CADA: Communication-Adaptive Distributed Adam

Tianyi Chen **Ziye Guo** **Yuejiao Sun** **Wotao Yin**

Rensselaer Polytechnic Institute University of California, Los Angeles

Abstract

Stochastic gradient descent (SGD) has taken the stage as the primary workhorse for large-scale machine learning. It is often used with its adaptive variants such as AdaGrad, Adam, and AMSGrad. This paper proposes an adaptive stochastic gradient descent method for distributed machine learning, which can be viewed as the communication-adaptive counterpart of the celebrated Adam method — justifying its name CADA. The key components of CADA are a set of new rules tailored for adaptive stochastic gradients that can be implemented to save communication upload. The new algorithms adaptively reuse the stale Adam gradients, thus saving communication, and still have convergence rates comparable to original Adam. In numerical experiments, CADA achieves impressive empirical performance in terms of total communication round reduction.

1 Introduction

Stochastic gradient descent (SGD) method [40] is prevalent in solving large-scale machine learning problems during the last decades. Although simple to use, the plain-vanilla SGD is often sensitive to the choice of hyper-parameters and sometimes suffer from the slow convergence. Among various efforts to improve SGD, adaptive methods such as AdaGrad [11], Adam [24] and AMSGrad [38] have well-documented empirical performance, especially in training deep neural networks.

To achieve "adaptivity," these algorithms adaptively adjust the *update direction* or tune the *learning rate*, or, the combination of both. While adaptive SGD methods have been mostly studied in the setting where data and computation are both centralized in a single node, their performance in the distributed learning setting is less understood. As this setting brings new challenges to machine learning, can we add an *additional dimension of adaptivity* to Adam in this regime?

In this context, we aim to develop a fully adaptive SGD algorithm tailored for the distributed learning. We consider the setting composed of a central server and a set of M workers in $\mathcal{M} := \{1, \ldots, M\}$, where each worker m has its local data ξ_m from a distribution Ξ_m. Workers may have different data distributions $\{\Xi_m\}$, and they collaboratively solve the following problem

$$\min_{\theta\in\mathbb{R}^p} \ \mathcal{L}(\theta) = \frac{1}{M}\sum_{m\in\mathcal{M}} \mathcal{L}_m(\theta) \tag{1a}$$

$$\text{with} \ \ \mathcal{L}_m(\theta) := \mathbb{E}_{\xi_m}\left[\ell(\theta;\xi_m)\right], m \in \mathcal{M} \tag{1b}$$

where $\theta \in \mathbb{R}^p$ is the sought variable and $\{\mathcal{L}_m, m\in\mathcal{M}\}$ are smooth (but not necessarily convex) functions. We focus on the setting where local data ξ_m at each worker m can not be uploaded to the server, and collaboration is needed through communication between the server and workers. This setting often emerges due to the data privacy concerns, e.g., federated learning [31, 19].

To solve (1), we can in principle apply the single-node version of the adaptive SGD methods such as Adam [24]: At iteration k, the server broadcasts θ^k to *all* the workers; each worker m computes $\nabla\ell(\theta^k;\xi_m^k)$ using a randomly selected sample or a minibatch of samples $\{\xi_m^k\} \sim \Xi_m$, and then uploads it to the server; and once receiving stochastic gradients from all workers, the server can simply use the aggregated stochastic gradient $\bar{\boldsymbol{\nabla}}^k = \frac{1}{M}\sum_{m\in\mathcal{M}} \nabla\ell(\theta^k;\xi_m^k)$ to update the parameter via the plain-vanilla single-node Adam. When $\nabla\ell(\theta^k;\xi_m^k)$ is an unbiased gradient of $\mathcal{L}_m(\theta)$, the convergence of this distributed implementation of Adam follows from the original ones [38, 8]. To implement this, however, *all* the workers have to *upload* the fresh $\{\nabla\ell(\theta^k;\xi_m^k)\}$ at each iteration. This prevents the efficient implementation of Adam in scenarios where the communication uplink and downlink are not symmet-

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

ric, and communication especially upload from workers and the server is costly; e.g., cellular networks [35]. Therefore, *our goal* is to endow an additional dimension of adaptivity to Adam for solving the distributed problem (1). In short, on top of its adaptive learning rate and update direction, we want Adam to be communication-adaptive.

1.1 Related work

To put our work in context, we review prior contributions that we group in two categories.

1.1.1 SGD with adaptive gradients

A variety of SGD variants have been developed recently, including momentum and acceleration [36, 33, 13]. However, these methods are relatively sensitive to the hyper-parameters such as stepsizes, and require significant efforts on finding the optimal parameters.

Adaptive learning rate. One limitation of SGD is that it scales the gradient uniformly in all directions by a pre-determined constant or a sequence of constants (a.k.a. learning rates). This may lead to poor performance when the training data are sparse [11]. To address this issue, adaptive learning rate methods have been developed that scale the gradient in an entry-wise manner by using past gradients, which include AdaGrad [11, 52], AdaDelta [58] and other variants [26]. This simple technique has improved the performance of SGD in some scenarios.

Adaptive SGD. Adaptive SGD methods achieve the best of both worlds, which update the search directions and the learning rates simultaneously using past gradients. Adam [24] and AMSGrad [38] are the representative ones in this category. While these methods are simple-to-use, analyzing their convergence is challenging [38, 48]. Their convergence in the nonconvex setting has been settled only recently [8, 9]. However, most adaptive SGD methods are studied in the single-node setting where data and computation are both centralized. Recently, adaptive SGD has been studied in the shared memory setting [55], where data is still centralized and the communication is not adaptive.

1.1.2 Communication-efficient distributed optimization

Popular communication-efficient distributed learning methods belong to two categories: c1) reduce the number of bits per communication round; and, c2) save the number of communication rounds.

For c1), methods are centered around the ideas of *quantization* and *sparsification*.

Reducing communication bits. Quantization has been successfully applied to distributed machine learning. The 1-bit and multi-bits quantization methods have been developed in [41, 2, 46]. More recently, signSGD with majority vote has been developed in [4]. Other advances of quantized gradient schemes include error compensation [54, 23], variance-reduced quantization [59, 15], and quantization to a ternary vector [53, 39]. Sparsification amounts to transmitting only gradient coordinates with large enough magnitudes exceeding a certain threshold [44, 1]. To avoid losing information of skipping communication, small gradient components will be accumulated and transmitted when they are large enough [29, 42, 3, 51, 17]. Other compression methods also include low-rank approximation [47] and sketching [16]. However, all these methods aim to resolve c1). In some cases, other latencies dominate the bandwidth-dependent transmission latency. This motivates c2).

