DOUBLESQUEEZE: Parallel Stochastic Gradient Descent with Double-pass Error-Compensated Compression

Hanlin Tang [1] **Xiangru Lian** [1] **Chen Yu** [1] **Tong Zhang** [2] **Ji Liu** [3 1]

Abstract

A standard approach in large scale machine learning is distributed stochastic gradient training, which requires the computation of aggregated stochastic gradients over multiple nodes on a network. Communication is a major bottleneck in such applications, and in recent years, compressed stochastic gradient methods such as QSGD (quantized SGD) and sparse SGD have been proposed to reduce communication. It was also shown that error compensation can be combined with compression to achieve better convergence in a scheme that each node compresses its local stochastic gradient and broadcast the result to all other nodes over the network in a single pass. However, such a single pass broadcast approach is not realistic in many practical implementations. For example, under the popular parameter-server model for distributed learning, the worker nodes need to send the compressed local gradients to the parameter server, which performs the aggregation. The parameter server has to compress the aggregated stochastic gradient again before sending it back to the worker nodes. In this work, we provide a detailed analysis on this two-pass communication model, with error-compensated compression both on the worker nodes and on the parameter server. We show that the error-compensated stochastic gradient algorithm admits three very nice properties: 1) it is compatible with an *arbitrary* compression technique; 2) it admits an improved convergence rate than the non error-compensated stochastic gradient methods such as QSGD and sparse SGD; 3) it admits linear speedup with respect to the number of workers. An empirical study is also conducted to validate our theoretical results.

[1]University of Rochester [2]Hong Kong University of Science and Technology [3]Seattle AI Lab, FeDA Lab, Kwai Inc. Correspondence to: Hanlin Tang <htang14@ur.rochester.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

1. Introduction

Large scale distributed machine learning on big data sets are important for many modern applications (Abadi et al., 2016a; Seide & Agarwal, 2016), and there are many methods being studied to improve the performance of distributed learning, such as communication efficient learning (Alistarh et al., 2017; Bernstein et al., 2018a; Seide et al., 2014), decentralized learning (He et al., 2018b; Lian et al., 2017), and asynchronous learning (Agarwal & Duchi, 2011; Lian et al., 2015; Recht et al., 2011). All these methods have been proved to be quite efficient in accelerating distributed learning under different seneario.

A widely used framework in distributed learning is data parallelism, where we assume that data are distributed over multiple nodes on a network, with a shared model that needs to be jointly optimized. Mathematically, the underlying problem can be posed as the following distributed optimization problem:

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}_i} F(\boldsymbol{x};\boldsymbol{\zeta}), \tag{1}$$

where n is the number of workers, $\mathcal{D}_i$ is the local data distribution for worker i (in other words, we do not assume that all nodes can access the same data set), and $F(\boldsymbol{x};\boldsymbol{\zeta})$ is the local loss function of model $\boldsymbol{x}$ given data $\boldsymbol{\zeta}$ for worker i.

A standard synchronized approach for solving (1) is parallel SGD (stochastic gradient descent) (Bottou & Bottou, 2010), where each worker i draws $\boldsymbol{\zeta}^{(i)}$ from $\mathcal{D}_i$, and compute the local stochastic gradient with respect to the shared parameter $\boldsymbol{x}$:

$$\boldsymbol{g}^{(i)} = \nabla F(\boldsymbol{x};\boldsymbol{\zeta}^{(i)}),$$

The local gradients are sent over the network, where the aggregated SGD is computed as:

$$\boldsymbol{g} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{g}^{(i)},$$

and the result are sent back to each local node.

The high communication cost is a main bottleneck for large scale distributed training. In order to alleviate this cost,

recently it has been suggested that each worker can send a compressed version of the local gradient (Alistarh et al., 2018; Stich et al., 2018; Zhang et al., 2017a), with methods such as quantization or sparsification. Specifically, let $Q_\omega[\cdot]$ be a compression operator, one will transmit

$$Q_\omega[\boldsymbol{g}^{(i)}]$$

to form aggregated gradient[1]. However, it is observed that such compression methods slow down the convergence due to the loss of information under compression. To remedy the problem, error compensation has been proposed (Seide et al., 2014), and successfully used in practical applications. The idea is to keep aggregated compression error in a vector $\boldsymbol{\delta}^{(i)}$, and send $Q_\omega\left[\boldsymbol{g}^{(i)}+\boldsymbol{\delta}^{(i)}\right]$, where we update $\boldsymbol{\delta}^{(i)}$ by using the following recursion at each time step

$$\boldsymbol{\delta}^{(i)}=\boldsymbol{g}^{(i)}+\boldsymbol{\delta}^{(i)}-Q_\omega\left[\boldsymbol{g}^{(i)}+\boldsymbol{\delta}^{(i)}\right].$$

It was recently shown that such methods can be effectively used to accelerate convergence when the compression ratio is high.

However, previous work assume that the error compensation is done only for each worker, $\boldsymbol{g}^{(i)}$, but not for the aggregated gradient $\boldsymbol{g}$ (Stich et al., 2018). This is impractical for real world applications, since it can save up to 50% bandwidth. For example, in the popular parameter server model, the aggregation of local gradient is done at the parameter server, and then sent back to each worker node. Although each worker can send sparsified local stochastic gradient to the parameter server, and thus reduce the communication cost. However, the aggregated gradient $\boldsymbol{g}$ can become dense, and to save the communication cost, it needs to be compressed again before sending back to the worker nodes (Wangni et al., 2018). In such case, it is necessary to incorporate error compensation on the parameter server as well because only the parameter server can keep track of the historic compression error. In this paper, we study an error compensated compression of stochastic gradient algorithm namely DOUBLESQUEEZE under this more realistic setting.

