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

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

(1)

where $N$ is the number of nodes/workers and each $f_i(\mathbf{x}) \triangleq \mathbb{E}_{\zeta_i \sim D_i}[F_i(\mathbf{x}; \zeta_i)]$ is a smooth non-convex function where $D_i$ 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 $G^i_t = \nabla F_i(\mathbf{x}_i^{(t-1)}; \zeta_i^t)$ with $\mathbb{E}_{\zeta_i^t \sim D_i}[G^i_t(\zeta_i^{(t-1)})] = \nabla f_i(\mathbf{x}_i^{(t-1)}), \forall i$ where $\zeta_i^{(t-1)} \overset{\Delta}{=} [\zeta_i^t]_{t \in \{1, 2, \ldots, N\}, \tau \in \{1, \ldots, t-1\}}$ 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.
Parallel Restarted SGD and Its Performance Analysis

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

**Assumption 1.**

1. **Smoothness:** Each function $f_i(x)$ is smooth with modulus $L$.

2. **Bounded variances and second moments:** There exits constants $\sigma > 0$ and $G > 0$ such that
   \[
   \mathbb{E}_{\zeta_i \sim D_i} \| \nabla f_i(x; \zeta_i) - \nabla f_i(x) \|^2 \leq \sigma^2, \forall x, \forall i
   \]
   \[
   \mathbb{E}_{\zeta_i \sim D_i} \| \nabla F_i(x; \zeta_i) \|^2 \leq G^2, \forall x, \forall i
   \]

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 $\mathbf{y}$ 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 $\mathbf{y}$ 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, 2018).
Algorithm 1 Parallel Restarted SGD (PR-SGD)

1: **Input:** Initialize $x_0^i = y \in \mathbb{R}^m$. Set learning rate $\gamma > 0$ and node synchronization interval (integer) $I > 0$
2: **for** $t = 1$ to $T$
3: Each node $i$ observes an unbiased stochastic gradient $G_t^i$ of $f_i(\cdot)$ at point $x_t^{i-1}$
4: **if** $t$ is a multiple of $I$, i.e., $t \mod I = 0$, then
5: Calculate node average $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} x_t^{i-1}$
6: Each node $i$ in parallel updates its local solution
   \[ x_t^i = \bar{y} - \gamma G_t^i, \quad \forall i \] (2)
7: **else**
8: Each node $i$ in parallel updates its local solution
   \[ x_t^i = x_{t-1}^i - \gamma G_t^i, \quad \forall i \] (3)
9: **end if**
10: **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
\[ x_t = \frac{1}{N}\sum_{i=1}^{N} x_t^i \] (4)
as the average of local solution $x_t^i$ over all $N$ nodes. It is immediate that
\[ x_t = x_{t-1} - \gamma \frac{1}{N}\sum_{i=1}^{N} G_t^i \] (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 averaged version of individual solutions, this paper focuses on the convergence rate analysis of $\bar{x}$ 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 $x_t$, is used. Since each worker in Algorithm 1 periodically restarts its SGD with the same initial point, deviations between each local solution $x_t^i$ and $\bar{x}$ are expected to be controlled by selecting a proper synchronization interval $I$. The following useful lemma relates quantity $E[\|x^0 - x_t^i\|^2]$ and algorithm parameter $I$. A similar lemma is proven in (Stich 2018).

**Lemma 1.** Under Assumption 1, Algorithm 1 ensures
\[ E[\|x_t - x^F\|^2] \leq 4\gamma^2 I^2 G^2, \forall i, \forall t \]
where $x_t^F$ 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 $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} x_t^{i-1}$ every $I$ iterations. Consider the largest $t_0 \leq t$ such that $y = x_0^i$ 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
\[ x_t^i = \bar{y} - \gamma \sum_{\tau=t_0+1}^{t} G_t^i \] (6)

