Parallel Restarted SGD with Faster Convergence and Less Communication: Demystifying Why Model Averaging Works for Deep Learning

Hao Yu, Sen Yang, Shenghuo Zhu
Machine Intelligence Technology Lab, Alibaba Group (U.S.) Inc., Bellevue, WA.

Abstract

In distributed training of deep neural networks, parallel mini-batch SGD is widely used to speed up the training process by using multiple workers. It uses multiple workers to sample local stochastic gradients in parallel, aggregates all gradients in a single server to obtain the average, and updates each worker's local model using a SGD update with the averaged gradient. Ideally, parallel mini-batch SGD can achieve a linear speed-up of the training time (with respect to the number of workers) compared with SGD over a single worker. However, such linear scalability in practice is significantly limited by the growing demand for gradient communication as more workers are involved. Model averaging, which **periodically** averages individual models trained over parallel workers, is another common practice used for distributed training of deep neural networks since (Zinkevich et al. 2010) (McDonald, Hall, and Mann 2010). Compared with parallel mini-batch SGD, the communication overhead of model averaging is significantly reduced. Impressively, tremendous experimental works have verified that model averaging can still achieve a good speed-up of the training time as long as the averaging interval is carefully controlled. However, it remains a mystery in theory why such a simple heuristic works so well. This paper provides a thorough and rigorous theoretical study on why model averaging can work as well as parallel mini-batch SGD with significantly less communication overhead.

Introduction

Consider the distributed training of deep neural networks over multiple workers (Dean et al. 2012), where all workers can access all or partial training data and aim to find a common model that yields the minimum training loss. Such a scenario can be modeled as the following distributed parallel non-convex optimization

$$\min_{\mathbf{x}\in\mathbb{R}^m} \quad f(\mathbf{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}) \tag{1}$$

where N is the number of nodes/workers and each $f_i(\mathbf{x}) \triangleq \mathbb{E}_{\zeta_i\sim\mathcal{D}_i}[F_i(\mathbf{x};\zeta_i)]$ is a smooth non-convex function where $\mathcal{D}_i$ can be possibly different for different i. Following the standard stochastic optimization setting, this paper assumes each worker can locally observe unbiased independent stochastic gradients (around the last iteration solution $\mathbf{x}_i^{t-1}$) given by $\mathbf{G}_i^t = \nabla F_i(\mathbf{x}_i^{t-1};\zeta_i^t)$ with $\mathbb{E}_{\zeta_i^t\sim\mathcal{D}_i}[\mathbf{G}_i^t|\boldsymbol{\zeta}^{[t-1]}] = \nabla f_i(\mathbf{x}_i^{t-1}), \forall i$ where $\boldsymbol{\zeta}^{[t-1]} \triangleq [\zeta_i^\tau]_{i\in\{1,2,\ldots,N\},\tau\in\{1,\ldots,t-1\}}$ denotes all the randomness up to iteration $t-1$.

One classical parallel method to solve problem (1) is to sample each worker's local stochastic gradient in parallel, aggregate all gradients in a single server to obtain the average, and update each worker's local solution using the averaged gradient in its SGD step[1] (Dekel et al. 2012) (Li et al. 2014). Such a classical method, called **parallel mini-batch SGD** in this paper, is conceptually equivalent to a single node Stochastic Gradient Descent (SGD) with a batch size N times large and achieves $O(1/\sqrt{NT})$ convergence with a linear speed-up with respect to (w.r.t.) the number of workers (Dekel et al. 2012). Since every iteration of parallel mini-batch SGD requires exchanging of local gradient information among all workers, the corresponding communication cost is heavy and often becomes a performance bottleneck.

There have been many attempts to reduce communication overhead in parallel mini-batch SGD. One notable method called **decentralized parallel SGD (D-PSGD)** is studied in (Lian et al. 2017)(Jiang et al. 2017) (Lian et al. 2018). Remarkably, D-PSGD can achieve the same $O(1/\sqrt{NT})$ convergence rate as parallel mini-batch SGD, i.e., the linear speed-up w.r.t. the number of workers is preserved, without requiring a single server to collect stochastic gradient information from local workers. However, since D-PSGD requires each worker to exchange their local solutions/gradients with its neighbors at **every** iteration, the total number of communication rounds in D-PSGD is the same as that in parallel mini-batch SGD. Another notable method to reduce communication overhead in parallel mini-batch SGD is to let each worker use compressed gradients rather than raw gradients for communication. For example, **quantized**

[1] Equivalently, we can let the server update its solution using the averaged gradient and broadcast this solution to all local workers. Another equivalent implementation is to let each worker take a single SGD step using its own gradient and send the updated local solution to the server; let the server calculate the average of all workers' updated solutions and refresh each worker's local solution with the averaged version.

SGD studied in (Seide et al. 2014)(Alistarh et al. 2017)(Wen et al. 2017) or **sparsified SGD** studied in (Strom 2015)(Dryden et al. 2016)(Aji and Heafield 2017) allow each worker to pass low bit quantized or sparsified gradients to the server at **every** iteration by sacrificing the convergence to a mild extent. Similarly to D-PSGD, such gradient compression based methods require message passing at every iteration and hence their total number of communication rounds is still the same as that in parallel mini-batch SGD.