The contribution of this paper can be summarized as follows:

- **Better tolerance to compression:** Our theoretical analysis suggests that the proposed DOUBLESQUEEZE enjoys a better tolerance than the non-error-compensated algorithms.
- **Optimal communication cost:** There are only n rounds of communication at each iteration (compared to Alistarh et al. (2018); Wu et al. (2018) where there are n^2 rounds) and we could ensure that all the information to be sent is compressed (compared to (Stich et al., 2018) where only half of the information send from workers to the server is compressed).
- **Prove for parallel case:** To the best of our knowledge, this is the first work that gives the convergence rate analysis for a parallel implementation of error-compensated SGD, and our result shows a linear speedup corresponding to the number of workers n. To the best of our knowledge, this is the first result to show the speedup property for error compensated algorithms.
- **Proof of acceleration for Non-Convex case:** To the best of our knowledge, this is the first work where the loss function in our work is only assumed to be non-convex, which is the case for most of the real world deep neural network, and still prove that the error-compensated SGD admits a factor of improvement over the non-compensated SGD. In Wu et al. (2018) they only consider the quadratic loss function and in Stich et al. (2018) they consider a strongly-convex loss function. Alistarh et al. (2018) considers a non-convex loss function but they did not prove an acceleration of error-compensated SGD.

Notations and definitions Throughout this paper, we use the following notations and definitions

- $\nabla f(\boldsymbol{x})$ denotes the gradient of a function $f(\cdot)$.
- f^* denotes the optimal solution to (1).
- $\|\cdot\|$ denotes the l_2 norm for vectors.
- $\|\cdot\|_2$ denotes the spectral norm for matrix.
- $f_i(\boldsymbol{x}):=\mathbb{E}_{\zeta\sim\mathcal{D}_i}F(\boldsymbol{x};\boldsymbol{\zeta})$.
- $\lesssim$ means "less than equal to up to a constant factor".

2. Related Work

2.1. Distributed Learning

Nowadays, distributed learning has been proved to the key strategy for accelerating the deep learning training. There are two kinds of designs for parallelism: *centralized* design (Agarwal & Duchi, 2011; Recht et al., 2011), where the network is designed to ensure that all workers can get information from all others, and *decentralized* design (He et al., 2018a; Li & Yan, 2017; Lian et al., 2017; 2018; Shi et al., 2015; Tang et al., 2018b), where each worker is only allowed to communicate with its neighbors.

[1] $Q_\omega[\cdot]$ could also include randomness.

Centralized Parallel Training In centralized parallel training, the network is designed to ensure that all workers can get information of all others. One key primitive in centralized training is to aggregate all local models or gradients. This primitive is called the collective communication operator in HPC literature (Thakur et al., 2005). There are different implementations for information aggregation in centralized systems. For example, the parameter server (Abadi et al., 2016b; Li et al., 2014) and AllReduce that averages models over a ring topology (Renggli et al., 2018; Seide & Agarwal, 2016).

Decentralized Parallel Training In decentralized parallel training, the network does not ensure that all workers could get information of all others in a single step. They can only communicate with their individual neighbors. Decentralized training can be divided into fixed topology algorithms and random topology algorithms. For fixed topology algorithms, the communication network is fixed, for example, (Jin et al., 2016; Lian et al., 2017; Shen et al., 2018; Tang et al., 2018b;c). For the random topology decentralized algorithms, the communication network changes over time, for example, (Lian et al., 2018; Nedić & Olshevsky, 2015; Nedic et al., 2017). All of these works provide rigorous analysis to show the convergence rate. For example, Lian et al. (2017) proves that the decentralized SGD achieves a comparable convergence rate to the centralized SGD algorithm, but significantly reduces the communication cost and is more suitable for training a large model.

There has been lots of works studying the implementation of distributed learning from different angles. such as differentially private distributed optimization (Jayaraman et al., 2018; Zhang et al., 2018), adaptive distributed ADMM (Xu et al., 2017), adaptive distributed SGD (Cutkosky & Busa-Fekete, 2018), non-smooth distributed optimization (Scaman et al., 2018), distributed proximal primal-dual algorithm (Hong et al., 2017), projection-free distributed online learning (Zhang et al., 2017b). Some works also investigate methods for parallel backpropgation (Huo et al., 2018; Li et al., 2018).

2.2. Compressed Communication Learning

In order to save the communication cost, a widely used approach is to compress the gradients (Shen et al., 2018). In Wang et al. (2017), the communication cost is reduced by sending a sparsified model from the parameter server to workers. An adaptive approach for doing the compression is proposed in Chen (2018). Alistarh et al. (2017) gives a theoretical analysis for QSGD and studies the tradeoff between local update and communication cost. Most of the previous work used unbiased quantizing operation (Jiang & Agrawal, 2018; Tang et al., 2018a; Wangni et al., 2018; Zhang et al., 2017a) to ensure the convergence of the algorithm. Some extension of communication efficient distributed learning, such as differentially private optimization (Agarwal et al., 2018), optimization on manifolds (Saparbayeva et al., 2018), compressed PCA (Garber et al., 2017), are also studied recently.

In Seide et al. (2014), a 1Bit-SGD was proposed to utilize only the sign of each element in the gradient vector for stochastic gradient descent. The convergence rate guarantee of 1Bit-SGD is studied recently in Bernstein et al. (2018a;b). In Wen et al. (2017), authors manipulate the 1Bit-SGD to ensure that the compressed is an unbiased estimation of the original gradient, and they prove that this unbiased 1Bit-SGD could ensure the algorithm to converge to the single minimum.

Some specific methods for implementing the compression for other distributed systems is also studied. In Suresh et al. (2017), authors proposed some communication efficient strategies distributed mean estimation. A Lazily Aggregated Gradient (**LAG**) strategy is studied to reduce the communication cost for Gradient Descent based distributed learning (Chen et al., 2018). Wang et al. (2018) proposed an atomic sparsification strategy for gradient sparsification.