Reducing communication rounds. One of the most popular techniques in this category is the periodic averaging, e.g., elastic averaging SGD [60], local SGD (a.k.a. FedAvg) [32, 28, 20, 43, 49, 57, 22, 14] or local momentum SGD [56, 50]. In local SGD, workers perform local model updates independently and the models are averaged periodically. Therefore, communication frequency is reduced. However, except [20, 49, 14], most of local SGD methods follow a predetermined communication schedule that is nonadaptive. Some of them are tailored for the *homogeneous* settings, where the data are independent and identically distributed over all workers. To tackle the heterogeneous case, FedProx has been developed in [25] by solving local subproblems, which pays the price of increasing local gradient computation. Communication-efficient optimization has also been studied in the control literature under the name event-triggered control; see e.g., [10, 34, 30]. However, this line of work primarily focuses on the consensus-based control tasks.

The most related line of work to this paper is the lazily aggregated gradient (LAG) approach [6]. In contrast to periodic communication, the communication in LAG is adaptive and tailored for the *heterogeneous* settings. Parameters in LAG are updated at the server, and workers only adaptively upload information that is determined to be informative enough. Unfortunately, while LAG has good performance in the deterministic settings (e.g., with full gradient), as shown in the recent work [7], its performance will be significantly degraded in the stochastic settings. In [7], LAG has been extended to the stochastic setting by incorporating the plain-vanilla SGD update. Considering the popularity of adaptive SGD in practice, this paper generalizes LAG to the regime of running distributed adaptive SGD such as Adam. Very recently, FedAvg with local adaptive SGD update has been proposed in [37], which sets a strong benchmark for communication-efficient learning. When the new

algorithm in [37] achieves the sweet spot between local SGD and adaptive momentum SGD, the proposed algorithm is very different from ours, and the *averaging period* and the selection of *participating workers* are nonadaptive.

1.2 Our approach

We develop a new adaptive SGD algorithm for distributed learning, called **C**ommunication-**A**daptive **D**istributed **A**dam (**CADA**). Akin to the dynamic scaling of every gradient coordinate in Adam, the key motivation of adaptive communication is that during distributed learning, not all communication rounds between the server and workers are equally important. So a natural solution is to use a condition that decides whether the communication is important or not, and then adjust the frequency of communication between a worker and the server. If some workers are not communicating, the server uses their stale information instead of the fresh ones. We will show that this adaptive communication technique can considerably reduce the less informative communication of distributed Adam.

Analogous to the original Adam [24] and its modified version AMSGrad [38], our new CADA approach also uses the exponentially weighted stochastic gradient h^{k+1} as the update direction of θ^{k+1}, and leverages the weighted stochastic gradient magnitude v^{k+1} to inversely scale the update direction h^{k+1}. Different from the direct distributed implementation of Adam that incorporates the fresh (thus unbiased) stochastic gradients $\bar{\boldsymbol{\nabla}}^k = \frac{1}{M}\sum_{m\in\mathcal{M}}\nabla\ell(\theta^k;\xi_m^k)$, CADA exponentially combines the aggregated stale stochastic gradients $\boldsymbol{\nabla}^k = \frac{1}{M}\sum_{m\in\mathcal{M}}\nabla\ell(\hat{\theta}_m^k;\hat{\xi}_m^k)$, where $\nabla\ell(\hat{\theta}_m^k;\hat{\xi}_m^k)$ is either the fresh stochastic gradient $\nabla\ell(\theta^k;\xi_m^k)$, or an old copy when $\hat{\theta}_m^k \neq \theta^k; \hat{\xi}_m^k \neq \xi_m^k$. Informally, with $\alpha_k > 0$ denoting the stepsize at iteration k, CADA has the following update

$$h^{k+1} = \beta_1 h^k + (1-\beta_1)\boldsymbol{\nabla}^k, \text{ with } \boldsymbol{\nabla}^k = \frac{1}{M}\sum_{m\in\mathcal{M}}\nabla\ell(\hat{\theta}_m^k;\hat{\xi}_m^k) \tag{2a}$$

$$v^{k+1} = \beta_2\hat{v}^k + (1-\beta_2)(\boldsymbol{\nabla}^k)^2 \tag{2b}$$

$$\theta^{k+1} = \theta^k - \alpha_k(\epsilon I + \hat{V}^{k+1})^{-\frac{1}{2}}h^{k+1} \tag{2c}$$

where $\beta_1, \beta_2 > 0$ are the momentum weights, $\hat{V}^{k+1} := \text{diag}(\hat{v}^{k+1})$ is a diagonal matrix whose diagonal vector is $\hat{v}^{k+1} := \max\{v^{k+1}, \hat{v}^k\}$, the constant is $\epsilon > 0$, and I is an identity matrix. To reduce the memory requirement of storing all the stale stochastic gradients $\{\nabla\ell(\theta^k;\xi_m^k)\}$, we can obtain $\boldsymbol{\nabla}^k$ by refining the previous aggregated stochastic gradients $\boldsymbol{\nabla}^{k-1}$ stored in the server via

$$\boldsymbol{\nabla}^k = \boldsymbol{\nabla}^{k-1} + \frac{1}{M}\sum_{m\in\mathcal{M}^k}\delta_m^k \tag{3}$$

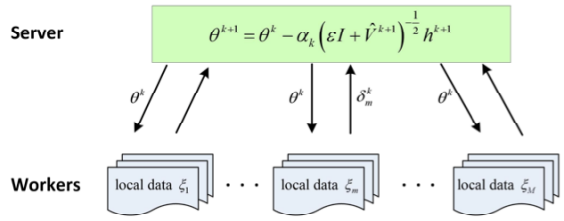

Figure 1: The CADA implementation.

where $\delta_m^k := \nabla\ell(\theta^k;\xi_m^k) - \nabla\ell(\hat{\theta}_m^k;\hat{\xi}_m^k)$ is the stochastic gradient innovation, and $\mathcal{M}^k$ is the set of workers that upload the stochastic gradient to the server at iteration k. Henceforth, $\hat{\theta}_m^k = \theta^k; \hat{\xi}_m^k = \xi_m^k, \forall m \in \mathcal{M}^k$ and $\hat{\theta}_m^k = \hat{\theta}_m^{k-1}; \hat{\xi}_m^k = \hat{\xi}_m^{k-1}, \forall m \notin \mathcal{M}^k$. See CADA's implementation in Figure 1.

Clearly, the selection of subset $\mathcal{M}^k$ is both critical and challenging. It is critical because it adaptively determines the number of communication rounds per iteration $|\mathcal{M}^k|$. However, it is challenging since 1) the staleness introduced in the Adam update will propagate not only through the momentum gradients but also the adaptive learning rate; 2) the importance of each communication round is dynamic, thus a fixed or nonadaptive condition is ineffective; and 3) the condition needs to be checked efficiently without extra overhead. To overcome these challenges, we develop two adaptive conditions to select $\mathcal{M}^k$ in CADA.

With details deferred to Section 2, the contributions of this paper are listed as follows.

c1) We introduce a novel communication-adaptive distributed Adam (CADA) approach that reuses stale stochastic gradients to reduce communication for distributed implementation of Adam.

c2) We develop a new Lyapunov function to establish convergence of CADA under both the nonconvex and Polyak-Łojasiewicz (PL) conditions even when the datasets are non-i.i.d. across workers. The convergence rate matches that of the original Adam.

c3) We confirm that our novel fully-adaptive CADA algorithms achieve at least 60% performance gains in terms of communication upload over some popular alternatives using numerical tests on logistic regression and neural network training.

