Adaptive Communication Strategies to Achieve the Best Error-Runtime Trade-off in Local-Update SGD

Anonymous Authors[1]

Abstract

Large-scale machine learning training, in particular distributed stochastic gradient descent, needs to be robust to inherent system variability such as node straggling and random communication delays. This work considers a distributed training framework where each worker node is allowed to perform local model updates and the resulting models are averaged periodically. We analyze the true speed of error convergence with respect to wall-clock time (instead of the number of iterations), and analyze how it is affected by the frequency of averaging. The main contribution is the design of AdaComm, *an adaptive communication strategy* that starts with infrequent averaging to save communication delay and improve convergence speed, and then increases the communication frequency in order to achieve a low error floor. Rigorous experiments on training deep neural networks show that AdaComm can take $3\times$ less time than fully synchronous SGD, and still reach the same final training loss.

1 Introduction

Stochastic gradient descent (SGD) is the backbone of state-of-the-art supervised learning, which is revolutionizing inference and decision-making in many diverse applications. Classical SGD was designed to be run on a single computing node, and its error-convergence with respect to the number of iterations has been extensively analyzed and improved via accelerated SGD methods. Due to the massive training data-sets and neural network architectures used today, it has became imperative to design distributed SGD implementations, where gradient computation and aggregation is parallelized across multiple worker nodes. Although parallelism boosts the amount of data processed per iteration, it exposes SGD to unpredictable node slowdown and communication delays stemming from variability in the computing infrastructure. Thus, there is a critical need to make distributed SGD fast, yet robust to system variability.

Need to Optimize Convergence in terms of Error versus Wall-clock Time. The convergence speed of distributed SGD is a product of two factors: 1) the error in the trained model versus the number of iterations, and 2) the number of iterations completed per second. Traditional single-node SGD analysis focuses on optimizing the first factor, because the second factor is generally a constant when SGD is run on a single dedicated server. In distributed SGD, which is often run on shared cloud infrastructure, the second factor depends on several aspects such as the number of worker nodes, their local computation and communication delays, and the protocol (synchronous, asynchronous or periodic) used to aggregate their gradients. Hence, in order to achieve the fastest convergence speed we need: 1) optimization techniques (eg. variable learning rate) to maximize the error-convergence rate with respect to iterations, and 2) scheduling techniques (eg. straggler mitigation, infrequent communication) to maximize the number of iterations completed per second. These directions are inter-dependent and need to be explored together rather than in isolation. While many works have advanced the first direction, the second is less explored from a theoretical point of view, and the juxtaposition of both is an unexplored problem.

Local-Update SGD to Reduce Communication Delays. A popular distributed SGD implementation is the parameter server framework (Dean et al., 2012; Cui et al., 2014; Li et al., 2014; Gupta et al., 2016; Mitliagkas et al., 2016) where in each iteration, worker nodes compute gradients on one mini-batch of data and a central parameter server aggregates these gradients (synchronously or asynchronously) and updates the parameter vector $\mathbf{x}$. The constant communication between the parameter server and worker nodes in each iteration can be expensive and slow in bandwidth-limited computed environments. Recently proposed distributed SGD frameworks such as Elastic-averaging (Zhang et al., 2015; Chaudhari et al., 2017), Federated Learning (McMahan et al., 2017; Smith et al., 2017b) and decentralized SGD (Lian et al., 2017; Jiang et al., 2017) save this communication cost by allowing worker nodes to perform

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Systems and Machine Learning (SysML) Conference. Do not distribute.

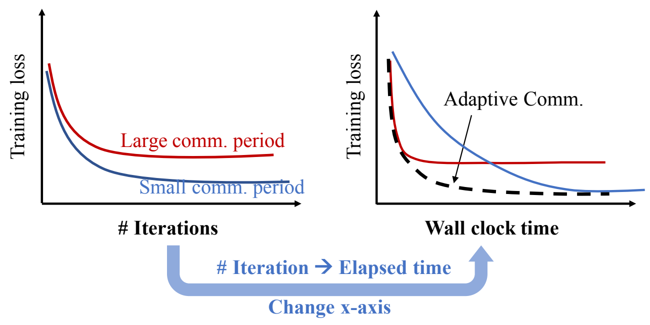


Figure 1. This work departs from the traditional view of considering error-convergence with respect to the number of iterations, and instead considers the true convergence in terms of error versus wall-clock time. Adaptive strategies that start with infrequent model-averaging and increase the communication frequency can achieve the best error-runtime trade-off.

