LENA: Communication-Efficient Distributed Learning with Self-Triggered Gradient Uploads

Hossein S. Ghadikolaei
KTH, Stockholm, Sweden

Sebastian U. Stich
EPFL, Lausanne, Switzerland

Martin Jaggi
EPFL, Lausanne, Switzerland

Abstract

In distributed optimization, parameter updates from the gradient computing node devices have to be aggregated in every iteration on the orchestrating server. When these updates are sent over an arbitrary commodity network, bandwidth and latency can be limiting factors. We propose a communication framework where nodes may skip unnecessary uploads. Every node locally accumulates an error vector in memory and self-triggers the upload of the memory contents to the parameter server using a significance filter. The server then uses a history of the nodes' gradients to update the parameter. We characterize the convergence rate of our algorithm in smooth settings (strongly-convex, convex, and nonconvex) and show that it enjoys the same convergence rate as when sending gradients every iteration, with substantially fewer uploads. Numerical experiments on real data indicate a significant reduction of used network resources (total communicated bits and latency), especially in large networks, compared to state-of-the-art algorithms. Our results provide important practical insights for using machine learning over resource-constrained networks, including Internet-of-Things and geo-separated datasets across the globe.

1 Introduction

We consider the training of central machine learning models where the private training data sets are distributed among M nodes (e.g. data centers or mobile devices). This problem can be formulated as the distributed optimization problem

$$\boldsymbol{w}^\star := \underset{\boldsymbol{w}\in\mathbb{R}^d}{\arg\min} \left[F(\boldsymbol{w}) := \frac{1}{M} \sum_{m\in[M]} f_m(\boldsymbol{w}) \right], \quad (1)$$

where here $f_m\colon \mathbb{R}^d \to \mathbb{R}$ denotes the loss function on node $m \in [M]$ (note that in general $f_i \neq f_j$, for $i \neq j$). Distributed stochastic gradient descent with learning rate $\alpha > 0$ addresses (1) by running iterations of the form

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \tfrac{\alpha}{M} \textstyle\sum_{m\in[M]} \boldsymbol{g}_m^t \,, \quad (2)$$

where $\boldsymbol{g}_m^t$ is an unbiased estimator of the local loss function $\nabla f_m(\boldsymbol{w}^t)$ (e.g. computed over a mini-batch of the local data on node $m \in [M]$). In a distributed computation setting, each update of the form (2) encounters non-negligible communication cost, to gather all the local gradients $\{\boldsymbol{g}_m^t\}_{m\in[M]}$ on the server and broadcast the updated global parameter $\boldsymbol{w}^{t+1}$. These costs become of paramount importance when implemented on resource-constrained networks with nodes distributed across the world (Hsieh et al., 2017) or with nodes wirelessly connected to the parameter server (Gündüz et al., 2019). Prior work addresses this communication bottleneck by introducing (i) gradient compression techniques, to reduce the number of bits per communication round (Alistarh et al., 2017; Mishchenko et al., 2019), and (ii) local updates, where communication rounds are skipped according to a prescribed schedule (McMahan et al., 2016; Stich, 2019a).

Whilst gradient compression techniques can drastically reduce the size of the uplink transmissions[1] (from $\mathcal{O}(d)$ to $\mathcal{O}(\log d)$ (Alistarh et al., 2017) or even to $\mathcal{O}(1)$ (Stich et al., 2018) per node), current approaches often neglect communication overheads, such as latency,

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

[1]For instance in the case of wireless networks, the uplink channel to send the gradients to the server is often the main bottleneck, since a limited resource should be shared among multiple transmitter nodes. On the other hand, the cost of broadcasting a new parameter to all nodes is almost identical to the cost of sending it to a subset of the nodes, due to the broadcast nature of wireless communications (El Gamal and Kim, 2011).

packet overheads in standard communication protocols, or hardware constraints such as power consumption. In many real-world networks, a few silent nodes that do not transmit any data can yield much more drastic speedups and savings of network resources than simply compressing every message down to a few bits (but keeping all nodes active) (Van Dam and Langendoen, 2003). Local update algorithms defer communication rounds by allowing the nodes to skip either a fixed number of uploads (McMahan et al., 2016; Stich, 2019a; Karimireddy et al., 2019a) or in adaptive schemes, such as lazily aggregated gradient (LAG) (Chen et al., 2018) and sparsified action regulated quantized SGD (SPARQ-SGD) (Singh et al., 2020), the frequencies are not fixed and the nodes send an upload only if their updates have a significant contribution to the centralized parameter updates.

We propose a novel event-triggered communication framework, called LENA, where nodes adaptively and only infrequently upload their computed gradients, without any explicit coordination with other nodes and not relying on a prescribed communication schedule. Our approach converge to a solution of (1) for any smooth f, whilst other 'ad-hoc' approaches (such as the (Kamp et al., 2019)) either only work for specific f or specific set of algorithms (the so-called f-proportional convex update).

A key mechanism in our scheme is that silent nodes still contribute to the parameter update on the server by means of an individual *drift* term which is applied by default when the server does not receive an update from a particular node. Distinctive to LAG, this drift term can be chosen arbitrarily while our framework still guarantees convergence. LENA substantially improves the performance of LAG in our experiments by as much as 99% fewer total uploaded bits.

Our main contributions are as follows:

- We present a general framework for communication constrained distributed optimization with drift compensated updates for silent workers. We prove convergence of LENA (for arbitrary drift compensation) on convex and nonconvex problems and show that it converges as fast as distributed SGD but with the possibility to skip unnecessary uploads.

- We evaluate several choices for the drift compensation terms in numerical experiments and give concrete suggestions which of them reduce the communication frequency by the most. We further argue that our method performs especially well in the challenging non-IID (heterogeneous) data setting that—conversely—is the most challenging for other local update schemes (Karimireddy et al., 2019a).

- We run comprehensive numerical analyses on real datasets to characterize the impact of various algorithm and networking parameters on the convergence and overheads of our algorithm. Our results indicate that LENA achieves the same accuracy as the state-of-the-art benchmarks with orders of magnitude fewer communicated bits and lower latency. All these gains improve with the network size, making our approach suitable for efficient distributed machine learning over a network of a massive number of workers.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 4 presents the problem setting, our proposed algorithm, and its theoretical convergence results. We present our experimental results in Section 5, and conclude the paper in Section 6. For the sake of readability, we have moved extra definitions, lemmas, extra numerical results, and all the proofs to the appendix.