By (5), we have $x_t^F = \bar{y} - \gamma \sum_{\tau=t_0+1}^{t} \frac{1}{N}\sum_{i=1}^{N} G_t^i$.
Thus, we have
\[ E[\|x_t^i - x_t^F\|^2] = E[\|\gamma \sum_{\tau=t_0+1}^{t} G_t^i - \gamma \sum_{\tau=t_0+1}^{t} G_t^i\|^2] \]
\[ = \gamma^2 E[\|\sum_{\tau=t_0+1}^{t} G_t^i\|^2] \]
\[ \leq 2\gamma^2 E[\sum_{\tau=t_0+1}^{t} \frac{1}{N}\sum_{i=1}^{N} G_t^i\|^2 + \sum_{\tau=t_0+1}^{t} \|G_t^i\|^2] \]
\[ \leq 2\gamma^2 E[\sum_{\tau=t_0+1}^{t} \frac{1}{N}\sum_{i=1}^{N} G_t^i\|^2 + \sum_{\tau=t_0+1}^{t} \|G_t^i\|^2] \]
\[ \leq 2\gamma^2 E[\sum_{\tau=t_0+1}^{t} \frac{1}{N}\sum_{i=1}^{N} \|G_t^i\|^2 + \sum_{\tau=t_0+1}^{t} \|G_t^i\|^2] \]
\[ \leq 4\gamma^2 I^2 G^2 \]
where (a)-(c) follows by using the inequality $\|\sum_{i=1}^{n} z_i\|^2 \leq n \sum_{i=1}^{n} \|z_i\|^2$ for any vectors $z_i$ 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 $0 < \gamma \leq \frac{1}{\nabla f^2}$ in Algorithm 1, then for all $T \geq 1$, we have
\[ \frac{1}{T} \sum_{t=1}^{T} E[\|\nabla f(x_{t-1})\|^2] \leq \frac{2}{\gamma^2} (f(x^0) - f^*) + 4\gamma^2 I^2 G^2 L^2 + \frac{L}{N} \sigma^2 \]
where $f^*$ is the minimum value of problem (1).
Proof. Fix $t \geq 1$. By the smoothness of $f$, we have
\[ 
\mathbb{E}[f(\mathbf{x}^t)] \leq \mathbb{E}[f(\mathbf{x}^{t-1})] + \mathbb{E}[(\nabla f(\mathbf{x}^{t-1}), \mathbf{x}^t - \mathbf{x}^{t-1})] + \frac{L}{2} \mathbb{E}[\|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2] 
\]
(7)

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

where (a) follows from (5); (b) follows by noting that $\mathbb{E}[\mathbf{G}_i^t] = \nabla f_i(\mathbf{x}^{t-1})$ 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 $\mathbf{G}_i^t - \nabla f_i(\mathbf{x}^{t-1})$ has 0 mean and is independent across nodes; and (d) follows from Assumption 1.

We further note that
\[ 
\mathbb{E}[\nabla f(\mathbf{x}^{t-1}), \mathbf{x}^t - \mathbf{x}^{t-1})] 
\]
\[ 
= -\gamma \mathbb{E}[\nabla f(\mathbf{x}^{t-1}), \sum_{i=1}^{N} \mathbf{G}_i^t] 
\]
\[ 
= -\gamma \mathbb{E}[\nabla f(\mathbf{x}^{t-1}), \sum_{i=1}^{N} \nabla f_i(\mathbf{x}^{t-1})] 
\]
\[ 
= -\gamma \mathbb{E}[\|\nabla f(\mathbf{x}^{t-1})\|^2 + \sum_{i=1}^{N} \|\nabla f_i(\mathbf{x}^{t-1})\|^2] 
\]
\[ 
= -\frac{\gamma}{2} \mathbb{E}[\|\nabla f(\mathbf{x}^{t-1})\|^2] - \gamma \mathbb{E}[\|\sum_{i=1}^{N} \nabla f_i(\mathbf{x}^{t-1})\|^2] 
\]
(9)

where (a) follows from (5); (b) follows because
\[ 
\mathbb{E}[\nabla f(\mathbf{x}^{t-1}), \sum_{i=1}^{N} \mathbf{G}_i^t] 
\]
\[ 
= \mathbb{E}[\mathbb{E}[\nabla f(\mathbf{x}^{t-1}), \sum_{i=1}^{N} \mathbf{G}_i^t]|\mathbf{x}^{t-1})] 
\]
\[ 
= \mathbb{E}[\nabla f(\mathbf{x}^{t-1}), \sum_{i=1}^{N} \mathbb{E}[\mathbf{G}_i^t]|\mathbf{x}^{t-1})] 
\]
\[ 
= \mathbb{E}[\nabla f(\mathbf{x}^{t-1}), \sum_{i=1}^{N} \nabla f_i(\mathbf{x}^{t-1})] 
\]
where the first equality follows by the iterated law of expectations, the second equality follows because $\mathbf{x}^{t-1}$ is determined by $\mathbf{z}^{t-1} = [\mathbf{z}_1^{t-1}, \ldots, \mathbf{z}_t^{t-1}]$ and the third equality follows by $\mathbb{E}[\mathbf{G}_i^t|\mathbf{z}^{t-1})] = \mathbb{E}[\nabla f_i(\mathbf{x}_i^{t-1}; \mathbf{z}_i^{t-1})|\mathbf{z}^{t-1})] = \nabla f_i(\mathbf{x}_i^{t-1}), and (c) follows from the basic identity $(\mathbf{z}_1, \mathbf{z}_2) = \frac{1}{2}(||\mathbf{z}_1||^2 + ||\mathbf{z}_2||^2 - ||\mathbf{z}_1 - \mathbf{z}_2||^2)$ for any two vectors $\mathbf{z}_1, \mathbf{z}_2$ of the same length.