2.3. Error-Compensated SGD

In Seide et al. (2014), authors used an error-compensate strategy to compensate the error for AllReduce 1Bit-SGD, and found in experiments that the accuracy drop could be minor as long as the error is compensated. Recently, Wu et al. (2018) studied an Error-Compensated SGD for quadratic optimization via adding two hyperparameters to compensate the error, but does not successfully prove the advantage of using error compensation theoretically. In Stich et al. (2018), authors adapted the error-compensate strategy for compressing the gradient, and proved that the error-compensating procedure could greatly reduce the influence of the quantization for non-parallel and strongly-convex loss functions. But their theoretical results are restricted to the compressing operators whose expectation compressing error cannot be larger than the magnitude of the original vector, which is not the case for some biased compressing methods, such as SignSGD (Bernstein et al., 2018a). Alistarh et al. (2018) studied the error-compensated SGD under a non-convex loss function, but did not prove that the error-compensate method could admit a factor of acceleration compared to the non-compensate ones. All of those works did not prove a linear speedup corresponding to the number of workers n for a parallel learning case.

3. Parallel Error-Compensated Algorithms

In this section, we will introduce the parallel error-compensated SGD algorithm, namely DOUBLESQUEEZE.

Algorithm 1 DOUBLESQUEEZE

Input: Initialize $\boldsymbol{x}_0$, learning rate γ, initial error $\boldsymbol{\delta} = \mathbf{0}$, and number of total iterations T.
for $t = 1, 2, \cdots, T$ **do**
  On worker
    Compute the error-compensated stochastic gradient $\boldsymbol{v}^{(i)} \leftarrow \nabla F(\boldsymbol{x}; \boldsymbol{\zeta}^{(i)}) + \boldsymbol{\delta}^{(i)}$
    Compress $\boldsymbol{v}^{(i)}$ into $Q_\omega\left[\boldsymbol{v}^{(i)}\right]$
    Update the error $\boldsymbol{\delta}^{(i)} \leftarrow \boldsymbol{v} - Q_\omega\left[\boldsymbol{v}^{(i)}\right]$
    Send $Q_\omega\left[\boldsymbol{v}^{(i)}\right]$ to the parameter server
  On parameter server
    Average all gradients received from workers $\boldsymbol{v} \leftarrow \frac{1}{n}\sum_{i=1}^n Q_\omega\left[\boldsymbol{v}^{(i)}\right] + \boldsymbol{\delta}$
    Compress $\boldsymbol{v}$ into $Q_\omega\left[\boldsymbol{v}\right]$
    Update the error $\boldsymbol{\delta} \leftarrow \boldsymbol{v} - Q_\omega\left[\boldsymbol{v}\right]$
    Send $Q_\omega\left[\boldsymbol{v}\right]$ to workers
  On worker
    Update the local model $\boldsymbol{x} \leftarrow \boldsymbol{x} - \gamma Q_\omega\left[\boldsymbol{v}\right]$
end for
Output: $\boldsymbol{x}$

We first introduce the algorithm details. Next we will give its mathematical updating formulation from a global view of point in order to get a better understanding of the DOUBLESQUEEZE algorithm.

3.1. Algorithm Description

In this paper, we consider a parameter-server (PS) architecture for parallel training for simplicity – a parameter server and n workers, but the proposed DOUBLESQUEEZE algorithm is not limit to the parameter server architecture. DOUBLESQUEEZE essentially applies the error-compensate strategy on both workers and the parameter to ensure that all information communicated is compressed.

During the tth iteration, the key updating rule for DOUBLESQUEEZE is described below:

- (**Worker: Compute**) Each worker i computes the local stochastic gradient $\nabla F(\boldsymbol{x}_t; \boldsymbol{\zeta}_t^{(i)})$, based on the global model $\boldsymbol{x}_t$ and local sample $\boldsymbol{\zeta}_t^{(i)}$. Here i is the index for worker i and t is the index for iteration number.

- (**Worker: Compress**) Each worker i computes the error-compensated stochastic gradient

$$\boldsymbol{v}_t^{(i)} = \nabla F(\boldsymbol{x}_t; \boldsymbol{\zeta}_t^{(i)}) + \boldsymbol{\delta}_{t-1}^{(i)}, \tag{2}$$

  and update the local error of tth step $\boldsymbol{\delta}_t^{(i)}$ according to

$$\boldsymbol{\delta}_t^{(i)} = \boldsymbol{v}_t^{(i)} - Q_{\omega_t^{(i)}}[\boldsymbol{v}_t^{(i)}], \tag{3}$$

  where $Q_{\omega_t^{(i)}}[\boldsymbol{v}_t^{(i)}]$ is the compressed error-compensated stochastic gradient.

- (**Parameter server: Compress**) All workers send $Q_{\omega_t^{(i)}}\left[\boldsymbol{v}_t^{(i)}\right]$ to the parameter server, then the parameter server average all $Q_{\omega_t^{(i)}}\left[\boldsymbol{v}_t^{(i)}\right]$s and update the global error-compensated stochastic gradient $\boldsymbol{v}_t$, together with the global error $\boldsymbol{\delta}_t$ according to

$$\begin{aligned}\boldsymbol{v}_t =& \boldsymbol{\delta}_{t-1} + \frac{1}{n}\sum_{i=1}^n Q_{\omega_t^{(i)}}\left[\boldsymbol{v}_t^{(i)}\right] \\ \boldsymbol{\delta}_t =& \boldsymbol{v}_t - Q_{\omega_t}\left[\boldsymbol{v}_t\right].\end{aligned} \tag{4}$$

- (**Worker: Update**) The parameter server sends $Q_{\omega_t}[\boldsymbol{v}_t]$ to all workers. Then each worker updates its local model using

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \gamma Q_{\omega_t}[\boldsymbol{v}_t],$$

  where γ is the learning rate.

It is worth noting that all information exchanged between workers and parameter server under the DOUBLESQUEEZE framework is compressed. As a result, the required bandwidth could be extremely low (much lower than 10%). Comparing to some recent error compensated algorithms (Stich et al., 2018), they only compress the gradient sent from the worker to the PS and still send dense vector from PS to workers, which can only save bandwidth up to 50%.