Recall that parallel mini-batch SGD can be equivalently interpreted as a procedure where at each iteration each local worker first takes a single SGD step and then replaces its own solution by the average of individual solutions. With a motivation to reduce the number of inter-node communication rounds, a lot of works suggest to reduce the frequency of averaging individual solutions in parallel mini-batch SGD. Such a method is known as **model averaging** and has been widely used in practical training of deep neural networks. Model averaging can at least date back to (Zinkevich et al. 2010) (McDonald, Hall, and Mann 2010) where individual models are averaged only at the last iteration before which all workers simply run SGD in parallel. The method in (Zinkevich et al. 2010) (McDonald, Hall, and Mann 2010), referred to as **one-shot averaging**, uses only one single communication step at the end and is numerically shown to have good solution quality in many applications. However, it is unclear whether the one-shot averaging can preserve the linear speed-up w.r.t. the number of workers. In fact, (Zhang et al. 2016) shows that one-shot averaging can yield inaccurate solutions for certain non-convex optimization. As a remedy, (Zhang et al. 2016) suggests more frequent averaging should be used to improve the performance. However, the understanding on how averaging frequency can affect the performance of parallel SGD is quite limited in the current literature. Work (Zhou and Cong 2017) proves that by averaging local worker solutions only every I iterations, parallel SGD has convergence rate $O(1/\sqrt{(N/I)T})$ for non-convex optimization. That is, the convergence slows down by a factor of I by saving I times inter-node communication.[2] A recent exciting result reported in (Stich 2018) proves that for strongly-convex minimization, model averaging can achieve a linear speed-up w.r.t. N as long as the averaging (communication) step is performed once at least every $I = O(\sqrt{T}/\sqrt{N})$ iterations. Work (Stich 2018) provides the first theoretical analysis that demonstrates the possibility of achieving the same linear speedup attained by parallel mini-batch SGD with strictly less communication for **strongly-convex** stochastic optimization. However, it remains as an open question in (Stich 2018) whether it is possible to achieve $O(1/\sqrt{NT})$ convergence for non-convex optimization, which is the case of deep learning.

On the other hand, many experimental works (Povey, Zhang, and Khudanpur 2015) (Chen and Huo 2016) (McMahan et al. 2017) (Su, Chen, and Xu 2018) (Kamp et al. 2018) (Lin, Stich, and Jaggi 2018) observe that model averaging can achieve a superior performance for various deep learning applications. One may be curious whether these positive experimental results are merely coincidences for special case examples or can be attained universally. In this paper, we shall show that model averaging indeed can achieve $O(1/\sqrt{NT})$ convergence for non-convex optimization by averaging only every $I = O(T^{1/4}/N^{3/4})$ iterations. That is, the same $O(1/\sqrt{NT})$ convergence is preserved for non-convex optimization while communication overhead is saved by a factor of $O(T^{1/4}/N^{3/4})$. To our knowledge, this paper is the first[3] to present provable convergence rate guarantees (with the linear speed-up w.r.t. number of workers and less communication) of model averaging for non-convex optimization such as deep learning and provide guidelines on how often averaging is needed without losing the linear speed-up.

Besides reducing the communication cost, the method of model averaging also has the advantage of reducing privacy and security risks in the **federated learning** scenario recently proposed by Google in (McMahan et al. 2017). This is because model averaging only passes deep learning models, which preserve good differential privacy, and does not pass raw data or gradients owned by each individual worker.

Parallel Restarted SGD and Its Performance Analysis

Throughout this paper, we assume problem (1) satisfies the following assumption.

Assumption 1.

1. **Smoothness:** *Each function $f_i(\mathbf{x})$ is smooth with modulus L.*
2. **Bounded variances and second moments:** *There exits constants $\sigma > 0$ and $G > 0$ such that*

$$\mathbb{E}_{\zeta_i \sim \mathcal{D}_i} \|\nabla F_i(\mathbf{x}; \zeta_i) - \nabla f_i(\mathbf{x})\|^2 \leq \sigma^2, \forall \mathbf{x}, \forall i$$

$$\mathbb{E}_{\zeta_i \sim \mathcal{D}_i} \|\nabla F_i(\mathbf{x}; \zeta_i)\|^2 \leq G^2, \forall \mathbf{x}, \forall i$$

Consider the simple parallel SGD described in Algorithm 1. If we divide iteration indices into epochs of length I, then in each epochs all N workers are running SGD in parallel with the same initial point $\overline{\mathbf{y}}$ that is the average of final individual solutions from the previous epoch. This is why we call Algorithm 1 "**Parallel Restarted SGD**". The "model averaging" technique used as a common practice for training deep neural networks can be viewed as a special case since Algorithm 1 calculates the model average to obtain $\overline{\mathbf{y}}$ every I iterations and performs local SGDs at each worker otherwise. Such an algorithm is different from elastic averaging SGD (EASGD) proposed in (Zhang, Choromanska,

[2] In this paper, we shall show that if I is chosen as $I = O(T^{1/4}/N^{3/4})$, parallel SGD for non-convex optimization does not sacrifice any factor in its convergence rate.

[3] After the preprint (Yu, Yang, and Zhu 2018) of this paper is posted on ArXiv in July 2018, another work (Wang and Joshi 2018) subsequently analyzes the convergence rate of model averaging for non-convex optimization. Their independent analysis relaxes our bounded second moment assumption but further assumes all $f_i(\mathbf{x})$ in formulation (1) are identical, i.e, all workers must access a common training set when training deep neural networks.

Algorithm 1 Parallel Restarted SGD (PR-SGD)

Input: Initialize $\mathbf{x}_i^0 = \overline{\mathbf{y}} \in \mathbb{R}^m$. Set learning rate $\gamma > 0$ and node synchronization interval (integer) $I > 0$
for $t = 1$ to T **do**
  Each node i observes an unbiased stochastic gradient $\mathbf{G}_i^t$ of $f_i(\cdot)$ at point $\mathbf{x}_i^{t-1}$
  if t is a multiple of I, i.e., $t \bmod I = 0$, **then**
    Calculate node average $\overline{\mathbf{y}} \triangleq \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{t-1}$
    Each node i in parallel updates its local solution

$$\mathbf{x}_i^t = \overline{\mathbf{y}} - \gamma \mathbf{G}_i^t, \quad \forall i \tag{2}$$

  else
    Each node i in parallel updates its local solution

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} - \gamma \mathbf{G}_i^t, \quad \forall i \tag{3}$$

  end if
end for

and LeCun 2015) which periodically drags each local solution towards their average using a controlled weight. Note that synchronization (of iterations) across N workers is not necessary inside each epoch of Algorithm 1. Furthermore, inter-node communication is only needed to calculate the initial point at the beginning of each epoch and is longer needed inside each epoch. As a consequence, Algorithm 1 with $I > 1$ reduces its number of communication rounds by a factor of I when compared with the classical parallel mini-batch SGD. The linear speed-up property (w.r.t. number of workers) with $I > 1$ is recently proven only for strongly convex optimization in (Stich 2018). However, there is no theoretical guarantee on whether the linear speed-up with $I > 1$ can be preserved for non-convex optimization, which is the case of deep neural networks.