local updates to the parameter $\mathbf{x}$ instead of just computing gradients. The resulting locally trained models (which are different due to variability in training data across nodes) are periodically averaged through a central server, or via direct inter-worker communication. Periodic averaging has been shown to offer significant speedup in deep neural network training (Moritz et al., 2015; Zhang et al., 2016; Su & Chen, 2015; Zhou & Cong, 2017; Lin et al., 2018).

Error-Runtime Trade-offs in Local-Update SGD. While local updates reduce the communication-delay incurred per iteration, discrepancies between the models can result in an inferior error-convergence. For example, consider the case of periodic averaging SGD where each of m worker nodes makes τ local updates, and the resulting models are averaged after every τ iterations. A larger value of τ leads to slower convergence with respect to the number of iterations as illustrated in Figure 1. However, if we look at the *true convergence with respect to the wall-clock time*, then a larger τ, that is, less frequent averaging, saves communication delay and reduces the runtime per iteration. While some recent theoretical works (Zhou & Cong, 2017; Yu et al., 2018; Wang & Joshi, 2018; Stich, 2018) study this dependence of the error-convergence with respect to the number of iterations as τ varies, achieving a provably-optimal speed-up in the true convergence with respect to wall-clock time is an open problem that we aim to address in this work.

Need for Adaptive Communication Strategies. In the error-runtime in Figure 1, we observe a trade-off between the convergence speed and the error floor when the number of local updates τ is varied. A larger τ gives a faster initial drop in the training loss but results in a higher error floor. This calls for *adaptive communication strategies* that start with a larger τ and gradually decrease it as the model reaches closer to convergence. Such an adaptive strategy will offer a win-win in the error-runtime trade-off by achieving fast convergence as well as low error floor. To the best of our knowledge, this is the first work to propose an adaptive communication frequency strategy.

Main Contributions. In this paper we consider periodic-averaging distributed SGD (PASGD), where each worker node performs τ local updates by processing one mini-batch of data per iteration. A fusion node takes a simple average of these local models and then the worker nodes start with the averaged model and perform the next τ local updates. The main contributions are as follows:

1. We provide the first runtime analysis of local-update SGD algorithms by modeling local computing time and communication delays as random variables, and quantifying the runtime speed-up in comparison with fully synchronous SGD. A novel insight from this analysis is that periodic averaging strategy not only reduces the communication delay but also mitigates the stragglers.

2. Combining the runtime analysis and previous error-convergence analysis of PASGD, we can obtain the error-runtime trade-off for different values of τ. Using this combined error-runtime trade-off, we derive an expression of the optimal communication period, which can serve as a useful guideline in practice.

3. We present a convergence analysis for PASGD with variable communication period τ and variable learning rate η, generalizing previous works (Zhou & Cong, 2017; Wang & Joshi, 2018). This analysis shows that decaying τ provides similar convergence benefits as decaying learning rate, the difference being that varying τ improves the true convergence with respect to the wall-clock time. Adaptive communication can also be used in conjunction with existing learning rate schedules.

4. Based on the observations in runtime and convergence analysis, we develop an adaptive communication scheme: AdaComm. Experiments on training VGG-16 and ResNet-50 deep neural networks and different settings (with/without momentum, fixed/decaying learning rate) show that AdaComm can give a $3\times$ runtime speed-up and still reach the same low training loss as fully synchronous SGD.

Although we focus on periodic simple-averaging of local models, the insights on error-runtime trade-offs and adaptive communication strategies are directly extendable to other communication-efficient SGD algorithms including Federated Learning (McMahan et al., 2017), Elastic-Averaging (Zhang et al., 2015) and Decentralized averaging (Jiang et al., 2017; Lian et al., 2017), as well as synchronous/asynchronous distributed SGD with a central parameter server (Dean et al., 2012; Cui et al., 2014; Dutta et al., 2018).

