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). Standard approaches inherently assume separate computation and communication, where local gradient estimates are compressed and communicated to the PS over the wireless multiple access channel (MAC). Following this digital approach, we introduce D-DSGD, assuming that the workers operate on the boundary of the MAC capacity region at each iteration of the DSGD algorithm, and employ gradient quantization and error accumulation to transmit their gradient 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. In A-DSGD, the workers first sparsify their gradient estimates (with error accumulation), and then project them to a lower dimensional space imposed by the available channel bandwidth. These projections are transmitted directly over the MAC without employing any digital code. Numerical results show that A-DSGD converges much faster than D-DSGD thanks to its more efficient use of the limited bandwidth and the natural alignment of the gradient estimates over the channel. 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 (keeping the total size of the dataset constant), while D-DSGD deteriorates, limiting the ability of the latter in harnessing the computation power of too many edge devices. We highlight that the lack of quantization and channel encoding/decoding operations in A-DSGD further speeds up communication, making it very attractive for low-latency machine learning applications at the wireless network edge.
I. INTRODUCTION

Many emerging technologies involve massive amount of data collection, and collaborative intelligence that can process and make sense of this data. Internet of things (IoT), autonomous driving, or extended reality (XR) technologies are prime examples, where data from sensors must be continuously collected, communicated, and processed to make inferences about the state of a system, or predictions about its future states. Many specialized machine learning algorithms are being developed tailored for various types of sensor data; however, the current trend focuses on centralized algorithms, where a powerful learning algorithm, often a neural network, is trained on a massive dataset. While this inherently assumes the availability of data at a central processor, in the case of wireless edge devices, transmitting 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. Moreover, communication is typically more costly compared to processing; therefore, a much more desirable and practically viable alternative is to develop distributed machine learning techniques that can exploit the local processing capabilities of edge devices, requiring limited communications (see [1] for a survey of applications of edge intelligence and existing approaches to enable it). In this paper, we consider machine learning at the wireless network edge, where distributed processors with their 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

\[
F(\theta) = \frac{1}{N} \sum_{n=1}^{N} f(\theta, u_n),
\]

where \( \theta \in \mathbb{R}^d \) denotes the model parameters to be optimized, \( u_n \in \mathbb{R}^d \) is the \( n \)-th training data sample, for \( n \in [N] \triangleq \{1, \ldots, N\} \), and \( f(\cdot) \) is the loss function defined by the learning model. The minimization of (1) is typically carried out through iterative gradient descent (GD), in which the model parameters at the \( t \)-th iteration, \( \theta_t \), are updated according to

\[
\theta_{t+1} = \theta_t - \eta_t \nabla F(\theta_t) = \theta_t - \eta_t \frac{1}{N} \sum_{n=1}^{N} \nabla f(\theta_t, u_n),
\]

where \( \eta_t \) is the learning rate at iteration \( t \). However, in the case of massive datasets each iteration of GD becomes prohibitively demanding. Instead, in stochastic GD (SGD) the parameter vector
is updated with a stochastic gradient

$$\theta_{t+1} = \theta_t - \eta_t \cdot g(\theta_t)$$

(3)

which satisfies $E[g(\theta_t)] = \nabla F(\theta_t)$. SGD also allows parallelization when the dataset is distributed across tens or even hundreds of computation servers, called the workers. In distributed SGD (DSGD), workers process data samples in parallel while maintaining a globally consistent parameter vector $\theta_t$. In each iteration, worker $m$ computes a gradient vector based on the global parameter vector with respect to its local dataset, denoted by $B_{m,t}$, and sends the result to the parameter server (PS), also referred to as the master node, which stores and updates the global parameter vector. Once the PS receives the computed gradients from all the workers, it updates the global parameter vector according to

$$\theta_{t+1} = \theta_t - \frac{\eta_t}{M} \sum_{m=1}^{M} g_m(\theta_t),$$

(4)

where $M$ denotes the number of workers, and $g_m(\theta_t) \triangleq \frac{1}{|B_{m,t}|} \sum_{u_n \in B_{m,t}} \nabla f(\theta_t, u_n)$ is the stochastic gradient of the current model computed at worker $m$, $m \in [M]$, using the locally available portion of the dataset, $B_{m,t}$. Ideally, the data parallelism with DSGD should speed up the process $M$ times providing linear scalability. However, in practice, it suffers from extensive communications from the workers to the PS, which is to maintain a consistent global model, and communication is reported as the major bottleneck of DSGD [2–4]. There is no doubt that the communication will be an even bigger hurdle in wireless edge learning due to stringent bandwidth and energy constraints on the workers.

Numerous studies have been dedicated to the reduction of the communication load of DSGD; however, these works ignore the communication channel, and simply focus on reducing the amount of data that needs to be transmitted from 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, respectively. We will show that analog “over-the-air” computation can significantly speed up wireless DSGD, particularly in bandwidth-limited and low-power settings, typically experienced by wireless edge devices.
A. Prior Works

Extensive efforts have been made in recent years to speed up large-scale distributed learning. Research in this direction can be categorized into two: those reducing the computation time at each worker, and those reducing the communication time between the workers and the PS.

The techniques aiming to reduce the computation time can be further categorized as coded and uncoded computation techniques. Most coded computation techniques in the literature are designed to carry out GD without being limited by straggling workers. The main goal is to recover gradient \( \nabla F(\theta_t) \) from computations executed by a limited number of workers, by jointly designing the distribution of the data samples across the workers, pre-processing of the computed gradients at the workers, and post-processing of the received computations by the PS [5]–[10]. Thus, the PS does not need to wait for the computations from all the workers to update the global parameter vector.

While coded computation [5] and gradient coding [6] approaches allow mitigating persistent stragglers, they cannot exploit partial computations executed by non-persistent stragglers. Instead, several other works have focused on uncoded computations, which allows utilizing partial computations performed by slow workers instead of discarding them [11]–[16]. As highlighted in [14] and [16], unlike most coded GD approaches, uncoded computations also allow implementing DSGD, which, in turn, reduces either the computation load of the workers, or the communication load. A coded DSGD algorithm is also proposed in [17].

In many practical implementations, however, bandwidth of the communication channel from the workers to the PS turns out to be the main bottleneck for speeding up distributed learning [3], [18]. Therefore, reducing the communication requirements of DSGD is as important as reducing the average computation time. To reduce the communication load, 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 gradient vectors by quantizing each of their entries to a finite-bit low precision value [2], [4], [18]–[24]. Sparsification reduces the communication time by transmitting only some values of the gradient vectors [3], [25]–[31]. Sparsification can be considered as another way of lossy compression, but it is assumed that the chosen entries of the gradient vectors are transmitted reliably, e.g., at a very high resolution. Another approach is to reduce the frequency of communication from the workers by allowing local parameter updates [3], [32]–[35].
B. Our Contributions

Most of the current literature on distributed machine learning consider interference-and-error-free links from the workers to the PS to communicate their local gradient estimates possibly in compressed form to reduce the amount of information that must be transmitted. Due to the prevalence of wireless networks and the increasing availability of edge devices, e.g., mobile phones, sensors, etc., for large-scale data collection and learning, we consider a wireless multiple-access channel (MAC) from the workers to the PS, through which the PS receives the gradients computed by the workers at each iteration of the DSGD algorithm. The standard approach to this problem, aligned with the aforementioned studies on distributed machine learning, is a separate approach to computation and communication.