Fix iteration index t, we define

$$\overline{\mathbf{x}}^t \triangleq \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^t \tag{4}$$

as the average of local solution $\mathbf{x}_i^t$ over all N nodes. It is immediate that

$$\overline{\mathbf{x}}^t = \overline{\mathbf{x}}^{t-1} - \gamma \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^t \tag{5}$$

Inspired by earlier works on distributed stochastic optimization (Zhang, Wainwright, and Duchi 2012) (Lian et al. 2017) (Mania et al. 2017) (Stich 2018) where convergence analysis is performed for an aggregated version of individual solutions, this paper focuses on the convergence rate analysis of $\overline{\mathbf{x}}^t$ defined in (4). An interesting observation from (5) is: Workers in Algorithm 1 run their local SGD independently for most iterations, however, they still jointly update their node average using a dynamic similar to SGD.The main issue in (5) is an "inaccurate" stochastic gradient, which is a simple average of individual stochastic gradients at points different from $\overline{\mathbf{x}}^t$, is used. Since each worker in Algorithm 1 periodically restarts its SGD with the same initial point, deviations between each local solution $\mathbf{x}_i^t$ and $\overline{\mathbf{x}}^t$ are expected to be controlled by selecting a proper synchronization interval I. The following useful lemma relates quantity $\mathbb{E}[\|\overline{\mathbf{x}}^t - \mathbf{x}_i^t\|^2]$ and algorithm parameter I. A similar lemma is proven in (Stich 2018).

Lemma 1. *Under Assumption 1, Algorithm 1 ensures*

$$\mathbb{E}[\|\overline{\mathbf{x}}^t - \mathbf{x}_i^t\|^2] \leq 4\gamma^2 I^2 G^2, \forall i, \forall t$$

where $\overline{\mathbf{x}}^t$ is defined in (4) *and G is the constant defined in Assumption 1.*

Proof. Fix $t \geq 1$ and $i \in \{1, 2, \ldots, N\}$. Note that Algorithm 1 calculates the node average $\overline{\mathbf{y}} \triangleq \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{t-1}$ every I iterations. Consider the largest $t_0 \leq t$ such that $\overline{\mathbf{y}} = \overline{\mathbf{x}}^{t_0}$ at iteration t_0 in Algorithm 1. (Note that such t_0 must exist and $t - t_0 \leq I$.) We further note, from the update equations (2) and (3) in Algorithm 1, that

$$\mathbf{x}_i^t = \overline{\mathbf{y}} - \gamma \sum_{\tau=t_0+1}^{t} \mathbf{G}_i^\tau \tag{6}$$

By (5), we have $\overline{\mathbf{x}}^t = \overline{\mathbf{y}} - \gamma \sum_{\tau=t_0+1}^t \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^\tau$.

Thus, we have

$$\begin{aligned}
&\mathbb{E}[\|\mathbf{x}_i^t - \overline{\mathbf{x}}^t\|^2] \\
=&\mathbb{E}[\|\gamma \sum_{\tau=t_0+1}^{t} \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^\tau - \gamma \sum_{\tau=t_0+1}^{t} \mathbf{G}_i^\tau\|^2] \\
=&\gamma^2 \mathbb{E}[\| \sum_{\tau=t_0+1}^{t} \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^\tau - \sum_{\tau=t_0+1}^{t} \mathbf{G}_i^\tau\|^2] \\
\overset{(a)}{\leq}&2\gamma^2 \mathbb{E}[\| \sum_{\tau=t_0+1}^{t} \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^\tau\|^2 + \| \sum_{\tau=t_0+1}^{t} \mathbf{G}_i^\tau\|^2] \\
\overset{(b)}{\leq}&2\gamma^2 (t-t_0) \mathbb{E}[\sum_{\tau=t_0+1}^{t} \| \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^\tau\|^2 + \sum_{\tau=t_0+1}^{t} \|\mathbf{G}_i^\tau\|^2] \\
\overset{(c)}{\leq}&2\gamma^2 (t-t_0) \mathbb{E}[\sum_{\tau=t_0+1}^{t} (\frac{1}{N}\sum_{i=1}^N \|\mathbf{G}_i^\tau\|^2) + \sum_{\tau=t_0+1}^{t} \|\mathbf{G}_i^\tau\|^2] \\
\overset{(d)}{\leq}&4\gamma^2 I^2 G^2
\end{aligned}$$

where (a)-(c) follows by using the inequality $\| \sum_{i=1}^n \mathbf{z}_i\|^2 \leq n \sum_{i=1}^n \|\mathbf{z}_i\|^2$ for any vectors $\mathbf{z}_i$ and any positive integer n (using $n = 2$ in (a), $n = t - t_0$ in (b) and $n = N$ in (c)); and (d) follows from Assumption 1. □

Theorem 1. *Consider problem* (1) *under Assumption 1. If $0 < \gamma \leq \frac{1}{L}$ in Algorithm 1, then for all $T \geq 1$, we have*

$$\frac{1}{T}\sum_{t=1}^T \mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2] \leq \frac{2}{\gamma T}(f(\overline{\mathbf{x}}^0) - f^*) + 4\gamma^2 I^2 G^2 L^2 + \frac{L}{N}\gamma\sigma^2$$

where f^* is the minimum value of problem (1).