2 Problem Framework

Empirical Risk Minimization via Mini-batch SGD. Our objective is to minimize an objective function $F(\mathbf{x})$, the empirical risk function, with respect to model parameters denoted by $\mathbf{x} \in \mathbb{R}^d$. The training dataset is denoted by $\mathcal{S} = \{s_1, \dots, s_N\}$, where s_i represents the i-th labeled data point. The objective function can be expressed as the empirical risk calculated using the training data and is given by

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left[F(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}; s_i) \right] \tag{1}$$

where $f(\mathbf{x}; s_i)$ is the composite loss function at the i^{th} data point. In classic mini-batch stochastic gradient descent (SGD) (Dekel et al., 2012), updates to the parameter vector $\mathbf{x}$ are performed as follows. If $\xi_k \subset S$ represents a randomly sampled mini-batch, then the update rule is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta g(\mathbf{x}_k; \xi_k) \tag{2}$$

where η denotes the learning rate and the stochastic gradient is defined as: $g(\mathbf{x}; \xi) = \frac{1}{|\xi|}\sum_{s_i \in \xi} \nabla f(\mathbf{x}; s_i)$. For simplicity, we will use $g(\mathbf{x}_k)$ instead of $g(\mathbf{x}_k; \xi_k)$ in the rest of the paper. A complete review of convergence properties of serial SGD can be found in (Bottou et al., 2018).

Periodic Averaging SGD (PASGD). We consider a distributed SGD framework with m worker nodes where all workers can communicate with others via a central server or via direct inter-worker communication. In periodic averaging SGD, all workers start at the same initial point $\mathbf{x}_1$. Each worker performs τ local mini-batch SGD updates according to (2), and the local models are averaged by a fusion node or by performing an all-node broadcast. The workers then update their local models with the averaged model, as illustrated in Figure 2. Thus, the overall update rule at the i^{th} worker is given by

$$\mathbf{x}_{k+1}^{(i)} = \begin{cases} \frac{1}{m}\sum_{j=1}^{m}[\mathbf{x}_k^{(j)} - \eta g(\mathbf{x}_k^{(j)})], & k \bmod \tau = 0 \\ \mathbf{x}_k^{(i)} - \eta g(\mathbf{x}_k^{(i)}), & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbf{x}_k^{(i)}$ denote the model parameters in the i-th worker after k iterations and τ is defined as the communication period. Note that the iteration index k corresponds to the local iterations, and not the number of averaging steps.

Special Case ($\tau = 1$): Fully Synchronous SGD. When $\tau = 1$, that is, the local models are synchronized after every iteration, periodic-averaging SGD is equivalent to fully synchronous SGD which has the update rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \left[\frac{1}{m}\sum_{i=1}^{m} g(\mathbf{x}_k; \xi_k^{(i)}) \right]. \tag{4}$$

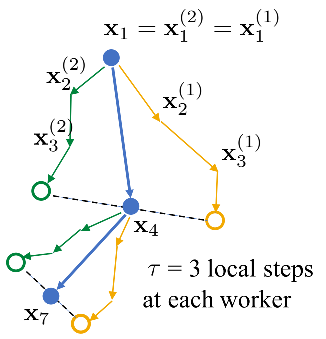


Figure 2. Illustration of Periodic averaging SGD (PASGD) in the model parameter space for $m = 2$ workers. The discrepancy between the local models increases with the number of local updates, $\tau = 3$.

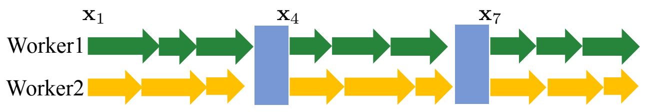


Figure 3. Illustration of PASGD in the time space for $m = 2$ and $\tau = 3$. Lengths of the colored arrows at the i^{th} worker are $Y_{i,k}$, the local-update times, which are i.i.d. across workers and updates. The blue block represents the communication delay for each model-averaging step.

The analysis of fully synchronous SGD is identical to serial SGD with m-fold large mini-batch size.