Notation: Normal font w or W, bold font small-case $\boldsymbol{w}$, and bold-font capital letter $\boldsymbol{W}$ denote scalar, vector, matrix, and set, respectively. We let $[N] = \{1, 2, \dots, N\}$ for any integer N. We denote by $\|\cdot\|$ the l_2 norm and by $\boldsymbol{w}^{\mathrm{T}}$ the transpose of $\boldsymbol{w}$.

2 Related Work

Communication-efficient distributed optimization addresses the tradeoff between computational gain due to parallel processing and the resulting communication overhead. In general, there are two complementary approaches to address the communication bottleneck: compressing messages in every communication round and skipping some communication rounds (Tang et al., 2020).

2.1 Compressing Messages in Every Communication Round

A prominent approach is to use a lossy-compression (realized through quantization with few bits) of the parameter and gradient vectors in every iteration to save the communication resources. This approach has been applied to both deterministic (Jordan et al., 2018; Magnússon et al., 2017; Magnusson et al., 2020) and stochastic (Alistarh et al., 2017; Bernstein et al., 2018; Wen et al., 2017; De Sa et al., 2018; Mishchenko et al., 2019) algorithms.

Seide et al. (2014) and Bernstein et al. (2018) proposed 1bitSGD and sign-SGD, respectively, in which only the sign of each coordinate is used for the updates. Surprisingly, such extreme quantization may not hurt the convergence in deep learning problems, as shown in those papers. The convergence of 1-bit quantization is also shown in the proximal methods (Xu and Kamilov, 2019) and for distributed methods (Magnússon et al., 2017; Mishchenko et al., 2019). Alistarh et al. (2017) proposed multiple levels for the quantization, and Lin et al. (2018) added momentum correction and local gradient clipping to further reduce the communication overhead. Sparsification is an alternative approach for lossy-compression in which only a subset of "most prominent" coordinates of the gradients will be communicated to the parameter server while the rest being dropped. An example approach includes sending only Top-k values of the gradient vector (in absolute value) as well as sending k randomly selected coordinates, Rand-k (Alistarh et al., 2018; Stich et al., 2018). Acharya et al. (2019) introduced a family of sparsified mirror descent algorithms to solve a distributed learning problem with $o(d)$ communications.

To maintain the asymptotic convergence rate with compressed messages, error compensation (Stich et al., 2018; Alistarh et al., 2018; Karimireddy et al., 2019b; Stich and Karimireddy, 2019; Basu et al., 2019) and adaptive quantizers (Pu et al., 2017; McGlohon and Patterson, 2016; De Sa et al., 2018; Magnusson et al., 2020; Ghadikolaei and Magnusson, 2020) have been proposed in the literature. Tang et al. (2019) combined error compensation and lossy quantization to quantize both gradients in uplink and parameters in downlink. Phuong et al. (2020) proposed an algorithm where every node uploads a quantized version of its gradient with probability 0.3 (hardcoded in the algorithm). Wangni et al. (2019) introduced the concept of trajectory normalized gradients, where all nodes share in advance a reference gradient vector, and they share only the changes. The reference can dynamically change based on the local optimization landscape, which makes it similar to adaptive quantization algorithms, but also a good measure for detecting "significant" changes, as we use in this paper.

All these approaches alleviate the communication between the server and the nodes at every iteration. However, every upload is often subject to extra overheads (e.g., extra bits for channel encoding and packet headers, and extra latency for queuing and network congestion), and reducing the number of payload bits cannot reduce those overheads (El Gamal and Kim, 2011). In fact, extreme payload quantization may not bring any noticeable reduction in the total number of transmitted bits or latency (Magnusson et al., 2020). When many nodes are connected to the server via a faulty wireless channel or public internet, like massive edge computing scenarios, additional functionalities of communication protocols (like adaptive transmission modes, TCP window size, and queue policy of intermediate routers) may rise to straggler and missing updates, which may substantially affect the overall performance of distributed learning algorithm.

2.2 Eliminating Some Communication Rounds

The second category includes algorithms that eliminate communication between some of the nodes and the server in some iterations.

Predefined intermittent communication schedule: Local update algorithms can reduce the communication frequency by allowing nodes to perform several epochs over their local data and upload the final parameter (Stich, 2019a; Hsieh et al., 2017; Singh et al., 2020) or gradient (Zhu et al., 2019) with a predefined upload schedule for a global update. Zhang et al. (2016) showed that the global averaging step acts like a variance-reducing mechanism, and its proper frequency depends on the noise of local gradients. However, it is often hard to find a proper communication schedule that still allows convergence without increasing the total number of iterations. Moreover, the upload frequency should be substantially increased for non-IID data to avoid a huge drift of local updates.

Adaptive intermittent communication: Hsieh et al. (2017) proposes the notion of significance filter in local SGD by which every node uploads its parameter (for the averaging step) only when there is a significant change in the parameter vector, measured by the Euclidean norm. This algorithm is extended in concurrent parallel work by Singh et al. (2020); Singh et al. (2020) by adding an error feedback framework and momentum. Kamp et al. (2019) modified the upload criteria to the average divergence of the parameter vector over a window of k iterations. However, these approaches are more suited for IID data where the upload threshold does not need to consider local geometry of the optimization landscape. In particular, a worker with the least smooth optimization landscape forces a small upload threshold for all workers, leading to many unnecessary uploads of workers with smoother landscapes.

Zhu et al. (2019) introduced a new algorithm in which the parameters can be updated based on the delayed and temporally sparse global gradient aggregations. The authors introduced error feedback to correct the steps in which only local information is available. Moreover, this approach may adverse the effect of the straggler problem, since the slowest processor may need more time to run more local computations. Chen et al. (2018) proposed lazily aggregated gradient (LAG) for communication-efficient distributed learning with full gradients. In LAG, each node reports its gradient vector to the server only if the changes to the gradient from the last step is large enough. That way, some nodes may skip sending their gradients at some iterations, which saves communication resources. Li et al. (2019) and Chen et al. (2020) extended LAG to stochastic gradient, Sun et al. (2019) added gradient quantization per upload, and Singh et al. (2020) added error feedback and extended LAG to a decentralized setting in SPARQ-SGD. Our algorithm introduces the concept of drift function, which generalizes iterations of SPARQ-SGD. Moreover, unlike SPARQ-SGD, we do not rely on a pre-engineered schedule (or decision thresholds) to trigger the uploads, which substantially improves its applicability in practice. The concept of using a local drift to to trigger data upload has been also investigated for linear regression (Gabel et al., 2015). However, its extension to the nonconvex setting and nodes with non-IID data is not trivial. Moreover, Heemels et al. (2012) introduced the concept of locally event-triggered control in a different context.

