A Double Residual Compression Algorithm for Efficient Distributed Learning

Xiaorui Liu, Yao Li, Jiliang Tang and Ming Yan
{xiaorui, liyao6, tangjili, myan}@msu.edu
Michigan State University

Abstract

Large-scale machine learning models are often trained by parallel stochastic gradient descent algorithms. However, the communication cost of gradient aggregation and model synchronization between the master and worker nodes becomes the major obstacle for efficient learning as the number of workers and the dimension of the model increase. In this paper, we propose DORE, a DOuble REsidual compression stochastic gradient descent algorithm, to reduce over 95% of the overall communication such that the obstacle can be immensely mitigated. Our theoretical analyses demonstrate that the proposed strategy has superior convergence properties for both strongly convex and nonconvex objective functions. The experimental results validate that DORE achieves the best communication efficiency while maintaining similar model accuracy and convergence speed in comparison with start-of-the-art baselines.

1 Introduction

Stochastic gradient algorithms (Bottou, 2010) are efficient at minimizing the objective function $f : \mathbb{R}^d \to \mathbb{R}$ which is usually defined as $f(\mathbf{x}) := \mathbb{E}_{\xi\sim\mathcal{D}}[\ell(\mathbf{x}, \xi)]$, where $\ell(\mathbf{x}, \xi)$ is the objective function defined on data sample ξ and model parameter $\mathbf{x}$. A basic stochastic gradient descent (SGD) repeats the gradient "descent" step $\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma\mathbf{g}(\mathbf{x}^k)$ where $\mathbf{x}_k$ is the current iteration and γ is the step size. The stochastic gradient $\mathbf{g}(\mathbf{x}^k)$ is computed based on an i.i.d. sampled mini-batch from the distribution of the training data $\mathcal{D}$ and serves as the estimator of the full gradient $\nabla f(\mathbf{x}^k)$. In the context of large-scale machine learning, the number of data samples and the model size are usually very large. Distributed learning utilizes a large number of computers/cores to perform the stochastic algorithms aiming at reducing the training time. It has attracted extensive attention due to the demand for highly efficient model training (Abadi et al., 2016; Chen et al., 2015; Li et al., 2014; You et al., 2018).

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

In this paper, we focus on the data-parallel SGD (Dean et al., 2012; Lian et al., 2015; Zinkevich et al., 2010), which provides a scalable solution to speed up the training process by distributing the whole data to multiple computing nodes. The objective can be written as:

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}}\, f(\mathbf{x}) + R(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i}[\ell(\mathbf{x}, \xi)]}_{:=f_i(\mathbf{x})} + R(\mathbf{x}),$$

where each $f_i(\mathbf{x})$ is a local objective function of the worker node i defined based on the allocated data under distribution $\mathcal{D}_i$ and $R : \mathbb{R}^d \to \mathbb{R}$ is usually a closed convex regularizer.

In the well-known parameter server framework (Li et al., 2014; Zinkevich et al., 2010), during each iteration, each worker node evaluates its own stochastic gradient $\{\widetilde{\nabla} f_i(\mathbf{x}^k)\}_{i=1}^n$ and send it to the master node, which collects all gradients and calculates their average $(1/n)\sum_{i=1}^n \widetilde{\nabla} f_i(\mathbf{x}^k)$. Then the master node further takes the gradient descent step with the averaged gradient and broadcasts the new model parameter $\mathbf{x}^{k+1}$ to all worker nodes. It makes use of the computational resources from all nodes. In reality, the network bandwidth is often limited. Thus, the communication cost for the gradient transmission and model synchronization becomes the dominating bottlenecks as the number of nodes and the model size increase, which hinders the scalability and efficiency of SGD.

One common way to reduce the communication cost is to compress the gradient information by either gradient sparsification or quantization (Alistarh et al., 2017;

Seide et al., 2014; Stich et al., 2018; Strom, 2015; Wang et al., 2017; Wangni et al., 2018; Wen et al., 2017; Wu et al., 2018) such that many fewer bits of information are needed to be transmitted. However, little attention has been paid on how to reduce the communication cost for model synchronization and the corresponding theoretical guarantees. Obviously, the model shares the same size as the gradient, so does the communication cost. Thus, merely compressing the gradient can reduce at most 50% of the communication cost, which suggests the importance of model compression. Notably, the compression of model parameters is much more challenging than gradient compression. One key obstacle is that its compression error cannot be well controlled by the step size γ and thus it cannot diminish like that in the gradient compression (Tang et al., 2018). In this paper, we aim to bridge this gap by investigating algorithms to compress the full communication in the optimization process and understanding their theoretical properties. Our contributions can be summarized as:

- We proposed DORE, which can compress both the gradient and the model information such that more than 95% of the communication cost can be reduced.
- We provided theoretical analyses to guarantee the convergence of DORE under strongly convex and nonconvex assumptions without the bounded gradient assumption.
- Our experiments demonstrate the superior efficiency of DORE comparing with the state-of-art baselines without degrading the convergence speed and the model accuracy.

2 Background

Recently, many works try to reduce the communication cost to speed up the distributed learning, especially for deep learning applications, where the size of the model is typically very large (so is the size of the gradient) while the network bandwidth is relatively limited. Below we briefly review relevant papers.

Gradient quantization and sparsification. Recent works (Alistarh et al., 2017; Seide et al., 2014; Wen et al., 2017; Mishchenko et al., 2019; Bernstein et al., 2018) have shown that the information of the gradient can be quantized into a lower-precision vector such that fewer bits are needed in communication without loss of accuracy. Seide et al. (2014) proposed 1Bit SGD that keeps the sign of each element in the gradient only. It empirically works well, and Bernstein et al. (2018) provided theoretical analysis systematically. QSGD (Alistarh et al., 2017) utilizes an unbiased multi-level random quantization to compress the gradient while Terngrad (Wen et al., 2017) quantizes the gradient into ternary numbers $\{0, \pm 1\}$. In DIANA (Mishchenko et al., 2019), the gradient difference is compressed and communicated contributing to the estimator of the gradient in the master node.

