Trading Redundancy for Communication: Speeding up Distributed SGD for Non-convex Optimization

Farzin Haddadpour [1] Mohammad Mahdi Kamani [2] Mehrdad Mahdavi [1] Viveck R. Cadambe [1]

Abstract

Communication overhead is one of the key challenges that hinders the scalability of distributed optimization algorithms to train large neural networks. In recent years, there has been a great deal of research to alleviate communication cost by compressing the gradient vector or using local updates and periodic model averaging. In this paper, we advocate the use of redundancy towards communication-efficient distributed stochastic algorithms for non-convex optimization. In particular, we, both theoretically and practically, show that by properly infusing redundancy to the training data with model averaging, it is possible to significantly reduce the number of communication rounds. To be more precise, we show that redundancy reduces residual error in local averaging, thereby reaching the same level of accuracy with fewer rounds of communication as compared with previous algorithms. Empirical studies on CIFAR10, CIFAR100 and ImageNet datasets in a distributed environment complement our theoretical results; they show that our algorithms have additional beneficial aspects including tolerance to failures, as well as greater gradient diversity.

1. Introduction

The Stochastic Gradient Descent (SGD) is a fundamental tool for solving numerous optimization problems in machine learning. SGD is an iterative method in which the model parameters $\mathbf{x}^{(t)}, \mathbf{x}^{(t+1)}$ respectively at the tth and $(t+1)$st iteration are related as:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta\tilde{\mathbf{g}}^{(t)}, \tag{1}$$

where $\tilde{\mathbf{g}}^{(t)}$ is the stochastic gradient of the cost function. Requiring immense computational resources, large scale machine learning models such as deep neural networks which use SGD are, nowadays, trained in distributed systems on several computation nodes.

In particular, the dataset $\mathcal{D}$ is partitioned into shards and distributed to computation nodes. Each node evaluates the partial gradients over a randomly sampled mini-batch from its shard. In fully synchronous SGD, the computation nodes, after evaluation of gradients over the sampled mini-batch, exchanges their updates in every iteration to ensure that all nodes have the same updated model. Thus, implementing fully synchronous SGD to update the model as (1) incurs significant amount of communication both in terms of number of rounds and data exchange. This communication cost is, in fact, among the primary obstacles towards scaling distributed SGD.

A promising strand of research that aims at mitigating communication bottlenecks in large scale SGD is model averaging, where each node locally updates the model sequentially on different data partitions, and the models of the different nodes are averaged periodically (McDonald et al., 2010; Wang & Joshi, 2018; Yu et al., 2018; Zhang et al., 2016; Zhou & Cong, 2017; Zinkevich et al., 2010). Because of local updates, the model averaging approach reduces the number of rounds of communication in training and can therefore be much faster in practice. However, since the model for every iteration is not updated based on the entire data, it suffers from a residual error with respect to fully synchronous SGD; this residual error lowers the accuracy in training for model averaging. In this paper, we develop a new approach towards model averaging by infusing redundancy in the data stored at the computation nodes. Our approach retains the benefits of model averaging - lower number of rounds of communication - and yet leads to improved model accuracy via residual error reduction.

In summary, the main contributions of the present work are as follows:

- We present in Section 3, the Redundancy-Infused SGD (RI-SGD) algorithm, which is a variation of distributed SGD. In RI-SGD, the data is partitioned into shards; each node has its own shard of data, and is allowed to partially access the shards of a subset of other workers

[1] School of Electrical Engineering and Computer Science [2] College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA, USA. Correspondence to: Farzin Haddadpour <fxh18@psu.edu>.

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

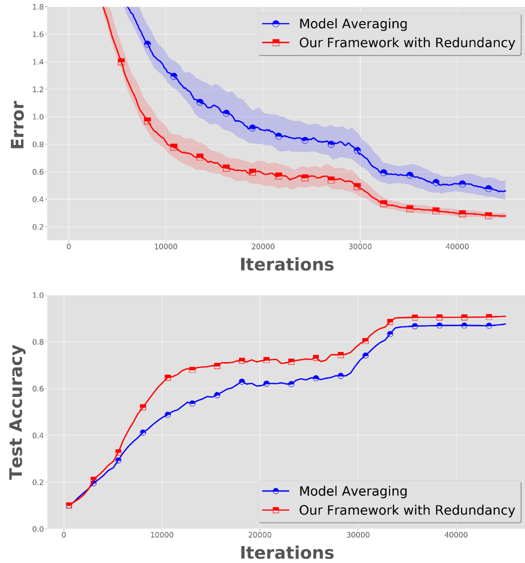

Figure 1. Training CIFAR10 using ResNet44 on 4 parallel GPUs, using a generic Model Averaging introduced in (Yu et al., 2018), and our framework with 25% redundancy infused. In both runs the number for local updates is 50. For more details, see the experiment setup in Section 5.

to the level of imposed redundancy budget (either explicitly by data replication or implicitly by sampling data from other workers' shard). RI-SGD operates by performing local updates and periodically averaging the models across the nodes.

- In Section 4, we perform a convergence analysis of RI-SGD for non-convex Lipschitz-continuous loss functions. Under certain assumptions, our bounds show that the error converges inversely proportional to the number of local updates. Importantly, for a fixed number of number of iterations and local updates, our algorithm can be designed to have lower residual error as compared with the previous algorithms where there is no redundancy. The main technical contribution of our paper is an error analysis in distributed local SGD with non-identical and non-uniform sampling at the different nodes. As a special case, this includes the practical case where data is partitioned among the computation nodes, and each node draws mini-batches from the locally stored data. The important consequences of our analysis are (a) the role of diversity in the residual error, (b) the role of redundancy reducing this error to speed up convergence and (c) fault tolerance.