2 CADA: Communication-Adaptive Distributed Adam

In this section, we revisit the recent LAG method [6] and provide insights why it does not work well in stochastic settings, and then develop our communication-adaptive distributed Adam approach. To be more precise in our notations, we henceforth

use $\tau_m^k \geq 0$ for the *staleness or age of the information* from worker m used by the server at iteration k, e.g., $\hat{\theta}_m^k = \theta^{k-\tau_m^k}$. An age of 0 means "fresh."

2.1 The ineffectiveness of LAG with stochastic gradients

The LAG method [6] modifies the distributed gradient descent update. Instead of communicating with all workers per iteration, LAG selects the subset of workers $\mathcal{M}^k$ to obtain *fresh* full gradients and reuses stale full gradients from others, that is

$$\theta^{k+1} = \theta^k - \frac{\eta_k}{M} \sum_{m \in \mathcal{M} \setminus \mathcal{M}^k} \nabla \mathcal{L}_m(\theta^{k-\tau_m^k}) - \frac{\eta_k}{M} \sum_{m \in \mathcal{M}^k} \nabla \mathcal{L}_m(\theta^k) \tag{4}$$

where $\mathcal{M}^k$ is adaptively decided by comparing the gradient difference $\|\nabla \mathcal{L}_m(\theta^k) - \nabla \mathcal{L}_m(\theta^{k-\tau_m^k})\|$. Following this principle, the direct (or "naive") stochastic version of LAG selects the subset of workers $\mathcal{M}^k$ to obtain *fresh* stochastic gradients $\nabla \mathcal{L}_m\big(\theta^k; \xi_m^k\big)$, $m \in \mathcal{M}^k$. The **stochastic LAG** also follows the distributed SGD update, but it selects $\mathcal{M}^k$ by: if worker m finds the innovation of the fresh stochastic gradient $\nabla \ell(\theta^k; \xi_m^k)$ is small such that it satisfies

$$\left\| \nabla \ell(\theta^k; \xi_m^k) - \nabla \ell(\theta^{k-\tau_m^k}; \xi_m^{k-\tau_m^k}) \right\|^2 \leq \frac{c}{d_{\max}} \sum_{d=1}^{d_{\max}} \left\| \theta^{k+1-d} - \theta^{k-d} \right\|^2 \tag{5}$$

where $c \geq 0$ and $d_{\max}$ are pre-fixed constants, then worker m reuses the old gradient, $m \in \mathcal{M} \setminus \mathcal{M}^k$, and sets the staleness $\tau_m^{k+1} = \tau_m^k + 1$; otherwise, worker m uploads the fresh gradient, and sets $\tau_m^{k+1} = 1$.

If the stochastic gradients were full gradients, LAG condition (5) *compares the error induced by using the stale gradients and the progress of the distributed gradient descent algorithm*, which has proved to be effective in skipping redundant communication [6]. Nevertheless, the observation here is that the two stochastic gradients (5) are evaluated on not just two different iterates (θ^k and $\theta^{k-\tau_m^k}$) but also two different samples (ξ_m^k and $\xi_m^{k-\tau_m^k}$) thus two different loss functions.

This subtle difference leads to the ineffectiveness of (5). We can see this by expanding the left-hand-side (LHS) of (5) by (see details in supplemental material)

$$\mathbb{E}\left[\|\nabla \ell(\theta^k; \xi_m^k) - \nabla \ell(\theta^{k-\tau_m^k}; \xi_m^{k-\tau_m^k})\|^2 \right] \tag{6a}$$

$$\geq \frac{1}{2} \mathbb{E}\Big[\big\| \nabla \ell(\theta^k; \xi_m^k) - \nabla \mathcal{L}_m(\theta^k) \big\|^2 \Big] \tag{6b}$$

$$+ \frac{1}{2} \mathbb{E}\Big[\big[\big\| \nabla \ell(\theta^{k-\tau_m^k}; \xi_m^{k-\tau_m^k}) - \nabla \mathcal{L}_m(\theta^{k-\tau_m^k}) \big\|^2 \big] \Big] \tag{6c}$$

$$- \mathbb{E}[\|\nabla \mathcal{L}_m(\theta^k) - \nabla \mathcal{L}_m(\theta^{k-\tau_m^k})\|^2]. \tag{6d}$$

Even if θ^k converges, e.g., $\theta^k \to \theta^*$, and thus the right-hand-side (RHS) of (5) $\left\|\theta^{k+1-d} - \theta^{k-d}\right\|^2 \to 0$, the LHS of (5) does not, because the variance inherited in (6b) and (6c) does not vanish yet the gradient difference at the same function (6d) diminishes. Therefore, the key insight here is that the non-diminishing variance of stochastic gradients makes the LAG rule (5) ineffective eventually. This will also be verified in our simulations when we compare CADA with stochastic LAG.

2.2 Algorithm development of CADA

In this section, we formally develop our CADA method, and present the intuition behind its design.

To overcome the limitations of LAG in stochastic settings, the key of the CADA design is to *reduce the variance of the innovation measure* in the adaptive condition. We introduce two CADA variants, both of which follow the update (2), but they differ in the variance-reduced communication rules.

The first one termed **CADA1** will calculate two stochastic gradient innovations with one $\tilde{\delta}_m^k := \nabla \ell(\theta^k; \xi_m^k) - \nabla \ell(\tilde{\theta}; \xi_m^k)$ at the sample ξ_m^k, and one $\tilde{\delta}_m^{k-\tau_m^k} := \nabla \ell(\theta^{k-\tau_m^k}; \xi_m^{k-\tau_m^k}) - \nabla \ell(\tilde{\theta}; \xi_m^{k-\tau_m^k})$ at the sample $\xi_m^{k-\tau_m^k}$, where $\tilde{\theta}$ is a snapshot of the previous iterate θ that will be updated every D iterations. As we will show in (8), $\tilde{\delta}_m^k - \tilde{\delta}_m^{k-\tau_m^k}$ can be viewed as the difference of two variance-reduced gradients calculated at θ^k and $\theta^{k-\tau_m^k}$. Using $\tilde{\delta}_m^k - \tilde{\delta}_m^{k-\tau_m^k}$ as the error induced by using stale information, CADA1 will exclude worker m from $\mathcal{M}^k$ if worker m finds

$$\left\| \tilde{\delta}_m^k - \tilde{\delta}_m^{k-\tau_m^k} \right\|^2 \leq \frac{c}{d_{\max}} \sum_{d=1}^{d_{\max}} \left\| \theta^{k+1-d} - \theta^{k-d} \right\|^2. \tag{7}$$

In (7), we use the change of parameter θ^k averaged over the past $d_{\max}$ consecutive iterations to measure the progress of algorithm. Intuitively, if (7) is satisfied, the error induced by using stale information will not large affect the learning algorithm. In this case, worker m does not upload, and the staleness of information from worker m increases by $\tau_m^{k+1} = \tau_m^k + 1$; otherwise, worker m belongs to $\mathcal{M}^k$, uploads the stochastic gradient innovation δ_m^k, and resets $\tau_m^{k+1} = 1$.