Another effective strategy to reduce the communication cost is sparsification. Wangni et al. (2018) proposed a convex optimization formulation to minimize the coding length of stochastic gradients. A more aggressive sparsification method is to keep the elements with relatively larger magnitude in gradients, such as top-k sparsification (Stich et al., 2018; Strom, 2015; Aji and Heafield, 2017).

Model synchronization. The typical way for model synchronization is to broadcast model parameters to all worker nodes. Some works (Wang et al., 2017; Jordan et al., 2019) have been proposed to reduce model size by enforcing sparsity, but it cannot be applied to general optimization problems. Some alternatives including QSGD (Alistarh et al., 2017) and ECQ-SGD (Wu et al., 2018) choose to broadcast all quantized gradients to all other workers such that every worker can perform model update independently. However, all-to-all communication is not efficient since the number of transmitted bits increases dramatically in large-scale networks. DoubleSqueeze (Tang et al., 2019) applies compression on the averaged gradient with error compensation to speed up model synchronization.

Error compensation. Seide et al. (2014) applied error compensation on 1Bit-SGD and achieved negligible loss of accuracy empirically. Recently, error compensation was further studied (Wu et al., 2018; Stich et al., 2018; Karimireddy et al., 2019) to mitigate the error caused by compression. The general idea is to add the compressed error to the next compression step:

$$\hat{\mathbf{g}} = Q(\mathbf{g} + \mathbf{e}), \quad \mathbf{e} = (\mathbf{g} + \mathbf{e}) - \hat{\mathbf{g}}.$$

However, to the best of our knowledge, most of the algorithms with error compensation (Wu et al., 2018; Stich et al., 2018; Karimireddy et al., 2019; Tang et al., 2019) need to assume bounded gradient, i.e., $\mathbb{E}\|\mathbf{g}\|^2 \leq B$, and the convergence rate depends on this bound.

Contributions of DORE. The most related papers to DORE are DIANA (Mishchenko et al., 2019) and DoubleSqueeze (Tang et al., 2019). Similarly, DIANA compresses gradient difference on the worker side and achieves good convergence rate. However, it doesn't consider the compression in model synchronization, so at most 50% of the communication cost can be saved. DoubleSqueeze applies compression with error compensation on both worker and server sides, but it only considers non-convex objective functions. Moreover, its analysis relies on a bounded gradient assumption, i.e., $\mathbb{E}\|\mathbf{g}\|^2 \leq B$, and the convergence error has a dependency on the gradient bound like most existed error compensation works.

In general, the uniform bound on the norm of the stochastic gradient is a strong assumption which might not hold in some cases. For example, it is violated in the strongly convex case (Nguyen et al., 2018; Gower et al., 2019). In this paper, we design DORE, the first algorithm which utilizes gradient and model compression with error compensation without assuming bounded gradients. Unlike existing error compensation works, we provide a linear convergence rate to the $\mathcal{O}(\sigma)$ neighborhood of the optimal solution for strongly convex functions and a sublinear rate to the stationary point for nonconvex functions with linear speedup. In Table 1, we compare the asymptotic convergence rates of different quantized SGDs with DORE.

3 Double Residual Compression SGD

In this section, we introduce the proposed DOuble REsidual compression SGD (DORE) algorithm. Before that, we introduce a common assumption for the compression operator.

In this work, we adopt an assumption from (Alistarh et al., 2017; Wen et al., 2017; Mishchenko et al., 2019) that the compression variance is linearly proportional to the magnitude.

Assumption 1. *The stochastic compression operator* $Q : \mathbb{R}^d \to \mathbb{R}^d$ *is unbiased, i.e.,* $\mathbb{E}Q(\mathbf{x}) = \mathbf{x}$ *and satisfies*

$$\mathbb{E}\|Q(\mathbf{x}) - \mathbf{x}\|^2 \leq C\|\mathbf{x}\|^2, \tag{1}$$

for a nonnegative constant C *that is independent of* $\mathbf{x}$*. We use* $\hat{\mathbf{x}}$ *to denote the compressed* $\mathbf{x}$*, i.e.,* $\hat{\mathbf{x}} \sim Q(\mathbf{x})$*.*

Many feasible compression operators can be applied to our algorithm since our theoretical analyses are built on this common assumption. Some examples of feasible stochastic compression operators include:

- *No Compression:* $C = 0$ when there is no compression.
- *Stochastic Quantization:* A real number $x \in [a, b], (a < b)$ is set to be a with probability $\frac{b-x}{b-a}$ and b with probability $\frac{x-a}{b-a}$, where a and b are predefined quantization levels (Alistarh et al., 2017). It satisfies Assumption 1 when $ab > 0$ and $a < b$.
- *Stochastic Sparsification:* A real number x is set to be 0 with probability $1 - p$ and $\frac{x}{p}$ with probability p (Wen et al., 2017). It satisfies Assumption 1 with $C = (1/p) - 1$.