- In Section 5, we demonstrate the accuracy and convergence properties empirically - see for e.g., Fig. 1. Our results show that with some redundancy, both training and testing can be significantly more accurate as compared with conventional algorithms for the same communication. Depending on the minibatch size, RI-SGD can incur an additional computation cost due to the redundancy in the data. Despite this, for a fixed level of accuracy, our experiments demonstrate that RI-SGD trains faster (lower wall clock times) as compared with the no-redundancy case. This speed up is due to communication being a more severe bottleneck than computation. Furthermore, even when we reduce the minibatch size of RI-SGD inversely proportional to the amount of redundancy to make its computation burden equal to the no-redundancy counterpart, our experiments reveal that RI-SGD can still lead to increased accuracy due to greater diversity (see Section 5 for more details). We also show a positive byproduct of redundancy in the form of increased fault-tolerance in our paper.

2. Related work

The primary approach to distribute the training process across several workers is the parameter-server framework (Cui et al., 2014; Dean et al., 2012; Li et al.). These methods suffer from synchronization delay and high communication cost. A line of work is to mitigate the synchronization delay via asynchronous model update methods (Cui et al., 2014; Gupta et al., 2016; Mitliagkas et al., 2016; Recht et al., 2011). These methods, though faster than synchronized methods, lead to convergence error issues due to stale gradients. This line of work is orthogonal to our work as we focus on communication issues.

Communication bottlenecks in distributed SGD manifest in two ways; first, the number of bits communicated can be significant, and second, the number of rounds of communication can be large. Several studies aim to minimize the number of transmitted bits in each iteration by *quantizing* (Alistarh et al., 2017; Bernstein et al., 2018; Seide et al., 2014; Wangni et al., 2018) or *sparsifying* (Aji & Heafield, 2017; Dryden et al., 2016; Strom, 2015) the gradient vector. Another approach to reducing communication is *decentralized parallel SGD* (Jiang et al., 2017; Jin et al., 2016; Lian et al., 2017), which performs decentralized training with sparse-connected network of computation nodes. All these methods are complementary to ours, as we focus on approaches that have fewer number of communication rounds relative to the number of iterations. Similar to us, references (Haddadpour et al., 2018; Ye & Abbe, 2018) use redundancy to reduce communication costs. However their technical content is orthogonal to ours since they focus on exact gradient descent and quadratic cost functions.

We focus on model averaging methods where nodes to perform updates locally based on their own model. These methods were first introduced under the name of *one-shot* *averaging* in (McDonald et al., 2010; Zinkevich et al., 2010), wherein they show numerically that model averaging works well in several optimization problems. Model averaging is shown to experimentally work well in (Chen & Huo, 2016; Kamp et al., 2018; Lin et al., 2018; McMahan et al., 2016; Povey et al., 2014; Su & Chen, 2015) by reducing communication costs; it is also shown to improve privacy and security in the federated learning (McMahan et al., 2016). However, it is also known (Zhang et al., 2016) that one-shot averaging leads to inaccurate models in non-convex optimization settings. The study of the benefits and trade-offs in model averaging, in particular its convergence properties, have been explored from a theoretical/analytical viewpoint in (Chaudhari et al., 2017; Wang & Joshi, 2018; Yu et al., 2018; Zhang et al., 2015; Zhou & Cong, 2017).

Periodic averaging, a generalization of one-shot averaging where the models are averaged across nodes afters every τ local updates; here we focus on a common periodic averaging technique sometimes referred to as parallel restarted SGD (PR-SGD)[1] (Yu et al., 2018). Reference (Zhou & Cong, 2017) demonstrated that the convergence error of PR-SGD grows with the number of local updates. This method is empirically shown to work well in (Chaudhari et al., 2017). Recently, (Yu et al., 2018) improved the convergence rate of (Zhou & Cong, 2017) showing that the same convergence rate as fully synchronous SGD can be achieved by properly tuning the number of local updates. Nonetheless, in (Yu et al., 2018; Zhou & Cong, 2017), the residual error grows quadratically in τ. Recently, (Wang & Joshi, 2018) demonstrated that the residual error can be linear in the number of local updates τ, if every node can sample mini-batches from the entire data, rather than their own locally stored partition. Interestingly, (Wang & Joshi, 2018) also removed the bounded gradient assumption which was used in (Yu et al., 2018; Zhou & Cong, 2017).

Our algorithm can be viewed as a redundancy-infused variant of PR-SGD discussed before. In our work, we show a residual error that grows linearly in τ similar to (Wang & Joshi, 2018). Importantly, we do not assume that each node draws samples from the entire data. Rather, we make the more practical assumption that each node only samples from the partitions that it stores locally. We discuss further comparisons of our work with (Wang & Joshi, 2018; Zhou & Cong, 2017) in Appendix C.

3. RI-SGD: Redundancy Infused SGD

Consider a setting with p distributed *worker* nodes, each of which has a disjoint chunk of data, $\mathcal{D}_j$ for $1 \le j \le p$. We are interested in solving the following optimization problem in a distributed manner

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) \triangleq \frac{1}{p}\sum_{j=1}^{p} f(\mathbf{x};\mathcal{D}_j), \tag{2}$$

where the function $f(\cdot;\mathcal{D}_j)$ is the training loss over the samples in jth chunk $\mathcal{D}_j$. Recall that a commonly studied centralized approach towards solving this optimization problem is mini-batch stochastic gradient descent (SGD) for randomly sampled mini-batches $\xi^{(t)} \subset \mathcal{D} = \{\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_p\}$, which requires the same update rule as in (1), where $\tilde{\mathbf{g}}^{(t)} = \frac{1}{|\xi^{(t)}|}\nabla f(\mathbf{x}^{(t)};\xi^{(t)})$ is the stochastic unbiased gradient over a uniformly sampled mini-batch $\xi^{(t)} \subseteq \mathcal{D}$ from all the data samples $\mathcal{D}$. Here we study a decentralized approach to the above problem in homogeneous redundancy setting and defer the heterogeneous setting to Appendix D.