Local Computation Times and Communication Delay. In order to analyze the effect of τ on the expected runtime per iteration, we consider the following delay model. The time taken by the i^{th} worker to compute a mini-batch gradient at the k^{th} local-step is modeled a random variable $Y_{i,k} \sim F_Y$, assumed to be i.i.d. across workers and mini-batches. The communication delay is a random variable D for each all-node broadcast, as illustrated in Figure 3. The value of random variable D can depend on the number of workers as follows.

$$D = D_0 \cdot s(m) \tag{5}$$

where D_0 represents the time taken for each inter-node communication, and $s(m)$ describes how the delay scales with the number of workers, which depends on the implementation and system characteristics. For example, in the parameter server framework, the communication delay can be proportional to $2\log_2(m)$ by exploiting a reduction tree structure (Iandola et al., 2016). We assume that $s(m)$ is known beforehand for the communication-efficient distributed SGD framework under consideration.

Convergence Criteria. In the error-convergence analysis, since the objective function is non-convex, we use the expected gradient norm as a an indicator of convergence following (Ghadimi & Lan, 2013; Bottou et al., 2018). We say the algorithm achieves an ϵ-suboptimal solution if:

$$\mathbb{E}\left[\min_{k\in[1,K]} \|\nabla F(\mathbf{x}_k)\|^2\right] \leq \epsilon. \tag{6}$$

When ϵ is arbitrarily small, this condition can guarantee the algorithm converges to a stationary point.

3 Jointly Analyzing Runtime and Error-Convergence

3.1 Runtime Analysis

We now present a comparison of the runtime per iteration of periodic averaging SGD with fully synchronous SGD to illustrate how increasing τ can lead to a large runtime speed-up. Another interesting effect of performing more local update τ is that it mitigates the slowdown due to straggling worker nodes.

Runtime Per Iteration of Fully Synchronous SGD. Fully synchronous SGD is equivalent to periodic averaging SGD with $\tau = 1$. Each of the m workers computes the gradient of one mini-batch and updates the parameter vector $\mathbf{x}$, which takes time $Y_{i,1}$ at the i^{th} worker[1]. After all workers finish their local updates, an all-node broadcast is performed to synchronize and average the models. Thus, the total time to complete each iteration is given by

$$T_{\text{sync}} = \max(Y_{1,1}, Y_{2,1}, \dots, Y_{m,1}) + D \tag{7}$$
$$\mathbb{E}\left[T_{\text{sync}}\right] = \mathbb{E}[Y_{m:m}] + \mathbb{E}[D] \tag{8}$$

where $Y_{i,1}$ are i.i.d. random variables with probability distribution F_Y and D is the communication delay. The term $Y_{m:m}$ denotes the highest order statistic of m i.i.d. random variables (David & Nagaraja, 2003).

Runtime Per Iteration of Periodic Averaging SGD (PASGD). In periodic averaging SGD, each worker performs τ local updates before communicating with other workers. Let us denote the average local computation time at the i^{th} worker by

$$\overline{Y}_i = \frac{Y_{i,1} + Y_{i,2} + \dots Y_{i,\tau}}{\tau} \tag{9}$$

Since the communication delay D is amortized over τ iterations, the average computation time per iteration is

$$T_{\text{P-Avg}} = \max(\overline{Y}_1, \overline{Y}_2, \dots, \overline{Y}_m) + \frac{D}{\tau} \tag{10}$$
$$\mathbb{E}[T_{\text{P-Avg}}] = \mathbb{E}[\overline{Y}_{m:m}] + \frac{\mathbb{E}[D]}{\tau} \tag{11}$$

[1]Instead of local updates, typical implementations of fully synchronous SGD have a central server that performs the update. Here we compare PASGD with fully synchronous SGD without a central parameter server.

The value of the first term $\overline{Y}_{m:m}$ and how it compares with $Y_{m:m}$ depends on the probability distribution F_Y of Y. We can obtain the following distribution-independent bound on the runtime of PASGD that only depends on the mean and the variance of Y.

Theorem 1 (Upper Bound on the Runtime per Iteration). *Suppose each worker takes time $Y \sim F_Y$ to compute gradients and perform a local update, which is i.i.d. across workers and mini-batches. The mean and the variance of Y are μ_Y and σ_Y^2 respectively. Then,*

$$\mathbb{E}\left[T_{\text{P-Avg}}\right] \leq \mu_Y + \sigma_Y \sqrt{\frac{m-1}{\tau}} + \frac{\mathbb{E}[D]}{\tau}. \tag{12}$$

The proof follows from the bound on expected order statistics given by (Arnold & Groeneveld, 1979). Observe that as we increase τ, the runtime bound in Theorem 1 decreases in two ways: 1) τ-fold reduction in the communication delay and 2) reduction in the variance of the maximum of the local computation times, that is, reduction in additional delay due to slow or straggling workers.