3 Algorithm — LENA

The error feedback framework closely follows the template of distributed SGD as given in (2), but instead of using the computed gradients $\{\boldsymbol{g}_m^t\}$ directly for the parameter update, the server uses *gradient estimates* $\{\boldsymbol{v}_m^t\}$ as well as aggregated errors $\{\boldsymbol{e}_m^t\}$ (Stich and Karimireddy, 2019):

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \frac{1}{M}\sum_{m\in[M]} \boldsymbol{v}_m^t\,, \tag{3a}$$

$$\boldsymbol{v}_m^t = \text{Estimate}\left(\boldsymbol{e}_m^t + \alpha \boldsymbol{g}_m^t\right)\,, \tag{3b}$$

$$\boldsymbol{e}_m^{t+1} = \boldsymbol{e}_m^t + \alpha \boldsymbol{g}_m^t - \boldsymbol{v}_m^t\,, \tag{3c}$$

where the Estimate operator can represent a compression algorithm or delayed uploads. We extend these iterates by removing the necessity of uploading at every round. To this end, we redefine $\boldsymbol{v}_m^t$ as

$$\boldsymbol{v}_m^t = \begin{cases} \boldsymbol{e}_m^t + \alpha \boldsymbol{g}_m^t & \text{if } \|\boldsymbol{e}_m^t + \alpha \boldsymbol{g}_m^t - \boldsymbol{v}_m^{t-1}\|^2 \geq \beta \|\boldsymbol{g}_m^{t-1}\|^2 \\ \boldsymbol{q}_m^t & \text{otherwise}\,, \end{cases} \tag{4}$$

where for node m at iteration t, $\boldsymbol{q}_m^t$ is the drift parameter (formally defined later), and β is a non-negative constant. When the upload is triggered according to (4), the server applies $\boldsymbol{e}_m^t + \alpha \boldsymbol{g}_m^t$, but when no upload is triggered the server simply uses the previously received $\boldsymbol{q}_m^t$ as this workers contribution (as we are using error-feedback, *any* arbitrary choice of $\boldsymbol{q}_m^t$ will yield convergence). The worker locally accumulates the update errors in the variable $\boldsymbol{e}_m^t$, accumulating the differences between the locally computed gradients and the applied updates. The memory is cleared after each upload. Our novel significance filter on line 8 depends on a parameter $\beta \geq 0$ that impacts how often the updates are triggered: for $\beta = 0$ in every iteration, but potentially less often for larger β (we will use the choice $\beta = 40$ in experiments, see also Appendix C for more discussions). Algorithm 1 presents the pseudocode of LENA. Notice that to make the upload criteria smoother, one may replace $\boldsymbol{g}_m^{t-1}$ by a running average of the gradients over a window, like (6). This change would not affect the theoretical results of our paper. We have investigated this case in the proofs.

Algorithm 1 LENA: distributed Learning with sElf-triggered grdieNt uploAds

```
1: Inputs: Number of iterations $T$, step size $\alpha > 0$, parameter $0 < \beta \leq 1$.
2: Initialize: Server: parameter $\boldsymbol{w}^0$; Nodes: memory $\{\boldsymbol{e}_m^0\}_m$, drift $\{\boldsymbol{q}_m^{-1}\}_m$, gradient estimate $\{\boldsymbol{v}_m^0\}_m$.
3: for $t = 0, 1, \ldots, T-1$ do
4:   Server broadcasts $\boldsymbol{w}^t$ to all nodes
5:   parallel for all node $m \in [M]$ do
6:     Randomly sample a local minibatch
7:     Compute stochastic gradient $\boldsymbol{g}_m^t$
8:     if $\|\boldsymbol{e}_m^t + \alpha\boldsymbol{g}_m^t - \boldsymbol{v}_m^{t-1}\|^2 \geq \beta\|\boldsymbol{g}_m^t\|^2$ then    ▷ upload trigger
9:       Set $\boldsymbol{v}_m^t = \boldsymbol{e}_m^t + \alpha\boldsymbol{g}_m^t$ and compute[a] $\boldsymbol{q}_m^t$
10:      Upload $(\boldsymbol{v}_m^t, \boldsymbol{q}_m^t)$ to the server
11:    else
12:      Set $\boldsymbol{q}_m^t = \boldsymbol{q}_m^{t-1}$, $\boldsymbol{v}_m^t = \boldsymbol{q}_m^t$,    ▷ no uploads
13:    end if
14:    Update local error $\boldsymbol{e}_m^{t+1} = \boldsymbol{e}_m^t + \alpha\boldsymbol{g}_m^t - \boldsymbol{v}_m^t$
15:  end parallel for
16:  Server updates model
         $$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \frac{1}{M}\sum_{m\in[M]} \boldsymbol{v}_m^t$$
17: end for
18: Return: $\boldsymbol{w}^T$
```

[a]Depending on the update scheme; for instance momentum averaging (6) and uniform averaging can be efficiently implemented by maintaining a running average locally on each node.