In homogeneous sample size setting, we assume that the data budget at each worker node is $\gamma \triangleq \frac{q}{p}$, where q is a predetermined integer satisfying $p \ge q \ge 1$ and p is the total number of chunks of the data set as well as the number of workers. The assumption that q is an integer is a mathematical simplification, as it is indicated in the experiments with non-integer choice of q. We assume that the jth worker receives $\mathcal{D}_j$ and $q-1$ other distinct chunks of data from $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{j-1}, \mathcal{D}_{j+1}, \ldots \mathcal{D}_p$. We define the redundancy factor as $\mu \triangleq \frac{q-1}{p} = \gamma - \frac{1}{p}$. We denote the data chunks at the jth worker as $\tilde{\mathcal{D}}_j \triangleq \{\mathcal{D}_{j,1}, \ldots, \mathcal{D}_{j,q}\}$ where $1 \le j \le p$ and $\mathcal{D}_{j,\ell} \in \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_p\}, \ell = 1, 2, \ldots, q$. We assume that the qth data chunk at each worker j is its own local data $\mathcal{D}_{j,q} = \mathcal{D}_j$. In our algorithm, denoted by RI-SGD(τ) where τ represents the number of local updates, at iteration t worker $j, 1 \le j \le p$ samples mini-batches $\xi_{j,i}^{(t)}$ identically and independently from data chunk $\mathcal{D}_{j,i}$ for $1 \le i \le q$. Then the samples are used to compute the stochastic gradient as follows:

$$\tilde{\mathbf{g}}_{j,i}^{(t)} = \nabla f(\mathbf{x}_j^{(t)};\xi_{j,i}^{(t)})\ ,\ \tilde{\mathbf{g}}_j^{(t)} \triangleq \sum_{i=1}^{q} \tilde{\mathbf{g}}_{j,i}^{(t)}, \tag{3}$$

Next, each worker, updates its own local version of the model $\mathbf{x}_j^{(t)}$ using $\mathbf{x}_j^{(t+1)} = \mathbf{x}_j^{(t)} - \eta\, \tilde{\mathbf{g}}_j^{(t)}$. After every τ iterations, we do the model averaging, where we average local versions of the model in all p workers. The pseudocode of RI-SGD is shown in Algorithm 1. We note that the introduced algorithm does not necessarily imply data replication in practice, although data replication leads to one natural way of implementing the algorithm. Specifically, as long as every worker can sample mini-batch samples from the data chunk of other workers (e.g., via network communication), our algorithm and results apply.

A few remarks and comparison with related works. The case of $\tau = 1$ corresponds to one round of communication for every iteration. The case of $\tau > 1$ is referred to

[1]The term PR-SGD also helps distinguish the technique we study from *elastic-averaging SGD* (Zhang et al., 2015), where nodes perform a weighted average rather than a simple average.

as periodic averaging in literature, and leads to lower communication cost albeit, with a residual error in the model. In our results in the next section, we show that the residual error grows linearly with τ, the number of local updates.The case of $\gamma = 1/p$ corresponds to PR-SGD (Yu et al., 2018); this algorithm is shown to have a residual error that is quadratic in τ. Thus, our analysis can be viewed as an improvement of the analysis of (Yu et al., 2018). In addition, we can reduce the residual error by tuning the parameter γ.

If $\gamma = 1$, every node has access to all the chunks. This case is comparable to (Wang & Joshi, 2018) as well to fully synchronous mini-batch SGD where $\tau = 1$, and the workers sample mini-batches from the entire data in every iteration. However, a subtle difference is that in our approach, we sample the data differently. The workers first sample mini-batches the various partitions of the data and then perform updates on the various mini-batches. Thus our approach leads to the samples being more uniformly spread across the different partitions across different iterations; the effect of this difference is explored empirically in Section 5. Similar to (Wang & Joshi, 2018), we show that our residual error only linear in τ, even if $\gamma < 1$ meaning that the workers do not have access to the entire data. The main technical difference between the $\gamma = 1$ case (e.g., (Wang & Joshi, 2018)) and the $\gamma < 1$ case is that in the latter case, since each worker samples a mini-batches from the its data chunks, instead of entire dataset, the mini-batch gradients are biased. This technical difference is the main challenge we overcome in our proofs of convergence analysis.

Algorithm 1 RI-SGD(τ): Redundancy Infused SGD

Offline phase:
The worker j receives set of random chunks $\tilde{\mathcal{D}}_j \triangleq \{\mathcal{D}_{j,1}, \dots, \mathcal{D}_{j,q}\}$, where $1 \leq j \leq p$ and $\mathcal{D}_{j,k} \neq \mathcal{D}_{j,l}$ for $k \neq l$ and $1 \leq k, l \leq q$ and $\mathcal{D}_{j,q} = \mathcal{D}_j$ for $1 \leq j \leq p$.
Online-phase:
  Inputs: $\{\tilde{\mathcal{D}}_1, \dots, \tilde{\mathcal{D}}_p\}$ and $\mathbf{x}^{(1)}$ as an initial global model.
  for $t = 1, 2, \dots, T$ **do**
    parallel for $j = 1, 2, \dots, p$ **do**
      j-th worker forms the set of mini-batches as $\tilde{\xi}_j^{(t)} \triangleq \{\xi_{j,1}^{(t)}, \dots, \xi_{j,q}^{(t)}\}$, where $\xi_{j,k}^{(t)}, k = 1, \dots, q$ is uniformly sampled from kth chunk
      Evaluates stochastic gradient over each mini-batches, $\tilde{\mathbf{g}}_{j,i}^{(t)}$ for $i = 1, \dots, q$, and computes $\tilde{\mathbf{g}}_j^{(t)}$ as in (3)
      if t divides τ **do**
        $\mathbf{x}_j^{(t+1)} = \frac{1}{p}\sum_{j=1}^{p}\left[\mathbf{x}_j^{(t)} - \eta\, \tilde{\mathbf{g}}_j^{(t)}\right]$
      else do
        $\mathbf{x}_j^{(t+1)} = \mathbf{x}_j^{(t)} - \eta\, \tilde{\mathbf{g}}_j^{(t)}$
      end if
    end parallel for
end
Output: $\bar{\mathbf{x}}^{(T)} = \frac{1}{p}\sum_{j=1}^{p}\mathbf{x}_j^{(T)}$

4. Convergence Analysis

In this section, we present the convergence analysis of the proposed algorithm. We first state the main assumptions and then present the convergence rates. All the proofs are deferred to the appendix. In what follows, we use $\mathbf{g}_j \triangleq \sum_{i=1}^{q}\mathbf{g}_{j,i}$ to denote the full batch gradients over all q data chunks at jth worker, where $\mathbf{g}_{j,i} \triangleq \nabla f(\cdot; \mathcal{D}_{j,i})$ is the full gradient over ith chunk.