Speed-up over fully synchronous SGD. We now evaluate the speed-up of periodic-averaging SGD over fully synchronous SGD for different Y and D to demonstrate how the relative value of computation versus communication delays affects the speed-up. Consider the simplest case where Y and D are constants and $\alpha = D/Y$, the communication/computation ratio. Besides systems aspects such as network bandwidth and computing capacity, for deep neural network training, this ratio α also depends on the size of the neural network model and the mini-batch size. See Figure 8 for a comparison of the communication/computation delays of common deep neural network architectures. Then $\overline{Y}$, $Y_{m:m}$, $\overline{Y}_{m:m}$ are all equal to Y, and the ratio of $\mathbb{E}[T_{\text{sync}}]$ and $\mathbb{E}[T_{\text{P-Avg}}]$ is given by

$$\frac{\mathbb{E}\left[T_{\text{sync}}\right]}{\mathbb{E}\left[T_{\text{P-Avg}}\right]} = \frac{Y + D}{Y + D/\tau} = \frac{1+\alpha}{1+\alpha/\tau} \tag{13}$$

Figure 4 shows the speed-up for different values of α and τ. *When D is comparable with Y ($\alpha = 0.9$), periodic-averaging SGD (PASGD) can be almost twice as fast as fully synchronous SGD.*

Straggler Mitigation due to Local Updates. Suppose that Y is exponentially distributed with mean y and variance y^2. For fully synchronous SGD, the term $\mathbb{E}[Y_{m:m}]$ in (8) is equal to $y\sum_{i=1}^{m} 1/i$, which is approximately equal to $y \log m$. Thus, the expected runtime per iteration of fully synchronous SGD (8) increases logarithmically with the number of workers m. Let us compare this with the scaling of the runtime of periodic-averaging SGD (11). Here, $\overline{Y}$ (9) is an Erlang random variable with mean y and variable y^2/τ. Since the variance is τ times smaller than that of

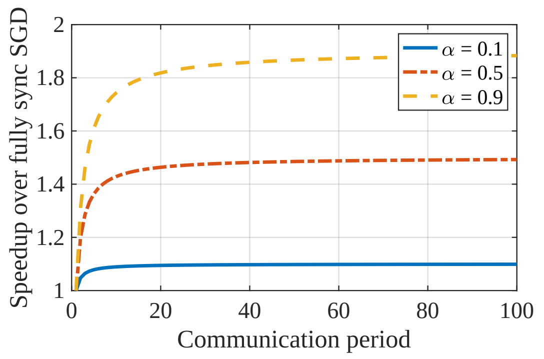


Figure 4. The speed-up offered by using periodic-averaging SGD increases with τ (the communication period) and with the communication/computation delay ratio $\alpha = D/Y$, where D is the all-node broadcast delay and Y is the time taken for each local update at a worker.

Y, the maximum order statistic $\mathbb{E}[\overline{Y}_{m:m}]$ is smaller than $\mathbb{E}[Y_{m:m}]$. Figure 5 shows the probability distribution of T_{sync} and $T_{\text{P-Avg}}$ for exponentially distributed Y. Observe that $T_{\text{P-Avg}}$ has a much lighter tail. This is because the effect of the variability in Y on $T_{\text{P-Avg}}$ is reduced due to the Y in (8) being replaced by $\overline{Y}$ (which has lower variance) in (11).

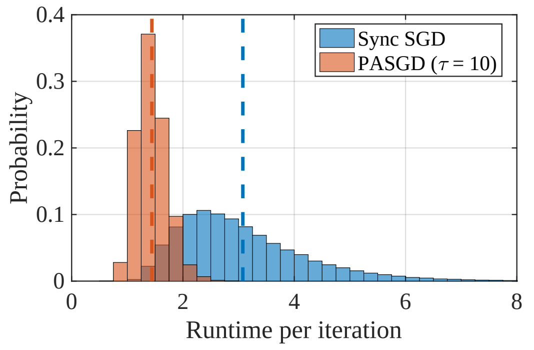


Figure 5. Probability distribution of runtime per iteration, where communication delay $D = 1$, mean computation time $y = 1$, and number of workers $m = 16$. Dash lines represent the mean values.