Choosing the drift parameter $\boldsymbol{q}_m^t$. Our framework supports arbitrary choices of $\boldsymbol{q}_m^t$ in the sense that they do not affect the convergence rates (convergence is ensured by our significance filter), but good choices of $\boldsymbol{q}_m^t$ can minimize the upload frequency and communication cost. We now give some intuition on good choices. Suppose an upload has been triggered at iteration t_m^i and consider the memory $\boldsymbol{e}_m^{t+1}$, at time

$t \geq t_m^i$, supposing no uploads have happened in between. Then

$$\begin{aligned} \boldsymbol{e}_m^{t+1} &= \sum_{i=t_m^i+1}^{t} \left(\alpha \boldsymbol{g}_m^i - \boldsymbol{q}_m^t\right) \\ &= \sum_{i=t_m^i+1}^{t} \left(\alpha \nabla f_m(\boldsymbol{w}^i) + \alpha \boldsymbol{\xi}_m^i - \boldsymbol{q}_m^t\right) . \end{aligned} \tag{5}$$

This equation implies that $\boldsymbol{q}_m^t$ should approximate the local gradient steps (i.e. the local *drift*). For instance in the noise-free case, we can choose $\boldsymbol{q}_m^t = \alpha \nabla f(\boldsymbol{w}^0)$ (similar as in LAG (Chen et al., 2018)), however simply choosing $\boldsymbol{q}_m^t = \alpha \boldsymbol{g}_m^0 = \alpha \nabla f(\boldsymbol{w}^0) + \alpha \boldsymbol{\xi}_m^0$ in the presence of noise might not be a good choice, as the noise will be too strong, $\mathbb{E}\,\|\boldsymbol{\xi}_m^0\|^2 = t\mathbb{E}\,\|\frac{1}{t}\sum_{i=0}^t \boldsymbol{\xi}_m^i\|^2$. Therefore, we should use a variance reduced estimator. Opposed to LASG (Chen et al., 2020) that proposes a full pass over the local data, we here propose smoothing with momentum averaging for tune-able parameter γ. Set $\boldsymbol{q}_m^t = \boldsymbol{m}_m^t$ in line 12, for local momentum on each node:

$$\boldsymbol{m}_m^{t+1} := (1-\gamma)\boldsymbol{m}_m^t + \gamma\alpha \boldsymbol{g}_m^t \,. \tag{6}$$

Alternatively, the workers could also send the uniform average of their computed gradient between subsequent uploads at iteration t_m^i and t_m^{i+1}. By observing

$$\tfrac{1}{t_m^{i+1}-t_m^i}\sum_{t=t_m^i+1}^{t_m^{i+1}} \alpha \boldsymbol{g}_m^t = \boldsymbol{v}_m^{t_m^{i+1}} - (t_m^{i+1} - t_m^i - 1)\boldsymbol{q}_m^{t_m^i} \,. \tag{7}$$

This average can be computed on the server side from the updates $\boldsymbol{v}_m^t$ alone. This saves half of the bandwidth per upload and gives a good performance in a high-noise regime. An alternative half-bandwidth option is $\boldsymbol{q}_m^t = \boldsymbol{v}_m^t$, which performed quite well (but not best) in our experiments.

Adding Quantization and Further Extensions. If communication was triggered in iteration t, the memory will be cleared, $\boldsymbol{e}_m^{t+1} = \boldsymbol{0}$. For our proof this is not a requirement, and any update that ensures contraction $\|\boldsymbol{e}_m^{t+1}\|^2 \leq (1-\beta)\|\boldsymbol{e}_m^t + \alpha \boldsymbol{g}_m^t\|^2$ is supported by our proof. This enables extensions, such as incorporating the complementary technique of quantization on top of our scheme to further save in bandwidth.

4 Main Assumptions and Results

In this section, we prove the convergence of Algorithm 1 for *any*, arbitrary, choice of drift parameter $\boldsymbol{q}_m^t$. We use the following assumptions:

Assumption 1 (Smoothness). *The loss functions $f_m\colon \mathbb{R}^d \to \mathbb{R}$ are L_m-smooth for all $m \in [M]$. Namely, there exists constants $L_m > 0$ such that $\forall m \in [M]$:*

$$\|\nabla f_m(\boldsymbol{w}) - \nabla f_m(\boldsymbol{v})\| \leq L_m \|\boldsymbol{w} - \boldsymbol{v}\| \,, \forall \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^d \,.$$

It follows that $F\colon \mathbb{R}^d \to \mathbb{R}$ is $\overline{L}$-smooth with $\overline{L} = \sum_{m\in[M]} L_m/M$.

For some results, we need the following assumption, sometimes called single point strong convexity. It is weaker than the standard convexity ($\mu = 0$) or strong-convexity ($\mu > 0$) notions, but stronger than the Polyak-Łojasiewicz (PL) condition ($2\mu(F(\boldsymbol{w}) - F^\star) \leq \|\nabla F(\boldsymbol{w})\|^2$, $\forall \boldsymbol{w} \in \mathbb{R}^d$).

Assumption 2 (Quasi-Strong convexity). *$F\colon \mathbb{R}^d \to \mathbb{R}$ is differentiable and μ-quasi convex with constant $\mu \geq 0$ with respect to a $\boldsymbol{w}^\star$, i.e.,*

$$F(\boldsymbol{w}) - F^\star \leq \langle \nabla F(\boldsymbol{w}), \boldsymbol{w} - \boldsymbol{w}^\star\rangle - \frac{\mu}{2}\|\boldsymbol{w} - \boldsymbol{w}^\star\|^2, \forall \boldsymbol{w} \in \mathbb{R}^d.$$

We assume uniformly bounded stochastic noise (Nemirovski and Yudin, 1983). Recent works proposed various relaxations of this assumption, e.g., allowing the noise to grow with the distance to the optimum (Bottou et al., 2018) or measuring the noise only locally, combined with stronger smoothness assumptions (Nguyen et al., 2019). Our results can be extended to these settings as well.

Assumption 3 (Bounded gradient noise). *For gradient oracle of the form $\boldsymbol{g}_m^t = \nabla f_m(\boldsymbol{w}^t) + \boldsymbol{\xi}_m^t$ and conditionally independent noise $\boldsymbol{\xi}_m^t$, we have for some constant $\sigma^2 \geq 0$ and all $t \geq 0, m \in [M]$:*

$$\mathbb{E}\left[\boldsymbol{\xi}_m^t \mid \boldsymbol{w}^t\right] = \boldsymbol{0}, \qquad \mathbb{E}\left[\left\|\boldsymbol{\xi}_m^t\right\|^2 \mid \boldsymbol{w}^t\right] = \sigma^2 \,. \tag{8}$$

We now state our main convergence theorem. We give a proof in Appendix B.