Substituting (8) and (9) into (7) yields
\[ 
\mathbb{E}[f(\mathbf{x}^t)] 
\]
\[ 
\leq \mathbb{E}[f(\mathbf{x}^{t-1})] - \frac{\gamma - \gamma^2 L}{2} \mathbb{E}[\|\sum_{i=1}^{N} \nabla f_i(\mathbf{x}^{t-1})\|^2] 
\]
\[ 
+ \frac{2\gamma}{2} \mathbb{E}[\|\nabla f(\mathbf{x}^{t-1})\|^2] + \frac{L}{2N} \gamma^2 \sigma^2 
\]
(10)

where (b) follows from $0 < \gamma \leq \frac{1}{2}$ and (a) follows because
\[ 
\mathbb{E}[\|\nabla f(\mathbf{x}^{t-1}) - \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\mathbf{x}^{t-1})\|^2] 
\]
\[ 
= \mathbb{E}[\|\sum_{i=1}^{N} \nabla f_i(\mathbf{x}^{t-1}) - \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\mathbf{x}^{t-1})\|^2] 
\]
\[ 
= \frac{1}{N^2} \mathbb{E}[\|\sum_{i=1}^{N} (\nabla f_i(\mathbf{x}^{t-1}) - \nabla f_i(\mathbf{x}^{t-1}))\|^2] 
\]
\[ 
\leq \frac{1}{N} \mathbb{E}[\|\sum_{i=1}^{N} \nabla f_i(\mathbf{x}^{t-1}) - \nabla f_i(\mathbf{x}^{t-1})\|^2] 
\]
\[ 
\leq \frac{L^2}{N} \sum_{i=1}^{N} \mathbb{E}[\|\mathbf{x}^{t-1} - \mathbf{x}^{t-1}\|^2] 
\]
\[ 
\leq 4\gamma^2 T^2 G^2 L^2 
\]
(11)

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
\[ 
\mathbb{E}[(\nabla f(\mathbf{x}^{t-1}))^2] 
\]
\[ 
\leq \frac{2}{\gamma} \left( \mathbb{E}[f(\mathbf{x}^{t-1})] - \mathbb{E}[f(\mathbf{x})] + 4\gamma^2 T^2 G^2 L^2 + \frac{L}{N} \gamma^2 \sigma^2 \right) 
\]
(12)

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

5696
yields
\[
\frac{1}{T} \sum_{t=1}^{T} E \left[ \| \nabla f(x_t) \|^2 \right] \\
\leq \frac{2}{\gamma T} \left( f(x_0) - E \left[ f(x_T) \right] \right) + 4 \gamma^2 I^2 G^2 L^2 + \frac{L}{N} \gamma \sigma^2 \\
\leq \frac{2}{\gamma T} \left( f(x_0) - f^* \right) + 4 \gamma^2 I^2 G^2 L^2 + \frac{L}{N} \gamma \sigma^2 \tag{a}
\]
where (a) follows because \( f^* \) is the minimum value of problem (1).

The next corollary follows by substituting suitable \( \gamma, I \) values into Theorem 1.

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

1. If we choose \( \gamma = \frac{\sqrt{N}}{T^{3/4}} \) in Algorithm 1, then we have
\[
\frac{1}{T} \sum_{t=1}^{T} E \left[ \| \nabla f(x_t) \|^2 \right] \leq \frac{2}{\sqrt{NT}} \left( f(x_0) - f^* \right) + \frac{4N}{\sqrt{T}} G^2 + \frac{1}{\sqrt{NT}} \sigma^2.
\]
2. If we further choose \( I \leq \frac{T^{1/4}}{N^{3/4}} \), then
\[
\frac{1}{T} \sum_{t=1}^{T} E \left[ \| \nabla f(x_t) \|^2 \right] \leq \frac{2}{\sqrt{NT}} \left( f(x_0) - f^* \right) + \frac{4}{\sqrt{NT}} G^2 + \frac{1}{\sqrt{NT}} \sigma^2 = O\left( \frac{1}{\sqrt{NT}} \right)
\]
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 \( x_t \) 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 \( O\left( \frac{1}{\sqrt{NT}} + \frac{N}{T} \right) \). 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 \( O\left( \frac{1}{\sqrt{NT}} \right) \). That is, our algorithm achieves a linear speedup 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).