Proof. Fix $t \geq 1$. By the smoothness of f , we have

$$\mathbb{E}[f(\overline{\mathbf{x}}^t)] \leq \mathbb{E}[f(\overline{\mathbf{x}}^{t-1})] + \mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1} \rangle] + \frac{L}{2}\mathbb{E}[\|\overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1}\|^2] \tag{7}$$

Note that

$$\begin{aligned}
\mathbb{E}[\|\overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1}\|^2] &\overset{(a)}{=} \gamma^2 \mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^t\|^2] \\
&\overset{(b)}{=} \gamma^2 \mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N (\mathbf{G}_i^t - \nabla f_i(\mathbf{x}_i^{t-1}))\|^2] + \gamma^2 \mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2] \\
&\overset{(c)}{=} \gamma^2 \frac{1}{N^2}\sum_{i=1}^N \mathbb{E}[\|\mathbf{G}_i^t - \nabla f_i(\mathbf{x}_i^{t-1})\|^2] + \gamma^2 \mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2] \\
&\overset{(d)}{\leq} \frac{1}{N}\gamma^2\sigma^2 + \gamma^2 \mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2]
\end{aligned} \tag{8}$$

where (a) follows from (5); (b) follows by noting that $\mathbb{E}[\mathbf{G}_i^t] = \nabla f_i(\mathbf{x}_i^{t-1})$ and applying the basic inequality $\mathbb{E}[\|\mathbf{Z}\|^2] = \mathbb{E}[\|\mathbf{Z} - \mathbb{E}[\mathbf{Z}]\|^2] + \|\mathbb{E}[\mathbf{Z}]\|^2$ that holds for any random vector $\mathbf{Z}$; (c) follows because each $\mathbf{G}_i^t - \nabla f_i(\mathbf{x}_i^{t-1})$ has $\mathbf{0}$ mean and is independent across nodes; and (d) follows from Assumption 1.

We further note that

$$\begin{aligned}
&\mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1} \rangle] \\
&\overset{(a)}{=} -\gamma \mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^t \rangle] \\
&\overset{(b)}{=} -\gamma \mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1}) \rangle] \\
&\overset{(c)}{=} -\frac{\gamma}{2}\mathbb{E}\big[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2 + \|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2 \\
&\qquad - \|\nabla f(\overline{\mathbf{x}}^{t-1}) - \frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2\big]
\end{aligned} \tag{9}$$

where (a) follows from (5); (b) follows because

$$\begin{aligned}
&\mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^t \rangle] \\
=&\mathbb{E}[\mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \frac{1}{N}\sum_{i=1}^N \mathbf{G}_i^t \rangle | \boldsymbol{\zeta}^{[t-1]}]] \\
=&\mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \frac{1}{N}\sum_{i=1}^N \mathbb{E}[\mathbf{G}_i^t | \boldsymbol{\zeta}^{[t-1]}] \rangle] \\
=&\mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1}) \rangle]
\end{aligned}$$

where the first equality follows by the iterated law of expectations, the second equality follows because $\overline{\mathbf{x}}^{t-1}$ is determined by $\boldsymbol{\zeta}^{[t-1]} = [\boldsymbol{\zeta}^1, \ldots, \boldsymbol{\zeta}^{t-1}]$ and the third equality follows by $\mathbb{E}[\mathbf{G}_i^t | \boldsymbol{\zeta}^{[t-1]}] = \mathbb{E}[\nabla F_i(\mathbf{x}_i^{t-1}; \zeta_i^t) | \boldsymbol{\zeta}^{[t-1]}] = \nabla f_i(\mathbf{x}_i^{t-1})$; and (c) follows from the basic identity $\langle \mathbf{z}_1, \mathbf{z}_2 \rangle = \frac{1}{2}\big(\|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 - \|\mathbf{z}_1 - \mathbf{z}_2\|^2\big)$ for any two vectors $\mathbf{z}_1, \mathbf{z}_2$ of the same length.

Substituting (8) and (9) into (7) yields

$$\begin{aligned}
&\mathbb{E}[f(\overline{\mathbf{x}}^t)] \\
\leq&\mathbb{E}[f(\overline{\mathbf{x}}^{t-1})] - \frac{\gamma - \gamma^2 L}{2}\mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2] \\
&- \frac{\gamma}{2}\mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2] \\
&+ \frac{\gamma}{2}\mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1}) - \frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2] + \frac{L}{2N}\gamma^2\sigma^2 \\
\overset{(a)}{\leq}&\mathbb{E}[f(\overline{\mathbf{x}}^{t-1})] - \frac{\gamma - \gamma^2 L}{2}\mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2] \\
&- \frac{\gamma}{2}\mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2] + 2\gamma^3 I^2 G^2 L^2 + \frac{L}{2N}\gamma^2\sigma^2 \qquad (10) \\
\overset{(b)}{\leq}&\mathbb{E}[f(\overline{\mathbf{x}}^{t-1})] - \frac{\gamma}{2}\mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2] + 2\gamma^3 I^2 G^2 L^2 + \frac{L}{2N}\gamma^2\sigma^2 \qquad (11)
\end{aligned}$$

where (b) follows from $0 < \gamma \leq \frac{1}{L}$ and (a) follows because

$$\begin{aligned}
&\mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1}) - \frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2] \\
=&\mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\overline{\mathbf{x}}^{t-1}) - \frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2] \\
=&\frac{1}{N^2}\mathbb{E}[\|\sum_{i=1}^N \big(\nabla f_i(\overline{\mathbf{x}}^{t-1}) - \nabla f_i(\mathbf{x}_i^{t-1})\big)\|^2] \\
\leq&\frac{1}{N}\mathbb{E}[\sum_{i=1}^N \|\nabla f_i(\overline{\mathbf{x}}^{t-1}) - \nabla f_i(\mathbf{x}_i^{t-1})\|^2] \\
\leq&L^2\frac{1}{N}\sum_{i=1}^N \mathbb{E}[\|\overline{\mathbf{x}}^{t-1} - \mathbf{x}_i^{t-1}\|^2] \\
\leq&4\gamma^2 I^2 G^2 L^2
\end{aligned}$$

where the first inequality follows by using $\|\sum_{i=1}^N \mathbf{z}_i\|^2 \leq N\sum_{i=1}^N \|\mathbf{z}_i\|^2$ for any vectors $\mathbf{z}_i$; the second inequality follows from the smoothness of each f_i by Assumption 1; and the third inequality follows from Lemma 1.