3.2 Joint Analysis with Error-convergence

In this subsection, we combine the runtime analysis with previous error-convergence analysis for PASGD (Wang & Joshi, 2018). Due to space limitations, we state the necessary theoretical assumptions in the Appendix; the assumptions are similar to previous works (Zhou & Cong, 2017; Wang & Joshi, 2018) on the convergence of local-update SGD algorithms.

Theorem 2 (**Error-runtime Convergence of PASGD**). *For PASGD, under certain assumptions (stated in the Appendix), if the learning rate satisfies $\eta L+\eta^2L^2\tau(\tau-1) \leq 1$ and all workers are initialized at the same point $\mathbf{x}_1$, then after total T wall-clock time, the minimal expected squared gradient norm within T time interval will be bounded by:*

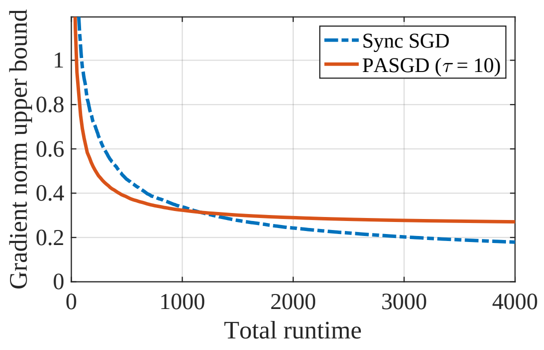


Figure 6. Illustration of theoretical error bound versus runtime in Theorem 2. The runtime per iteration is generated under the same parameters as Figure 5. Other constants in (14) are set as follows: $F(\mathbf{x}_1) = 1, F_{\text{inf}} = 0, \eta = 0.08, L = 1, \sigma^2 = 1$.

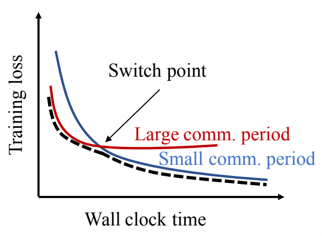
$$\frac{2\left[F(\mathbf{x}_1) - F_{inf}\right]}{\eta T}\left(\mathbb{E}\left[\overline{Y}_{m:m}\right] + \frac{\mathbb{E}\left[D\right]}{\tau}\right) + \frac{\eta L \sigma^2}{m} + \eta^2 L^2 \sigma^2 (\tau - 1) \quad (14)$$

where L is the Lipschitz constant of the objective function and σ^2 is the variance bound of mini-batch stochastic gradients.

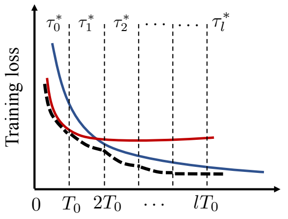
The proof of Theorem 2 is presented in the Appendix. From the optimization error upper bound (14), one can easily observe the error-runtime trade-off for different communication periods. While a larger τ reduces the runtime per iteration and let the first term in (14) become smaller, it also adds additional noise and increases the last term. In Figure 6, we plot theoretical bounds for both fully synchronous SGD ($\tau = 1$) and PASGD. It is shown that although PASGD with $\tau = 10$ starts with a rapid drop, it will eventually converge to a high error floor. This theoretical result is also corroborated by experiments in Section 5. Another direct outcome of Theorem 2 is the determination of the best communication period that balances the first and last terms in (14). We will discuss the selection of communication period later in Section 4.1.

4 AdaComm: Proposed Adaptive Communication Strategy

Inspired by the clear trade-off in the learning curve in Figure 6, it would be better to have an adaptive communication strategy that starts with infrequent communication to im-



(a) Switch between curves.



(b) Choose the best τ for each time interval.

Figure 7. Illustration of communication period adaptation strategies. Dash line denotes the learning curve using adaptive communication.

prove convergence speed, and then increases the frequency to achieve a low error floor. In this section, we are going to develop the proposed adaptive communication scheme.

