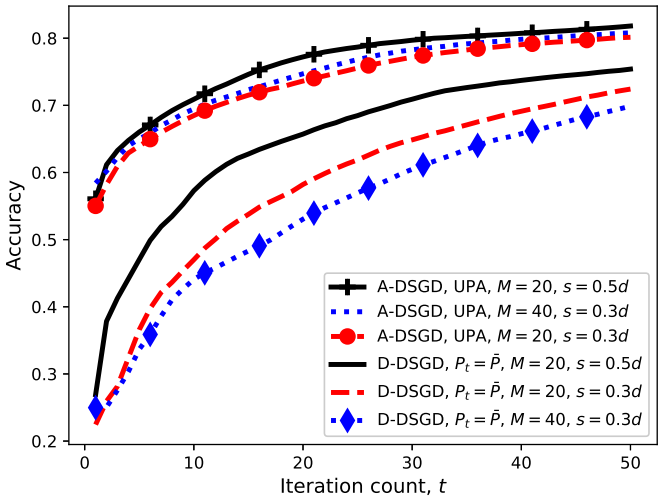

Fig. 2: Performance of the A-DSGD and D-DSGD algorithms for different (M, s) pairs.

D-DSGD, and the improvement increases remarkably with M. We further observe that reducing the available channel uses s from $s = 0.5d$ to $s = 0.3d$ degrades the performance of the D-DSGD algorithm considerably, whereas the sensitivity of A-DSGD to channel bandwidth is much weaker.

In the longer version of the paper [17], we have included more numerical results.

VI. Conclusions

We have studied distributed machine learning at the wireless edge, where M workers aim to minimize a loss function by performing DSGD with the help of a remote PS. Workers with limited datasets communicate with the PS over a MAC. PS updates the parameter vector, and shares it with the workers through a noiseless shared link. We consider both a digital approach (D-DSGD) that separates computation and communication, and an analog approach (A-DSGD) that exploits the superposition property of the wireless channel to have the average gradient at the PS computed over-the-air. In the D-DSGD scheme, the amount of information bits sent by each worker at each iteration can be adaptively adjusted with respect to the average transmit power constraint $\bar{P}$. In the A-DSGD scheme, we have proposed gradient sparsification followed by compressive sensing employing the same measurement matrix at all the workers in order to reduce the typically very large parameter vector dimension to the limited channel bandwidth. This analog approach allows a much more efficient use of the limited channel bandwidth, and benefits from the beamforming effect thanks to the identical distributions of the gradients across the workers. Numerical results have shown significant improvement in performance with the analog approach, particularly in the low-power and low-bandwidth regimes. We have also observed that, unlike D-DSGD, the performance of A-DSGD improves with the number of workers. Future work will include extending this framework to fading channels, and incorporating the computation time and energy.

References

[1] D. Alistarh, D. Grubic, J. Z. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via randomized quantization and encoding," in *NIPS*, Long Beach, CA, Dec. 2017, pp. 1709–1720.

[2] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv:1712.01887v2 [cs.CV]*, Feb. 2018.

[3] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *INTERSPEECH*, Singapore, Sep. 2014, pp. 1058–1062.

[4] M. M. Amiri and D. Gündüz, "Computation scheduling for distributed machine learning with straggling workers," *arXiv:1810.09992 [cs.DC]*.

[5] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *ICML*, Jul. 2015.

[6] N. Strom, "Scalable distributed DNN training using commodity gpu cloud computing," in *INTERSPEECH*, 2015.

[7] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv:1704.05021v2 [cs.CL]*, Jul. 2017.

[8] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," *arXiv:1805.08768v1 [cs.LG]*, May 2018.

[9] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017.

[10] T. Lin, S. U. Stich, and M. Jaggi, "Don't use large mini-batches, use local SGD," *arXiv:1808.07217v3 [cs.LG]*, Oct. 2018.

[11] G. Zhu, Y. Wang, and K. Huang, "Low-latency broadband analog aggregation for federated edge learning," *arXiv:1812.11494 [cs.IT]*.

[12] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *arXiv:1812.11750 [cs.LG]*, Jan. 2019.

[13] M. Goldenbaum and S. Stanczak, "Robust analog function computation via wireless multiple-access channels," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3863–3877, Sep. 2013.

[14] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci. USA*, vol. 106, no. 45, pp. 18 914–18 919, Nov. 2009.

[15] Y. LeCun, C. Cortes, and C. Burges, "The MNIST database of handwritten digits," *http://yann.lecun.com/exdb/mnist/*, 1998.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980v9 [cs.LG]*, Jan. 2017.

[17] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *arXiv:1901.00844 [cs.DC]*, Jan. 2019.