4.1. Assumptions

Our convergence analysis is based on the following standard assumptions in non-convex optimization.

Assumption 1 (Unbiased estimation over chunks)**.** *The stochastic gradient evaluated on a mini-batch $\xi_{j,i} \subset \mathcal{D}_{j,i}$ is an unbiased estimator of the partial full gradient, i.e. $\mathbb{E}_{\xi_{j,i}}\left[\tilde{\mathbf{g}}_{j,i}|\mathbf{x}_j\right] = \mathbf{g}_{j,i}$ for $1 \leq j \leq p$ and $1 \leq i \leq q$.*

Assumption 2 (Lipschitz continuity and lower boundedness)**.** *The objective function $F(\mathbf{x})$ is differentiable and L-smooth, i.e., $\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. The function value is bounded below by a scalar F^*.*

Assumption 3 (Bounded variance)**.** *The variance of stochastic gradients evaluated on a mini-batch $\xi_{j,i}$ for all $1 \leq j \leq p$ and $1 \leq i \leq q$ is bounded as*

$$\mathbb{E}_{\xi_{j,i}}\left[\|\tilde{\mathbf{g}}_{j,i} - \mathbf{g}_{j,i}\|^2|\right] \leq C_1\|\mathbf{g}_{j,i}\|^2 + \frac{C_2^2}{p} \tag{4}$$

where C_1 and C_2 are non-negative constants and inversely proportion to the mini-batch size.

Remark 1. *We note that, as p increases, the data chunk size reduces, and therefore, the mini-batch size increases relative the the size of the data chunk. As a result, the variance reduces; this justifies the assumption that the variance is inversely proportional to p. For further clarification, suppose that total number of the data points is n and let the data chunk at each worker node have $\frac{n}{p}$ data points. Then, let $p = \frac{n}{b}$ where is the mini-batch size at each worker node. In this case, each data chunk at workers will have $\frac{n}{p} = b$ data points and therefore $\tilde{\mathbf{g}}_{j,i} = \mathbf{g}_{j,i}$. Therefore, for larger p and fixed number of the data points the variance reduces.*

We make the following assumption about the partial full gradients among different workers.

Assumption 4. *We assume that the local full gradients over pair of chunks for every pair of workers satisfies $|\langle \mathbf{g}_{j,i}, \mathbf{g}_{j',i'}\rangle| \leq \beta, \quad \forall 1 \leq i, j, i', j' \leq p, i \neq i', j \neq j'$.*

We note that Assumption 4 is easily implied by the assumption of uniformly bounded gradient. To see this, we note that if $\|\mathbf{g}_{j,i}\| \leq \sqrt{\beta}$ holds, a simple application of Cauchy-Schwartz inequality implies that $|\langle \mathbf{g}_{j,i}, \mathbf{g}_{j',i'}\rangle| \leq \|\mathbf{g}_{j,i}\|\|\mathbf{g}_{j',i'}\| = \beta$. However, Assumption 4 does not necessarily imply uniformly bounded gradient assumption.

Finally, we note that since $F(\mathbf{x})$ is a non-convex function, SGD may converge to a local minimum or a saddle point. We say the algorithm achieves an ϵ-submoptimal (Alistarh et al., 2017; Ghadimi & Lan, 2013; Lian et al., 2017) solution if

$$\mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^{T}\|\nabla F(\bar{\mathbf{x}}^{(t)})\|^2\Big] \leq \epsilon,$$

where $\bar{\mathbf{x}}^{(t)}$ is average model at tth iteration. This condition guarantees convergence of the algorithm to a stationary point, which will be used to show the convergence rate.

4.2. Main results

We now turn to provide the convergence analysis for Algorithm 1. To illustrate our key technical contribution in convergence analysis, let us define an auxiliary variable $\bar{\mathbf{x}}^{(t)} = \frac{1}{p}\sum_{j=1}^{p}\mathbf{x}_j^{(t)}$, which is the average model across p different workers at iteration t. Using the definition of $\bar{\mathbf{x}}^{(t)}$, the update rule in Algorithm 1, becomes:

$$\bar{\mathbf{x}}^{(t+1)} = \bar{\mathbf{x}}^{(t)} - \eta\Big[\frac{1}{p}\sum_{j=1}^{p}\tilde{\mathbf{g}}_j^{(t)}\Big] \tag{5}$$

As mentioned earlier, when $\gamma < 1$, the mini-batch stochastic gradients used in local updates are a biased estimate of the full gradient. Specifically, the update rule in (5) can be re-written as

$$\bar{\mathbf{x}}^{(t+1)} = \bar{\mathbf{x}}^{(t)} - \eta\nabla F(\bar{\mathbf{x}}^{(t)}) + \eta\Big[\nabla F(\bar{\mathbf{x}}^{(t)}) - \frac{1}{p}\sum_{j=1}^{p}\tilde{\mathbf{g}}_j^{(t)}\Big],$$

which establishes a connection between our algorithm and the perturbed SGD with deviation $\Big(\nabla F(\bar{\mathbf{x}}^{(t)}) - \frac{1}{p}\sum_{j=1}^{p}\tilde{\mathbf{g}}_j^{(t)}\Big)$. We show that by introducing data redundancy among workers, we can reduce the variance of biased gradients and obtain the desired convergence rates.

We start by stating the convergence rate of RI-SGD(τ) algorithm.