The rationale of CADA1. In contrast to the non-vanishing variance in LAG rule (see (6)), the CADA1 rule (7) reduces its inherent variance. To see this, we can decompose the LHS of (7) as the difference of two *variance reduced* stochastic gradients at iteration k and $k - \tau_m^k$. Using the stochastic gradient in SVRG

Algorithm 1 Pseudo-code of CADA; red lines are run only by **CADA1**; blue lines are implemented only by **CADA2**; not both at the same time.

Input: delay counter $\{\tau_m^0\}$, stepsize α_k, constant threshold c, max delay D.
for $k = 0, 1, \ldots, K-1$ **do**
  Server broadcasts θ^k to all workers.
  All workers set $\tilde{\theta} = \theta^k$ if $k \bmod D = 0$.
  for Worker $m = 1, 2, \ldots, M$ **do in parallel**
    Compute $\nabla\ell(\theta^k; \xi_m^k)$ and $\nabla\ell(\tilde{\theta}; \xi_m^k)$.
    Check condition (7) with stored $\tilde{\delta}_m^{k-\tau_m^k}$.
    Compute $\nabla\ell(\theta^k; \xi_m^k)$ and $\nabla\ell(\theta_m^{k-\tau_m^k}; \xi_m^k)$.
    Check condition (10).
    if (7) or (10) is violated or $\tau_m^k \geq D$ **then**
      Upload δ_m^k. ▷ $\tau_m^{k+1} = 1$
    else
      Upload nothing. ▷ $\tau_m^{k+1} = \tau_m^k + 1$
    end if
  end for
  Server updates $\{h^k, v^k\}$ via (2a)-(2b).
  Server updates θ^k via (2c).
end for

as an example [18], the innovation can be written as

$$\begin{aligned}&\tilde{\delta}_m^k - \tilde{\delta}_m^{k-\tau_m^k} \\ &= \big(\nabla\ell(\theta^k; \xi_m^k) - \nabla\ell(\tilde{\theta}; \xi_m^k) + \nabla\mathcal{L}_m(\tilde{\theta})\big) \\ &- \Big(\nabla\ell(\theta^{k-\tau_m^k}; \xi_m^{k-\tau_m^k}) - \nabla\ell(\tilde{\theta}; \xi_m^{k-\tau_m^k}) + \nabla\mathcal{L}_m(\tilde{\theta})\Big).\end{aligned} \tag{8}$$

Define the minimizer of (1) as $\theta^\star$. With derivations given in the supplementary document, the expectation of the LHS of (7) can be *upper-bounded* by

$$\begin{aligned}\mathbb{E}\Big[\big\|\tilde{\delta}_m^k - \tilde{\delta}_m^{k-\tau_m^k}\big\|^2\Big] = \mathcal{O}\Big(&\mathbb{E}[\mathcal{L}(\theta^k)] - \mathcal{L}(\theta^\star) \\ &+ \mathbb{E}[\mathcal{L}(\theta^{k-\tau_m^k})] - \mathcal{L}(\theta^\star) + \mathbb{E}[\mathcal{L}(\tilde{\theta})] - \mathcal{L}(\theta^\star)\Big).\end{aligned} \tag{9}$$

If θ^k converges, e.g., $\theta^k, \theta^{k-\tau_m^k}, \tilde{\theta} \to \theta^*$, the RHS of (9) diminishes, and thus the LHS of (7) diminishes. This is in contrast to the LAG rule (6) *lower-bounded* by a non-vanishing value. Notice that while enjoying the benefit of variance reduction, our communication rule does not need to repeatedly calculate the full gradient $\nabla\mathcal{L}_m(\tilde{\theta})$, which is only used for illustration purpose.

In addition to (7), the second rule is termed **CADA2**. The key difference relative to CADA1 is that CADA2 uses $\nabla\ell(\theta^k; \xi_m^k) - \nabla\ell(\theta_m^{k-\tau_m^k}; \xi_m^k)$ to estimate the error of using stale information. CADA2 will reuse the stale stochastic gradient $\nabla\ell(\theta_m^{k-\tau_m^k}; \xi_m^{k-\tau_m^k})$ or exclude worker m from $\mathcal{M}^k$ if worker m finds

$$\begin{aligned}&\big\|\nabla\ell(\theta^k; \xi_m^k) - \nabla\ell(\theta_m^{k-\tau_m^k}; \xi_m^k)\big\|^2 \\ &\qquad \leq \frac{c}{d_{\max}} \sum_{d=1}^{d_{\max}} \big\|\theta^{k+1-d} - \theta^{k-d}\big\|^2.\end{aligned} \tag{10}$$

If (10) is satisfied, then worker m does not upload, and the staleness increases by $\tau_m^{k+1} = \tau_m^k + 1$; otherwise, worker m uploads the stochastic gradient innovation δ_m^k, and resets the staleness as $\tau_m^{k+1} = 1$. Notice that different from the naive LAG (5), the CADA condition (10) is evaluated at two different iterates but on the same sample ξ_m^k.

The rationale of CADA2. Similar to CADA1, the CADA2 rule (10) also reduces its inherent variance, since the LHS of (10) can be written as the difference between a *variance reduced* stochastic gradient and a *deterministic* gradient, that is

$$\begin{aligned}&\nabla\ell(\theta^k; \xi_m^k) - \nabla\ell(\theta^{k-\tau_m^k}; \xi_m^k) \\ &= \Big(\nabla\ell(\theta^k; \xi_m^k) - \nabla\ell(\theta^{k-\tau_m^k}; \xi_m^k) + \nabla\mathcal{L}_m(\theta^{k-\tau_m^k})\Big) \\ &\quad - \nabla\mathcal{L}_m(\theta^{k-\tau_m^k}).\end{aligned} \tag{11}$$

With derivations deferred to the supplementary document, similar to (9) we also have that $\mathbb{E}[\|\nabla\ell(\theta^k; \xi_m^k) - \nabla\ell(\theta^{k-\tau_m^k}; \xi_m^k)\|^2] \to 0$ as the iterate $\theta^k \to \theta^\star$.

For either (7) or (10), worker m can check it locally with small memory cost by recursively updating the RHS of (7) or (10). In addition, worker m will update the stochastic gradient if the staleness satisfies $\tau_m^k \geq D$. We summarize CADA in Algorithm 1.

Computational, memory and download cost. In CADA, checking (7) and (10) will double the computational cost (gradient evaluation) per iteration. Aware of this fact, we have compared the number of iterations and gradient evaluations in simulations (see Figures 2-5), which will demonstrate that CADA requires *fewer* iterations and also *fewer* gradient queries to achieve a target accuracy. Thus the extra computation is small. In addition, the extra memory for large $d_{\max}$ is low. To compute the RHS of (7) or (10), each worker only stores the norm of model changes ($d_{\max}$ **scalars**). Also note that the current CADA1 and CADA2 only save communication upload during distributed learning, but they can be extended to save both upload and download by adapting the server's rules of LAG [6].