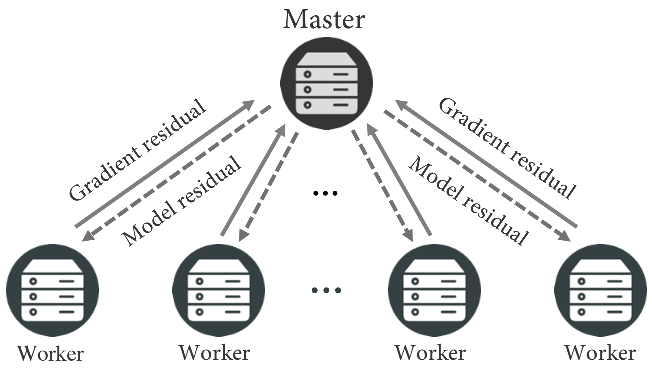


Figure 1: An Illustration of DORE

- *p-norm Quantization:* A vector $\mathbf{x}$ is quantized element-wisely by $Q_p(\mathbf{x}) = \|\mathbf{x}\|_p \operatorname{sign}(\mathbf{x}) \circ \xi$, where $\circ$ is the Hadamard product and ξ is a Bernoulli random vector satisfying $\xi_i \sim \text{Bernoulli}(\frac{|x_i|}{\|\mathbf{x}\|_p})$. It satisfies Assumption 1 with $C = \max_{\mathbf{x}\in\mathbb{R}^d} \frac{\|\mathbf{x}\|_1\|\mathbf{x}\|_p}{\|\mathbf{x}\|_2^2} - 1$ (Mishchenko et al., 2019). To decrease the constant C for a higher accuracy, a vector $\mathbf{x} \in \mathbb{R}^d$ can be further decomposed into blocks, i.e., $\mathbf{x} = (\mathbf{x}(1)^\top, \mathbf{x}(2)^\top, \cdots, \mathbf{x}(m)^\top)^\top$ with $\mathbf{x}(l) \in \mathbb{R}^{d_l}$ and $\sum_{l=1}^m d_l = d$, and the blocks can be compressed independently.

3.1 The Proposed DORE

Many previous works (Alistarh et al., 2017; Seide et al., 2014; Wen et al., 2017) reduce the communication cost of P-SGD by quantizing the stochastic gradient before sending it to the master node, but there are several intrinsic issues.

First, these algorithms will incur extra optimization error intrinsically. Let's consider the case when the algorithm converges to the optimal point $\mathbf{x}^*$ where we have $(1/n)\sum_{i=1}^n \nabla f_i(\mathbf{x}^*) = \mathbf{0}$. However, the data distributions may be different for different worker nodes in general, and thus we may have $\nabla f_i(\mathbf{x}^*) \neq \nabla f_j(\mathbf{x}^*), \forall i, j \in \{1, \ldots, n\}$ and $i \neq j$. In other words, each individual $\nabla f_i(\mathbf{x}^*)$ may be far away from zero. This will cause large compression variance according to Assumption 1, which indicates that the upper bound of compression variance $\mathbb{E}\|Q(\mathbf{x}) - \mathbf{x}\|^2$ is linearly proportional to the magnitude of $\mathbf{x}$.

Second, most existing algorithms (Seide et al., 2014; Alistarh et al., 2017; Wen et al., 2017; Bernstein et al., 2018; Wu et al., 2018; Mishchenko et al., 2019) need to broadcast the model or gradient to all worker nodes in each iteration. It is a considerable bottleneck for efficient optimization since the amount of bits to transmit is the same as the uncompressed gradient. DoubleSqueeze (Tang et al., 2019) is able to apply compression on both worker and server sides. However, its analysis depends on a strong assumption on bounded gradient. Meanwhile, no theoretical guarantees are provided for the convex problems.

We proposed DORE to address all aforementioned issues. Our motivation is that the gradient should change smoothly for smooth functions so that each worker node can keep a state variable $\mathbf{h}_i^k$ to track its previous gradient information. As a result, the residual between new gradient and the state $\mathbf{h}_i^k$ should decrease, and the compression variance of the residual can be well bounded. On the other hand, as the algorithm converges, the model would only change slightly. Therefore, we propose to compress the model residual such that the compression variance can be minimized and also well bounded. We also compensate the model residual compression error into next iteration to achieve a better convergence. Due to the advantages of the proposed double residual compression scheme, we can derive the fastest convergence rate through analyses without the bounded gradient assumption. Below are some key steps of our algorithm as showed in Algorithm 1 and Figure 1:

[lines 4-9]: each worker node sends the compressed gradient residual ($\hat{\Delta}_i^k$) to the master node and updates its state $\mathbf{h}_i^k$ with $\hat{\Delta}_i^k$;

[lines 13-15]: the master node gathers the compressed gradient residual ($\{\hat{\Delta}_i^k\}$) from all worker nodes and recovers the averaged gradient $\hat{\mathbf{g}}^k$ based on its state $\mathbf{h}^k$;

[lines 16]: the master node applies gradient descent algorithms (possibly with the proximal operator);

[lines 18-22]: the master node broadcasts the compressed model residual with error compensation ($\hat{\mathbf{q}}^k$) to all worker nodes and updates the model;

[lines 10-11]: each worker node receives the compressed model residual ($\hat{\mathbf{q}}^k$) and updates its model $\mathbf{x}_i^k$.

In the algorithm, the state $\mathbf{h}_i^k$ serves as an exponential moving average of the local gradient in expectation, i.e., $\mathbb{E}_Q \mathbf{h}_i^{k+1} = (1-\alpha)\mathbf{h}_i^k + \alpha \mathbf{g}_i^k$, as proved in Lemma 1. Therefore, as the iteration approaches the optimum, $\mathbf{h}_i^k$ will also approach the local gradient $\nabla f_i(\mathbf{x}^*)$ rapidly which contributes to small gradient residual and consequently small compression variance. Similar difference compression techniques are also proposed in DIANA and its variance-reduced variant (Mishchenko et al., 2019; Horváth et al., 2019).

[1]Equations in the curly bracket are just notations for the proof but does not need to computed actually.

3.2 Discussion

In this subsection, we provide more detailed discussions about DORE including model initialization, model update, the special smooth case as well as the compression rate of communication.