Proposition 1. *Let sequence $(\boldsymbol{w}_t)_{t\geq 0}$ follow the iterates of Algorithm 1, assume inequality $\|\nabla f_m(\boldsymbol{w})\|^2 \leq G^2$ hold for every m and $\boldsymbol{w}$ and some positive G, and set $R_0^2 := \|\boldsymbol{w}^0 - \boldsymbol{w}^\star\|^2$, $F_0 := f(\boldsymbol{w}^0) - f^\star$, and $C := \beta\overline{L}^2\left(G^2 + \sigma^2\right)/2$. Let Assumptions 1 and 3 hold.*

C1: If F satisfies Assumption 2 for $\mu > 0$, then there exists a step-size $\alpha \leq 1/4\overline{L}$ such that

$$\frac{1}{W_T}\sum_{t=0}^T w_t \mathbb{E}\, F(\boldsymbol{w}^t) - F^\star = \tilde{\mathcal{O}}\left(\frac{\sigma^2}{\mu M(T+1)} + \frac{C^2}{\mu^2(T+1)^2} + \overline{L}R_0^2 \exp\left[-\frac{\mu(T+1)}{4\overline{L}}\right]\right)$$

for $w_t = (1-\mu\alpha/2)^{-(t+1)}$ and $W_T = \sum_{t=0}^T w_t$. Here $\tilde{\mathcal{O}}(\cdot)$ hides logarithmic factors.

C2: If F satisfies Assumption 2 for $\mu = 0$, then there exists a step-size $\alpha \leq 1/4\overline{L}$ such that

$$\frac{1}{T+1}\sum_{t=0}^{T} \mathbb{E}\, F(\boldsymbol{w}^t) - F^\star \leq \frac{4\sigma R_0}{\sqrt{M(T+1)}} + \frac{4(CR_0^2)^{2/3}}{(T+1)^{2/3}} + \frac{8\overline{L}R_0^2}{T+1}\,,$$

C3: If F is an arbitrary smooth function, then there exists a step-size $\alpha \leq 1/2\overline{L}$ such that

$$\frac{1}{T+1}\sum_{t=0}^{T} \mathbb{E}\, \left\|\nabla F(\boldsymbol{w}^t)\right\|^2 \leq \frac{8\sigma\sqrt{\overline{L}F_0}}{\sqrt{M(T+1)}} + \frac{8(CF_0)^{2/3}}{(T+1)^{2/3}} + \frac{8\overline{L}F_0}{T+1}\,,$$

Our proof is based on the error-feedback framework (Mania et al., 2017; Stich and Karimireddy, 2019). We improve and extend over prior work which considered only the strongly convex setting (Cordonnier, 2018; Beznosikov et al., 2020). For instance, in the rates given in (Beznosikov et al., 2020) the noise is often divided by $1-\beta$, its Theorem 15 implies $\mathcal{O}\big(\frac{\sigma^2}{(1-\beta)\mu MT}\big)$ for some $\beta \in [0,1]$, showing linear speedup in M but a slowdown in β. In contrast, we can here separate the impact of these parameters and have β only impacting higher order terms. [2] This means that β does not affect the asymptotic convergence rates and we are in practice allowed to choose β relatively small (to reduce the triggering rate for the uploads). Moreover, notice that our approach reduces the number of computations similar to mini-batch SGD. Moreover, as we require less communication rounds, our speedup is indeed better than the synchronized mini-batch SGD one, as shown in Section 5.

Upload Intervals. Whilst the convergence result in Proposition 1 applies to general choices of $\boldsymbol{q}_m^t$, we need to focus on specific choices in order to estimate the communication savings. In Appendix B.6 we argue that for quasi-strongly convex functions and averaging based drift estimators the number of iterations between two consecutive uploads scales as

$$\beta \cdot \mathcal{O}\left(\min\left\{\frac{\|\nabla f_m(\boldsymbol{w}^\star)\|}{L_m\|\boldsymbol{w}^t - \boldsymbol{w}^\star\|}, \frac{\beta\left\|\nabla f_m(\boldsymbol{w}^\star)\right\|^2}{\sigma^2}\right\}\right). \tag{9}$$

That is, the frequency depends on function heterogeneity (magnitude of $\|\nabla f_m(\boldsymbol{w}^\star)\|$), linearly on β, on the distance to the optimal solution and the local smoothness. We see that the frequency decreases when approaching the optimum, but in the presence of noise it can never vanish. Whilst we do not give a formal proof, we conjecture that these arguments could be made precise, and we verify these observations in experiments.

Moreover, with non-IID datasets, the gradients of individual nodes would not vanish at $\mathbf{w}^\star$, but converges to a constant value, i.e., $\|\nabla f_m(\mathbf{w}^\star)\| = c_m$ for some positive constant c_m. However, the denominator of the first term would decrease, leading to a higher upload interval and lower upload frequency. Once the first term dominates (which ultimately happens if $\sigma \neq 0$), we converge to a constant upload frequency that depends on β and σ^2.

[2]The assumptions in (Beznosikov et al., 2020) are slightly weaker, but our restriction to G^2 bounded gradients here is not limiting, especially for weakly-convex and nonconvex setting and could be revoked with techniques used in (Stich and Karimireddy, 2019).

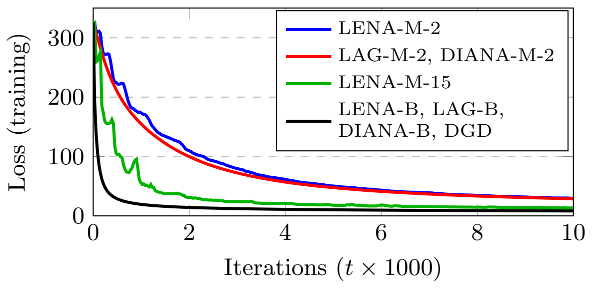
5 Experimental Results

In this section, we numerically characterize the convergence and communication complexity of LENA. We use the MNIST dataset, which has 60 000 training samples of dimension $d = 784$ and 10 classes corresponding to hand-written digits (LeCun et al., 2010). We split the dataset into M disjoint subsets and assume each node m has access to its private dataset. M is reported for every figure. To make the data distributions non-IID, we allocate only one label to one node. On an Nvidia 970GTX GPU, we have trained using the one-versus-all technique and logistic ridge regression objective.

In our experiments, we track the convergence of loss function and communication overhead for the training phase as well as the test accuracy. For benchmarks, we have implemented distributed gradient descent (DGD), LAG (Chen et al., 2018), and DIANA (Mishchenko et al., 2019) with QSGD compressor (Alistarh et al., 2017). Unless mentioned otherwise, we have used LENA with momentum, (6), for the drift parameter, $\beta = 40$, and run a grid search to optimize hyper-parameters of the algorithm (including step-sizes, momentum values, and quantizer level of QSGD). We list these parameters in the appendix.