3 Convergence Analysis of CADA

We present the convergence results of CADA. For all the results, we make some basic assumptions.

Assumption 1. *The loss function $\mathcal{L}(\theta)$ is smooth with the constant L.*

Assumption 2. *Samples $\xi_m^1, \xi_m^2, \ldots$ are independent, and the stochastic gradient $\nabla \ell(\theta; \xi_m^k)$ satisfies $\mathbb{E}_{\xi_m^k}[\nabla \ell(\theta; \xi_m^k)] = \nabla \mathcal{L}_m(\theta)$ and $\|\nabla \ell(\theta; \xi_m^k)\| \le \sigma_m$.*

Note that Assumptions 1-2 are standard in analyzing Adam and its variants [24, 38, 8, 55].

3.1 Key steps of Lyapunov analysis

The convergence results of CADA critically builds on the subsequent Lyapunov analysis. We will start with analyzing the expected descent in terms of $\mathcal{L}(\theta^k)$ by applying one step CADA update.

Lemma 1. *Under Assumptions 1 and 2, if $\alpha_{k+1} \le \alpha_k$, then $\{\theta^k\}$ generated by CADA satisfy*

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(\theta^{k+1})] - \mathbb{E}[\mathcal{L}(\theta^k)] \\
&\le -\alpha_k(1-\beta_1)\mathbb{E}\left[\left\langle \nabla \mathcal{L}(\theta^k), (\epsilon I + \hat{V}^{k-D})^{-\frac{1}{2}} \boldsymbol{\nabla}^k \right\rangle\right] \\
&- \alpha_k \beta_1 \mathbb{E}\left[\left\langle \nabla \mathcal{L}(\theta^{k-1}), (\epsilon I + \hat{V}^{k})^{-\frac{1}{2}} h^k \right\rangle\right] \\
&+ \left(\frac{L}{2} + \beta_1 L\right) \mathbb{E}\left[\|\theta^{k+1} - \theta^k\|^2\right] \\
&+ \alpha_k(2-\beta_1)\sigma^2 \mathbb{E}\Big[\sum_{i=1}^{p}\Big((\epsilon + \hat{v}_i^{k-D})^{-\frac{1}{2}} - (\epsilon + \hat{v}_i^{k+1})^{-\frac{1}{2}}\Big)\Big]
\end{aligned} \tag{12}$$

where p is the dimension of θ, σ is defined as $\sigma := \frac{1}{M}\sum_{m\in\mathcal{M}} \sigma_m$, and β_1, ϵ are parameters in (2).

Lemma 1 contains four terms in the RHS of (12): the first two terms quantify the correlations between the gradient direction $\nabla \mathcal{L}(\theta^k)$ and the *stale* stochastic gradient $\boldsymbol{\nabla}^k$ as well as the *state momentum* stochastic gradient h^k; the third term captures the drift of two consecutive iterates; and, the last term estimates the maximum drift of the adaptive stepsizes over $D+1$ iterations.

From Lemma 1, analyzing the progress of $\mathcal{L}(\theta^k)$ under CADA is challenging especially when the effects of staleness and the momentum couple with each other. Because the the state momentum gradient h^k is recursively updated by $\boldsymbol{\nabla}^k$, we will first need the following lemma to characterize the regularity of the stale aggregated stochastic gradients $\boldsymbol{\nabla}^k$, which lays the theoretical foundation for incorporating the properly controlled staleness into the Adam's momentum update.

Lemma 2. *Under Assumptions 1 and 2, if the stepsizes satisfy $\alpha_{k+1} \le \alpha_k \le 1/L$, then we have*

$$\begin{aligned}
&-\alpha_k \mathbb{E}\left[\left\langle \nabla \mathcal{L}(\theta^k), (\epsilon I + \hat{V}^{k-D})^{-\frac{1}{2}} \boldsymbol{\nabla}^k \right\rangle\right] \\
&\le -\frac{\alpha_k}{2}\mathbb{E}\left[\left\|\nabla \mathcal{L}(\theta^k)\right\|^2_{(\epsilon I + \hat{V}^{k-D})^{-\frac{1}{2}}}\right] + \frac{6DL\alpha_k^2\epsilon^{-\frac{1}{2}}}{M}\sum_{m\in\mathcal{M}}\sigma_m^2 \\
&+ \epsilon^{-\frac{1}{2}}\left(\frac{L}{12} + \frac{c}{2Ld_{\max}}\right)\sum_{d=1}^{D}\mathbb{E}\left[\|\theta^{k+1-d} - \theta^{k-d}\|^2\right].
\end{aligned} \tag{13}$$

Lemma 2 justifies the relevance of the stale yet properly selected stochastic gradients. Intuitively, the first term in the RHS of (13) resembles the descent of using SGD with the unbiased stochastic gradient, and the second and third terms will diminish if the stepsizes are diminishing since $\mathbb{E}\left[\|\theta^k - \theta^{k-1}\|^2\right] = \mathcal{O}(\alpha_k^2)$. This is achieved by our designed communication rules.

In view of Lemmas 1 and 2, we introduce the following **Lyapunov function**:

$$\begin{aligned}
\mathcal{V}^k &:= \mathcal{L}(\theta^k) - \mathcal{L}(\theta^\star) \\
&- \sum_{j=k}^{\infty} \alpha_j \beta_1^{j-k+1}\left\langle \nabla \mathcal{L}(\theta^{k-1}), (\epsilon I + \hat{V}^k)^{-\frac{1}{2}} h^k \right\rangle \\
&+ b_k \sum_{d=0}^{D}\sum_{i=1}^{p}(\epsilon + \hat{v}_i^{k-d})^{-\frac{1}{2}} + \sum_{d=1}^{D}\rho_d \|\theta^{k+1-d} - \theta^{k-d}\|^2
\end{aligned} \tag{14}$$

where $\theta^\star$ is the solution of (1), $\{b_k\}_{k=1}^K$ and $\{\rho_d\}_{d=1}^D$ are constants that will be specified in the proof.

The design of Lyapunov function in (14) is motivated by the progress of $\mathcal{L}(\theta^k)$ in Lemmas 1-2, and also coupled with our communication rules (7) and (10) that contain the parameter difference term. We find this new Lyapunov function can lead to a much simple proof of Adam and AMSGrad, which is of independent interest. The following lemma captures the progress of the Lyapunov function.