Initialization. It is important to take the identical initialization $\hat{\mathbf{x}}^0$ for all worker and master nodes. It is easy to be ensured by either setting the same random seed or broadcasting the model once at the beginning. In this way, although we don't need to broadcast the model parameters directly, every worker node updates the model $\hat{\mathbf{x}}^k$ in the same way. Thus we can keep their model parameters identical. Otherwise, the model inconsistency needs to be considered.

Model update. It is worth noting that although we can choose an accurate model $\mathbf{x}^{k+1}$ as the next iteration in the master node, we use $\hat{\mathbf{x}}^{k+1}$ instead. In this way, we can ensure that the gradient descent algorithm is applied based on the exact stochastic gradient which is evaluated on $\hat{\mathbf{x}}_i^k$ at each worker node. This dispels the intricacy to deal with inexact gradient evaluated on $\mathbf{x}^k$ and thus it simplifies the convergence analysis.

Smooth case. In the smooth case, i.e., $R = 0$, Algorithm 1 can be simplified. The master node quantizes the recovered averaged gradient with error compensation and broadcasts it to all worker nodes. The details of this simplified case can be found in Appendix A.4.

Compression rate. The compression of the gradient information can reduce at most 50% of the communication cost since it only considers compression during gradient aggregation while ignoring the model synchronization. However, DORE can further cut down the remaining 50% communication.

Taking the blockwise p-norm quantization as an example, every element of $\mathbf{x}$ can be represented by $\frac{3}{2}$ bits using the simple ternary coding $\{0, \pm 1\}$, along with one magnitude for each block. For example, if we consider the uniform block size b, the number of bits to represent a d-dimension vector of 32 bit float-point numbers can be reduced from $32d$ bits to $32\frac{d}{b} + \frac{3}{2}d$ bits. As long as the block size b is relatively large with respect to the constant 32, the cost $32\frac{d}{b}$ for storing the float-point number is relatively small such that the compression rate is close to $32d/(\frac{3}{2}d) \approx 21.3$ times (for example, 19.7 times when $b = 256$).

Applying this quantization, QSGD, Terngrad, MEM-SGD, and DIANA need to transmit $(32d + 32\frac{d}{b} + \frac{3}{2}d)$ bits per iteration and thus they are able to cut down 47% of the overall $2 \times 32d$ bits per iteration through gradient compression when $b = 256$. But with DORE, we only need to transmit $2(32\frac{d}{b} + \frac{3}{2}d)$ bits per iteration. Thus DORE can reduce over 95% of the total

Algorithm 1 The Proposed DORE.[1]

```
1: Input: Stepsize $\alpha, \beta, \gamma, \eta$, initialize $\mathbf{h}^0 = \mathbf{h}_i^0 = \mathbf{0}^d$, $\hat{\mathbf{x}}_i^0 = \hat{\mathbf{x}}^0$, $\forall i \in \{1, \dots, n\}$.
2: for $k = 1, 2, \cdots, K-1$ do
3:     For each worker $i \in \{1, 2, \cdots, n\}$:
4:     Sample $\mathbf{g}_i^k$ such that $\mathbb{E}[\mathbf{g}_i^k|\hat{\mathbf{x}}_i^k] = \nabla f_i(\hat{\mathbf{x}}_i^k)$
5:     Gradient residual: $\Delta_i^k = \mathbf{g}_i^k - \mathbf{h}_i^k$
6:     Compression: $\hat{\Delta}_i^k = Q(\Delta_i^k)$
7:     $\mathbf{h}_i^{k+1} = \mathbf{h}_i^k + \alpha\hat{\Delta}_i^k$
8:     $\{\ \hat{\mathbf{g}}_i^k = \mathbf{h}_i^k + \hat{\Delta}_i^k\ \}$
9:     Send $\hat{\Delta}_i^k$ to the master
10:    Receive $\hat{\mathbf{q}}^k$ from the master
11:    $\hat{\mathbf{x}}_i^{k+1} = \hat{\mathbf{x}}_i^k + \beta\hat{\mathbf{q}}^k$
12:    For the master:
13:    Receive $\{\hat{\Delta}_i^k\}$ from workers
14:    $\hat{\Delta}^k = 1/n\sum_i^n \hat{\Delta}_i^k$
15:    $\hat{\mathbf{g}}^k = \mathbf{h}^k + \hat{\Delta}^k\ \ \{= 1/n\sum_i^n \hat{\mathbf{g}}_i^k\ \}$
16:    $\mathbf{x}^{k+1} = \mathbf{prox}_{\gamma R}(\hat{\mathbf{x}}^k - \gamma\hat{\mathbf{g}}^k)$
17:    $\mathbf{h}^{k+1} = \mathbf{h}^k + \alpha\hat{\Delta}^k$
18:    Model residual: $\mathbf{q}^k = \mathbf{x}^{k+1} - \hat{\mathbf{x}}^k + \eta\mathbf{e}^k$
19:    Compression: $\hat{\mathbf{q}}^k = Q(\mathbf{q}^k)$
20:    $\mathbf{e}^{k+1} = \mathbf{q}^k - \hat{\mathbf{q}}^k$
21:    $\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \beta\hat{\mathbf{q}}^k$
22:    Broadcast $\hat{\mathbf{q}}^k$ to workers
23: end for
24: Output: $\hat{\mathbf{x}}^K$ or any $\hat{\mathbf{x}}_i^K$
```

communication by compressing both the gradient and model transmission. More efficient coding techniques such as Elias coding (Elias, 1975) can be applied to further reduce the number of bits per iteration.

4 Convergence Analysis

To show the convergence of DORE, we will make the following commonly used assumptions when needed.