We first follow this separation-based approach, and propose a digital DSGD scheme, which will be called D-DSGD. In this scheme, the workers first compress their gradient estimates to a finite number of bits. We then assume that some medium access scheme, e.g., time, frequency or code division multiple access, followed by error correction coding is employed to transmit the compressed gradient estimates to the PS in a reliable manner. Having received the gradient estimates from the workers, the PS computes their average as in (4) to update the parameter vector. In this work, to understand the performance limits of the digital approach, we assume capacity-achieving channel codes are utilized at the workers (operating on the boundary of the capacity region of the underlying MAC). The optimal solution for this scheme will require carefully allocating channel resources across the workers and the available power of each worker across iterations, together with an efficient gradient quantization scheme. For gradient compression, we will consider state-of-the-art quantization approaches together with local error accumulation [28].

It is known that separation-based approaches to distributed computing over wireless channels is sub-optimal. Instead, we propose an analog communication approach, in which the workers transmit their local gradient estimates directly over the wireless channel. This scheme is motivated by the fact that the PS is not interested in the individual gradient vectors, but in their average, and the wireless MAC automatically provides the PS with the sum of the gradients (plus a noise term). However, the bandwidth available at each iteration may not be sufficient to transmit the whole gradient vector in an uncoded fashion. Hence, to compress the gradients to the dimension of the limited bandwidth resources, we employ a compression scheme inspired by compressive
sensing, which we have previously used for analog image transmission over bandwidth-limited wireless channels [36]. In this analog computation scheme, called A-DSGD, workers first sparsify their local gradient estimates (after adding the accumulated local error). These sparsified gradient vectors are projected to the channel bandwidth using a pseudo-random measurement matrix, as in compressive sensing. Then, all the workers transmit the resultant real-valued vectors to the PS simultaneously in an analog fashion, by simply scaling them to meet the average transmit power constraints. The PS tries to reconstruct the sum of the actual sparse gradient vectors from its noisy observation. We use approximate message passing (AMP) algorithm to do this at the PS [37].

Numerical results show that the proposed analog distributed learning scheme A-DSGD has a significantly better convergence behaviour compared to its digital transmission counterpart. We also observe that the performance of the proposed A-DSGD algorithm improves as the number of workers increases, keeping the total size of the dataset constant, while the opposite happens for the digital scheme due to the inherent resource sharing approach. The improvement is more pronounced when the average transmit power is small compared to the noise power, which is the setting of interest for most applications involving low power sensor nodes. Furthermore, the performance of the A-DSGD algorithm degrades only slightly even with a significant reduction in the available average transmit power, and the degradation is negligible when the number of workers is relatively large. We argue that, these observations are due to the inherent ability of analog communications to benefits from the signal-superposition characteristic of the wireless MAC. Also, reduction in the communication bandwidth of the MAC deteriorates the performance of the D-DSGD algorithm much more compared to A-DSGD.

A similar over-the-air computation approach is considered in the federated learning context in a parallel work [38], which focuses on aligning the vectors received from different workers to have the same power level by performing power control and worker selection using orthogonal frequency-division multiplexing (OFDM).

The rest of the paper is organized as follows. We introduce the system model and the problem formulation in Section II. The proposed DGSD algorithms with digital and analog transmission approaches are elaborated in Section III and Section IV respectively. In Section V we present the numerical results. The paper is concluded in Section VI.
Fig. 1: Illustration of distributed machine learning at the wireless edge. Workers with limited local datasets collaborate through a PS to implement DSGD at the network edge, where the local gradient estimates of the workers are transmitted to the PS over a shared wireless MAC.

C. Notations

\( \mathbb{R} \) represents set of real values. For any positive integer \( i \), we define \( [i] \triangleq \{1, \ldots, i\} \), and \( \mathbf{1}_i \) denotes a column vector of dimension \( i \) with all entries 1. \( \mathcal{N}(0, \sigma^2) \) denotes a zero-mean normal distribution with variance \( \sigma^2 \). We represent the cardinality of a set by \( |\cdot| \). Notation \( \|\cdot\|_2 \) shows \( l_2 \) norm of a vector.

II. System Model

We consider distributed machine learning at the wireless network edge, where \( M \) wireless edge nodes, called the workers, employ SGD with the help of a remote PS, to which they are connected through a noisy wireless MAC (see Fig. II). Let \( B_{m,t} \) denote the set of data samples available at worker \( m \), \( m \in [M] \), and \( \mathbf{g}_m(\theta_t) \in \mathbb{R}^d \) be the stochastic gradient computed by worker \( m \) using local data samples. At each iteration of the DSGD algorithm in (4), the local gradient estimates of the workers are sent to the PS over \( s \) uses of a Gaussian MAC, characterized by:

\[
y_t = \sum_{m=1}^{M} \mathbf{x}_{m,t} + \mathbf{z}_t, \tag{5}
\]

where \( \mathbf{x}_{m,t} \in \mathbb{R}^s \) is the length-\( s \) channel input vector transmitted by worker \( m \) at iteration \( t \), \( y_t \) is the channel output received by the PS, and \( \mathbf{z}_t \in \mathbb{R}^s \) is the independent additive white Gaussian
noise (AWGN) vector with each entry independent and identically distributed (i.i.d.) according to $\mathcal{N}(0, \sigma^2)$. Since we focus on DSGD, the channel input vector of worker $m$ at iteration $t$ is a function of the current parameter vector $\theta_t$ and the local dataset $B_{m,t}$, and more specifically the current gradient estimate at worker $m$, $g_m(\theta_t)$, $m \in [M]$. A total average transmit power constraint is imposed:

$$\frac{1}{MT} \sum_{t=1}^{T} \sum_{m=1}^{M} \|x_{m,t}\|^2_2 \leq \bar{P},$$

averaged over iterations of the DSGD algorithm and the workers. The goal is to recover the average of the locally computed gradients $\frac{1}{M} \sum_{m=1}^{M} g_m(\theta_t)$ at the PS, and update the model parameter as in (4). However, due to the pre-processing performed at each worker and the noise added by the wireless channel, it is not possible to recover the average gradient perfectly at the PS, and instead, it uses a noisy estimate to update the model parameter vector; i.e., we have $\theta_{t+1} = \phi(\theta_t, y_t)$ for some update function $\phi : \mathbb{R}^d \times \mathbb{R}^s \rightarrow \mathbb{R}^d$. 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 global 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, $g_m(\theta_t)$, $m \in [M]$, to the PS with the goal of PS reconstructing their average can be considered as a distributed function computation problem over a MAC [39]. 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 transmitted simultaneously over the channel 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 wireless MAC, referred to as the digital DSGD (D-DSGD) algorithm. Since we do not know the variances of the gradient estimates at different workers, we allocate the power equally among the workers, so that worker $m$ sends $x_{m,t}$ with power $P_t$, i.e., $\|x_{m,t}\|^2_2 = P_t$, ...
where the $P_t$ values are chosen to satisfy the average transmit power constraint over $T$ iterations

$$\frac{1}{T} \sum_{t=1}^{T} P_t \leq \bar{P}. \quad (7)$$

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 from each of the workers over $s$ uses of the Gaussian MAC, described in (5), is upper bounded by

$$R_t = \frac{s}{2M} \log_2 \left( 1 + \frac{MP_t}{s\sigma^2} \right), \quad (8)$$

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 $s$.

**Remark 1.** We note that having a distinct sum power $MP_t$ at each iteration $t$ enables each user to transmit different numbers of bits at different iterations. This corresponds to a novel gradient compression scheme for DSGD, in which the workers can adjust over time the amount of information they send to the PS about their gradient estimates. They can send more information bits at the beginning of the DSGD algorithm when the gradient estimates have higher variances, and reduce the number of transmitted bits over time as the variance decreases. We observed empirically that this improves the performance compared to the standard approach in the literature, where the same compression scheme is applied at each iteration \[28\].