Dividing both sides of (11) by $\frac{\gamma}{2}$ and rearranging terms yields

$$\begin{aligned}
&\mathbb{E}\left[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2\right] \\
\leq&\frac{2}{\gamma}\left(\mathbb{E}\left[f(\overline{\mathbf{x}}^{t-1})\right] - \mathbb{E}\left[f(\overline{\mathbf{x}}^t)\right]\right) + 4\gamma^2 I^2 G^2 L^2 + \frac{L}{N}\gamma\sigma^2
\end{aligned} \tag{12}$$

Summing over $t \in \{1, \ldots, T\}$ and dividing both sides by T

yields

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2\right] \\
\leq&\frac{2}{\gamma T}\left(f(\overline{\mathbf{x}}^0)-\mathbb{E}\left[f(\overline{\mathbf{x}}^T)\right]\right)+4\gamma^2I^2G^2L^2+\frac{L}{N}\gamma\sigma^2 \\
\overset{(a)}{\leq}&\frac{2}{\gamma T}\left(f(\overline{\mathbf{x}}^0)-f^*\right)+4\gamma^2I^2G^2L^2+\frac{L}{N}\gamma\sigma^2
\end{aligned}$$

where (a) follows because f^* is the minimum value of problem (1). □

The next corollary follows by substituting suitable γ, I values into Theorem 1.

Corollary 1. *Consider problem* (1) *under Assumption 1. Let* $T \geq N$.

1. *If we choose* $\gamma = \frac{\sqrt{N}}{L\sqrt{T}}$ *in Algorithm 1, then we have* $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2] \leq \frac{2L}{\sqrt{NT}}\left(f(\overline{\mathbf{x}}^0)-f^*\right)+\frac{4N}{T}I^2G^2+\frac{1}{\sqrt{NT}}\sigma^2$.
2. *If we further choose* $I \leq \frac{T^{1/4}}{N^{3/4}}$, *then* $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2] \leq \frac{2L}{\sqrt{NT}}(f(\overline{\mathbf{x}}^0)-f^*)+\frac{4}{\sqrt{NT}}G^2+\frac{1}{\sqrt{NT}}\sigma^2 = O(\frac{1}{\sqrt{NT}})$ *where* f^* *is the minimum value of problem* (1)*.*

Remark 1. *For non-convex optimization, it is generally impossible to develop a convergence rate for objective values. In Theorem 1 and Corollary 1, we follow the convention in literature (Ghadimi and Lan 2013) (Lian et al. 2017) (Alistarh et al. 2017) to use the (average) expected squared gradient norm to characterize the convergence rate. Note that the average can be attained in expectation by taking each* $\overline{\mathbf{x}}^{t-1}$ *with an equal probability* $1/T$.

From Theorem 1 and Corollary 1, we have the following important observations:

- **Linear Speedup:** By part (1) of Corollary 1, Algorithm 1 with any fixed constant I has convergence rate $O(\frac{1}{\sqrt{NT}}+\frac{N}{T})$. If T is large enough, i.e., $T > N^3$, then the term $\frac{N}{T}$ is dominated by the term $\frac{1}{\sqrt{NT}}$ and hence Algorithm 1 has convergence rate $O(\frac{1}{\sqrt{NT}})$. That is, our algorithm achieves a linear speed-up with respect to the number of workers. Such linear speedup for stochastic non-convex optimization was previously attained by decentralized-parallel stochastic gradient descent (D-PSGD) considered in (Lian et al. 2017) by requiring at least $T > N^5$. See, e.g., Corollary 2 in (Lian et al. 2017).[4]
- **Communication Reduction:** Note that Algorithm 1 requires inter-node communication only at the iterations that are multiples of I. By Corollary 1, it suffices to choose any $I \leq \frac{T^{1/4}}{N^{3/4}}$ to ensure the $O(\frac{1}{\sqrt{NT}})$ convergence of our algorithm. That is, compared with parallel mini-batch SGD or the D-PSGD in (Lian et al. 2017), the number of communication rounds in our algorithm can be reduced by a factor $\frac{T^{1/4}}{N^{3/4}}$. Although Algorithm 1 does not describe how the node average $\overline{\mathbf{y}}$ is obtained at each node, in practice, the simplest way is to introduce a parameter server that collects all local solutions and broadcasts their average as in parallel mini-batch SGD (Li et al. 2014). Alternatively, we can perform an **all-reduce** operation on the local models(without introducing a server) such that all nodes obtain $\overline{\mathbf{y}}$ independently and simultaneously. (Using an all-reduce operation among all nodes to obtain gradients averages has been previously suggested in (Goyal et al. 2017) for distributed training of deep learning.)

[4] In fact, for a ring network considered in Theorem 3 in (Lian et al. 2017), D-PSGD requires a even larger $T > N^9$ since its implementation depends on the network topology. In contrast, the linear speedup of our algorithm is irrelevant to the network topology.

Extensions

Using Time-Varying Learning Rates

Note that Corollary 1 assumes time horizon T is known and uses a constant learning rate in Algorithm 1. In this subsection, we consider the scenario where the time horizon T is not known beforehand and develop a variant of Algorithm 1 with time-varying rates to achieve the same computation and communication complexity. Compared with Algorithm 1, Algorithm 2 has the advantage that its accuracy is being improved automatically as it runs longer.