Theorem 4.1. *For RI-SGD(τ), under Assumptions 1 - 4, if the learning rate satisfies* $\eta^2L^2\Big(2\tau C_1(\frac{p+1}{p}) + \tau(\tau-1)\Big) + \frac{\eta L(C_1+\gamma p^2)}{p} \leq 1$ *and all local model parameters are initialized at the same point* $\mathbf{x}^{(1)}$*, for the given budget* $\frac{1}{p} \leq \gamma \leq 1$ *the average-squared gradient after* T *iterations is bounded by:*

$$\begin{aligned}\mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^{T}\|\nabla F(\bar{\mathbf{x}}^{(t)})\|^2\Big] &\leq \frac{2\big[F(\bar{\mathbf{x}}^{(1)}) - F^*\big]}{\eta T} + \eta L C_2^2\frac{\gamma}{p}\\ &+ \gamma(\frac{p+1}{p})\eta^2L^2(\tau-1)C_2^2 + (\gamma-1)\frac{1}{T}\sum_{t=1}^{T}\tilde{\mathbf{G}}^{(t)}\\ &+ \frac{p^2}{2}\Big[3\gamma^2 - 4\gamma + (3-\frac{1}{p})\Big]\beta,\end{aligned} \tag{6}$$

where $\tilde{\mathbf{G}}^{(t)} \triangleq \sum_{j=1}^{p}\tilde{\mathbf{G}}_j^{(t)}$ *and* $\tilde{\mathbf{G}}_j^{(t)} \triangleq \min_{\mathcal{D}_i\in\mathcal{D}\setminus\tilde{\mathcal{D}}_j}\|\mathbf{g}_{j,i}^{(t)}\|^2$ *for* $1 \leq j \leq p$.

A few remarks regarding the convergence rate in Theorem 4.1 are in place. First, the obtained bound depends on the diversity of local gradients parameter β which is also illustrated in our experimental results as well, in particular Figure 3. Specifically, the rate increases with β, which means that for larger values of β, more rounds of communication are needed to achieve the same error. This is consistent with the result of (Yin et al., 2018), because the larger β implies smaller gradient diversity. We also note that due to our data allocation policy where each worker samples from multiple chunks of data instead of sampling from the entire dataset, the possibility of sampling the same data by multiple worker nodes decreases, leading to faster convergence rate.

Corollary 4.1.1. *The learning rate condition in Theorem 4.1 implies that* $\eta \leq \frac{1}{L\sqrt{2\tau C_1(\frac{p+1}{p})+\tau(\tau-1)}}$*. Therefore, for the larger* τ *we need to choose smaller learning rate.*

The following theorem strengthens Theorem 4.1 under a mild condition on the gradients, which specializes to the fully synchronous distributed SGD in case of full data redundancy.

Theorem 4.2. *For RI-SGD(τ), if for all* $1 \leq t \leq T$*, there exists constants* $\rho^{(t)} > 0$ *such that* $\sum_{j=1}^{p}\|\sum_i \mathbf{g}_{j,i}^{(t)} + \sum_i \mathbf{g}_{j,i}^{(t)}\|^2 \geq \rho^{(t)}\sum_{j=1}^{p}\Big(\sum_i\|\mathbf{g}_{j,i}^{(t)}\|^2\Big)$*, under Assumptions 1-4, the learning rate satisfies* $\eta^2L^2\Big(2\tau C_1(\frac{p+1}{p}) + \tau(\tau-1)\Big) + \frac{\eta L(C_1+\gamma p^2)}{p} < \rho^{(t)}$ *for* $1 \leq t \leq T$ *and all local model parameters are initialized at the same point* $\bar{\mathbf{x}}^{(1)}$*, for the given budget* $\frac{1}{p} \leq \gamma \leq 1$ *the average-squared gradient after* τ *iterations is bounded as follows:*

$$\begin{aligned}\mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^{T}\|\nabla F(\bar{\mathbf{x}}^{(t)})\|^2\Big] &\leq \frac{2\big[F(\bar{\mathbf{x}}^{(1)}) - F^*\big]}{\eta T} + \eta L C_2^2\frac{\gamma}{p}\\ &+ (\frac{p+1}{p})\gamma C_2^2\eta^2L^2(\tau-1) + 2p^2\Big[(1-\gamma)\Big]\beta\end{aligned} \tag{7}$$

Remark 2. *In Theorem 4.2, if we let* $\tau = 1$ *and* $\gamma = 1$*, we get the convergence error terms reduce to the convergence* *error of fully synchronous SGD (Bottou et al., 2018). If we let $\gamma = 1$ (i.e. full redundancy), the error bound in (7) reduces to the error bound provided by (Wang & Joshi, 2018) which is proved to achieve a convergence rate of $O(\frac{1}{\sqrt{pT}})$. Therefore, by letting $\gamma = 1 - O(\frac{1}{p^2\sqrt{pT}})$, and $\eta = \frac{\sqrt{p}}{L\sqrt{T}}$, we can achieve the convergence rate of $O(\frac{1}{\sqrt{pT}})$. This is shown more formally in Appendix G. On the other hand, if we let $\gamma = \frac{1}{p}$, (7) gives the error for the case where there is no redundancy and each worker applies local updates.*

These theoretical bounds are validated through experimental results in Section 5 where we compare RI-SGD to PR-SGD algorithm ($\mu = 0$) proposed in (Yu et al., 2018) and (Wang & Joshi, 2018), in which it is assumed that each worker can sample its mini-batches from the entire dataset. The detailed comparison of convergence of various algorithm can be found in Appendix C.

4.3. Distribution-aware convergence

Based on Assumption 4, we consider an upper bound β for the correlation between each pair of local full gradients of each data chunk $\mathbf{g}_{j,i}$. Hence, convergence rates presented in Theorems 4.1 and 4.2 do not explicitly reflect the effects of each pair individually. In order to further elaborate on the impact of distribution of data chunks on the final bound we need to bound each pair independently. To that end, we introduce an upper bound on the correlation between gradients of the main chunk at the jth worker (i.e. $\mathbf{g}_j = \nabla f(\cdot, \mathcal{D}_{j,q})$, recalling that the qth data chunk is local data) and gradients of data chunks not present in the jth worker (i.e. $\underline{\mathbf{g}}_{j,i} = \nabla f(\cdot, \mathcal{D}_i), \mathcal{D}_i \notin \tilde{\mathcal{D}}_j$), as stated below.