We will adopt the scheme proposed in \[28\] for gradient compression at each iteration of the DSGD scheme, as it provides the state-of-the-art in convergence speed with the minimum number of bits transmitted by each worker at each iteration. However, we modify this scheme by allowing different numbers of bits to be transmitted 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 \[4\], \[40\], where the accumulated error vector at worker $m$ until iteration $t$ is denoted by $\Delta_{m,t-1} \in \mathbb{R}^d$, where we set $\Delta_{m,0} = 0$, $\forall m \in [M]$. Hence, after the computation of the local gradient estimate for parameter vector $\theta_t$, i.e., $g_m(\theta_t)$, worker $m$ updates its estimate with the accumulated error as $g_m(\theta_t) + \Delta_{m,t-1}$, $m \in [M]$. At iteration $t$, worker $m$, $m \in [M]$, sets all but the highest $q_t$
and the smallest $q_t$ of the entries of its gradient estimate vector $g_m(\theta_t) + \Delta_{m,t-1}$, of dimension $d$, to zero, where $q_t \leq d/2$ (to have a communication-efficient scheme, in practice, the goal is to have $q_t \ll d, \forall t$). 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 $\mu^+_{m,t} > |\mu_{m,t}|$, 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 $\hat{g}_m(\theta_t)$, and worker $m$ updates the local accumulated error vector as $\Delta_{m,t} = g_m(\theta_t) + \Delta_{m,t-1} - \hat{g}_m(\theta_t)$, $m \in [M]$. It then aims to send $\hat{g}_m(\theta_t)$ 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 mean value (either $\mu^+_{m,t}$ or $|\mu^-_{m,t}|$) along with 1 bit indicating its sign. To send the positions of the non-zero entries, it is assumed in [28] that the distribution of the distances between the non-zero entries is geometrical with success probability $q_t$, which allows them to use Golomb encoding to send these distances with a total number of bits

$$b^* + \frac{1}{1 - (1 - q_t)^{2b^*}},$$

(9)

where $b^* = 1 + \log_2 \left( \frac{\log \left( \frac{d}{\log(1 - q_t)} \right)}{\log \left( \frac{\sqrt{5} - 1}{2} \right)} \right)$. However, we argue that, sending $\log_2 \left( \frac{d}{q_t} \right)$ bits to transmit the positions of the non-zero entries is sufficient regardless of the distribution of the positions. This can be achieved by simply enumerating all possible sparsity patterns. Thus, with the D-DSGD scheme, the total number of bits sent by each worker at iteration $t$ is given by

$$r_t = \log_2 \left( \frac{d}{q_t} \right) + 33,$$

(10)

where $q_t$ is chosen as the highest integer satisfying $r_t \leq R_t$.

We will also study the impact of introducing more workers into the system. With the reducing cost of sensing and computing devices, we can consider introducing more workers, each coming with its own power resources. We assume that the size of the total dataset remains constant, which allows each sensor to save computation time and energy. Assume that the number of workers is increased by a factor $\kappa > 1$. We assume that the total power consumed by the workers at iteration $t$ also increases by factor $\kappa$. We can see that the maximum number of bits that can be sent by each worker is strictly smaller in the new system. This means that the PS receives a less accurate gradient estimate from each worker, but from more workers. We will observe in Section V that the convergence speed of D-DSGD deteriorates with $K$ as the workers
end up sharing the limited channel resources. Numerical results for the D-DSGD scheme and its comparison to analog transmission are relegated to Section V.

IV. ANALOG DSGD (A-DSGD)

Next, we propose an analog DSGD algorithm, called A-DSGD, which does not employ any digital coding scheme, either for compression or channel coding, and instead all the workers transmit their gradient estimates simultaneously in an uncoded manner. This is motivated by the fact that, the PS is not interested in the individual gradient vectors, but only in their average. The underlying wireless MAC naturally provides the sum of the gradients, which is the only information required at the PS to update the parameter vector. See Algorithm 1 for a description of the A-DSGD scheme.

Similarly to the D-DSGD scheme, workers employ local error accumulation. Hence, after the computation of the local gradient estimate for parameter vector $\theta_t$, each worker updates its estimate with the accumulated error as $g^{ec}_{m}(\theta_t) \triangleq g_m(\theta_t) + \Delta_{m,t-1}$, $m \in [M]$.

The challenge in the analog transmission approach is to compress the gradient vectors to the available channel bandwidth. In many modern machine learning applications, such as deep neural networks, the parameter vector, and hence the gradient vectors, have extremely large dimensions, whereas the channel bandwidth, measured by parameter $s$, is small due to the bandwidth limitations, and to limit the latency of each DSGD iteration. Thus, transmitting all the model parameters one-by-one in an uncoded/analog fashion is not possible as we typically have $d >> s$.

Lossy compression at any required level is at least theoretically possible in the digital domain. For the analog scheme, in order to reduce the dimensionality of the gradient vector to that of the channel, the workers apply gradient sparsification. In particular, worker $m$ sets all but the $k$ elements of the error-compensated resulting vector $g^{ec}_{m}(\theta_t)$ with the highest magnitudes to zero. We denote the sparse vector at worker $m$ by $g^{sp}_{m}(\theta_t), m \in [M]$. This $k$-level sparsification is represented by function sparse$_k$ in Algorithm 1 i.e., $g^{sp}_{m}(\theta_t) = \text{sparse}_k(g^{ec}_{m}(\theta_t))$. The accumulated error at worker $m$, $m \in [M]$, is then updated according to

$$\Delta_{m,t} = g^{ec}_{m}(\theta_t) - g^{sp}_{m}(\theta_t) = g_m(\theta_t) + \Delta_{m,t-1} - \text{sparse}_k(g_m(\theta_t) + \Delta_{m,t-1}).$$ (11)

We would like to transmit only the non-zero entries of these sparse vectors. However, simply ignoring the zero elements would require transmitting their indexes to the PS separately. To
avoid this additional data transmission, we will employ a random projection matrix, similarly to compressive sensing. A similar idea is recently used in [36] for analog image transmission over a bandwidth-limited channel.

Assuming i.i.d. data samples across workers (as standard in the literature), the local gradient estimates are expected to be close to the true gradient for sufficiently large $|B_{m,t}|$, and hence, they are expected to have a similar sparsity pattern as well. 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 $\tilde{g}_m(\theta_t) \triangleq A_\tilde{s} g_m^{sp}(\theta_t) \in \mathbb{R}_{\tilde{s}}$, and transmits $x_{m,t}(\theta_t) \triangleq \left[ \sqrt{\alpha_{m,t}} g_m^{sp}(\theta_t)^T a_{m,t} \right]^T$, where $a_{m,t} \in \mathbb{R}^{s-\tilde{s}}$, over the MAC, $m \in [M]$, while satisfying the average power constraint (6).