Lemma 3. *Under Assumptions 1-2, if $\{b_k\}_{k=1}^K$ and $\{\rho_d\}_{d=1}^D$ in (14) are chosen properly, we have*

$$\begin{aligned}
&\mathbb{E}[\mathcal{V}^{k+1}] - \mathbb{E}[\mathcal{V}^k] \\
&\le -\frac{\alpha_k(1-\beta_1)}{2}\left(\epsilon + \frac{\sigma^2}{1-\beta_2}\right)^{-\frac{1}{2}}\mathbb{E}\left[\left\|\nabla \mathcal{L}(\theta^k)\right\|^2\right] + \alpha_k^2 C_0
\end{aligned} \tag{15}$$

where the constant C_0 depends on the CADA and problem parameters $c, \beta_1, \beta_2, \epsilon, D$, and $L, \{\sigma_m^2\}$.

The first term in the RHS of (15) is strictly negative, and the second term is positive but potentially small since it is $\mathcal{O}(\alpha_k^2)$ with $\alpha_k \to 0$. This implies that the

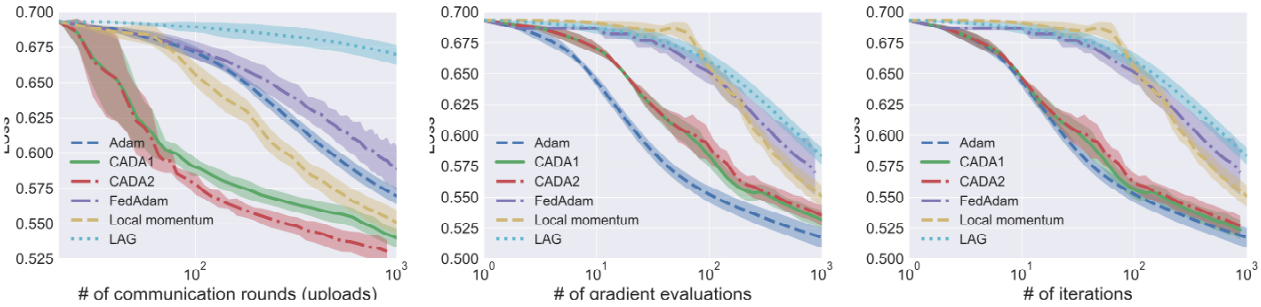

Figure 2: Logistic regression training loss on *covtype* dataset averaged over 10 Monte Carlo runs.

function $\mathcal{V}^k$ will eventually converge if we choose the stepsizes appropriately. Lemma 3 is a generalization of SGD's descent lemma. If we set $\beta_1 = \beta_2 = 0$ in (2) and $b_k = 0, \rho_d = 0, \forall d, k$ in (14), then Lemma 3 reduces to that of SGD in terms of $\mathcal{L}(\theta^k)$; see e.g., [5, Lemma 4.4].

3.2 Main convergence results

Building upon our Lyapunov analysis, we first present the convergence in nonconvex case.

Theorem 1 (nonconvex)**.** *Under Assumptions 1, 2, if we choose* $\alpha_k = \alpha = \mathcal{O}(\frac{1}{\sqrt{K}})$ *and* $\beta_1 < \sqrt{\beta_2} < 1$*, then the iterates* $\{\theta^k\}$ *generated by CADA satisfy*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla\mathcal{L}(\theta^k)\|^2\right] = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right). \tag{16}$$

From Theorem 1, the convergence rate of CADA in terms of the average gradient norms is $\mathcal{O}(1/\sqrt{K})$, which matches that of the plain-vanilla Adam [38, 8]. Unfortunately, due to the complicated nature of Adam-type analysis, the bound in (16) does not achieve the linear speed-up as analyzed for asynchronous nonadaptive SGD such as [27]. However, our analysis is tailored for adaptive SGD and does not make any assumption on the asynchrony, e.g., the set of uploading workers are independent from the past or even independent and identically distributed.

Next we present the convergence results under a slightly stronger assumption on the loss $\mathcal{L}(\theta)$.

Assumption 3. *The loss function* $\mathcal{L}(\theta)$ *satisfies the Polyak-Łojasiewicz (PL) condition with the constant* $\mu > 0$*, that is* $\mathcal{L}(\theta) - \mathcal{L}(\theta^*) \le \frac{1}{2\mu}\|\mathcal{L}(\theta)\|^2$.

The PL condition is weaker than the strongly convexity, which does not even require convexity [21]. And it is satisfied by a wider range of problems such as least squares for underdetermined linear systems, logistic regression, and also certain types of neural networks.

We next establish the convergence of CADA under this condition.

Theorem 2 (PL-condition)**.** *Under Assumptions 1-3, if we choose the stepsize as* $\alpha_k = \frac{2}{\mu(k+K_0)}$ *for a given constant* K_0*, then* θ^K *generated by Algorithm 1 satisfies*

$$\mathbb{E}\left[\mathcal{L}(\theta^K)\right] - \mathcal{L}(\theta^\star) = \mathcal{O}\left(\frac{1}{K}\right). \tag{17}$$

Theorem 2 implies that under the PL-condition of the loss function, the CADA algorithm can achieve the global convergence in terms of the loss function, with a fast rate $\mathcal{O}(1/K)$. Compared with the previous analysis for deterministic gradient-based LAG [6] and quantized LAG [45], as we highlighted in Section 3.1, the analysis for CADA is more involved, since it needs to deal with not only the outdated gradients but also the *stochastic momentum* gradients and the *adaptive* matrix learning rates. We tackle this issue by i) considering a new set of communication rules (7) and (10) with reduced variance; and, ii) incorporating the effect of momentum gradients and the drift of adaptive learning rates in the new Lyapunov function (14).

4 Simulations

In order to verify our analysis and show the empirical performance of CADA, we conduct simulations using logistic regression and training neural networks. Data are distributed across $M = 10$ workers during all tests. We benchmark CADA with some popular methods such as Adam [24], stochastic version of LAG [6], local momentum [56] and the state-of-the-art FedAdam [37]. For local momentum and FedAdam, workers perform model update independently, which are averaged over all workers every H iterations. In simulations, critical parameters are optimized for each algorithm by a grid-search. All experiments are run on a workstation with an Intel i9-9960x CPU with 128GB memory and four NVIDIA RTX 2080Ti GPUs each with 11GB memory

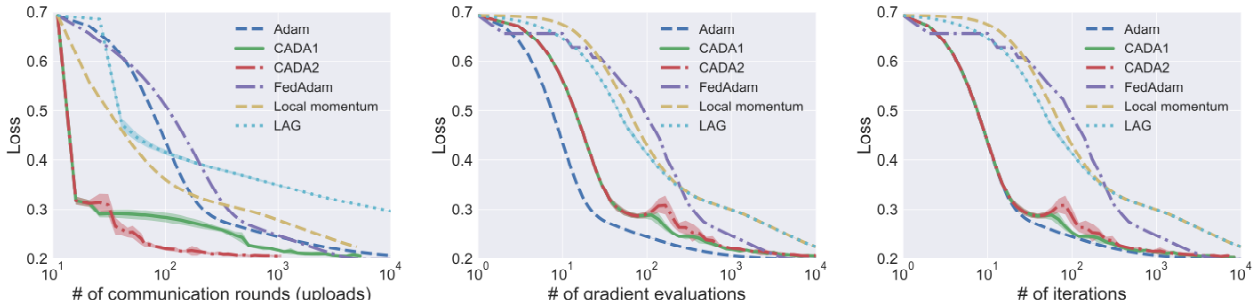

Figure 3: Logistic regression training loss on *ijcnn1* dataset averaged over 10 Monte Carlo runs.