Algorithm 2 PR-SGD with Time-Varying Learning Rates

1: **Input:** Set time-varying epoch learning rates $\gamma^s > 0$.
2: **Initialize:** Initialize $\mathbf{x}_i^{0,K^0} = \overline{\mathbf{x}}^0 \in \mathbb{R}^m$.
3: **for** epoch index $s = 1$ to S **do**
4: Set epoch length K^s and initialize $\mathbf{x}_i^{s,0} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i^{s-1,K^{s-1}}$ to be the node average of local worker solutions from the last epoch.
5: **for** $k = 1$ to K^s **do**
6: Each node i observes an unbiased gradient $\mathbf{G}_i^{s,k}$ of $f_i(\cdot)$ at point $\mathbf{x}_i^{s,k-1}$ and in parallel updates

$$\mathbf{x}_i^{s,k} = \mathbf{x}_i^{s,k-1} - \gamma^s\mathbf{G}_i^{s,k}, \quad \forall i \tag{13}$$

7: **end for**
8: **end for**

Although Algorithm 2 introduces the concept of epoch for the convenience of description, we note that it is nothing but a parallel restarted SGD where each worker restarts itself every epoch using the node average of the last epoch's final solutions as the initial point. If we sequentially reindex $\{\mathbf{x}_i^{s,k}\}_{s\in\{1,\ldots,S\},k\in\{1,\ldots,K^s\}}$ as $\mathbf{x}_i^t$ (note that all $\mathbf{x}_i^{s,0}$ are ignored since $\mathbf{x}_i^{s,0} = \mathbf{x}_i^{s-1,K^{s-1}}$), then Algorithm 2 is mathematically equivalent to Algorithm 1 except that time-varying learning rates γ^s are used in different epochs. Similarly to (4), we can define $\overline{\mathbf{x}}^{s,k}$ via $\overline{\mathbf{x}}^{s,k} \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i^{s,k}$ and have

$$\overline{\mathbf{x}}^{s,k} = \overline{\mathbf{x}}^{s,k-1} - \gamma\frac{1}{N}\sum_{i=1}^{N}\mathbf{G}_i^{s,k} \tag{14}$$

Theorem 2. *Consider problem* (1) *under Assumption 1. If we choose* $K^s = \lceil \frac{s^{1/3}}{N} \rceil$ *and* $\gamma^s = \frac{N}{s^{2/3}}$ *in Algorithm 2, then for all* $S \geq 1$*, we have*[5]

$$\frac{1}{\sum_{s=1}^{S}\sum_{k=1}^{K^s}\gamma^s}\sum_{s=1}^{S}\sum_{k=1}^{K^s}\mathbb{E}\left[\gamma^s\|\nabla f(\overline{\mathbf{x}}^{s,k-1})\|^2\right] \leq \widetilde{O}(\frac{1}{\sqrt{NT}})$$

where $T = \sum_{s=1}^{S} K^s$.

Proof. See our ArXiv full version. □

Asynchronous Implementations in Heterogeneous Networks

Algorithm 1 requires all workers to compute the average of individual solutions every I iterations and synchronization among local workers are not needed before averaging. However, the fastest worker still needs to wait until all the other workers finish I iterations of SGD even if it finishes its own I iteration SGD much earlier. (See Figure 1 for a 2 worker example where one worker is significantly faster than the other. Note that orange "syn" rectangles represent the procedures to compute the node average.) As a consequence, the computation capability of faster workers is wasted. Such an issue can arise quite often in heterogeneous networks where nodes are equipped with different hardwares. Intuitively, if

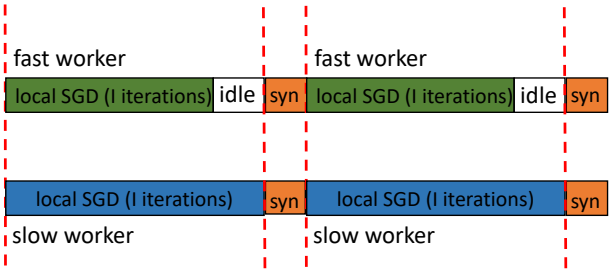

Figure 1: An illustration of Algorithm 1 implemented in a 2 worker heterogeneous network. Orange "syn" rectangles represent the procedures to compute the node average.

one worker finishes its I iteration local SGD earlier, to avoid wasting its computation capability, we might want to let this worker continue running its local SGD until all the other workers finish their I iteration local SGD. However, such a method can drag the node average too far towards the local solution at the fastest worker. Note that if $f_i(\cdot)$ in (1) are significantly different from each other such that the minimizer of $f_i(\cdot)$ at the i-th worker, which is the fastest one, deviates the true minimizer of problem (1) too much, then dragging the node average towards the fastest worker's local solution is undesired. In this subsection, we further assume that problem (1) satisfies the following assumption:

Assumption 2. *The distributions* $\mathcal{D}_i$ *in the definition of each* $f_i(\mathbf{x}) \triangleq \mathbb{E}_{\zeta_i \sim \mathcal{D}_i}[F_i(\mathbf{x};\zeta_i)]$ *in* (1) *are identical.*

[5] A logarithm factor $\log(NT)$ is hidden in the notation $\widetilde{O}(\cdot)$.

Note that Assumption 2 is satisfied if all local workers can access a common training data set or each local training data set is obtained from uniform sampling from the global training set. Consider the restarted local SGD for heterogeneous networks described in Algorithm 3. Note that if $I_i \equiv I, \forall i$ for some fixed constant I, then Algorithm 3 degrades to Algorithm 1. In practice, if the hardware configurations or measurements (from previous experiments) of each local worker are known, we can predetermine the value of each I_i, i.e., if worker i is two times faster than worker j, then $I_i = 2I_j$. Alternatively, under a more practical implementation, we can set a fixed time duration for each epoch and let each local worker keep running its local SGD until the given time elapses. By doing so, within the same time duration, the faster a worker is, the more SGD iterations it runs. In contrast, if we apply Algorithm 1 in this setting, then all local workers have to run the same number of SGD iterations as that can be run by the slowest worker within the given time interval. This subsection shows that, under Assumption 2, Algorithm 3 can achieve a better performance than Algorithm 1 in heterogeneous networks where some workers are much faster than others.