The PS receives

$$y(\theta_t) = \sum_{m=1}^M x_{m,t}(\theta_t) + z_t = \left[ A_{\tilde{s}} \sum_{m=1}^M \sqrt{\alpha_{m,t}} g_m^{sp}(\theta_t) \right] \sum_{m=1}^M a_{m,t} + z_t.$$ (12)

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

A. Equal Power Allocation (EPA)

In the EPA scheme, we set $\tilde{s} = s$, and at iteration $t$, worker $m$ computes $\tilde{g}_m(\theta_t) = A_s g_m^{sp}(\theta_t)$, $m \in [M]$, and scales its computed low-dimensional gradient vector $\tilde{g}_m(\theta_t)$ with the same factor $\sqrt{\alpha_t}$, which is known by the workers and the PS, and sends $x_{m,t}(\theta_t) = \sqrt{\alpha_t \tilde{g}_m(\theta_t)}$, i.e., $a_{m,t} = \emptyset$. The scaling factor $\sqrt{\alpha_t}$ is chosen to satisfy the average power constraint

$$\frac{1}{MT} \sum_{t=1}^T \sum_{m=1}^M \alpha_t \|g_m^{sp}(\theta_t)\|^2 \leq \bar{P}.\quad (13)$$

Thus, the received vector at the PS is given by

$$y(\theta_t) = \sqrt{\alpha_t} \sum_{m=1}^M \tilde{g}_m(\theta_t) + z_t.$$ (14)

Since $\alpha_t$ is known also at the PS, it can normalize the received vector to obtain:

$$\frac{1}{M \sqrt{\alpha_t}} y(\theta_t) = \frac{1}{M} \sum_{m=1}^M \tilde{g}_m(\theta_t) + \frac{1}{M \sqrt{\alpha_t}} z_t = A_s \frac{1}{M} \sum_{m=1}^M g_m^{sp}(\theta_t) + \frac{1}{M \sqrt{\alpha_t}} z_t.\quad (15)$$
Algorithm 1 A-DSGD

1: Initialize $\theta_1 = 0$ and $\Delta_{1,0} = \cdots = \Delta_{M,0} = 0$
2: for $t = 1, \ldots, T$ do
   
   - Workers do:
     3: for $m = 1, \ldots, M$ in parallel do
     4: Compute $g_m(\theta_t)$ with respect to $u_i \in B_{m,t}$
     5: $g_m^c(\theta_t) = g_m(\theta_t) + \Delta_{m,t-1}$
     6: $g_m^{sp}(\theta_t) = \text{sparsity}(g_m^c(\theta_t))$
     7: $\Delta_{m,t} = g_m^c(\theta_t) - g_m^{sp}(\theta_t)$
     8: EPA:
       9: $\hat{g}_m(\theta_t) = A_s g_m^{sp}(\theta_t)$
       10: Transmit $x_{m,t}(\theta_t) = \sqrt{\alpha_{m,t}} \hat{g}_m(\theta_t)$
     11: UPA:
       12: $\hat{g}_m(\theta_t) = A_{s-1} g_m^{sp}(\theta_t)$
       13: Transmit $x_{m,t}(\theta_t) = \left[ \sqrt{\alpha_{m,t}} g_m(\theta_t)^T \sqrt{\alpha_{m,t}} \right]^T$
     14: end for
   
   - PS does:
     15: EPA:
       16: $\frac{1}{M \sqrt{\alpha_t}} y(\theta_t) = \frac{1}{M} \sum_{m=1}^M g_m^{sp}(\theta_t) + \frac{1}{M \sqrt{\alpha_t}} z_t$
       17: $\hat{g}_{EPA}(\theta_t) = \text{AMP}_{A_s} \left( \frac{1}{M \sqrt{\alpha_t}} y(\theta_t) \right)$
       18: $\theta_{t+1} = \theta_t - \eta_t \cdot \hat{g}_{EPA}(\theta_t)$
     19: UPA:
       20: $\frac{1}{y_s(\theta_t) y^{s-1}(\theta_t)} = A_{s-1} \sum_{m=1}^M \frac{1}{\sum_{t=1}^T \sqrt{\alpha_{m,t} + z_{t,s}^2}} g_m^{sp}(\theta_t) + \frac{1}{\sum_{t=1}^T \sqrt{\alpha_{m,t} + z_{t,s}^2}} z_t^{s-1}$
       21: $\hat{g}_{UPA}(\theta_t) = \text{AMP}_{A_{s-1}} \left( \frac{1}{y_s(\theta_t) y^{s-1}(\theta_t)} \right)$
       22: $\theta_{t+1} = \theta_t - \eta_t \cdot \hat{g}_{UPA}(\theta_t)$
     23: end for

The goal of the PS is to recover $\frac{1}{M} \sum_{m=1}^M g_m^{sp}(\theta_t)$ from its noisy observation above. For this, we employ the approximate message passing (AMP) algorithm [37]. The AMP algorithm is represented by the AMP$_{A_s}$ function in Algorithm 1. The estimate $\hat{g}_{EPA}(\theta_t)$ is then used to update the model parameters as follows:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \hat{g}_{EPA}(\theta_t).$$  \hfill (16)

B. Unequal Power Allocation (UPA)

With the UPA scheme, we set $\bar{s} = s - 1$, which requires $s \geq 2$. At iteration $t$, we set $\alpha_{m,t} = \sqrt{\alpha_{m,t}}$, and worker $m$ computes $\hat{g}_m(\theta_t) = A_{s-1} g_m^{sp}(\theta_t)$, and sends vector $x_{m,t}(\theta_t) = \left[ \sqrt{\alpha_{m,t}} g_m(\theta_t)^T \sqrt{\alpha_{m,t}} \right]^T$ with the same power $P_t = ||x_{m,t}(\theta_t)||_2^2$ satisfying the average power
constraint $\frac{1}{T} \sum_{t=1}^{T} P_t \leq \bar{P}$, for $m \in [M]$. Accordingly, scaling factor $\sqrt{\alpha_{m,t}}$ is determined to satisfy

$$P_t = \alpha_{m,t} \left( \| \bar{g}_m(\theta_t) \|^2_2 + 1 \right), \quad (17)$$

where the term on the left hand side is $\| x_{m,t}(\theta_t) \|^2_2$, and it yields

$$\alpha_{m,t} = \frac{P_t}{\| \bar{g}_m(\theta_t) \|^2_2 + 1}, \quad \text{for } m \in [M]. \quad (18)$$

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

$$y(\theta_t) = \left[ A_{s-1} \sum_{m=1}^{M} \alpha_{m,t} \bar{g}_m^{sp}(\theta_t) \right] + z_t, \quad (19)$$

where $\alpha_{m,t}$, $m \in [M]$, is replaced by (18). For $i \in [s]$, we define

$$y^i(\theta_t) \triangleq \begin{bmatrix} y_1(\theta_t) & y_2(\theta_t) & \cdots & y_i(\theta_t) \end{bmatrix}^T \quad (20)$$

$$z^i_t \triangleq \begin{bmatrix} z_{t,1} & z_{t,2} & \cdots & z_{t,i} \end{bmatrix}^T, \quad (21)$$

where $y_j(\theta_t)$ and $z_{t,j}$ denote the $j$-th element of $y(\theta_t)$ and $z_t$, respectively. Thus, we have

$$y^{s-1}(\theta_t) = A_{s-1} \sum_{m=1}^{M} \sqrt{\alpha_{m,t}} \bar{g}_m^{sp}(\theta_t) + z^{s-1}_t, \quad (22a)$$

$$y_s(\theta_t) = \sum_{m=1}^{M} \alpha_{m,t} + z_{t,s}. \quad (22b)$$

Note that the goal is to recover $\frac{1}{M} \sum_{m=1}^{M} \bar{g}_m^{sp}(\theta_t)$ at the PS, while, from $y^{s-1}(\theta_t)$ given in (22a), the PS observes a noisy version of the weighted sum $\sum_{m=1}^{M} \sqrt{\alpha_{m,t}} \bar{g}_m^{sp}(\theta_t)$ projected to a low-dimensional vector through $A_{s-1}$. According to (18), each value of $\| \bar{g}_m(\theta_t) \|^2_2$ results in a distinct scaling factor $\alpha_{m,t}$. However, due to the independence of data samples, for large enough $d$ and $|B_{m,t}|$, the values of $\| \bar{g}_m(\theta_t) \|^2_2$, $\forall m \in [M]$, 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 $y^{s-1}(\theta_t)$ at the PS, given in (22a), 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 through $y_s(\theta_t)$ given in (22b). The resulting scaled vector at the PS is given by

$$\frac{1}{y_s(\theta_t)}y^{s-1}(\theta_t) = \frac{1}{y_s(\theta_t)} \left( A_{s-1} \sum_{m=1}^{M} \sqrt{\alpha_{m,t}} g_{m}^{sp}(\theta_t) + z^{s-1}_{t,s} \right) = A_{s-1} \sum_{m=1}^{M} \sqrt{\alpha_{m,t}} g_{m}^{sp}(\theta_t) + \frac{1}{\sum_{i=1}^{M} \sqrt{\alpha_{i,t}} + z^{s-1}_{t,s}} z^{s-1}_{t,s},$$