Assumption 5. *For $1 \le j \le p$ we assume an upper bound on the inner products of gradients of each worker's data chunk and data chunks not presented in that worker as follows:*

$$|\langle \mathbf{g}_j, \underline{\mathbf{g}}_{j,i} \rangle| \le \beta_{j,i}, \tag{8}$$

We note that by setting $\max_{j,i} \beta_{j,i} = \beta$, we can get the results which only depends on the number of mini-batches rather than the chunks. In Appendix F we derive the convergence of RI-SGD based on Assumption 5, where the convergence is stated in terms of gradient dissimilarity between different shards of data.

4.4. Robustness to node failure

While performing distributed optimization using iterative methods such as distributed fully synchronous scheme, worker nodes have to exchange their gradients. If at each round of communication certain *straggling* workers have delays (or failed) in their computational task, this can cause huge delay in entire optimization process. One popular approach (Chen et al., 2016) to deal with straggler nodes is to ignore them. Our single and multi-round approach can outperform a strategy that ignores stragglers by tuning the degree of redundancy. To see this, let us assume the number of failures is s. If we ignore failures in algorithm proposed in (Wang & Joshi, 2018), the convergence error upper bound becomes $\mathcal{E}(\tau, s) \triangleq \frac{2\left[F(\mathbf{x}^{(1)}) - F^*\right]}{\eta(\tau+T)} + \frac{\eta L C_2^2}{p-s} + \eta^2 L^2 C_2^2(\tau - 1)$ Now, if we consider the convergence error of RI-SGD(τ), we can see that it introduces the additional parameter γ, and ignoring the stragglers in this algorithm results in convergence error

$$\mathcal{E}_{\text{RI-SGD}}(\tau, s, \gamma) = \frac{2\left[F(\mathbf{x}^{(1)}) - F^*\right]}{\eta T} + \eta L C_2^2 \frac{\gamma}{(p-s)} + \frac{(\gamma-1)}{p-s}\frac{p}{T}\sum_{t=1}^{T}\tilde{\mathbf{G}}^{(t)} + \frac{q(p-s+1)}{(p-s)^2}\eta^2 C_2^2 L^2(\tau - 1) + \frac{p^2}{2}\left[3\gamma^2 - 4\gamma + (3 - \frac{1}{p})\right]\beta$$

where $\tilde{\mathbf{G}}^{(t)} = \sum_{j=1}^{p-s} \tilde{\mathbf{G}}_j$. Then, by proper choice of $\gamma = \gamma_s$, we can satisfy $\mathcal{E}(\tau, s) > \mathcal{E}_{\text{RI-SGD}}(\tau, s, \gamma_s)$, which indicates that taking advantage of the data redundancy, RI-SGD(τ) is resilient against failures.

4.5. Redundancy and intra-node gradient diversity

As we discussed in the convergence of RI-SGD, by inducing redundancy among workers, we can effectively reduce the variance in local updates and reduce the number of communication rounds and yet enjoying faster convergence. We can also investigate the effectiveness of redundancy infusion from gradient diversity perspective introduced in (Yin et al., 2018). In Appendix E, we mathematically demonstrate the intimate connection between redundancy and gradient diversity and show that under mild assumptions, infusing redundancy increases the intra-node gradient diversity– the effective quantity for the mini-batch size used for local updates at each worker node. A direct implication is that the effective size of mini-batch can be increased without any saturation or decay in performance.

5. Experiments

In this section, we experimentally evaluate the performance of our RI-SGD algorithm with parameters redundancy $\mu = \gamma - 1/p = \frac{q-1}{p}$ and step size τ. We compare our algorithm with fully synchronous SGD (SyncSGD), ($\tau = 1, \mu = 0$), PR-SGD ($\mu = 0$, variable τ) and cooperative SGD (Wang & Joshi, 2018), where they consider each node has access to the whole dataset for mini-batch sampling. We first describe the setup in which we conduct the experiment.

5.1. Experimental setup

We conduct different types of experiments on clusters with 4 and 8 GPUs (Tesla $V100$ and GeForce GTX 1080 Ti), to

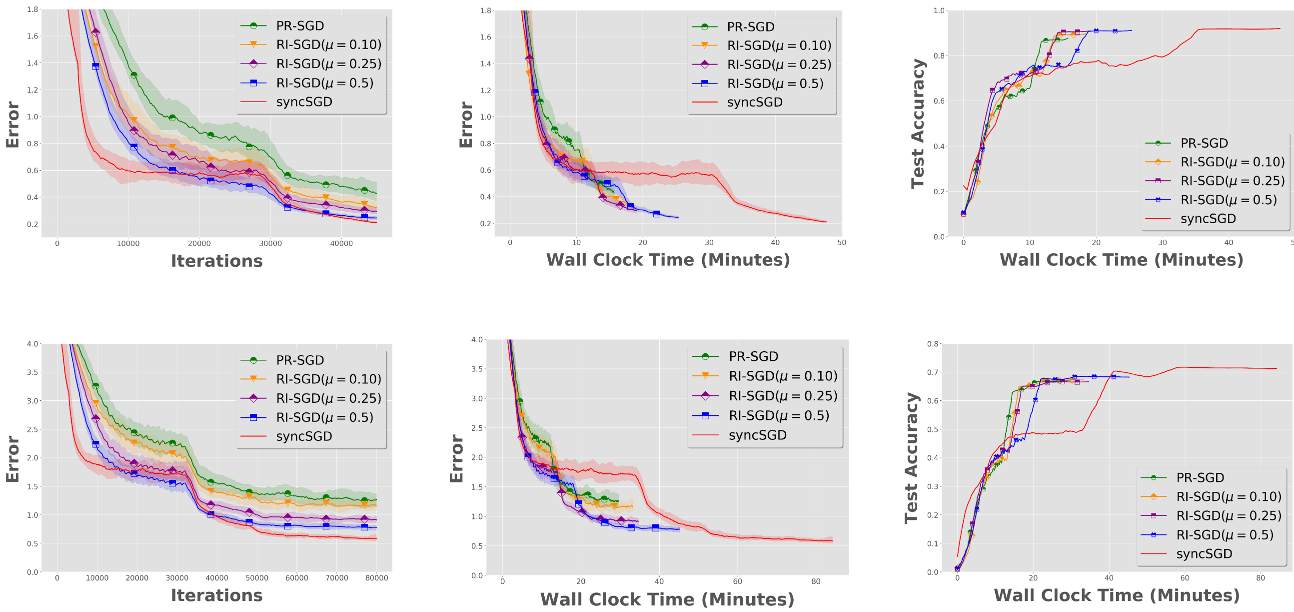

Figure 2. Training error and test accuracy for CIFAR10 and CIFAR100 on ResNet using syncSGD and RI-SGD with $\mu \in \{0.0, 0.1, 0.25\}$ and $\tau = 50$ for both datasets. Note that RI-SGD($\mu = 0$) is PR-SGD. Top row is for CIFAR10 dataset, and bottom row is for CIFAR100 dataset.