The basic idea to adapt the communication is to choose the communication period that minimizes the optimization error at each wall-clock time. One way to achieve the idea is switching between the learning curves at their intersections. However, without prior knowledge of various curves, it would be difficult to determine the switch points.

Instead, we divide the whole training procedure into uniform wall-clock time intervals with the same length T_0. At the beginning of each time interval, we select the best value of τ that has the fastest decay rate in the next T_0 wall-clock time. If the interval length T_0 is small enough and the best choice of communication period for each interval can be precisely estimated, then this adaptive scheme should achieve a win-win in the error-runtime trade-off as illustrated in Figure 7.

After setting the interval length, the next question is how to estimate the best communication period for each time interval. In Section 4.1 we use the error-runtime analysis in Section 3.2 to find the best τ at each time.

4.1 Determining the Best Communication Period for Each Time Interval

From Theorem 2, it can be observed that there is an optimal value τ^* that minimizes the optimization error bound at given wall-clock time. In particular, consider the simplest setting where Y and D are constants. Then, by minimizing the upper bound (14) over τ, we obtain the following.

Theorem 3. *For PASGD, under the same assumptions as Theorem 2, the optimization error upper bound in (14) at time T is minimized when the communication period is*

$$\tau^* = \sqrt{\frac{2(F(\mathbf{x}_1) - F_{inf})D}{\eta^3 L^2 \sigma^2 T}}. \tag{15}$$

The proof is straightforward by setting the derivative of (14) to zero. We present the details in the Appendix. Suppose all workers starts from the same initial point $\mathbf{x}_1 = \mathbf{x}_{t=0}$ where subscript t denotes the wall-clock time. Directly applying Theorem 3 to the first time interval, then the best choice of communication period is:

$$\tau_0 = \sqrt{\frac{2(F(\mathbf{x}_{t=0}) - F_{\text{inf}})D}{\eta^3 L^2 \sigma^2 T_0}}. \tag{16}$$

Similarly, for the l-th time interval, workers can be viewed as restarting training at a new initial point $\mathbf{x}_{t=lT_0}$. Applying Theorem 3 again, we have

$$\tau_l = \sqrt{\frac{2(F(\mathbf{x}_{t=lT_0}) - F_{\text{inf}})D}{\eta^3 L^2 \sigma^2 T_0}}. \tag{17}$$

Comparing (16) and (17), it is easy to see the generated communication period sequence decreases along with the objective value $F(\mathbf{x}_t)$. This result is consistent with the intuition that the trade-off between error-convergence and communication-efficiency varies over time. Compared to the initial phase of training, the benefit of using a large communication period diminishes as the model reaches close to convergence. At this later stage, a lower error floor is more preferable to speeding up the runtime.

Practical SGD implementations generally decay the learning rate or increase the mini-batch size (Smith et al., 2017a; Goyal et al., 2017), in order to reduce the variance of the gradient updates. As we saw from the convergence analysis Theorem 2, performing local updates adds noise in stochastic gradients, resulting in a higher error floor at the end of training. Decaying the communication period can gradually reduce the variance of gradients and yield a similar improvement in convergence. Thus, *adaptive communication strategies* are similar in spirit to decaying learning rate or increasing mini-batch size. *The key difference is that here we are optimizing the true error convergence with respect to wall-clock time rather than the number iterations.*

4.2 Practical Considerations

Although (16) and (17) provide useful insights about how to adapt τ over time, it is still difficult to directly use them in practice due to the Lipschitz constant L and the gradient variance bound σ^2 being unknown. For deep neural networks, estimating these constants can be difficult and unreliable due to the highly non-convex and high-dimensional loss surface. As an alternative, we propose a simpler rule where we approximate F_{inf} by 0, and divide (17) by (16) to obtain the basic communication period update rule:

$$\textbf{Basic update rule} \quad \tau_l = \left\lceil \sqrt{\frac{F(\mathbf{x}_{t=lT_0})}{F(\mathbf{x}_{t=0})}} \tau_0 \right\rceil \tag{18}$$

where $\lceil a \rceil$ is the ceil function to round a to the nearest integer $\geq a$. Since the objective function values (i.e., training loss) $F(\mathbf{x}_{t=lT_0})$ and $F(\mathbf{x}_{t=0})$ can be easily obtained in the training, the only remaining thing now is to determine the initial communication period τ_0. We obtain a heuristic estimate of τ_0 by a simple grid search over different τ run for one or two epochs each.