(23)

where $\alpha_{m,t}, m \in [M]$, is given in (18). By our choice, the PS tries to recover $\frac{1}{M} \sum_{m=1}^{M} g_{m}^{sp}(\theta_t)$ from $y^{s-1}(\theta_t)/y_s(\theta_t)$ knowing the measurement matrix $A_{s-1}$. The PS estimates $g_{UPA}(\theta_t)$ using the AMP algorithm. The estimate $\hat{g}_{UPA}(\theta_t)$ is then used to update the model parameter as follows:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \hat{g}_{UPA}(\theta_t).$$

(24)

**Remark 2.** We remark here that, with SGD the empirical variance of the stochastic gradient vectors reduce over time approaching zero asymptotically. The power should be allocated over iterations taking into account this decaying behaviour of gradient variance, while making sure that the noise term would not become dominant over time. To reduce the variation in the scaling factors $\alpha_{m,t}, \forall m \in [M]$, which is particularly efficient for the UPA power allocation scheme, variance reduction techniques can be used [41]. We also note that setting $P_t = \bar{P}, \forall t$, results in a special case of the UPA scheme, where the power is allocated uniformly over time to be resistant against the noise term.

**Remark 3.** We observe that, as opposed to the D-DSGD algorithm, increasing the number of workers $M$ can help increase the convergence speed for the A-DSGD algorithm. This is due to the fact that having more signals superposed over the MAC leads to more robust transmission against noise, particularly when the ratio $\bar{P}/s\sigma^2$ is relatively small, as we will observe in Fig. 6. Furthermore, for a higher $M$ value, $\frac{1}{M} \sum_{m=1}^{M} g_{m}^{sp}(\theta_t)$ provides a better estimation of the average of the actual gradient estimates $\frac{1}{M} \sum_{m=1}^{M} g_{m}(\theta_t)$, and also receiving information from higher number of workers can make the estimation of $\frac{1}{M} \sum_{m=1}^{M} g_{m}^{sp}(\theta_t)$, and consequently $\frac{1}{M} \sum_{m=1}^{M} g_{m}(\theta_t)$, more reliable.

**Remark 4.** We remark that in the considered model the main limitation is the channel bandwidth,
In the proposed A-DSGD algorithm, first a $k$-level sparsification is applied at worker $m$, resulting in vector $g_{m}^{sp}(\theta_{t})$, $m \in [M]$. Thus, $k$ can take different values satisfying $k < s$ leading to a tradeoff. For a relatively small value of $k$, $\frac{1}{M} \sum_{m=1}^{M} g_{m}^{sp}(\theta_{t})$ can be more reliably recovered from $\frac{1}{M} \sum_{m=1}^{M} \tilde{g}_{m}(\theta_{t})$; however it may not provide an accurate estimate of the actual average gradient $\frac{1}{M} \sum_{m=1}^{M} g_{m}(\theta_{t})$. Whereas, with a higher $k$ value, $\frac{1}{M} \sum_{m=1}^{M} g_{m}^{sp}(\theta_{t})$ provides a better estimate of $\frac{1}{M} \sum_{m=1}^{M} g_{m}(\theta_{t})$, but reliable recovery of $\frac{1}{M} \sum_{m=1}^{M} g_{m}^{sp}(\theta_{t})$ from the vector $\frac{1}{M} \sum_{m=1}^{M} \tilde{g}_{m}(\theta_{t})$ is less likely.

**Remark 5.** The proposed A-DSGD algorithm mainly focuses on the analog transmission from the workers, where the dimension of the gradient vectors is reduced utilizing the compressive sensing technique, leading to a more efficient communication scheme with smaller bandwidth requirement. While our focus in this paper has been on the transmission of the gradients, we can apply the existing schemes in the literature that trade-off an increase in the computation load at each worker with a reduction in the communication load. Such schemes include introducing communication delay, where each worker performs SGD algorithm updating the model parameter locally multiple times, and communicates only after multiple local iterations [32]. Moreover, applying momentum correction [3] improves the convergence speed of the DSGD algorithms with communication delay.

**C. Mean-Removal for Efficient Transmission**

To have a more efficient usage of the available power, each worker can remove the mean value of its gradient estimate before scaling and sending it. We define the mean value of $\tilde{g}_{m}(\theta_{t}) = A_{\tilde{s}}g_{m}(\theta_{t})$ as

$$\mu_{m,t} \triangleq \frac{1}{\tilde{s}} \sum_{i=1}^{\tilde{s}} \tilde{g}_{m,i}(\theta_{t}), \quad \text{for } m \in [M], \quad (25)$$

where $\tilde{g}_{m,i}(\theta_{t})$ is the $i$-th entry of vector $\tilde{g}_{m}(\theta_{t})$, $i \in [\tilde{s}]$. We also define $\tilde{g}_{m}^{az}(\theta_{t}) \triangleq \tilde{g}_{m}(\theta_{t}) - \mu_{m,t} 1_{\tilde{s}}$, $m \in [M]$. The power of vector $\tilde{g}_{m}^{az}(\theta_{t})$ is given by

$$\|\tilde{g}_{m}^{az}(\theta_{t})\|_2^2 = \|\tilde{g}_{m}(\theta_{t})\|_2^2 - (2\tilde{s} - 1)\mu_{m,t}^2. \quad (26)$$

For transmission over the MAC, we consider mean-removal EPA (MR-EPA) and mean-removal UPA (MR-UPA) transmission schemes described next.
In the MR-EPA scheme requiring \( s \geq 2 \), we consider \( \bar{s} = s - 1 \). At iteration \( t \), we set \( \alpha_{m,t} = \sqrt{\alpha_t^{az}} \mu_{m,t} \), and worker \( m \) computes \( \tilde{g}_m (\theta_t) = A_{s-1} g_{m}^{sp} (\theta_t) \), and sends vector \( x_{m,t} (\theta_t) = \left[ \sqrt{\alpha_t^{az}} g_m^{az} (\theta_t) \quad \sqrt{\alpha_t^{az}} \mu_{m,t} \right]^T \) with power

\[
\| x_{m,t} (\theta_t) \|_2^2 = \alpha_t^{az} (\| \tilde{g}_m (\theta_t) \|_2^2 - 2(s - 2)\mu_{m,t}^2), \quad \text{for } m \in [M].
\] (27)