test the performance of the algorithm. We use CIFAR10 and CIFAR100[2], as well as ImageNet[3] datasets, and choose ResNet (He et al., 2016) as the base model for training. We use Tensorflow (Abadi et al., 2016) to develop RI-SGD and Sync-SGD in the distributed environment. In each experiment, the batch size for original Sync-SGD is 128 (unless otherwise stated), and we will do the training on $45K$ and $80K$ iterations for CIFAR10 and CIFAR100 datasets, respectively. For ImageNet dataset, we do the training on both 4 and 8 GPUs, with batch size of the Sync-SGD set to 128 and 256, and for $1.1M$ and $550K$ iterations, respectively. We present the results of experiments on CIFAR datasets on the next section, and you can find the results on ImageNet dataset on Appendix A. All the experiments are based on heterogeneous mini-batch samples introduced in Section D. Hence, the redundancy value for each worker node is picked from a normal distribution with the mean of the reported redundancy value in the experiments (except for $\mu = 0$ or no redundancy).

5.2. Experimental results

For RI-SGD we have two hyperparameters we can tune, the redundancy μ and number of local updates τ. One exploration is to see the effect of redundancy alone without any averaging step in the middle of training. This scenario has been explained in Appendix A, and the results indicate that redundancy can, indeed, improve the convergence rate with much less time compared to SyncSGD. After that, we should find the optimum number for local updates, τ. This number introduces a trade-off in training, in which as it goes up, the accuracy goes down, however the number of communication rounds and hence the time for execution reduces. Based on this experiment's results in Appendix A, it seems that in CIFAR datasets, this number should be around $\tau = 50$ to have the optimal trade-off.

Now, in the main experiment, we will set $\tau = 50$ for both datasets. We change the redundancy with values $\mu \in \{0.0, 0.1, 0.25, 0.5\}$ in order to see its effects on the convergence. In our experiment we replicate data chunks cyclically, however, in general we can follow any replication strategy as long as the redundancy budget is satisfied for all workers. For instance, in our setup because we have 4 GPUS, in $\mu = 0.25$, the nodes have $\{\mathcal{D}_1, \mathcal{D}_2\}, \{\mathcal{D}_2, \mathcal{D}_3\}, \{\mathcal{D}_3, \mathcal{D}_4\}, \{\mathcal{D}_4, \mathcal{D}_1\}$ respectively. For $\mu = 0.1$, the redundant part is obtained as random sub-sampling, e.g., the first node has $\mathcal{D}_1$ and a random subset of $\mathcal{D}_2$ to ensure that $\mu = 0.1$. As it is shown in Figure 2, by increasing redundancy rate we get quite close to the baseline SyncSGD in terms of convergence, in an accelerated rate. We beat SyncSGD, not only in training, but also in test accuracy. We reach almost the same accuracy level way sooner. Note that significant improvements in speed of training and testing occur even though RI-SGD has higher computation cost, due to significantly lower time spent in communication.

[2] Available at `https://www.cs.toronto.edu/~kriz/cifar.html`

[3] Available at `https://www.kaggle.com/c/imagenet-object-localization-challenge`

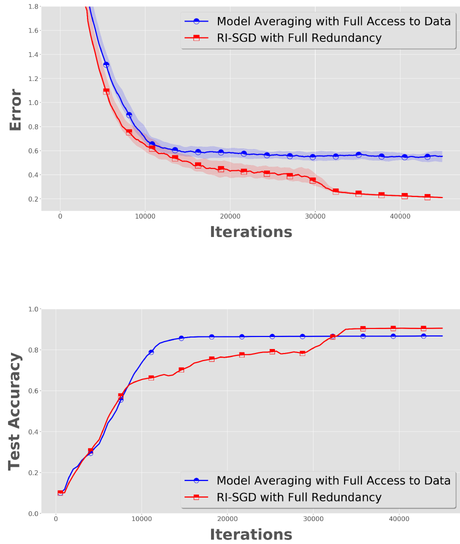

Figure 3. Training CIFAR10 using ResNet44 with $\tau = 50$ for both cases. Comparing the case, where each node has access to the entire dataset versus RI-SGD with full redundancy. Redundancy can lead to a better performance by introducing diversity in training data input. Note that in this experiment, both runs have the batch size of 512.

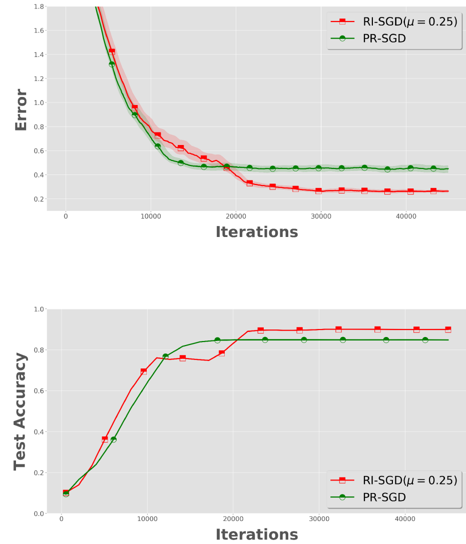

Figure 4. Training CIFAR10 using ResNet44 with $\tau = 50$ for both cases. Comparing PR-SGD with RI-SGD($\mu = 0.25$). Redundancy can lead to a better performance by introducing diversity using redundancy in training data input and escape the saturation in training. Note that in this experiment, both runs have the batch size of 512.