Assumption 2. *Each worker node samples an unbiased estimator of the gradient stochastically with bounded variance, i.e., for $i = 1, 2, \cdots, n$ and $\forall \mathbf{x} \in \mathbb{R}^d$,*

$$\mathbb{E}[\mathbf{g}_i|\mathbf{x}] = \nabla f_i(\mathbf{x}), \quad \mathbb{E}\|\mathbf{g}_i - \nabla f_i(\mathbf{x})\|^2 \le \sigma_i^2, \tag{2}$$

where $\mathbf{g}_i$ is the estimator of ∇f_i at $\mathbf{x}$. In addition, we define $\sigma^2 = \frac{1}{n}\sum_{i=1}^n \sigma_i^2$.

Assumption 3. *Each f_i is L-Lipschitz differentiable, i.e., for $i = 1, 2, \cdots, n$ and $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,*

$$f_i(\mathbf{x}) \le f_i(\mathbf{y}) + \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2. \tag{3}$$

Assumption 4. *Each f_i is μ-strongly convex ($\mu \ge 0$), i.e., for $i = 1, 2, \cdots, n$ and $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,*

$$f_i(\mathbf{x}) \ge f_i(\mathbf{y}) + \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2. \tag{4}$$

For simplicity, we use the same compression operator for all worker nodes, and the master node can apply a different compression operator. We denote the constants in Assumption 1 as C_q and C_q^m for the worker and master nodes, respectively. Then we set α and β in both algorithms to satisfy

$$\frac{1-\sqrt{1-\frac{4C_q(C_q+1)}{nc}}}{2(C_q+1)} \le \alpha \le \frac{1+\sqrt{1-\frac{4C_q(C_q+1)}{nc}}}{2(C_q+1)}, \qquad 0 < \beta \le \frac{1}{C_q^m+1}, \tag{5}$$

with $c \ge \frac{4C_q(C_q+1)}{n}$. We consider two scenarios in the following two subsections: f is strongly convex with a convex regularizer R and f is non-convex with $R = 0$.

4.1 The strongly convex case

Theorem 1. *Under Assumptions 1-4, if α and β in Algorithm 1 satisfy (5), η and γ satisfy*

$$\eta < \min\left(\frac{-C_q^m+\sqrt{(C_q^m)^2+4(1-(C_q^m+1)\beta)}}{2C_q^m}, \frac{4\mu L}{(\mu+L)^2(1+c\alpha)-4\mu L}\right), \tag{6}$$

$$\frac{\eta(\mu+L)}{2(1+\eta)\mu L} \le \gamma \le \frac{2}{(1+c\alpha)(\mu+L)}, \tag{7}$$

then we have

$$\mathbf{V}^{k+1} \le \rho^k\mathbf{V}^1 + \frac{(1+\eta)(1+nc\alpha)}{n(1-\rho)}\beta\gamma^2\sigma^2, \tag{8}$$

with

$$\mathbf{V}^k = \beta(1-(C_q^m+1)\beta)\mathbb{E}\|\mathbf{q}^{k-1}\|^2 + \mathbb{E}\|\hat{\mathbf{x}}^k - \mathbf{x}^*\|^2 + \frac{(1+\eta)c\beta\gamma^2}{n}\sum_{i=1}^n \mathbb{E}\|\mathbf{h}_i^k - \nabla f_i(\mathbf{x}^*)\|^2,$$

$$\rho = \max\left(\frac{(\eta^2+\eta)C_q^m}{1-(C_q^m+1)\beta}, 1+\eta\beta - \frac{2(1+\eta)\beta\gamma\mu L}{\mu+L}, 1-\alpha\right) < 1.$$

Corollary 1. *When there is no error compensation and we set $\eta = 0$, then $\rho = \max(1 - \frac{2\beta\gamma\mu L}{\mu+L}, 1-\alpha)$. If we further set*

$$\alpha = \frac{1}{2(C_q+1)}, \quad \beta = \frac{1}{C_q^m+1}, \quad c = \frac{4C_q(C_q+1)}{n}, \tag{9}$$

and choose the largest step-size $\gamma = \frac{2}{(\mu+L)(1+2C_q/n)}$, the convergent factor is

$$(1-\rho)^{-1} = \max\left(2(C_q+1), (C_q^m+1)\frac{(\mu+L)^2}{2\mu L}\left(\frac{1}{2}+\frac{C_q}{n}\right)\right). \tag{10}$$

Remark 1. *In particular, suppose $\{\Delta_i\}_{i=1}^n$ are compressed using the Bernoulli p-norm quantization with the largest block size $d_{\max}$, then $C_q = \frac{1}{\alpha^w} - 1$, with $\alpha^w = \min_{\mathbf{0}\ne\mathbf{x}\in\mathbb{R}^{d_{\max}}} \frac{\|\mathbf{x}\|_2^2}{\|\mathbf{x}\|_1\|\mathbf{x}\|_p} \le 1$. Similarly, $\mathbf{q}$ is compressed using the Bernoulli p-norm quantization with $C_q^m = \frac{1}{\alpha^m} - 1$. Then the linear convergent factor is*

$$(1-\rho)^{-1} = \max\left\{\frac{2}{\alpha^w}, \frac{1}{\alpha^m}\frac{(\mu+L)^2}{\mu L}\left(\frac{1}{2}-\frac{2}{n}+\frac{2}{n\alpha^w}\right)\right\}. \tag{11}$$

While the result of DIANA in (Mishchenko et al., 2019) is $\max\left\{\frac{2}{\alpha^w}, \frac{\mu+L}{\mu}\left(\frac{1}{2}-\frac{1}{n}+\frac{1}{n\alpha^w}\right)\right\}$, *which is better than* (11) *with* $\alpha^m = 1$ *(no compression for the model). When there is no compression for* Δ_i, *i.e.,* $\alpha^w = 1$, *the algorithm reduces to the gradient descent, and the linear convergent factor is the same as that of the gradient descent for strongly convex functions.*