The scaling factors \( \alpha_t^{az}, \forall t \), are chosen to satisfy the average power constraint given in (6). Remember that the instantaneous transmit power at worker \( m \) with EPA is \( \alpha_t \| \tilde{g}_m (\theta_t) \|_2^2 \), which, compared to (27), indicates that removing the mean reduces the transmit power by \( 2\alpha_t^{az}(s - 2)\mu_{m,t}^2, m \in [M] \). The received vector at the PS is given by

\[
y(\theta_t) = \left[ \sqrt{\alpha_t^{az}} \sum_{m=1}^{M} \tilde{g}_m^{az} (\theta_t) \right] + z_t = \left[ \sqrt{\alpha_t^{az}} \sum_{m=1}^{M} (A_{s-1} g_{m}^{sp} (\theta_t) - \mu_{m,t} 1_{s-1}) \right] + z_t,
\] (28)

where we have

\[
y^{s-1}(\theta_t) = \sqrt{\alpha_t^{az}} \left( A_{s-1} \sum_{m=1}^{M} g_{m}^{sp} (\theta_t) - \sum_{m=1}^{M} \mu_{m,t} 1_{s-1} \right) + z_t^{s-1}
\] (29a)

\[
y_s (\theta_t) = \sqrt{\alpha_t^{az}} \sum_{m=1}^{M} \mu_{m,t} + z_{t,s}.
\] (29b)

In order to recover \( \frac{1}{M} \sum_{m=1}^{M} g_{m}^{sp} (\theta_t) \) from \( y^{s-1}(\theta_t) \), given in (29a), we first need to cancel the term \( -\sqrt{\alpha_t^{az}} \sum_{m=1}^{M} \mu_{m,t} 1_{s-1} \), whose noisy version is received through \( y_s (\theta_t) \), given in (29b). Dividing the resultant vector by \( M \sqrt{\alpha_t^{az}} \) yields the following

\[
\frac{1}{M \sqrt{\alpha_t^{az}}} (y^{s-1}(\theta_t) + y_s (\theta_t) 1_{s-1}) = \frac{1}{M} \left( A_{s-1} \sum_{m=1}^{M} g_{m}^{sp} (\theta_t) + \frac{z_{t,s} 1_{s-1} + z_t^{s-1}}{\sqrt{\alpha_t^{az}}} \right).
\] (30)

The PS tries to recover \( \frac{1}{M} \sum_{m=1}^{M} g_{m}^{sp} (\theta_t) \) by applying the AMP algorithm on the expression on the left hand side of (30).

On the other hand, with MR-UPA, we set \( \bar{s} = s - 2 \), which requires \( s \geq 3 \). We also set \( \alpha_{m,t} = \left[ \sqrt{\alpha_{m,t}^{az}} \mu_{m,t} \quad \sqrt{\alpha_{m,t}^{az}} \right]^T \), and after computing \( \tilde{g}_m (\theta_t) = A_{s-2} g_{m}^{sp} (\theta_t) \), worker \( m \) sends

\[
x_{m,t} (\theta_t) = \left[ \sqrt{\alpha_{m,t}^{az}} g_m^{az} (\theta_t) \quad \sqrt{\alpha_{m,t}^{az}} \mu_{m,t} \sqrt{\alpha_{m,t}^{az}} \right]^T, \quad \text{for } m \in [M],
\] (31)
with power

\[
\|\mathbf{x}_{m,t}(\theta_t)\|^2 = \alpha_{m,t}^{az} \left(\|\mathbf{g}_m(\theta_t)\|^2 - 2(s-3)\mu_{m,t}^2 + 1\right),
\]

(32)

which is chosen to be equal to \(P_t\), such that \(\frac{1}{T} \sum_{t=1}^{T} P_t \leq \bar{P}\). Thus, we have

\[
\alpha_{m,t}^{az} = \frac{P_t}{\|\mathbf{g}_m(\theta_t)\|^2 - 2(s-3)\mu_{m,t}^2 + 1}, \quad \text{for} \ m \in [M].
\]

(33)

Compared with the instantaneous transmit power at worker \(m\) with UPA, given in (17), we observe that removing the mean reduces the transmit power by

\[
2\alpha_{m,t}^{az}(s-3)\mu_{m,t}, \quad m \in [M].
\]

(34)

The received vector at the PS is given by

\[
y(\theta_t) = \left[ \sum_{m=1}^{M} \sqrt{\alpha_{m,t}^{az}} \mathbf{g}_m(\theta_t) \right] + \mathbf{z}_t = \left[ \sum_{m=1}^{M} \sqrt{\alpha_{m,t}^{az}} (A_{s-2}\mathbf{g}_m(\theta_t) - \mu_{m,t}1_{s-2}) \right] + \mathbf{z}_t,
\]

where we have

\[\begin{align*}
y_{s-2}(\theta_t) &= A_{s-2} \sum_{m=1}^{M} \sqrt{\alpha_{m,t}^{az}} \mathbf{g}_m(\theta_t) - \sum_{m=1}^{M} \sqrt{\alpha_{m,t}^{az}} \mu_{m,t}1_{s-2} + \mathbf{z}_{t,s}^{s-2}; \\
y_{s-1}(\theta_t) &= \sum_{m=1}^{M} \sqrt{\alpha_{m,t}^{az}} \mu_{m,t} + \mathbf{z}_{t,s-1}; \\
y_s(\theta_t) &= \sum_{m=1}^{M} \sqrt{\alpha_{m,t}^{az}} + \mathbf{z}_{t,s}.
\end{align*}\]

(35)

The PS performs AMP to recover \(\frac{1}{M} \sum_{m=1}^{M} \mathbf{g}_m(\theta_t)\) from the following vector

\[
\frac{1}{y_s(\theta_t)} (y_{s-2}(\theta_t) + y_{s-1}(\theta_t)1_{s-2}) = A_{s-2} \sum_{m=1}^{M} \sqrt{\alpha_{m,t}^{az}} \mathbf{g}_m(\theta_t) + \frac{\mathbf{z}_{t,s-1}1_{s-2} + \mathbf{z}_{t,s}^{s-2}}{\sum_{i=1}^{M} \sqrt{\alpha_{i,t}^{az}} + \mathbf{z}_{t,s}}.
\]

(36)

Comparison between the digital and analog approaches and the impact of various power allocation schemes are studied in the next section through numerical simulations.
V. NUMERICAL RESULTS

In this section, we evaluate the performances of the proposed A-DSGD and D-DSGD algorithms numerically. We consider a logistic regression problem, in which the goal is to minimize the empirical loss function

\[ F(\theta) = -\frac{1}{N} \sum_{i=1}^{N} v_i \log(h_\theta(u_i)) + (1 - v_i) \log(1 - h_\theta(u_i)), \tag{37} \]

where \( h_\theta : \mathbb{R}^d \to \mathbb{R} \) is given by

\[ h_\theta(u) = \frac{1}{1 + e^{-\theta^T u}} \tag{38} \]

and \( v_i \in \{0, 1\} \) is the label corresponding to the \( i \)-th data sample \( u_i \in \mathbb{R}^d, i \in [N] \). We have, for \( i \in [N] \),

\[ f(\theta, u_i) = - (v_i \log(h_\theta(u_i)) + (1 - v_i) \log(1 - h_\theta(u_i))) \tag{39} \]

and

\[ \nabla f(\theta, u_i) = (h_\theta(u_i) - v_i) u_i. \tag{40} \]

At each iteration \( t \), worker \( m \), for \( m \in [M] \), computes

\[ g_m(\theta_t) = \frac{1}{|B_{m,t}|} \sum_{i \in B_{m,t}} \nabla f(\theta_t, u_i) = \frac{1}{|B_{m,t}|} \sum_{i \in B_{m,t}} (h_{\theta_t}(u_i) - v_i) u_i, \tag{41} \]

thus, it requires data samples \( u_i \) and their corresponding labels \( v_i, \forall i \in B_{m,t} \).