3.2. Compression options

Note that here unlike many existing work (Alistarh et al., 2017; Jiang & Agrawal, 2018), we do not require the compression to be unbiased, which means we do not assume $\mathbb{E}_\omega Q_\omega[\boldsymbol{x}] = \boldsymbol{x}$. So the choice of compression in our framework is pretty flexible. We list a few commonly options for $Q_\omega[\cdot]$[2]:

- **Randomized Quantization:** (Alistarh et al., 2017; Zhang et al., 2017a) For any real number $z \in [a, b]$ (a, b are pre-designed low-bit number), with probability $\frac{b-z}{b-a}$ compress p into a, and with probability $\frac{z-a}{b-a}$ compress z into b. This compression operator is unbiased.

- **1-Bit Quantization:** Compress a vector $\boldsymbol{x}$ into $\|\boldsymbol{x}\| sign(\boldsymbol{x})$, where $sign(\boldsymbol{x})$ is a vector whose element take the sign of the corresponding element in $\boldsymbol{x}$ (see Bernstein et al. (2018a)). This compression operator is biased.

[2]Deterministic operator can be considered as a special case of the randomized operator.

- **Clipping:** For any real number z, directly set its lower k bits into zero. For example, deterministically compress 1.23456 into 1.2 with its lower 4 bits set to zero. This compression operator is biased.
- **Top$-k$ sparsification:** (Stich et al., 2018) For any vector $\boldsymbol{x}$, compress $\boldsymbol{x}$ by retaining the top k largest elements of this vector and set the others to zero. This compression operator is biased.
- **Randomized Sparsification:** (Wangni et al., 2018) For any real number z, with probability p set z to 0 and $\frac{z}{p}$ with probability p. This is also an unbiased compression operator.

3.3. Mathematical form of the updating rule by DOUBLESQUEEZE

Below we are going to prove that the updating rule of DOUBLESQUEEZE admits the form

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \gamma \nabla f(\boldsymbol{x}_t) + \gamma \boldsymbol{\xi}_t - \gamma \Omega_{t-1} + \gamma \Omega_t, \quad (5)$$

where

$$\begin{aligned} \Omega_t :=& \boldsymbol{\delta}_t + \frac{1}{n}\sum_{i=1}^n \boldsymbol{\delta}_t^{(i)} \\ \boldsymbol{\xi}_t :=& \frac{1}{n}\sum_{i=1}^n \left(\nabla f(\boldsymbol{x}_t) - \nabla F\left(\boldsymbol{x}_t; \boldsymbol{\zeta}_t^{(i)}\right)\right). \end{aligned} \quad (6)$$

Here $\boldsymbol{\delta}_t^{(i)}$ and $\boldsymbol{\delta}_t$ are computed according to (3) and (4).

According to the algorithm description in Section 3.1, we know that the updating rule for the global model $\boldsymbol{x}_t$ can be written as

$$\begin{aligned} & \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \\ =& -\gamma Q_{\omega_t}\left[\boldsymbol{\delta}_{t-1} + \frac{1}{n}\sum_{i=1}^n Q_{\omega_t^{(i)}}\left[\boldsymbol{v}_t^{(i)}\right]\right] \\ =& -\gamma\left(\left(\boldsymbol{\delta}_{t-1} + \frac{1}{n}\sum_{i=1}^n Q_{\omega_t^{(i)}}\left[\boldsymbol{v}_t^{(i)}\right]\right) - \boldsymbol{\delta}_t\right) \quad \text{(from (4))} \\ =& -\frac{\gamma}{n}\sum_{i=1}^n Q_{\omega_t^{(i)}}\left[\boldsymbol{v}^{(i)}\right] - \gamma\boldsymbol{\delta}_{t-1} + \gamma\boldsymbol{\delta}_t \\ =& -\frac{\gamma}{n}\sum_{i=1}^n \left(\boldsymbol{v}_t^{(i)} - \boldsymbol{\delta}_t^{(i)}\right) - \gamma\boldsymbol{\delta}_{t-1} + \gamma\boldsymbol{\delta}_t \quad \text{(from (3))} \\ =& -\frac{\gamma}{n}\sum_{i=1}^n \left(\nabla F(\boldsymbol{x}_t; \boldsymbol{\zeta}_t^{(i)}) + \boldsymbol{\delta}_{t-1}^{(i)} - \boldsymbol{\delta}_t^{(i)}\right) \\ & - \gamma\boldsymbol{\delta}_{t-1} + \gamma\boldsymbol{\delta}_t \quad \text{(from (2))} \\ =& -\frac{\gamma}{n}\sum_{i=1}^n \nabla F(\boldsymbol{x}_t; \boldsymbol{\zeta}_t^{(i)}) - \gamma\Omega_{t-1} + \gamma\Omega_t \\ =& -\gamma\nabla f(\boldsymbol{x}_t) + \gamma\boldsymbol{\xi}_t - \gamma\Omega_{t-1} + \gamma\Omega_t. \end{aligned}$$

4. Convergence Analysis

In this section, we are going to give the convergence rate of DOUBLESQUEEZE, and from the theoretical result we shall see that DOUBLESQUEEZE is quite efficient in the way that it could reduce the side effect of the compression. For the convenience of further discussion, we first introduce some assumptions that are necessary for theoretical analysis.

Assumption 1. *We make the following assumptions:*

1. ***Lipschitzian gradient:*** *$f(\cdot)$ is assumed to be with L-Lipschitzian gradients, which means*
$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \le L\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \forall \boldsymbol{y},$$
2. ***Bounded variance:*** *The variance of the stochastic gradient is bounded*
$$\mathbb{E}_{\zeta\sim\mathcal{D}_i}\|\nabla F(\boldsymbol{x}; \boldsymbol{\zeta}) - \nabla f(\boldsymbol{x})\|^2 \le \sigma^2, \quad \forall \boldsymbol{x}, \forall i.$$
3. ***Bounded magnitude of error for $Q_\omega[\cdot]$:*** *The magnitude of worker's local errors $\boldsymbol{\delta}_t^{(i)}$ (defined in (3)), and the server's global error $\boldsymbol{\delta}_t$ (defined in (4)), are assumed to be bounded by a constant ϵ*
$$\begin{aligned} \mathbb{E}_\omega\left\|\boldsymbol{\delta}_t^{(i)}\right\| \le& \frac{\epsilon}{2}, \quad \forall t, \forall i, \\ \mathbb{E}_\omega\left\|\boldsymbol{\delta}_t\right\| \le& \frac{\epsilon}{2}, \quad \forall t. \end{aligned}$$