Remark 2. *Although error compensation often improves the convergence empirically, in theory, no compensation, i.e.,* $\eta = 0$, *provides the best convergence rate. This is because we don't have much information of the error being compensated. Filling this gap will be an interesting future direction.*

4.2 The nonconvex case with $R = 0$

Theorem 2. *Under Assumptions 1-3 and the additional assumption that each worker samples the gradient from the full dataset, we set* α *and* β *according to* (5)*. By choosing*

$$\gamma \leq \min\left\{\frac{-1+\sqrt{1+\frac{48L^2\beta^2(C_q^m+1)^2}{C_q^m}}}{12L\beta(C_q^m+1)}, \frac{1}{6L\beta(1+c\alpha)(C_q^m+1)}\right\},$$

we have

$$\frac{\frac{\beta}{2}-3(1+c\alpha)(C_q^m+1)L\beta^2\gamma}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 \leq \frac{\Lambda^1-\Lambda^{K+1}}{\gamma K} + \frac{3(C_q^m+1)(1+nc\alpha)L\beta^2\sigma^2\gamma}{n}, \quad (12)$$

where

$$\Lambda^k = (C_q^m+1)L\beta^2\|\mathbf{q}^{k-1}\|^2 + f(\hat{\mathbf{x}}^k) - f^* + 3c(C_q^m+1)L\beta^2\gamma^2\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\|\mathbf{h}_i^k\|^2. \quad (13)$$

Corollary 2. *Let* $\alpha = \frac{1}{2(C_q+1)}, \beta = \frac{1}{C_q^m+1}$, *and* $c = \frac{4C_q(C_q+1)}{n}$, *then* $1 + nc\alpha$ *is a fixed constant. If* $\gamma = \frac{1}{12L(1+c\alpha)(1+\sqrt{K/n})}$, *when* K *is relatively large, we have*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 \lesssim \frac{1}{K} + \frac{1}{\sqrt{Kn}}. \quad (14)$$

Remark 3. *The dominant term in* (14) *is* $O(1/\sqrt{Kn})$, *which implies that the sample complexity of each worker node is* $O(1/(n\epsilon^2))$ *in average to achieve an* ϵ*-accurate solution. It shows that, same as DoubleSqueeze in Tang et al. (2019), DORE is able to perform linear speedup. Furthermore, this convergence result is the same as the P-SGD without compression. Note that DoubleSqueeze has an extra term* $(1/K)^{\frac{2}{3}}$, *and its convergence requires the bounded variance of the compression operator.*

5 Experiment

In this section, we validate the theoretical results and demonstrate the superior performance of DORE. Our experimental results demonstrate that (1) DORE achieves similar convergence speed as full-precision SGD and state-of-art quantized SGD baselines and (2) its iteration time is much smaller than most existing algorithms, supporting the superior communication efficiency of DORE.

To make a fair comparison, we choose the same Bernoulli ∞-norm quantization as described in Section 3 and the quantization block size is 256 for all experiments if not being explicitly stated because ∞-norm quantization is unbiased and commonly used. The parameters α, β, η for DORE are chosen to be $0.1, 1$ and 1, respectively.

The baselines we choose to compare include SGD, QSGD (Alistarh et al., 2017), MEM-SGD (Stich et al., 2018), DIANA (Mishchenko et al., 2019), DoubleSqueeze and DoubleSqueeze (topk) (Tang et al., 2019). SGD is the vanilla SGD without any compression and QSGD quantizes the gradient directly. MEM-SGD is the QSGD with error compensation. DIANA, which only compresses and transmits the gradient difference, is a special case of the proposed DORE. DoubleSqueeze quantizes both the gradient on the workers and the averaged gradient on the server with error compensation. Although DoubleSqueeze is claimed to work well with both biased and unbiased compression, in our experiment it converges much slower and suffers the loss of accuracy with unbiased compression. Thus, we also compare with DoubleSqueeze using the Top-k compression as presented in Tang et al. (2019).

5.1 Strongly convex

To verify the convergence for strongly convex and smooth objective functions, we conduct the experiment on a linear regression problem: $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda\|\mathbf{x}\|^2$. The data matrix $\mathbf{A} \in \mathbb{R}^{1200\times 500}$ and optimal solution $\mathbf{x}_* \in \mathbb{R}^{500}$ are randomly synthesized. Then we generate the prediction $\mathbf{b}$ by sampling from a Gaussian distribution whose mean is $\mathbf{A}\mathbf{x}_*$. The rows of the data matrix $\mathbf{A}$ are allocated evenly to 20 worker nodes. To better verify the linear convergence to the $\mathcal{O}(\sigma)$ neighborhood around the optimal solution, we take the full gradient in each node for all algorithms to exclude the effect of the gradient variance ($\sigma = 0$).

As showed in Figure 3, with full gradient and a constant learning rate, DORE converges linearly, same as SGD and DIANA, but QSGD, MEM-SGD, DoubleSqueeze, as well as DoubleSqueeze (topk) converge to a neighborhood of the optimal point. This is be-

Algorithm	Compression	Compression Assumed	Linear rate	Nonconvex Rate
QSGD	Grad	2-norm Quantization	N/A	$\frac{1}{K} + B$
DIANA	Grad	p-norm Quantization	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$
DoubleSqueeze	Grad+Model	Bounded Variance	N/A	$\frac{1}{\sqrt{Kn}} + \frac{1}{K^{2/3}} + \frac{1}{K}$
DORE	Grad+Model	Assumption 1	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$

Table 1: A comparison between related algorithms. DORE is able to converges linearly to the $\mathcal{O}(\sigma)$ neighborhood of optimal point like full-precision SGD and DIANA in the strongly convex case while achieving much better communication efficiency. DORE also admits linear speedup in the nonconvex case like DoubleSqueeze but DORE doesn't require the assumptions of bounded compression error or bounded gradient.