Algorithm 3 PR-SGD in Heterogeneous Networks

1: **Input:** Set learning rate $\gamma > 0$ and epoch length of each worker i as I_i.
2: **Initialize:** Initialize $\mathbf{x}_i^{0,I_i} = \overline{\mathbf{x}}^0 \in \mathbb{R}^m$.
3: **for** epoch index $s = 1$ to S **do**
4: Initialize $\mathbf{x}_i^{s,0} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i^{s-1,I_i}$ as the node average of local worker solutions from the last epoch.
5: Each worker i in parallel runs its local SGD for I_i iterations via:

$$\mathbf{x}_i^{s,k} = \mathbf{x}_i^{s,k-1} - \gamma \mathbf{G}_i^{s,k}, \quad \forall i \tag{15}$$

where $\mathbf{G}_i^{s,k}$ is an unbiased stochastic gradient at point $\mathbf{x}_i^{s,k-1}$.
6: **end for**

Without loss of generality, this subsection always indexes local workers in a decreasing order of speed. That is, worker 1 is the fastest while worker N is the slowest. If we run Algorithm 3 by specifying a fixed wall clock time duration for each epoch, during which each local worker keeps running its local SGD, then we have $I_1 \geq I_2 \geq \cdots \geq I_N$. Fix epoch index s, for all $i \neq 1$, variables $\mathbf{x}_i^{s,k}$ with $k > I_i$ is never used. However, for the convenience of analysis, we define

$$\mathbf{x}_i^{s,k} \triangleq \mathbf{x}_i^{s,k-1}, \forall i \neq 1, \forall k \in \{I_i + 1, \ldots, I_1\}.$$

Conceptually, the above equation can be interpreted as assuming worker i, which is slower than worker 1, runs extra $I_1 - I_i$ iterations of SGD by using 0 as an imaginary stochastic gradient (with no computation cost). See Figure 2 for a 2 worker example where $I_1 = 16$ and $I_2 = 8$. Using the definition $\overline{\mathbf{x}}^{s,k} \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i^{s,k}$, we have

$$\overline{\mathbf{x}}^{s,k} = \overline{\mathbf{x}}^{s,k-1} + \gamma\frac{1}{N}\sum_{i:I_i \geq k}\mathbf{G}_i^{s,k}, \forall s, \forall k \in \{1, 2, \ldots, I_1\}.$$

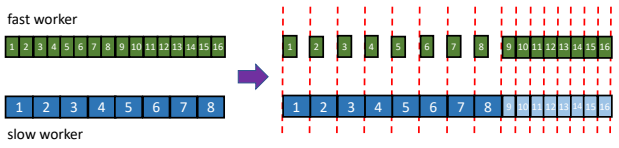

Figure 2: **Left**: A typical epoch of Algorithm 3 in a heterogeneous network with 2 workers. A wider rectangle means the SGD iteration takes a longer wall clock time. **Right**: Imagined extra SGD iterations with a 0 stochastic gradient (in light blue rectangles) are added for the slow worker.

Theorem 3. *Consider problem* (1) *under Assumptions 1 and 2. Suppose all workers are indexed in a decreasing order of their speed, i.e., worker* 1 *is the fastest and worker* N *is the slowest. If* $0 < \gamma \leq \frac{1}{L}$ *in Algorithm 3, then for all* $S \geq 1$,

$$\frac{1}{S\frac{1}{N}\sum_{i=1}^{N} I_i} \sum_{s=1}^{S} \sum_{k=1}^{I_1} \frac{j_k}{N} \mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{s,k-1})\|^2] \leq \frac{2}{\gamma S \frac{1}{N}\sum_{i=1}^{N} I_i}(f(\overline{\mathbf{x}}^0) - f^*) + 4\gamma^2 I_1^2 G^2 L^2 + \frac{L}{N}\gamma\sigma^2 \tag{16}$$

where j_k *for each given* k *is the largest integer in* $\{1, 2, \ldots, N\}$ *such that* $k \leq I_{j_k}$*(That is, for each fixed* k*,* j_k *is the number of workers that are still using sampled true stochastic gradients to update their local solutions at iteration* k*.); and* f^* *is the minimum value of problem* (1).

Proof. See our ArXiv full version. □

The next corollary shows that Algorithm 3 in heterogeneous networks can ensure the convergence and preserve the same $O(1/\sqrt{NT})$ convergence rate with the same $O(\frac{T^{1/4}}{N^{3/4}})$ communication reduction.

Corollary 2. *Consider problem* (1) *under Assumptions 1 and 2. Let* $T \geq N$*. If we use* $\gamma = \Theta(\frac{\sqrt{N}}{\sqrt{T}})$ *such that* $\gamma \leq \frac{1}{L}$*,* $I_i = \Theta\big(\frac{T^{1/4}}{N^{3/4}}\big), \forall i$ *and* $S = \frac{T}{I_N}$ *in Algorithm 3, then*

$$\frac{1}{S\frac{1}{N}\sum_{i=1}^{N} I_i} \sum_{s=1}^{S} \sum_{k=1}^{I_1} \frac{j_k}{N} \mathbb{E}\left[\|\nabla f(\overline{\mathbf{x}}^{s,k-1})\|^2\right] \leq O(\frac{1}{\sqrt{NT}})$$

where j_k *for each given* k *is the largest integer in* $\{1, 2, \ldots, N\}$ *such that* $k \leq I_{j_k}$.