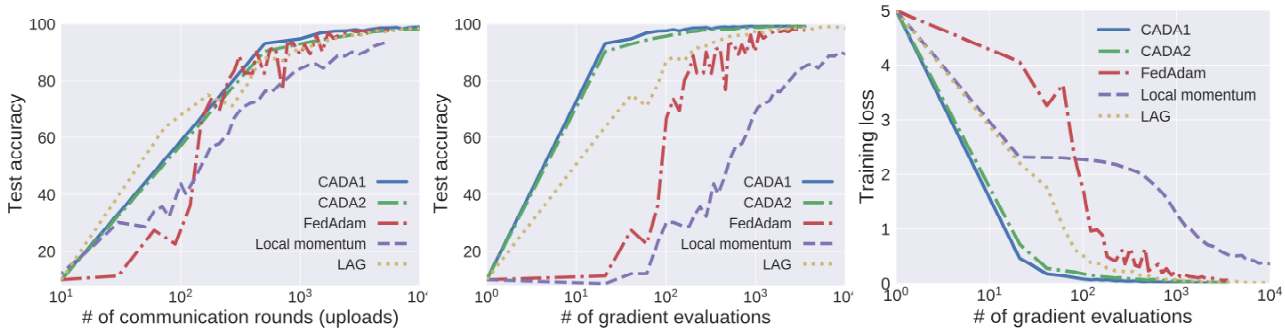

Figure 4: Training Neural network for classification on *mnist* dataset.

using Python 3.6. Due to space limitation, please see the detailed choice of parameters in the *supplementary document*. The code can be found in

https://github.com/ChrisYZZ/CADA-master

4.1 Logistic regression

For CADA, the maximal delay is $D = 100$ and $d_{\max} = 10$. For local momentum and FedAdam, we manually optimize the averaging period as $H = 10$ for *ijcnn1* and $H = 20$ for *covtype*. Results are averaged over 10 Monte Carlo runs.

Tests on logistic regression are reported in Figures 2-3. In our tests, two CADA variants achieve the similar iteration complexity as the original Adam and outperform all other baselines in most cases. Since our CADA requires two gradient evaluations per iteration, the gradient complexity (e.g., computational complexity) of CADA is higher than Adam, but still smaller than that of other baselines. For logistic regression task, CADA1 and CADA2 save the number of communication uploads by at least one order of magnitude.

4.2 Training neural networks

We train a neural network with two convolution-ELU-maxpooling layers followed by two fully-connected layers for 10 classes classification on *mnist*. We use the popular *ResNet20* model on *CIFAR10* dataset, which has 20 and roughly 0.27 million parameters. We searched the best values of H from the grid $\{1, 4, 6, 8, 16\}$ to optimize the testing accuracy vs communication rounds for each algorithm. In CADA, the maximum delay is $D = 50$ and the average interval $d_{\max} = 10$. See tests under different H in the supplementary material.

Tests on training neural networks are reported in Figures 4-5. In *mnist*, CADA1 and CADA2 save the number of communication uploads by roughly 60% than local momentum and slightly more than FedAdam. In *cifar10*, CADA1 and CADA2 achieve competitive performance relative to the state-of-the-art algorithms FedAdam and local momentum. We found that if we further enlarge H, FedAdam and local momentum converge fast at the beginning, but reached worse test accuracy (e.g., 5%-15%). It is also evident that the CADA1 and CADA2 rules achieve more communication reduction than the direct stochastic version of LAG, which verifies the intuition in Section 2.1.

5 Conclusions

While Adaptive SGD methods have been widely applied in the single-node setting, their performance in

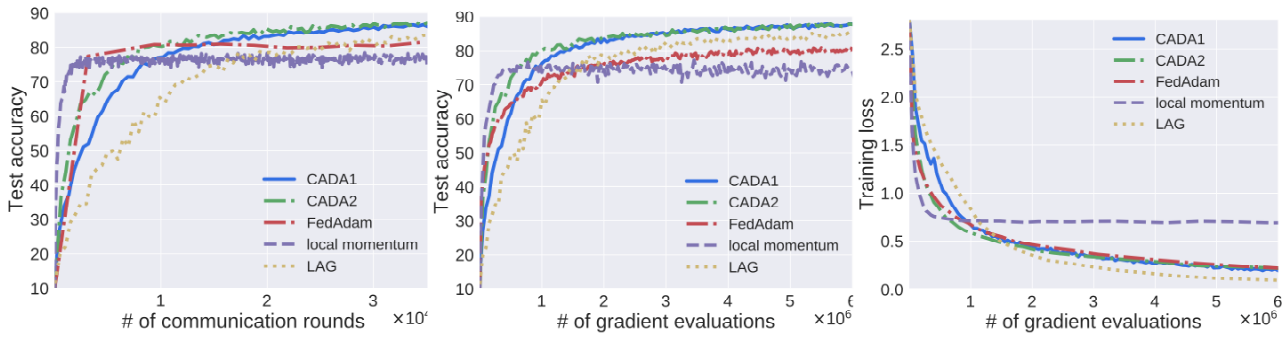

Figure 5: Training Neural network for classification on *cifar10* dataset.

the distributed learning setting is less understood. In this paper, we have developed a communication-adaptive distributed Adam method that we term CADA, which endows an additional dimension of adaptivity to Adam tailored for its distributed implementation. CADA method leverages a set of adaptive communication rules to detect and then skip less informative communication rounds between the server and workers during distributed learning. All CADA variants are simple to implement, and have convergence rate comparable to the original Adam.

Acknowledgment

The work of T. Chen and Z. Guo was partially supported by the RPI-IBM Artificial Intelligence Research Collaboration (AIRC). The work of Y. Sun was partially supported by ONR Grant N000141712162 and AFOSR MURI FA9550-18-1-0502.

References

[1] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proc. Conf. Empirical Methods Natural Language Process.*, pages 440–445, Copenhagen, Denmark, Sep 2017.

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Proc. Conf. Neural Info. Process. Syst.*, pages 1709–1720, Long Beach, CA, Dec 2017.

[3] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Proc. Conf. Neural Info. Process. Syst.*, pages 5973–5983, Montreal, Canada, Dec 2018.

[4] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. SignSGD: Compressed optimisation for non-convex problems. In *Proc. Intl. Conf. Machine Learn.*, pages 559–568, Stockholm, Sweden, Jul 2018.

[5] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[6] Tianyi Chen, Georgios Giannakis, Tao Sun, and Wotao Yin. LAG: Lazily aggregated gradient for communication-efficient distributed learning. In *Proc. Conf. Neural Info. Process. Syst.*, pages 5050–5060, Montreal, Canada, Dec 2018.