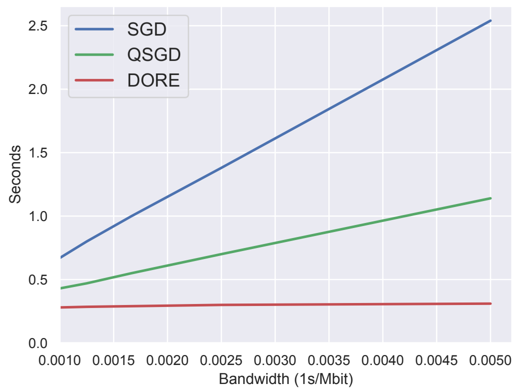


Figure 2: Per iteration time cost on Resnet18 for SGD, QSGD, and DORE. It is tested in a shared cluster environment connected by Gigabit Ethernet interface. DORE speeds up the training process significantly by mitigating the communication bottleneck.

cause these algorithms assume the bounded gradient and their convergence errors depend on that bound. Although they converge to the optimal solution using a diminishing step size, their converge rates will be much slower.

In addition, we also validate that the norms of the gradient and model residual decrease exponentially, which explains the linear convergence behavior of DORE. For more details, please refer to Appendix A.1.

5.2 Nonconvex

To verify the convergence in the nonconvex case, we test the proposed DORE with two classical deep neural networks on two representative datasets, respectively, i.e., LeNet (Lecun et al., 1998) on MNIST and Resnet18 (He et al., 2016) on CIFAR10. In the experiment, we use 1 parameter server and 10 workers, each of which is equipped with an NVIDIA Tesla K80 GPU. The batch size for each worker node is 256. We use 0.1 and 0.01 as the initial learning rates for LeNet and Resnet18, and decrease them by a factor of 0.1 after every 25 and 100 epochs, respectively. All parameter settings are the same for all algorithms.

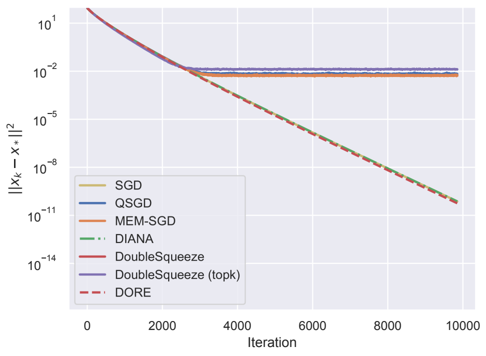
Figures 4 and 5 show the training loss and test loss for each epoch during the training of LeNet on the MNIST dataset and Resnet18 on CIFAR10 dataset. The results indicate that in the nonconvex case, even with both compressed gradient and model information, DORE can still achieve similar convergence speed as full-precision SGD and other quantized SGD variants. DORE achieves much better convergence speed than DoubleSqueeze using the same compression method and converges similarly with DoubleSqueeze with Topk compression as presented in (Tang et al., 2019). We also validate via parameter sensitivity in Appendix A.3 that DORE performs consistently well under different parameter settings such as compression block size, α, β and η.

5.3 Communication efficiency

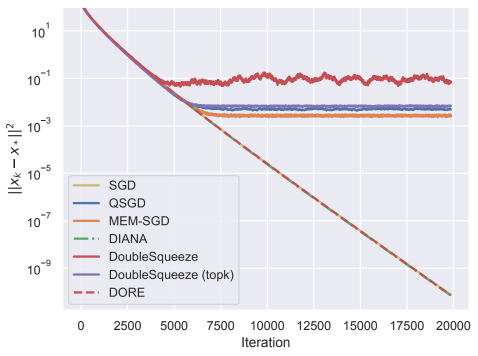
In terms of communication cost, DORE enjoys the benefit of extremely efficient communication. As one example, under the same setting as the Resnet18 experiment described in the previous section, we test the time cost per iteration for SGD, QSGD, and DORE under varied network bandwidth. We didn't test MEM-SGD, DIANA, and DoubleSqueeze because MEM-SGD, DIANA have similar time cost as QSGD while DoubleSqueeze has similar time cost as DORE. The result showed in Figure 2 indicates that as the bandwidth becomes worse, with both gradient and model compression, the advantage of DORE becomes more remarkable compared to the baselines that don't apply compression for model synchronization. In Appendix A.2, we also demonstrate the communication efficiency in terms of communication bits and running time, which clearly suggests the benefit of the proposed algorithm.

6 Conclusion

Communication cost is the severe bottleneck for distributed training of modern large-scale machine learning models. Extensive works have compressed the gradient information to be transferred during the training

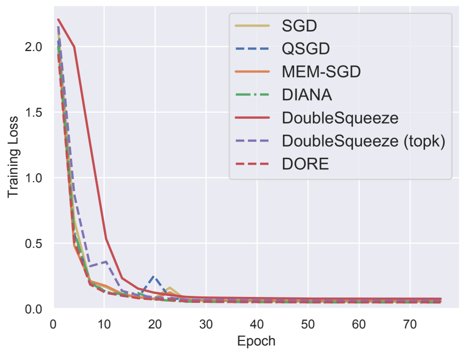


(a) Learning rate=0.05

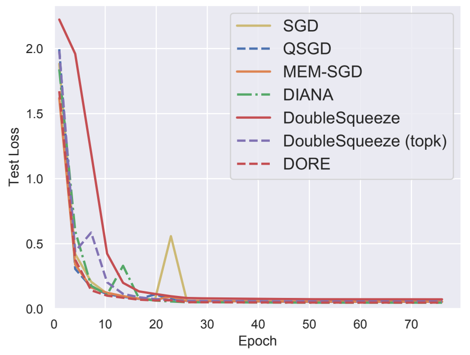


(b) Learning rate=0.025

Figure 3: Linear regression on synthetic data. When the learning rate is 0.05, DoubleSqueeze diverges. In both cases, DORE, SGD, and DIANA converge linearly to the optimal point, while QSGD, MEM-SGD, DoubleSqueeze, and DoubleSqueeze (topk) only converge to the neighborhood even when full gradient is available.