Here the first and second assumptions are commonly used for non-convex convergence analysis. The third assumption is used to restrict the compression. It can be obtained from the following commonly used assumptions (Stich et al., 2018):

$$\begin{aligned} \mathbb{E}\|C_\omega[\boldsymbol{x}] - \boldsymbol{x}\|^2 \le& \alpha^2\|\boldsymbol{x}\|^2, \quad \alpha \in [0,1), \forall \boldsymbol{x} \\ \|\nabla f(\boldsymbol{x})\|^2 \le& G^2, \quad \forall \boldsymbol{x}, \end{aligned}$$

where α is a constant specifies the compression level and is not required to be bounded in $[0, 1)$. Because

$$\mathbb{E}\|\boldsymbol{\delta}_t\|^2 = \mathbb{E}\|C_\omega[\boldsymbol{g}_t - \boldsymbol{\delta}_{t-1}] - \boldsymbol{g}_t + \boldsymbol{\delta}_{t-1}\|^2, \quad (7)$$

where $\boldsymbol{g}_t := \frac{1}{n}\sum_{i=1}^n \nabla F(\boldsymbol{x}_t; \boldsymbol{\xi}_t)$ is the sum of all stochastic gradient at each iteration, then from (7) we have

$$\begin{aligned} & \mathbb{E}\|\boldsymbol{\delta}_t\|^2 \\ \le& \mathbb{E}\alpha^2\|\boldsymbol{g}_t - \boldsymbol{\delta}_{t-1}\|^2 \\ \le& (1+\rho)\alpha^2\mathbb{E}\|\boldsymbol{g}_t\|^2 + \left(1 + \frac{1}{\rho}\right)\alpha^2\mathbb{E}\|\boldsymbol{\delta}_{t-1}\|^2 \\ \le& (1+\rho)\alpha^2\mathbb{E}\|\boldsymbol{g}_t\|^2 + \left((1+\rho)\alpha^2\right)^2\mathbb{E}\|\boldsymbol{g}_{t-1}\|^2 \\ & + \left(\left(1 + \frac{1}{\rho}\right)\alpha^2\right)^2\mathbb{E}\|\boldsymbol{\delta}_{t-2}\|^2 \end{aligned}$$

$$\leq \sum_{s=1}^{t} \left((1+\rho)\,\alpha^2\right)^{t-s} \mathbb{E}\|\boldsymbol{g}_s\|^2 + \left(\left(1+\frac{1}{\rho}\right)\alpha^2\right)^{t} \mathbb{E}\|\boldsymbol{\delta}_0\|^2$$
$$\leq \sum_{s=1}^{t} \left((1+\rho)\,\alpha^2\right)^{t-s} \mathbb{E}\|\boldsymbol{g}_s\|^2 \quad (\boldsymbol{\delta}_0 = \mathbf{0})$$
$$\leq \frac{G^2 + \frac{\sigma^2}{n}}{1-(1+\rho)\,\alpha^2}. \quad \left(\mathbb{E}\|\boldsymbol{g}_t\|^2 \leq G^2 + \frac{\sigma^2}{n}\right)$$

Here $\rho > 0$ can be any positive constant. So $\|\boldsymbol{\delta}_t\|^2$ would be bounded as long as $\alpha < \frac{1}{\sqrt{1+\rho}}$, which is equivalent to $\alpha \in [0, 1)$ since ρ can be any positive number.

Next we are ready to present the main theorem for DOUBLESQUEEZE.

Theorem 1. *Under Assumption 1, for* DOUBLESQUEEZE*, we have the following convergence rate*

$$\left(\frac{\gamma}{2} - \frac{L\gamma^2}{2}\right) \sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla f(\boldsymbol{x}_t)\right\|^2$$
$$\leq \mathbb{E}f(\boldsymbol{x}_0) - \mathbb{E}f(\boldsymbol{x}^*) + \frac{L\gamma^2\sigma^2 T}{2n} + 2L^2\epsilon^2\gamma^3 T.$$

Given the generic result in Theorem 1, we obtain the convergence rate for DOUBLESQUEEZE with appropriately chosen the learning rate γ.

Corollary 2. *Under Assumption 1, for* DOUBLESQUEEZE*, choosing*

$$\gamma = \frac{1}{4L + \sigma\sqrt{\frac{T}{n}} + \epsilon^{\frac{2}{3}} T^{\frac{1}{3}}},$$

we have the following convergence rate

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 \lesssim \frac{\sigma}{\sqrt{nT}} + \frac{\epsilon^{\frac{2}{3}}}{T^{\frac{2}{3}}} + \frac{1}{T},$$

where we treat $f(\boldsymbol{x}_1) - f^*$ and L as constants.

This result suggests that

- (**Comparison to SGD**) DoubleSqueeze essentially admits the same convergence rate as SGD in the sense that both of them admit the asymptotical convergence rate $O(1/\sqrt{T})$;
- (**Linear Speedup**) The asymptotical convergence rate of DOUBLESQUEEZE is $O(1/\sqrt{nT})$, the same convergence rate as Parallel SGD. It implies that the averaged sample complexity is $O(1/(n\epsilon^2))$. To the best of our knowledge, this is the first analysis to show the linear speedup for the error compensated type of algorithms.
- (**Advantage over non error-compensated SGD (Alistarh et al., 2017; Wangni et al., 2018)**) For non error-compensated SGD, there is no guarantee for convergence in general unless the compression operator is unbiased. Using the existing analysis for SGD's convergence rate