5.1 Convergence with Intermittent Uploads

Figure 1 compares the convergence and number of uploaded bits of benchmark algorithms for full batch (labeled by '-B') and minibatch of size 2 and 15 (labeled by '-M-2' and '-M-15') at every node. With full batch, the loss curves in terms of iterations of all the algorithms coincide. With stochastic noise, LENA closely follows the loss of LAG and DIANA with noticeably fewer uploads. To illustrate, we have depicted the up-



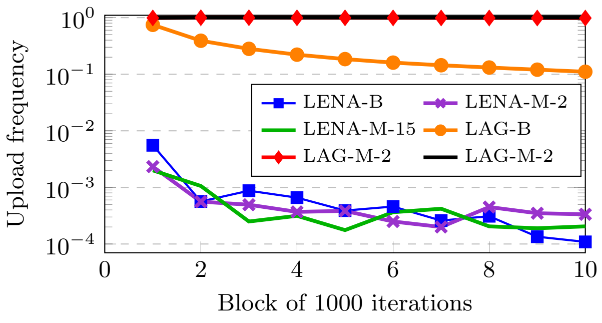
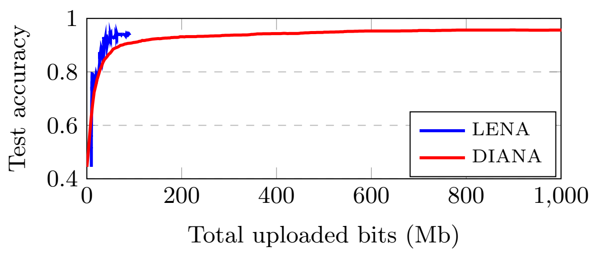
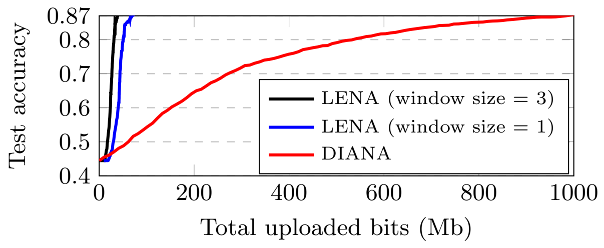


Figure 1: Convergence of benchmark algorithms. '-B' and '-M-x' correspond to full batch and minibatch of size x. DIANA and DGD, omitted from right figure, have an upload frequency of 1.



(a) Full batch, $M = 200$.



(b) Minibatch of size 3, $M = 200$.

Figure 2: Test accuracy vs. bits. DGD and LAG are not depicted due to significantly different scales. In the setting of (a), LAG and DGD upload 11.5 Gb and 46.7 Gb to reach 0.95 test accuracy. In (b), LAG requires 39.75 Gb to reach 0.86 accuracy.

load frequency of LENA and LAG, averaged over all nodes for every 1000 iterations. The upload frequency of LENA-B decreases over time and becomes the smallest, while it reaches constant levels for LENA-M-2/15, with a frequency proportional to the noise (which is lower for 'M-15'), in agreement with (9). The upload frequency of LAG is decreasing over time, though it is significantly higher than that of LENA (around 2 orders of magnitude). Compared to DIANA, every node in LENA sends more bits per upload, however, it enjoys very sparse communication when the iterates evolve.

To better analyze the communication overhead, at every iteration we monitor the classification accuracy of the test data as well as the upload bits so far, depicted in Figure 2. We have dropped LAG and DGD from this figure as they require orders of magnitude more uploads for the same decision accuracy. From these figures, LENA has a higher upload overhead at early iterations, compared to DIANA. However, as the iteration evolves and the test accuracy improves, the infrequent uploads of LENA lead to more bandwidth saving than the gradient compression of DIANA. This gap is more prominent with a small minibatch size where we need more iterations to achieve a high test accuracy. In particular, LENA requires 51.3 Mb to reach an accuracy of 0.86, whereas DIANA needs 879 Mb. Now, if we change the upload rule to 'upload once condition in line 8 of LENA is satisfied for 3 times', we can further eliminate some unnecessary uploads that happen due to gradient noise and obtain the black curve. In that case, only 37 Mb is enough to reach 0.86 test accuracy. It corresponds to 95% saving compared to DIANA, and 99.9% saving compared to LAG. These gains, in general, depend on minibatch size (compare Figure 2(a) and Figure 2(b)) and the choice of the drift function (see Figure 4).

5.2 Latency Performance

Here, we analyze the latency (due to communication) to achieve a certain accuracy. Again, we only focus on the uplink as all our benchmarks behave similarly for the downlink.

Local area networks: When the nodes can form a local area network, realized by either wired or wireless communication links, the server can determine an upload schedule to resolve the so-called channel contention (Bertsekas et al., 2004). Round-robin time division multiple access (TDMA) is the most commonly used scheduling protocol in which the server sequentially ping the nodes to get their vectors (Bertsekas et al., 2004). Let ν_m^t be the number of bits uploaded by node m at iteration t, and r be the channel transmission rate, assumed to be constant for the sake of simplicity. The communication latency is then

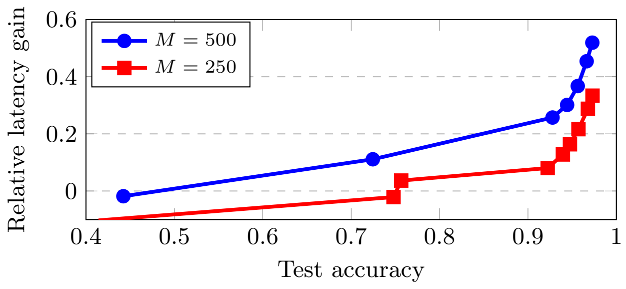


Figure 3: Comparison of LENA and DIANA on a deep learning model. It shows relative latency gain (LENA w.r.t. DIANA) to reach a certain test accuracy.

$\sum_{m\in[M]}\sum_{t\in[T]}\nu_m^t/r$ for the first T iterations. It is a linear function of the total uploaded bits, visualized in Figure 2, and we can expect similar gains as in Figure 2.