Proof. This simply follows by substituting specific values of γ, I_i, S into (16) in Theorem 3. □

Remark 2. *Note that once* I_i *values are known, then* j_k *for any* k *in Theorem 3 and Corollary 2 are also available by its definition. To appreciate the implication of Theorem 2, we recall that Algorithm 1 is a special case of Algorithm 3 with* $I_i \equiv I_N, \forall i$*, i.e., all workers can only run the same number (determined by the slowest worker) of SGD iterations in each epoch. In this perspective, Theorem 1 (with* $I = I_N$*) implies that the performance of Algorithm 1 is given by*

$$\frac{1}{SI_N} \sum_{s=1}^{S} \sum_{k=1}^{I_N} \mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{s,k-1})\|^2] \leq \frac{2}{\gamma S I_N}(f(\overline{\mathbf{x}}^0) - f^*) + 4\gamma^2 I_N^2 G^2 L^2 + \frac{L}{N}\gamma\sigma^2 \tag{17}$$

Note that the left sides of (16) *and* (17) *(weighted average expressions) can be attained by taking each* $\overline{\mathbf{x}}^{s,k-1}$ *randomly with a probability equal to the normalized weight in the summation. The first error term in* (16) *is strictly smaller than that in* (17) *while the second error term in* (16) *is larger than that in* (17)*. Note that the constant factor* $f(\overline{\mathbf{x}}^0) - f^*$ *in the first error term in* (17) *is large when a poor initial point* $\overline{\mathbf{x}}^0$ *is chosen (and dominates the second error term if* $f(\overline{\mathbf{x}}^0) - f^* \geq 2\gamma^3 I_N^3 G^2 L^2 S$*). So the main message of Theorem 3 is that if a poor initial point is selected, Algorithm 2 can possibly converges faster than Algorithm 1 (at least for the first few epochs) in a heterogeneous network.*

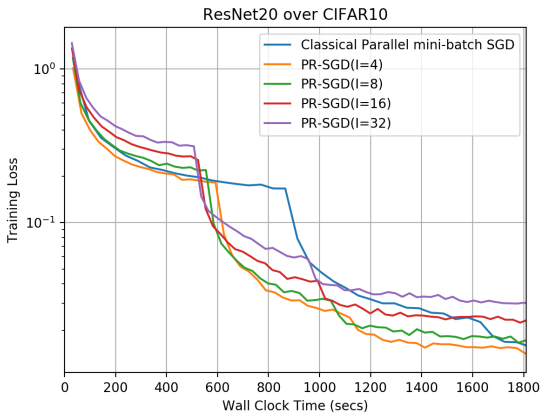

Figure 3: Training loss of ResNet20 over CIFAR10 on a machine with 8 P100 GPUs. In all schemes, each worker uses a local batch size 32 and momentum 0.9. The initial learning at each worker is 0.1 and is divided by 10 when 8 workers together access 150 epochs and 275 epochs of training data.

Experiment

The superior training speed-up performance of model averaging has been empirically observed in various deep learning scenarios, e.g., CNN for MNIST in (Zhang et al. 2016)(Kamp et al. 2018)(McMahan et al. 2017); VGG for CIFAR10 in (Zhou and Cong 2017); DNN-GMM for speech recognition in (Chen and Huo 2016) (Su, Chen, and Xu 2018); and LSTM for language modeling in (McMahan et al. 2017). A thorough empirical study of ResNet over CIFAR and ImageNet is also available in the recent work (Lin, Stich, and Jaggi 2018). We compare model averaging, i.e., PR-SGD (Algorithm 1) with $I \in \{4, 8, 16, 32\}$) with the classical parallel mini-batch SGD[6] by training ResNet20 with

[6]The classical parallel mini-batch SGD is equivalent to Algorithm 1 with $I = 1$. Our implementation with Horovod uses the more efficient "all-reduce"method rather than the "parameter server" method to synchronize information between workers.

CIFAR10 on a machine with 8 P100 GPUs. Figure 3 plots the training loss convergence. See our ArXiv full version for a figure of test accuracy. Our implementation uses Horovod (Sergeev and Del Balso 2018) for inter-worker communication and uses PyTorch 0.4 for algorithm implementations.

Conclusion

This paper studies parallel restarted SGD, which is a theoretical abstraction of the "model averaging" practice widely used in training deep neural networks. This paper shows that parallel restarted SGD can achieve $O(1/\sqrt{NT})$ convergence for non-convex optimization with a number of communication rounds reduced by a factor $O(T^{1/4})$ compared with that required by the classical parallel mini-batch SGD.

References

Aji, A. F., and Heafield, K. 2017. Sparse communication for distributed gradient descent. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 (NIPS)*.

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

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Senior, A.; Tucker, P.; Yang, K.; Le, Q. V.; et al. 2012. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Dekel, O.; Gilad-Bachrach, R.; Shamir, O.; and Xiao, L. 2012. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research* 13(165–202).

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

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

Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, large minibatch SGD: training imagenet in 1 hour. *arXiv:1706.02677*.

Jiang, Z.; Balu, A.; Hegde, C.; and Sarkar, S. 2017. Collaborative deep learning in fixed topology networks. In *Advances in Neural Information Processing Systems (NIPS)*.

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

Li, M.; Andersen, D. G.; Smola, A. J.; and Yu, K. 2014. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems (NIPS)*.

Lian, X.; Zhang, C.; Zhang, H.; Hsieh, C.-J.; Zhang, W.; and Liu, J. 2017. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*.

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

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

Mania, H.; Pan, X.; Papailiopoulos, D.; Recht, B.; Ramchandran, K.; and Jordan, M. I. 2017. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization* 27(4):2202–2229.

McDonald, R.; Hall, K.; and Mann, G. 2010. Distributed training strategies for the structured perceptron. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; et al. 2017. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Povey, D.; Zhang, X.; and Khudanpur, S. 2015. Parallel training of DNNs with natural gradient and parameter averaging. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Sergeev, A., and Del Balso, M. 2018. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv:1802.05799*.

Stich, S. U. 2018. Local SGD converges fast and communicates little. *arXiv:1805.09767*.

Strom, N. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*.

Su, H.; Chen, H.; and Xu, H. 2018. Experiments on parallel training of deep neural network using model averaging. *arXiv:1507.01239v3*.

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

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 (NIPS)*.

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

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

Zhang, S.; Choromanska, A. E.; and LeCun, Y. 2015. Deep learning with elastic averaging SGD. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhang, Y.; Wainwright, M. J.; and Duchi, J. C. 2012. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems (NIPS)*.

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

Zinkevich, M.; Weimer, M.; Li, L.; and Smola, A. J. 2010. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*.