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

[8] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *Proc. Intl. Conf. Learn. Representations*, New Orleans, LA, May 2019.

[9] Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. On the convergence of Adam and Adagrad. *arXiv preprint:2003.02395*, March 2020.

[10] Dimos V Dimarogonas, Emilio Frazzoli, and Karl H Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Trans. Automatic Control*, 57(5):1291–1297, November 2011.

[11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Machine Learning Res.*, 12(Jul):2121–2159, 2011.

[12] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[13] Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

[14] Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. In *Proc. Conf. Neural Info. Process. Syst.*, pages 11080–11092, Vancouver, Canada, December 2019.

[15] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint:1904.05115*, April 2019.

[16] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, et al. Communication-efficient distributed SGD with sketching. In *Proc. Conf. Neural Info. Process. Syst.*, pages 13144–13154, Vancouver, Canada, December 2019.

[17] Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Proc. Conf. Neural Info. Process. Syst.*, pages 2525–2536, Montreal, Canada, Dec 2018.

[18] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proc. Conf. Neural Info. Process. Syst.*, pages 315–323, 2013.

[19] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint:1912.04977*, December 2019.

[20] Michael Kamp, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel. Efficient decentralized deep learning by dynamic model averaging. In *Euro. Conf. Machine Learn. Knowledge Disc. Data.,*, pages 393–409, Dublin, Ireland, 2018.

[21] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Proc. Euro. Conf. Machine Learn.*, pages 795–811, Riva del Garda, Italy, September 2016.

[22] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. In *Proc. Intl. Conf. Machine Learn.*, July 2020.

[23] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *Proc. Intl. Conf. Machine Learn.*, pages 3252–3261, Long Beach, CA, June 2019.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*, December 2014.

[25] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[26] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *Proc. Intl. Conf. on Artif. Intell. and Stat.*, pages 983–992, Okinawa, Japan, April 2019.

[27] Xiangru Lian, Huan Zhang, Cho-Jui Hsieh, Yijun Huang, and Ji Liu. A comprehensive linear speedup analysis for asynchronous stochastic parallel optimization from zeroth-order to first-order. In *Proc. Conf. Neural Info. Process. Syst.*, pages 3054–3062, Barcelona, Spain, December 2016.

[28] Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local SGD. In *Proc. Intl. Conf. Learn. Representations*, Addis Ababa, Ethiopia, April 2020.

[29] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *Proc. Intl. Conf. Learn. Representations*, Vancouver, Canada, Apr 2018.

[30] Yaohua Liu, Cameron Nowzari, Zhi Tian, and Qing Ling. Asynchronous periodic event-triggered coordination of multi-agent systems. In *Proc. IEEE Conf. Decision and Control*, pages 6696–6701, Melbourne, Australia, December 2017.

[31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. Intl. Conf. Artificial Intell. and Stat.*, pages 1273–1282, Fort Lauderdale, FL, April 2017.

[32] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. Intl. Conf. on Artif. Intell. and Stat.*, pages 1273–1282, Fort Lauderdale, Florida, Apr 2017.

[33] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Doklady AN USSR*, volume 269, pages 543–547, 1983.

[34] Cameron Nowzari, Eloy Garcia, and Jorge Cortés. Event-triggered communication and control of networked systems for multi-agent consensus. *Automatica*, 105:1–27, July 2019.

[35] Jihong Park, Sumudu Samarakoon, Mehdi Bennis, and Mérouane Debbah. Wireless network intelligence at the edge. *Proc. of the IEEE*, 107(11):2204–2239, November 2019.

[36] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[37] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint:2003.00295*, March 2020.

[38] Sashank Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *Proc. Intl. Conf. Learn. Representations*, Vancouver, Canada, April 2018.

[39] Amirhossein Reisizadeh, Hossein Taheri, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. Robust and communication-efficient collaborative learning. In *Proc. Conf. Neural Info. Process. Syst.*, pages 8386–8397, Vancouver, Canada, December 2019.

[40] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, September 1951.

[41] 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 *Proc. Conf. Intl. Speech Comm. Assoc.*, Singapore, Sept 2014.

[42] Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Proc. Conf. Neural Info. Process. Syst.*, pages 4447–4458, Montreal, Canada, Dec 2018.

[43] Sebastian Urban Stich. Local SGD converges fast and communicates little. In *Proc. Intl. Conf. Learn. Representations*, New Orleans, LA, May 2019.

[44] Nikko Strom. Scalable distributed DNN training using commodity GPU cloud computing. In *Proc. Conf. Intl. Speech Comm. Assoc.*, Dresden, Germany, September 2015.

[45] Jun Sun, Tianyi Chen, Georgios Giannakis, and Zaiyue Yang. Communication-efficient distributed learning via lazily aggregated quantized gradients. In *Proc. Conf. Neural Info. Process. Syst.*, page to appear, Vancouver, Canada, Dec 2019.

[46] Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. Communication compression for decentralized training. In *Proc. Conf. Neural Info. Process. Syst.*, pages 7652–7662, Montreal, Canada, December 2018.

[47] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In *Proc. Conf. Neural Info. Process. Syst.*, pages 14236–14245, Vancouver, Canada, December 2019.

[48] Guanghui Wang, Shiyin Lu, Weiwei Tu, and Lijun Zhang. SAdam: A variant of adam for strongly convex functions. In *Proc. Intl. Conf. Learn. Representations*, 2020.

[49] Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. In *ICML Workshop on Coding Theory for Large-Scale ML*, Long Beach, CA, June 2019.

[50] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. SlowMo: Improving communication-efficient distributed SGD with slow momentum. In *Proc. Intl. Conf. Learn. Representations*, 2020.

[51] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Proc. Conf. Neural Info. Process. Syst.*, pages 1299–1309, Montreal, Canada, Dec 2018.

[52] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *Proc. Intl. Conf. Machine Learn.*, pages 6677–6686, Long Beach, CA, June 2019.

[53] 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 *Proc. Conf. Neural Info. Process. Syst.*, pages 1509–1519, Long Beach, CA, Dec 2017.

[54] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *Proc. Intl. Conf. Machine Learn.*, pages 5325–5333, Stockholm, Sweden, Jul 2018.

[55] Yangyang Xu, Colin Sutcher-Shepard, Yibo Xu, and Jie Chen. Asynchronous parallel adaptive stochastic gradient methods. *arXiv preprint:2002.09095*, February 2020.

[56] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In *Proc. Intl. Conf. Machine Learn.*, pages 7184–7193, Long Beach, CA, June 2019.

[57] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proc. AAAI Conf. Artificial Intell.*, volume 33, pages 5693–5700, 2019.

[58] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint:1212.5701*, December 2012.

[59] Hantian Zhang, Jerry Li, Kaan Kara, Dan Alistarh, Ji Liu, and Ce Zhang. Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In *Proc. Intl. Conf. Machine Learn.*, pages 4035–4043, Sydney, Australia, Aug 2017.

[60] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging SGD. In *Proc. Conf. Neural Info. Process. Syst.*, pages 685–693, Montreal, Canada, Dec 2015.