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 \lesssim \frac{\sigma'}{\sqrt{nT}} + \frac{1}{T}$$

where σ' is the stochastic variance, it is not hard to obtain the following convergence rate for unbiased compressed SGD:
(one-pass compressed SGD on workers such as QSGD (Alistarh et al., 2017) and sparse SGD (Wangni et al., 2018))

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 \lesssim \frac{\sigma}{\sqrt{nT}} + \frac{\epsilon}{\sqrt{nT}} + \frac{1}{T}$$

(double-pass compressed SGD on workers and the parameter server)

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 \lesssim \frac{\sigma}{\sqrt{T}} + \frac{\epsilon}{\sqrt{T}} + \frac{1}{T}.$$

Note that ϵ measures the upper bound of the (stochastic) compression variance. Therefore, when ϵ is dominant, the convergence rate for DOUBLESQUEEZE has a much better dependence on ϵ in terms of iteration number T. It means that DOUBLESQUEEZE has a much better tolerance on the compression variance or bias. It makes sense since DOUBLESQUEEZE does not drop any information in stochastic gradients just delay to update some portion in them.

5. Experiments

We validate our theory with experiments that compared DOUBLESQUEEZE with other compression implementations. We run experiments with 1 parameter server and 8 workers, and show that, the DOUBLESQUEEZE converges similar to SGD without compression, but runs much faster than vanilla SGD and other compressed SGD algorithms when bandwidth is limited.

5.1. Experiment setting

Datasets and models We evaluate DOUBLESQUEEZE by training ResNet-18 (He et al., 2016) on CIFAR-10. The model size is about 44MB.

Implementations and setups We evaluate five SGD implementations:

1. **DOUBLESQUEEZE.** Both workers and the parameter server compress gradients. The error caused by

compression are saved and used to compensate new gradients as shown in Algorithm 1. We evaluate DOUBLESQUEEZE with two compression methods:

- *1-bit compression*: The gradients are quantized into 1-bit representation (containing the sign of each element). Accompanying the vector, a scaling factor is computed as

$$\frac{\text{magnitude of compensated gradient}}{\text{magnitude of quantized gradient}}.$$

  The scaling factor is multiplied onto the quantized gradient whenever the quantized gradient is used, so that the recovered gradient has the same magnitude of the compensated gradient.
- *Top-k compression*: The compensated gradients are compressed so that only the largest k elements (in the sense of absolute value) are kept, and all other elements are set to 0.

2. **QSGD (Alistarh et al., 2017).** The workers quantize the gradients into a tenary representation, where each element is in the set $\{-1, 0, 1\}$. Assuming the element with maximum absolute value in a gradient vector is m, for any other element e, it has a probability of $|e|/|m|$ to be quantized to $sign(e)$, and a probability of $1 - |e|/|m|$ to be quantized to 0. A scaling factor like the one in DOUBLESQUEEZE is computed as

$$\frac{\text{magnitude of original gradient}}{\text{magnitude of compressed gradient}}.$$

   The parameter server aggregates the gradients and sends the aggregated gradient back to all workers without compression.

3. **Vanilla SGD.** This is the common centralized parallel SGD implementation without compression, where the parameter server aggregates all gradients and sends it back to each worker.

4. **MEM-SGD.** As in DOUBLESQUEEZE, workers do both compression and compensation. However, the parameter server aggregates all gradients and sends it back to all workers without compression as shown in Stich et al. (2018). For MEM-SGD, we also evaluate both 1-bit compression and top-k compression methods.

5. **Top-k SGD.** This is vanilla SGD with top-k compression in each worker, without compensation.

For more direct comparison, no momentum and weight decay are used in the optimization process. The learning rate starts with 0.1 and is reduced by a factor of 10 every 160 epochs. The batch size is set to 256 on each worker. Each worker computes gradients on a Nvidia 1080Ti.

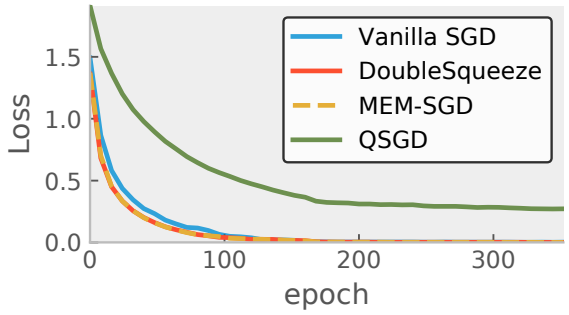

Figure 1: Training loss w.r.t. epochs for DOUBLESQUEEZE (1-bit compression), QSGD, Vanilla SGD, and MEM-SGD (1-bit compression) on CIFAR-10.

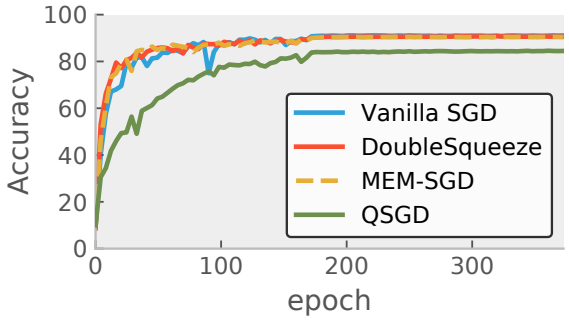

Figure 2: Testing accuracy w.r.t. epochs for DOUBLESQUEEZE (1-bit compression), QSGD, Vanilla SGD, and MEM-SGD (1-bit compression) on CIFAR-10.

5.2. Experiment results

The empirical study is conducted on two compression approaches: 1-bit compression and top-k compression.

1-bit compression We apply the 1-bit compression to DOUBLESQUEEZE, MEM-SGD, QSGD, and report results for the training loss w.r.t. epochs in Figure 1. The result shows that with 1-bit compression DOUBLESQUEEZE and MEM-SGD converge similarly w.r.t. epochs as Vanilla SGD, while QSGD converges much slower due to the lack of compensation. For testing accuracy, we have similar results, as shown in Figure 2.

While DOUBLESQUEEZE, MEM-SGD, and Vanilla SGD converges similarly w.r.t. epochs, when network bandwidth is limited, DOUBLESQUEEZE can be much faster than other algorithms as shown in Figure 3.