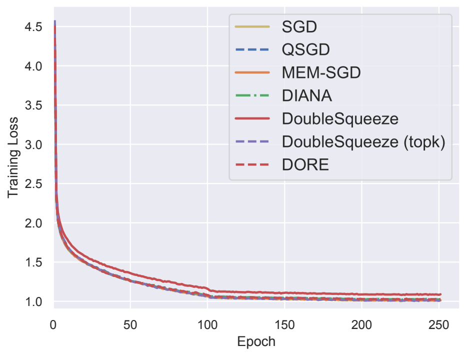


(a) Training loss

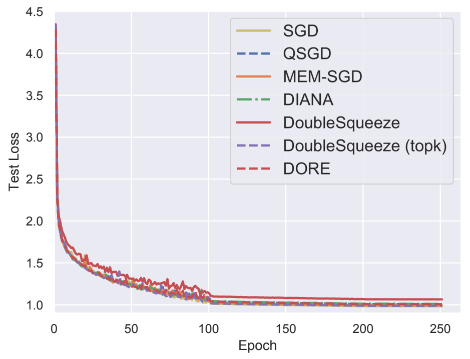


(b) Test Loss

Figure 4: LeNet trained on MNIST. DORE converges similarly as most baselines. It outperforms DoubleSqueeze using the same compression method while has similar performance as DoubleSqueeze (topk).



(a) Training Loss



(b) Test Loss

Figure 5: Resnet18 trained on CIFAR10. DORE achieves similar convergence and accuracy as most baselines. DoubeSuqeeze converges slower and suffers from the higher loss but it works well with topk compression.

process, but model compression is rather limited due to its intrinsic difficulty. In this paper, we proposed the Double Residual Compression SGD named DORE to compress both gradient and model communication that can mitigate this bottleneck prominently. The theoretical analyses suggest good convergence rate of DORE under weak assumptions. Furthermore, DORE is able to reduce 95% of the communication cost while maintaining similar convergence rate and model accuracy compared with the full-precision SGD.

7 Acknowledgement

Xiaorui Liu and Jiliang Tang are supported by the National Science Foundation (NSF) under grant numbers IIS-1714741, IIS-1715940, IIS-1845081, IIS-1907704, IIS-1928278 and CNS-1815636 and a grant from Criteo Faculty Research Award. Yao Li and Ming Yan are supported by the National Science Foundation (NSF) under grant number DMS-1621798.

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. (2016). TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, OSDI'16, pages 265–283.

Aji, A. F. and Heafield, K. (2017). Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445, Copenhagen, Denmark. Association for Computational Linguistics.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. (2017). QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720.

Bernstein, J., Wang, Y., Azizzadenesheli, K., and Anandkumar, A. (2018). SIGNSGD: compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 559–568.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Lechevallier, Y. and Saporta, G., editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg. Physica-Verlag HD.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015). MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274.

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. (2012). Large scale distributed deep networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1223–1231, USA.

Elias, P. (1975). Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203.

Gower, R. M., Loizou, N., Qian, X., Sailanbayev, A., Shulgin, E., and Richtárik, P. (2019). SGD: General analysis and improved rates. *arXiv preprint arXiv:1901.09401.*

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Horváth, S., Kovalev, D., Mishchenko, K., Stich, S., and Richtárik, P. (2019). Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115.*

Jordan, M. I., Lee, J. D., and Yang, Y. (2019). Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, 114(526):668–681.

Karimireddy, S. P., Rebjock, Q., Stich, S. U., and Jaggi, M. (2019). Error feedback fixes SignSGD and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3252–3261. PMLR.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. (2014). Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 583–598, Berkeley, CA, USA. USENIX Association.

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

Mishchenko, K., Gorbunov, E., Takáč, M., and Richtárik, P. (2019). Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269.*

Nguyen, L., NGUYEN, P. H., van Dijk, M., Richtarik, P., Scheinberg, K., and Takac, M. (2018). SGD and hogwild! Convergence without the bounded gradients assumption. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3750–3758, Stockholmsmässan, Stockholm Sweden. PMLR.

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

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. (2018). Sparsified SGD with memory. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4452–4463, USA. Curran Associates Inc.

Strom, N. (2015). Scalable distributed DNN training using commodity GPU cloud computing. In *INTERSPEECH*, pages 1488–1492.

Tang, H., Gan, S., Zhang, C., Zhang, T., and Liu, J. (2018). Communication compression for decentralized training. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 7663–7673.

Tang, H., Yu, C., Lian, X., Zhang, T., and Liu, J. (2019). DoubleSqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6155–6165.

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

Wangni, J., Wang, J., Liu, J., and Zhang, T. (2018). Gradient sparsification for communication-efficient distributed optimization. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 1306–1316, USA. Curran Associates Inc.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. (2017). TernGrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pages 1509–1519.

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

You, Y., Zhang, Z., Hsieh, C.-J., Demmel, J., and Keutzer, K. (2018). ImageNet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, ICPP 2018, pages 1:1–1:10, New York, NY, USA. ACM.

Zinkevich, M., Weimer, M., Li, L., and Smola, A. J. (2010). Parallelized stochastic gradient descent. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 2595–2603. Curran Associates, Inc.