We consider a toy example, in which we fix a parameter vector \( \theta^* \in \mathbb{R}^d \), and we independently generate \( N \) data samples \( u_1, \ldots, u_N \) with corresponding labels \( v_i = (\text{sign}(u_i^T \theta^*) + 1)/2 \), where function \( \text{sign}(x), x \in \mathbb{R}, \) is defined as follows:

\[ \text{sign}(x) \triangleq \begin{cases} 1, \quad \text{if } x \geq 0, \\ -1, \quad \text{if } x < 0, \end{cases} \tag{42} \]

We add 5\% of random noise to the labels, i.e., the value of the \( i \)-th label is changed from \( v_i \) to \( 1 - v_i \), for \( i \in S \), where the set \( S \subset [N] \) with \( |S| = 0.05N \) is chosen at random. We also generate \( N' \) random test data samples \( u'_1, \ldots, u'_{N'} \) independently, where the label corresponding to \( u'_i \) is \( v'_i = (\text{sign}(u'_i^T \theta^*) + 1)/2 \), for \( i \in [N'] \). At iteration \( t \), after obtaining the parameter
vector $\theta_t$ using any DSGD scheme, $F(\theta_t)$, given in (37), provides the loss function, and defining $S' \triangleq \{ i : \left( \text{sign} \left( \mathbf{w}_i^T \theta_t \right) + 1 \right) / 2 \neq v'_i \}$, the accuracy is obtained by $|S'| / N'$.

For simplicity, we assume throughout the experiments that $|B_{m,t}| = B$, $\forall m, t$, and data samples and their corresponding labels available at each worker remain fixed over time. We also assume that $N = MB$, and there are no overlaps between the datasets of the workers, i.e., $B_{i,t} \cap B_{j,t} = \emptyset$, $\forall i, j \in [M]$ where $i \neq j$. We set the channel noise variance to $\sigma^2 = 1$. The network is trained utilizing the ADAM optimizer [42], and the number of test data samples is assumed to be $N' = N/2$.

When performing the A-DSGD algorithm with the EPA (UPA) power allocation scheme, we start the algorithm by using MR-EPA (MR-UPA) and after 20 iterations we continue the algorithm utilizing EPA (UPA). The reason is that the gradients are more aggressive at the beginning of the A-DSGD algorithm, and their mean values are expected to be relatively diverse, however, over time, they converge to zero.

In Fig. 2, we compare the performance of the A-DSGD algorithm with both EPA and UPA with D-DSGD algorithm for different values of the available average transmit power $\bar{P} \in \{455, 1170\}$. Since, for the digital approach, we need to have $r_t \leq R_t$, we set number of channel uses $s$ and $P$ to relatively high values and number of workers $M$ to a relatively small value to make sure that $q_t \geq 1$, $\forall t$, i.e., each worker can transmit at least one information bit at each iteration. Accordingly, we consider a distributed computation system with $N = 12500$ samples, $M = 25$ workers, $B = 500$ data samples available at each worker, dimension of the parameter vector $d = 1500$, and number of channel uses $s = 0.4d = 600$. We set a fixed ratio $k = s/2 = 300$ for sparsification. 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 $\bar{P} = 1170$ since it is very close to the one with $\bar{P} = 455$ 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. Therefore, analog computation approach is particularly attractive for learning across low-power devices as it allows them to align their limited transmission powers to dominate the noise term. For the UPA, we set $P_t = \bar{P}$, $\forall t$, which satisfies the average power constraint, and for the EPA, we set $\alpha_t = \sqrt{500 + 5t}/4$ and $\alpha_t = \sqrt{1000 + 13t}/2$ resulting $\bar{P} = 455$ and $\bar{P} = 1170$, respectively. For each average power constraint $\bar{P}$, we consider two different power allocation schemes for transmission with the D-DSGD algorithm: in the first
Fig. 2: Performance of the A-DSGD and D-DSGD algorithms for different $\bar{P}$ values.

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 leading to a distinct $P_t$ value at each iteration $t$. Observe that, for the D-DSGD algorithm, letting $P_t$ vary over time improves the performance, particularly at the initial iterations and for smaller $\bar{P}$ values; however, with further iterations, the gap between the two decreases, which is due to the reduction in the variance of the gradient vectors causing the noise term to be relatively more dominant with the time-varying power allocation compared to setting $P_t = \bar{P}$, $\forall t$.

In Fig. 3 we compare the performance of the A-DSGD algorithm with UPA and the D-DSGD algorithm, where, for both analog and digital communications, we set $P_t = \bar{P}$, $\forall t$, for different $M$ and $s$ values. We consider a setting with parameters $N = 3000$, $d = 1500$, and $\bar{P} = 400$. We consider two different wireless networks $(M, B) = \{(20, 150), (30, 100)\}$ assuming that the total size of the dataset remains the same independent of the number of workers, and for each, we consider two different values of number of channel uses $s \in \{2d/5, d/2\} = \{600, 750\}$, and a fixed ratio $k = s/2 \in \{300, 375\}$. As it can be seen, for $s = 2d/5$, increasing $M$ by a factor of 1.5 significantly deteriorates the performance of D-DSGD. Accordingly, the performance of D-DSGD algorithm is highly 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. 2. We can conclude that the digital scheme prefers to have a smaller number of workers, which are then allocated more
Iteration, $t$

Loss function, $F(\theta_t)$

Accuracy

DDSGD, $P_t = \bar{P}$, $M = 20$, $B = 150$, $s = d/5$

DDSGD, $P_t = \bar{P}$, $M = 20$, $B = 150$, $s = d/2$

DDSGD, $P_t = \bar{P}$, $M = 20$, $B = 150$, $s = 2d/5$

ADSGD, MR-UPA, UPA, $M = 20$, $B = 150$, $s = d/5$

ADSGD, MR-UPA, UPA, $M = 20$, $B = 150$, $s = d/2$

Fig. 3: Performance of the A-DSGD and D-DSGD algorithms for different $(M, s)$ pairs.

channel resources to be able to transmit their gradient estimates to the PS more accurately. However, this means that D-DSGD cannot harvest the computation power of many edge devices; and its performance compared to A-DSGD will become even poorer when the computation time and energy is also taken into account. On the other hand, we observe that the performance of A-DSGD improves slightly by increasing $M$ from $M = 20$ to $M = 30$ when $s = 2d/5$, and is significantly superior compared to D-DSGD, and the improvement increases remarkably with $M$. We will investigate the effect of increasing $M$ on A-DSGD in more details in Fig. 6 and we will observe that increasing $M$ improves the performance of A-DSGD. We further observe from Fig. 3 that reducing the available channel uses $s$ from $s = d/2$ to $s = 2d/5$ degrades the performance of the D-DSGD algorithm considerably, whereas the sensitivity of A-DSGD to channel bandwidth is much weaker.

In Fig. 4 and Fig. 5 we compare the performance of the A-DSGD algorithm for different $s$ values. We consider a system with $N = 10000$, $M = 100$, $B = 100$, and $d = 1000$. We consider $s \in \{d/8, d/4, d/2\} = \{125, 250, 500\}$, and a fixed ratio $k = 4s/5 \in \{100, 200, 400\}$. For the EPA scheme, which is only considered for the case $s = 100$ channel uses, we set $\alpha_t = \sqrt{1/5 + t/1000}$ leading to $P = 2.7$. For the UPA, we set $P_t = P = 2.7, \forall t$. In Fig. 4 we consider different $s$ values for each iteration of the A-DSGD algorithm, and plot the performance with respect to the iteration count $t$. This may correspond to allocating different
Fig. 4: Performance of the A-DSGD algorithm as a function of the iteration count for different number of channel uses \( s \in \{d/8, d/4, d/2\} \) per iteration.