Top-k compression For the top-k compression method, we choose $k = 300000$, which is about $1/32$ of the number of parameters in the model. We report results for the training loss and testing accuracy w.r.t. epochs in Figure 4 and

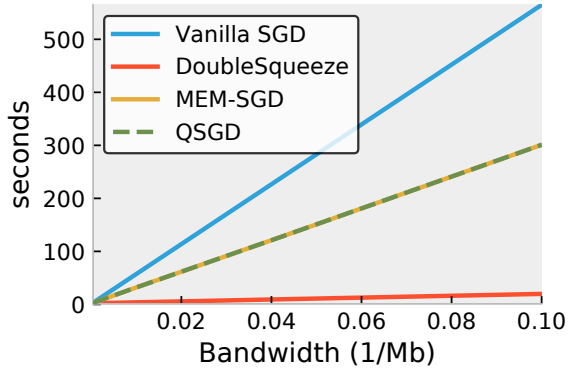

Figure 3: Per iteration time cost for DOUBLESQUEEZE (1-bit compression), QSGD, MEM-SGD (1-bit compression), and Vanilla SGD, under different network environments. The x-axis represents the inverse of the bandwidth of the parameter server. The y-axis is the number of seconds needed to finish one iteration.

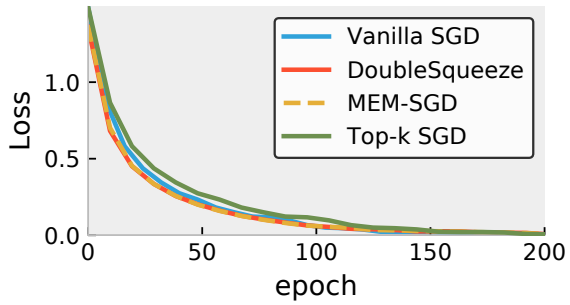

Figure 4: Training loss w.r.t. epochs for DOUBLESQUEEZE (top-k compression), Top-k SGD, Vanilla SGD, and MEM-SGD (top-k compression) on CIFAR-10. $k = 300000$.

Figure 5, respectively, for Vanilla SGD, DOUBLESQUEEZE, MEM-SGD, and Top-k SGD. With the top-k compression, all methods converge similarly w.r.t. epochs. The Top-k SGD method converges a little bit slower.

Similar to what we observed in the 1-bit compression experiment, when network bandwidth is limited, DOUBLESQUEEZE can be much faster than other algorithms as shown in Figure 6.

6. Conclusion

In this paper, we study an error-compensated SGD algorithm, namely DOUBLESQUEEZE that performs the compression on both the worker's side and the parameter server's side, to ensure that all information exchanged over the network is compressed. As a result, this approach can significantly save the bandwidth, unlike many existing error compensated algorithms that can only save bandwidth up to 50%. Theoretical convergence for DOUBLESQUEEZE

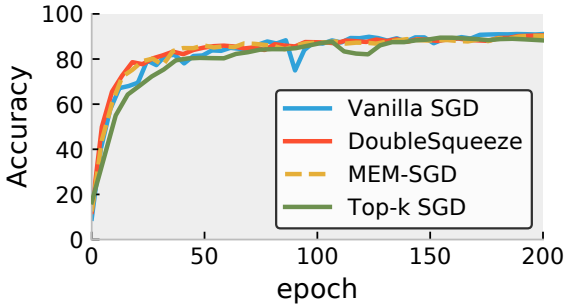

Figure 5: Testing accuracy w.r.t. epochs for DOUBLESQUEEZE (top-k compression), Top-k SGD, Vanilla SGD, and MEM-SGD (top-k compression) on CIFAR-10. $k = 300000$.

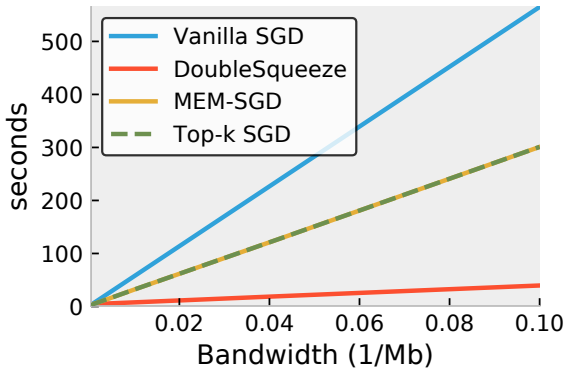

Figure 6: Per iteration time cost for DOUBLESQUEEZE (top-k compression), Top-k SGD, MEM-SGD (top-k compression), and Vanilla SGD, under different network environments. The x-axis represents the inverse of the bandwidth of the parameter server. The y-axis is the number of seconds needed to finish one iteration.

is also provided. It implies that DOUBLESQUEEZE admits the linear speedup corresponding to the number of workers, and has a better tolerance to the compression bias and noise than those non-error-compensated approaches. Empirical study is also conducted to validate the DOUBLESQUEEZE algorithm.

Acknowledgements

This project is in part supported by NSF CCF1718513, IBM faculty award, and NEC fellowship.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pp. 265–283, Berkeley, CA, USA, 2016a. USENIX Association. ISBN 978-1-931971-33-1.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016b.

Agarwal, A. and Duchi, J. C. Distributed delayed stochastic optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 873–881. Curran Associates, Inc., 2011.

Agarwal, N., Suresh, A. T., Yu, F. X. X., Kumar, S., and McMahan, B. cpsgd: Communication-efficient and differentially-private distributed sgd. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 7575–7586. Curran Associates, Inc., 2018.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: communication-efficient SGD via gradient quantization and encoding. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1707–1718, 2017.

Alistarh, D., Hoefler, T., Johansson, M., Konstantinov, N., Khirirat, S., and Renggli, C. The convergence of sparsified gradient methods. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 5977–5987. Curran Associates, Inc., 2018.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: compressed optimisation for non-convex problems. In *ICML*, 2018a.