In the next experiment, we show that redundancy can be beneficial in introducing gradient diversity (Yin et al., 2018) in training. To that end, we compare RI-SGD with two different extreme cases of distributed local SGD. These two extreme cases are when each node has a full access to the dataset and samples mini-batches from the entire dataset (e.g. cooperative SGD (Wang & Joshi, 2018)) or each node has only access to its chunk of the data for mini-batch sampling (e.g. PR-SGD (Yu et al., 2018)). For the first one, we compare it with RI-SGD with full redundancy, where each node not only has access to its portion of data, but also can get mini-batches from every other node's data chunk. Note that the main difference between the two schemes is the nature of the sampling; in RI-SGD, each node's data chunk contributes to the mini-batch equally, while in cooperative SGD we randomly sample mini-batches from the entire dataset. In this systematic redundancy infusion regime of RI-SGD using non-i.i.d. sampling, the chances of having overlap in input data for different nodes are less than the former case. In Figure 3, it is shown that RI-SGD with full redundancy can outperform the simple model averaging with every node having access to all data. To compare RI-SGD with PR-SGD, which is basically RI-SGD with $\mu = 0$, we choose mini-batch size to be 512 and then compare it with RI-SGD($\mu = 0.25$), with the same mini-batch size. As it is depicted in Figure 4, redundancy can help to escape saturation by introducing gradient diversity, hence, can achieve a better performance.

6. Conclusion

In this paper, we advocate the use of data redundancy in periodic model averaging techniques for distributed SGD. Through a theoretical convergence proof, we show that redundancy reduces residual error as compared with conventional algorithms where there is no redundancy. Through experimental results, we show that the redundancy, is well worth the implied higher computation cost. The residual error reduction leads to lower communication overheads and therefore faster convergence. Furthermore, we also demonstrate empirically that our redundancy infused approach has higher diversity, leading to superior algorithms as compared to the previous ones where nodes can sample from the complete data.

This work leaves some interesting directions as future work. First, it would be interesting to investigate the significance of redundancy in federated optimization to reduce sensitivity of SGD to the variance in data. Furthermore, understanding more effective replication schemes, e.g. sketching methods, when data in different workers has some low-rank structure is worthy of investigation.

Acknowledgement

This work was partially supported by the NSF CCF 1553248 and NSF CCF 1763657 grants. The first author would like to thank Dr. Kang Wook Lee to bring the paper (McMahan et al., 2016) to his attention.

References

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, 2016.

Aji, A. F. and Heafield, K. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

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

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. Signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 559–568, 2018.

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

Chaudhari, P., Baldassi, C., Zecchina, R., Soatto, S., Talwalkar, A., and Oberman, A. Parle: parallelizing stochastic gradient descent. *arXiv preprint arXiv:1707.00424*, 2017.

Chen, J., Pan, X., Monga, R., Bengio, S., and Jozefowicz, R. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

Chen, K. and Huo, Q. Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 5880–5884. IEEE, 2016.

Cui, H., Cipar, J., Ho, Q., Kim, J. K., Lee, S., Kumar, A., Wei, J., Dai, W., Ganger, G. R., Gibbons, P. B., et al. Exploiting bounded staleness to speed up big data analytics. In *USENIX Annual Technical Conference*, pp. 37–48, 2014.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012.

Dryden, N., Moon, T., Jacobs, S. A., and Van Essen, B. Communication quantization for data-parallel training of deep neural networks. In *Machine Learning in HPC Environments (MLHPC), Workshop on*, pp. 1–8. IEEE, 2016.

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

Gupta, S., Zhang, W., and Wang, F. Model accuracy and runtime tradeoff in distributed deep learning: A systematic study. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pp. 171–180. IEEE, 2016.

Haddadpour, F., Yang, Y., Cadambe, V., and Grover, P. Cross-iteration coded computing. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 196–203. IEEE, 2018.

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.

Jiang, Z., Balu, A., Hegde, C., and Sarkar, S. Collaborative deep learning in fixed topology networks. In *Advances in Neural Information Processing Systems*, pp. 5904–5914, 2017.

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

Kamp, M., Adilova, L., Sicking, J., Hüger, F., Schlicht, P., Wirtz, T., and Wrobel, S. Efficient decentralized deep learning by dynamic model averaging. *arXiv preprint arXiv:1807.03210*, 2018.

Li, M., Andersen, D. G., and Park, J. W. Scaling distributed machine learning with the parameter server.

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.

Lin, T., Stich, S. U., and Jaggi, M. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018.

McDonald, R., Hall, K., and Mann, G. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 456–464. Association for Computational Linguistics, 2010.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Mitliagkas, I., Zhang, C., Hadjis, S., and Ré, C. Asynchrony begets momentum, with an application to deep learning. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, pp. 997–1004. IEEE, 2016.

Povey, D., Zhang, X., and Khudanpur, S. Parallel training of dnns with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*, 2014.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pp. 693–701, 2011.

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Strom, N. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Su, H. and Chen, H. Experiments on parallel training of deep neural network using model averaging. *arXiv preprint arXiv:1507.01239*, 2015.

Wang, J. and Joshi, G. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1306–1316, 2018.

Ye, M. and Abbe, E. Communication-computation efficient gradient coding. *arXiv preprint arXiv:1802.03475*, 2018.

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1998–2007, 2018.

Yu, H., Yang, S., and Zhu, S. Parallel restarted sgd for non-convex optimization with faster convergence and less communication. *arXiv preprint arXiv:1807.06629*, 2018.

Zhang, J., De Sa, C., Mitliagkas, I., and Ré, C. Parallel sgd: When does averaging help? *arXiv preprint arXiv:1606.07365*, 2016.

Zhang, S., Choromanska, A. E., and LeCun, Y. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, pp. 685–693, 2015.

Zhou, F. and Cong, G. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. *arXiv preprint arXiv:1708.01012*, 2017.

Zinkevich, M., Weimer, M., Li, L., and Smola, A. J. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pp. 2595–2603, 2010.