Fig. 5: Performance of the A-DSGD algorithm as a function of the normalized time \( t_s \) for different number of channel uses \( s \in \{d/8, d/4, d/2\} \).

number of channel resources for each iteration, for example, allocating more sub-bands for the computation tasks through OFDM. On the other hand, in Fig. 5 we limit the number of channel uses at each communication round, and assume that the transmission time at each iteration linearly increases with \( s \), and evaluate the performance of the A-DSGD algorithm with respect
to the normalized transmission time $ts$. This setting assumes a total number of channel uses dedicated for the whole computation tasks through time-division multiple-access (TDMA), and therefore, assigning more resources to each iteration, i.e., increasing $s$, means that less number of iterations can be implemented by the same time. Observe from Fig. 4 that, as expected, utilizing more channel uses can significantly improve the performance both in terms of the convergence rate and the accuracy. Furthermore, by the choice of the parameters for EPA and UPA power allocation schemes, we can see that EPA slightly outperforms UPA. On the other hand, we see from Fig. 5 that the performance of the A-DSGD algorithm degrades slightly by reducing the available number of channel uses from $s = 500$ to $s = 125$ with respect to the normalized transmission time. This means that the convergence is faster if more accurate gradient estimations are transmitted at each iteration, compared to carrying out more iterations with less accurate communication of the local gradient estimates. However, we note here that we do not consider the computation time. The computation time in the workers remain mostly the same independent of $s$, as the same number of gradients are computed. The computation time at the PS increases slightly with $s$ due to the increasing number of parameters input to the AMP reconstruction algorithm.

In Fig. 6, we investigate the performance of the A-DSGD algorithm with UPA power allocation scheme for different $M$ and $\bar{P}$ values. We consider a distributed computation system with $N = 10000$, $d = 1000$, and $s = 0.15d = 150$. We set $k = s/3 = 50$. We consider three different networks $(M, B) = \{(1000, 10), (100, 100), (50, 200)\}$, and for each, we consider two different average transmit power values $\bar{P} \in \{2, 100\}$, and we set $P_t = \bar{P}, \forall t$. As it can be seen, for the smaller $\bar{P}$ value, increasing $M$ leads to a significant performance improvement in terms of the convergence speed and the accuracy. This is due to the fact that, in analog transmission of the gradients, adding more signals over the MAC increases the robustness of the estimation against the noise term. Although the accuracy of individual gradient estimates degrade with $M$ (due to the reduction in the sample size available at each worker), A-DSGD algorithm benefits from the additional power introduced by each worker. Furthermore, even for large enough $\bar{P}$, increasing $M$ slightly improves the performance. We observe that, when the signal can dominate the noise term, increasing $\bar{P}$ provides a negligible performance improvement, e.g., for the case of $M = 1000$. Otherwise, increasing $\bar{P}$ can make the signal term dominant compared to the noise, which leads to a more significant performance improvement, e.g., for cases $M = 100$ and $M = 50$. 
VI. CONCLUSIONS

We have studied distributed machine learning at the wireless network edge, where $M$ wireless nodes, referred to as the workers, aim to minimize an empirical loss function collaboratively by performing DSGD with the help of a remote PS. Workers have their own local datasets, and they communicate with the PS over a wireless MAC. The PS updates the global parameter vector as a function of the noisy channel output it receives from the workers. We have assumed that the communication from the PS to the workers is noiseless, so the updated parameter vector is shared with the workers in a lossless fashion.

As opposed to standard approach to distributed machine learning, which ignores the channel aspects, and simply aims at reducing the communication load by compressing the gradients at each iteration to a prefixed level, here we incorporate the wireless channel characteristics and constraints into the system design. 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 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 $\bar{P}$, and each worker digitizes its computed gradient utilizing the state-of-the-art DSGD algorithm followed by capacity-achieving channel coding. We have shown that, under a finite power constraint, the
convergence speed can be improved by allocating more power to earlier iterations when the gradient variance is high.

In the A-DSGD scheme, we have proposed gradient sparsification followed by compressive sensing employing the same measurement matrix at all the workers. This allowed reducing the typically very large parameter vector dimension to the limited channel bandwidth. The workers then transmit these compressed gradient vectors simultaneously over the MAC, which naturally adds them. This analog approach allows a much more efficient use of the available limited channel bandwidth, and benefits from the “beamforming effect” thanks to the highly correlated 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 the performance of A-DSGD improves with the number of workers, whereas the performance of the D-DSGD algorithm deteriorates as more workers are introduced. Future work will include extending this framework to fading channels, and incorporating the computation time and energy into the optimization framework.

REFERENCES

[1] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, “Wireless network intelligence at the edge,” CoRR, vol. abs/1812.02858, 2018. [Online]. Available: http://arxiv.org/abs/1812.02858

[2] D. Alistarh, D. Grubic, J. Z. Li, R. Tomioka, and M. Vojnovic, “QSGD: Communication-efficient SGD via randomized quantization and encoding,” in Advances in Neural Information Processing Systems, Long Beach, CA, USA, Dec. 2017, pp. 1709–1720.

[3] 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.

[4] 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.

[5] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” IEEE Trans. Inform. Theory, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[6] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, “Gradient coding: Avoiding stragglers in distributed learning,” in Proc. of International Conference on Machine Learning, Sydney, Australia, Aug. 2017, pp. 3368–3376.

[7] W. Halbawi, N. A. Ruhi, F. Salehi, and B. Hassibi, “Improving distributed gradient descent using Reed-Solomon codes,” arXiv:1706.05436v1 [cs.IT], Jun. 2017.

[8] G. Joshi, S. Dutta, S. Ghosh, P. Dube, and P. Nagpurkar, “Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD,” arXiv:1803.01113v3 [cs.IT], May 2018.

[9] R. K. Maity, A. S. Rawat, and A. Mazumdar, “Robust gradient descent via moment encoding with LDPC codes,” arXiv:1805.08327v1 [cs.IT], May 2018.
[31] Y. Tsuzuku, H. Imachi, and T. Akiba, “Variance-based gradient compression for efficient distributed deep learning,” arXiv:1802.06058v2 [cs.LG], Feb. 2018.

[32] 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.

[33] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” arXiv:1610.05492v2 [cs.LG], Oct. 2017.

[34] S. U. Stich, “Local SGD converges fast and communicates little,” arXiv:1805.09767v2 [math.OC], Jun. 2018.

[35] T. Lin, S. U. Stich, and M. Jaggi, “Don’t use large mini-batches, use local SGD,” arXiv:1808.07217v3 [cs.LG], Oct. 2018.

[36] T-Y. Tung and D. Gündüz, “SparseCast: Hybrid digital-analog wireless image transmission exploiting frequency-domain sparsity,” IEEE Commun. Lett., vol. 22, no. 12, pp. 2451–2454, Dec. 2018.

[37] D. L. Donoho, A. Maleki, and A. Montanari, “Message-passing algorithms for compressed sensing,” Proc. Nat. Acad. Sci. USA, vol. 106, no. 45, pp. 18914–18919, Nov. 2009.

[38] G. Zhu, Y. Wang, and K. Huang, “Low-latency broadband analog aggregation for federated edge learning,” to be submitted.

[39] 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.

[40] N. Strom, “Scalable distributed DNN training using commodity GPU cloud computing,” in INTERSPEECH, 2015, pp. 1488–1492.

[41] R. Johnson and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” in Proc. International Conference on Neural Information Processing Systems (NIPS), Nevada, USA, Dec. 2013.

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