Bernstein, J., Zhao, J., Azizzadenesheli, K., and Anandkumar, A. signsgd with majority vote is communication efficient and byzantine fault tolerant. 10 2018b.

Bottou, L. and Bottou, L. Large-scale machine learning with stochastic gradient descent. *IN COMPSTAT*, 2010. doi: 10.1.1.419.462.

Chen, C.-Y. *AdaComp : Adaptive Residual Gradient Compression for Data-Parallel Distributed Training.; AAAI*. 2018.

Chen, T., Giannakis, G., Sun, T., and Yin, W. Lag: Lazily aggregated gradient for communication-efficient distributed learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 5055–5065. Curran Associates, Inc., 2018.

Cutkosky, A. and Busa-Fekete, R. Distributed stochastic optimization via adaptive sgd. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 1914–1923. Curran Associates, Inc., 2018.

Garber, D., Shamir, O., and Srebro, N. Communication-efficient algorithms for distributed stochastic principal component analysis. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1203–1212, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, L., Bian, A., and Jaggi, M. Cola: Decentralized linear learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 4541–4551. Curran Associates, Inc., 2018a.

He, L., Bian, A., and Jaggi, M. Cola: Decentralized linear learning. In *Advances in Neural Information Processing Systems*, pp. 4541–4551, 2018b.

Hong, M., Hajinezhad, D., and Zhao, M.-M. Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1529–1538, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Huo, Z., Gu, B., qian Yang, and Huang, H. Decoupled parallel backpropagation with convergence guarantee. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80

of *Proceedings of Machine Learning Research*, pp. 2098–2106, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Jayaraman, B., Wang, L., Evans, D., and Gu, Q. Distributed learning without distress: Privacy-preserving empirical risk minimization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6346–6357. Curran Associates, Inc., 2018.

Jiang, P. and Agrawal, G. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2530–2541. Curran Associates, Inc., 2018.

Jin, P. H., Yuan, Q., Iandola, F., and Keutzer, K. How to scale distributed deep learning? *arXiv preprint arXiv:1611.04581*, 2016.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pp. 583–598, 2014.

Li, Y., Yu, M., Li, S., Avestimehr, S., Kim, N. S., and Schwing, A. Pipe-sgd: A decentralized pipelined sgd framework for distributed deep net training. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8056–8067. Curran Associates, Inc., 2018.

Li, Z. and Yan, M. A primal-dual algorithm with optimal stepsizes and its application in decentralized consensus optimization. *arXiv preprint arXiv:1711.06785*, 2017.

Lian, X., Huang, Y., Li, Y., and Liu, J. Asynchronous parallel stochastic gradient for nonconvex optimization. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2737–2745. Curran Associates, Inc., 2015.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 5330–5340, 2017.

Lian, X., Zhang, W., Zhang, C., and Liu, J. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, 2018.

Nedić, A. and Olshevsky, A. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2015.

Nedic, A., Olshevsky, A., and Shi, W. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4): 2597–2633, 2017.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 693–701. Curran Associates, Inc., 2011.

Renggli, C., Alistarh, D., and Hoefler, T. Sparcml: High-performance sparse communication for machine learning. *arXiv preprint arXiv:1802.08021*, 2018.

Saparbayeva, B., Zhang, M., and Lin, L. Communication efficient parallel algorithms for optimization on manifolds. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3578–3588. Curran Associates, Inc., 2018.

Scaman, K., Bach, F., Bubeck, S., Massoulié, L., and Lee, Y. T. Optimal algorithms for non-smooth distributed optimization in networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2745–2754. Curran Associates, Inc., 2018.

Seide, F. and Agarwal, A. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2135–2135. ACM, 2016.

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns. In *Interspeech 2014*, September 2014.

Shen, Z., Mokhtari, A., Zhou, T., Zhao, P., and Qian, H. Towards more efficient stochastic decentralized learning: Faster convergence and sparse communication. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4624–4633, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Shi, W., Ling, Q., Wu, G., and Yin, W. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified sgd with memory. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.),

Advances in Neural Information Processing Systems 31, pp. 4452–4463. Curran Associates, Inc., 2018.

Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, H. B. Distributed mean estimation with limited communication. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3329–3337, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Tang, H., Gan, S., Zhang, C., Zhang, T., and Liu, J. Communication compression for decentralized training. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 7663–7673. Curran Associates, Inc., 2018a.

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. d^2: Decentralized training over decentralized data. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4848–4856, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR.

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. D2: Decentralized training over decentralized data. *arXiv preprint arXiv:1803.07068*, 2018c.

Thakur, R., Rabenseifner, R., and Gropp, W. Optimization of collective communication operations in mpich. *The International Journal of High Performance Computing Applications*, 19(1):49–66, 2005.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient learning via atomic sparsification. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 9872–9883. Curran Associates, Inc., 2018.

Wang, J., Kolar, M., Srebro, N., and Zhang, T. Efficient distributed learning with sparsity. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3636–3645, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 1306–1316. Curran Associates, Inc., 2018.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1509–1519. Curran Associates, Inc., 2017.

Wu, J., Huang, W., Huang, J., and Zhang, T. Error compensated quantized SGD and its applications to large-scale distributed optimization. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5325–5333, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Xu, Z., Taylor, G., Li, H., Figueiredo, M. A. T., Yuan, X., and Goldstein, T. Adaptive consensus ADMM for distributed optimization. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3841–3850, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Zhang, H., Li, J., Kara, K., Alistarh, D., Liu, J., and Zhang, C. ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 4035–4043, International Convention Centre, Sydney, Australia, 06–11 Aug 2017a. PMLR.

Zhang, W., Zhao, P., Zhu, W., Hoi, S. C. H., and Zhang, T. Projection-free distributed online learning in networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 4054–4062, International Convention Centre, Sydney, Australia, 06–11 Aug 2017b. PMLR.

Zhang, X., Khalili, M. M., and Liu, M. Improving the privacy and accuracy of ADMM-based distributed algorithms. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5796–5805, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.