- **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 \( O\left( \frac{1}{\sqrt{NT}} \right) \) 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 \( T^{3/4} \). Although Algorithm 1 does not describe how the node average \( \overline{x} \) 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 \( \overline{x} \) 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.)

**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 \( \gamma^k > 0 \).
2. **Initialize:** Initialize \( x^{0,K^0}_s = x^0 \in \mathbb{R}^m \).
3. **for** epoch index \( s = 1 \) to \( S \) **do**
4. **Set** epoch length \( K^s \) and initialize \( x^{s,0}_s = \frac{1}{K^s} \sum_{i=1}^{K^s} x^{s-1,k}_{s-1,K^s-1} \) 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 \( G^{s,k}_i \) of \( f_i(\cdot) \) at point \( x^{s,k}_{i-1} \) and in parallel updates
\[
x^{s,k}_i = x^{s,k-1}_i - \gamma^k G^{s,k}_i, \quad \forall i \tag{13}
\]
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 index \( \{x^{s,k}_i \}_{s \in \{1,...,S\},k \in \{1,...,K^s\}} \) as \( x^s \) (note that all \( x^{s,0}_i \) are ignored since \( x^{s,0}_i = x^{s-1,k}_{s-1,K^s-1} \)), then Algorithm 2 is mathematically equivalent to Algorithm 1 except that time-varying learning rates \( \gamma^k \) are used in different epochs. Similarly to (4), we can define \( \bar{x}^{s,k} \) via \( \bar{x}^{s,k} = \frac{1}{N} \sum_{i=1}^{N} x^{s,k}_i \) and have
\[
\bar{x}^{s,k} = \bar{x}^{s,k-1} - \frac{1}{N} \sum_{i=1}^{N} G^{s,k}_i \tag{14}
\]
Theorem 2. Consider problem (1) under Assumption 1. If we choose $K^s = \lceil s^{1/3} N \rceil$ and $\gamma^s = \frac{N}{s^{1/3}}$ in Algorithm 2, then for all $S \geq 1$, we have
\[ \frac{1}{\sum_{s=1}^{S} \sum_{k=1}^{K^s} \gamma^s} \sum_{s=1}^{S} \sum_{k=1}^{K^s} \mathbb{E} [\gamma^s \| \nabla f(x_i^{(s,k)}) \|^2] \leq \tilde{O}(\frac{1}{\sqrt{NT}}) \]
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 fasted 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 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 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 $D_i$ in the definition of each $f_i(\cdot)$ in (1) are identical.

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 $I_i \equiv I_i$, $\forall i$ for some fixed constant $I_i$, 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.

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 $I_1 \geq I_2 \cdots \geq I_N$. Fix epoch index $s$, for all $i \neq 1$, variables $x_i^{s,k}$ with $k > I_i$ is never used. However, for the convenience of analysis, we define $x_i^{s,k} \triangleq x_i^{s,k-1}$, $\forall i \neq 1$, $\forall k \in \{I_i+1, \ldots, I_1\}$.

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 $I_1 = 10$ and $I_2 = 8$. Using the definition $x_i^{s,k} \triangleq \frac{1}{N} \sum_{i=1}^{N} x_i^{s,k}$, we have $x_i^{s,k} = x_i^{s,k-1} + \gamma \sum_{i:I_i \geq k} G_i^{s,k}$, $\forall s, \forall k \in \{1, 2, \ldots, I_1\}$.

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.

![Diagram of Algorithm 1 in a 2 worker heterogeneous network.](image-url)
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 $0 < \gamma \leq \frac{1}{T}$ in Algorithm 3, then for all $S \geq 1$,

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

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. \qed

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 $O(1/\sqrt{NT})$ communication reduction.

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

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

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 $\gamma, I_i, S$ into (16) in Theorem 3. \qed

**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 $I = I_N$)

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 rate 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., PRSGD (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 $O(1/\sqrt{T})$ 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**