Wide area networks: When the nodes are scattered across the globe (like different AWS regions), uploading data could be very time-consuming due to the huge geographical distance and limited bandwidth (Hsieh et al., 2017). To give a perspective, this latency can be as high as 270 ms for a few bytes of data, compared to only 2 μs of uploads inside a cluster (Zhu et al., 2019). When we use public internet as the main communication infrastructure of edge computing, the uplink latency will be further affected by the temporal network congestion. To experiment with the performance of our benchmarks on geo-separated datasets, we assume that the link bandwidth between edge devices and the server is a value in $\{1, 5, 10, 100, 1000\}$ Mbps to consider heterogeneous types of connections. To every node, we assign a randomly drawn number from this set with probability distribution denoted in Figure 4. The total upload latency of node m is linear in its packet size (uploaded vector), $\{\nu_m^t\}_{t\geq 0}$, and inversely proportional to the link bandwidth (Prasad et al., 2003). We repeat the experiment over 100 realizations of the link bandwidths and depict the difference between latency of benchmarks and that of LENA, in Figure 4. For a small network ($M = 10$), compression using DIANA seems better than gradient skipping with LENA. However, as the network gets bigger toward realistic sized edge computing scenarios, LENA outperforms other benchmarks. Compared to LENA, the benchmarks exhibit a higher extra latency as M grows larger. LENA enjoys a similar performance gain when most of the nodes have low communication bandwidth, shown in Figure 4(b). Moreover, LENA is more robust to channel failure and stragglers than the benchmarks since it requires much fewer uploads compared to the benchmarks. Next, we have applied LENA to a deep neural network. The networks includes two convolutional layers, followed by two fully connected layers. We have used ReLU activation function, dropout on the hidden layers, and cross-entropy loss to learn the classes of the MNIST dataset. Figure 3 shows the relative latency gain of LENA over DIANA using the rate and probability models of Figure 4. Further, we have tested LENA on CIFAR10 dataset to train VGG13 model and observed similar performance gain and insights as of 3. These results are reported in Appendix.

Finally, Figures 4(c) and 4(d) show the impact of drift functions on the achievable latency. A properly tuned momentum function can reduce the accumulated error in updating with old uploads and achieve a better latency. However, when the gradient noise becomes so small (large minibatch), LENA-v may be a better option due to sending only one vector, $\boldsymbol{v}_m^t$, per upload instead of two vectors, $\boldsymbol{v}_m^t$ and $\boldsymbol{q}_m^t$, of the momentum function. We have provided more numerical results and discussions in Appendix C.

6 Conclusions

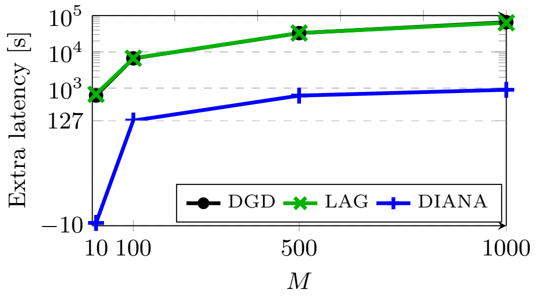
We addressed the problem of communication-efficient distributed optimization over a network. We proposed LENA in which every node locally decides to skip some insignificant uploads. The server then uses the so-called drift term to update the parameter with some silent nodes. We studied the convergence rate of LENA and showed that it obtains the same accuracy as state-of-the-art algorithms with order(s) of magnitude fewer uploads and a significantly smaller latency. These gains are more prominent in larger networks, highlighting the suitability of our approaches on massive edge computing scenarios with constrained resources.

References

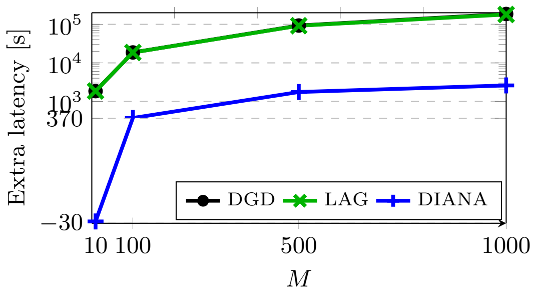
Jayadev Acharya, Chris De Sa, Dylan Foster, and Karthik Sridharan. Distributed learning with sublinear communication. In *Proceedings of the 36th International Conference on Machine Learning*, pages 40–50, 2019.

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

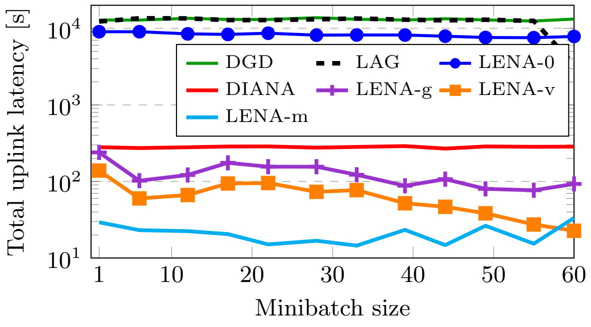
Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pages 5973–5983, 2018.



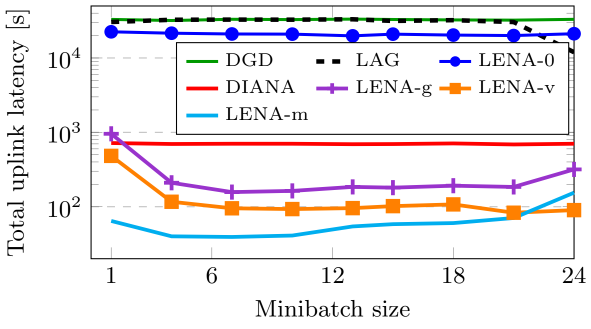
(a) Probability vector: $[0.2, 0.2, 0.2, 0.2, 0.2]$.



(b) Probability vector: $[0.7, 0.2, 0.06, 0.03, 0.01]$.



(c) $M = 200$, $p = [0.2, 0.2, 0.2, 0.2, 0.2]$.



(d) $M = 500$, $p = [0.2, 0.2, 0.2, 0.2, 0.2]$.

Figure 4: Latency performance. In (a) and (b), every node uses 10% of its local dataset as the minibatch size, and the curves show extra latency in comparison to LENA. In (c) and (d), labels 'LENA-0', 'LENA-g', 'LENA-v', and 'LENA-m' correspond to $\boldsymbol{q}_m^t = \boldsymbol{0}$, $\boldsymbol{q}_m^t = \alpha \boldsymbol{g}_m^t$, $\boldsymbol{q}_m^t = \boldsymbol{v}_m^t$, and $\boldsymbol{q}_m^t = \boldsymbol{m}_m^t$ upon every upload at iteration t.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas N. Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, page 14668–14679, 2019.

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

Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. *Data networks, second edition*, volume 2. Prentice-Hall International New Jersey, 2004.

Aleksandr Beznosikov, Samuel Horváth, Peter Richtárik, and Mher Safaryan. On biased compression for distributed learning. *arXiv preprint arXiv:2002.12410*, 2020.

L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Tianyi Chen, Georgios Giannakis, Tao Sun, and Wotao Yin. LAG: Lazily aggregated gradient for communication-efficient distributed learning. In *Advances in Neural Information Processing Systems*, pages 5050–5060, 2018.

Tianyi Chen, Yuejiao Sun, and Wotao Yin. LASG: Lazily aggregated stochastic gradients for communication-efficient distributed learning. *arXiv preprint arXiv:2002.11360*, 2020.

Jean-Baptiste Cordonnier. Convex optimization using sparsified stochastic gradient descent with memory. Master's thesis, EPFL, 2018.

Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R Aberger, Kunle Olukotun, and Christopher Ré. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

Abbas El Gamal and Young-Han Kim. *Network information theory*. Cambridge university press, 2011.

Moshe Gabel, Daniel Keren, and Assaf Schuster. Monitoring least squares models of distributed streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 319–328, 2015.

Hossein S. Ghadikolaei and Sindri Magnusson. Communication-efficient variance-reduced stochastic gradient descent. *IFAC World Congress*, 2020.

Deniz Gündüz, Paul de Kerret, Nicholas D Sidiropoulos, David Gesbert, Chandra R Murthy, and Mihaela van der Schaar. Machine learning in the air. *IEEE Journal on Selected Areas in Communications*, 37 (10):2184–2199, 2019.

W. Heemels, K. H. Johansson, and Paulo Tabuada. An introduction to event triggered and self-triggered control. In *IEEE Conference on Decision and Contro*, pages 3270–3285, 2012.

Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. Gaia: Geo-distributed machine learning approaching LAN speeds. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 629–647, 2017.

Michael I Jordan, Jason D Lee, and Yun Yang. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, 2018.

Michael Kamp, Mario Boley, Michael Mock, Daniel Keren, Assaf Schuster, and Izchak Sharfman. Adaptive communication bounds for distributed online learning. *arXiv preprint arXiv:1911.12896*, 2019.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. *arXiv preprint arXiv:1910.06378*, 2019a.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3252–3261, 2019b.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U Stich. A unified theory of decentralized SGD with changing topology and local updates. *arXiv preprint arXiv:2003.10422*, 2020.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Weiyu Li, Tianyi Chen, Liping Li, and Qing Ling. Communication-censored distributed stochastic gradient descent. *arXiv preprint arXiv:1909.03631*, 2019.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

S. Magnusson, H. S. Ghadikolaei, and N. Li. On maintaining linear convergence of distributed learning and optimization under limited communication. *IEEE Transactions on Signal Processing*, 68:6101–6116, Oct 2020.

Sindri Magnússon, Chinwendu Enyioha, Na Li, Carlo Fischione, and Vahid Tarokh. Convergence of limited communications gradient methods. *IEEE Transactions on Automatic Control*, 2017.

Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization*, 27(4):2202–2229, 2017.

Neil McGlohon and Stacy Patterson. Distributed semi-stochastic optimization with quantization refinement. In *2016 American Control Conference (ACC)*, pages 7159–7164. IEEE, 2016.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 1273–1282, 2017.

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

Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* John Wiley & Sons, Inc., New York, NY, USA, 1983.

Lam M. Nguyen, Phuong Ha Nguyen, Peter Richtárik, Katya Scheinberg, Martin Takáč, and Marten van Dijk. New convergence aspects of stochastic gradient algorithms. *Journal of Machine Learning Research*, 20(176):1–49, 2019.

Tran Thi Phuong et al. Distributed sgd with flexible gradient compression. *IEEE Access*, 8:64707–64717, 2020.

Ravi Prasad, Constantinos Dovrolis, Margaret Murray, and KC Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE network*, 17(6):27–35, 2003.

Ye Pu, Melanie N Zeilinger, and Colin N Jones. Quantization design for distributed optimization. *IEEE Transactions on Automatic Control*, 62(5): 2107–2120, 2017.

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

N. Singh, D. Data, J. George, and S. Diggavi. Sparq-sgd: Event-triggered and compressed communication in decentralized optimization. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3449–3456, 2020.

Navjot Singh, Deepesh Data, Jemin George, and Suhas Diggavi. Squarm-sgd: Communication-efficient momentum sgd for decentralized optimization. *arXiv preprint arXiv:2005.07041*, 2020.

Sebastian U. Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations (ICLR)*, 2019a.

Sebastian U Stich. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019b.

Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems 31*, pages 4447–4458. Curran Associates, Inc., 2018.

Jun Sun, Tianyi Chen, Georgios B Giannakis, and Zaiyue Yang. Communication-efficient distributed learning via lazily aggregated quantized gradients. In *Advances in Neural Information Processing Systems*, 2019.

Hanlin Tang, Xiangru Lian, Tong Zhang, and Ji Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1905.05957*, 2019.

Zhenheng Tang, Shaohuai Shi, Xiaowen Chu, Wei Wang, and Bo Li. Communication-efficient distributed deep learning: A comprehensive survey. *arXiv preprint arXiv:2003.06307*, 2020.

Tijs Van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of 1st International Conference on Embedded Networked Sensor Systems*, pages 171–180, 2003.

Jianqiao Wangni, Ke Li, Jianbo Shi, and Jitendra Malik. Trajectory normalized gradients for distributed optimization. *arXiv preprint arXiv:1901.08227*, 2019.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pages 1509–1519, 2017.

X. Xu and U. S. Kamilov. SignProx: One-bit proximal algorithm for nonconvex stochastic optimization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7800–7804, 2019.

Jian Zhang, Christopher De Sa, Ioannis Mitliagkas, and Christopher Ré. Parallel sgd: When does averaging help? *arXiv preprint arXiv:1606.07365*, 2016.

Ligeng Zhu, Yao Lu, Yujun Lin, and Song Han. Distributed training across the world. In *Advances in Neural Information Processing Systems, Workshop on Systems for ML*, 2019.