4.3 Refinements to the Proposed Adaptive Strategy

4.3.1 Faster Decay When Training Saturates

The communication period update rule (18) tends to give a decreasing sequence $\{\tau_l\}$. Nonetheless, it is possible that the best value of τ_l for next time interval is larger than the current one due to random noise in the training process. Besides, when the training loss can get stuck on plateaus and decrease very slowly, (18) will result in τ_l saturating at the same value for a long time. To address this issue, we borrow a idea used in classic SGD where the learning rate is decayed by a factor γ when the training loss saturates for several epochs (Goyal et al., 2017). Similarly, in the our scheme, the communication period will be multiplied by $\gamma < 1$ when the τ_l given by (18) is not strictly less than τ_{l-1}. To be specific, the communication period for the l^{th} time interval will be determined as follows:

$$\tau_l = \begin{cases} \left\lceil \sqrt{\frac{F(\mathbf{x}_{t=lT_0})}{F(\mathbf{x}_{t=0})}}\tau_0 \right\rceil, & \text{if } \left\lceil \sqrt{\frac{F(\mathbf{x}_{t=lT_0})}{F(\mathbf{x}_{t=0})}}\tau_0 \right\rceil < \tau_{l-1} \\ \gamma\tau_{l-1}, & \text{otherwise} \end{cases}. \tag{19}$$

In the experiments, $\gamma = 1/2$ turns out to be a good choice. One can obtain a more aggressive decay in τ_l by either reducing the value of γ or introducing a slack variable s in the condition, such as $\lceil \sqrt{\frac{F(\mathbf{x}_{t=lT_0})}{F(\mathbf{x}_{t=0})}}\tau_0 \rceil + s < \tau_{l-1}$.

4.3.2 Incorporating Adaptive Learning Rate

So far we consider a fixed learning rate η for the local SGD updates at the workers. We now present an adaptive communication strategy that adjusts τ_l for a given variable learning rate schedule, in order to obtain the best error-runtime trade-off. Suppose η_l denotes the learning rate for the l^{th} time interval. Then, combining (16) and (17) again, we have

$$\tau_l = \left\lceil \sqrt{\frac{\eta_0^3}{\eta_l^3}\frac{F(\overline{\mathbf{x}}_{t=lT_0})}{F(\overline{\mathbf{x}}_{t=0})}}\tau_0 \right\rceil. \tag{20}$$

Observe that when the learning rate becomes smaller, the communication period τ_l increases. This result corresponds the intuition that a small learning rate reduces the discrepancy between the local models, and hence is more tolerant to large communication periods.

(20) states that the communication period should be proportional to $(\eta_0/\eta_l)^{3/2}$. However, in practice, it is common to decay the learning rate 10 times after some given number of epochs. The dramatic change of learning rate may push the communication period to an unreasonably large value. In the experiments with momentum SGD, we observe that when applying (20), the communication period can increase to $\tau = 1000$ which causes the training loss to diverge.

To avoid this issue, we propose the adaptive strategy given by (21) below. This strategy can also be justified by theoretical analysis. Suppose that in l^{th} time interval, the objective function has a local Lipschitz smoothness L_l. Then, by using the approximation $\eta_l L_l \approx 1$, which is common in SGD literature (Balles et al., 2016), we derive the following adaptive strategy:

$$\tau_l = \left\lceil \sqrt{\frac{\eta_0^3 L_0^2}{\eta_l^3 L_l^2}\frac{F(\mathbf{x}_{t=lT_0})}{F(\mathbf{x}_{t=0})}}\tau_0 \right\rceil \approx \left\lceil \sqrt{\frac{\eta_0}{\eta_l}\frac{F(\mathbf{x}_{t=lT_0})}{F(\mathbf{x}_{t=0})}}\tau_0 \right\rceil. \tag{21}$$

Apart from coupling the communication period with learning rate, when to decay the learning rate is another key design factor. In order to eliminate the noise introduced by local updates, we choose to first gradually decay the communication period to 1 and then decay the learning rate as usual. For example, if the learning rate is scheduled to be decayed at the 80^{th} epoch but at that time the communication period τ is still larger than 1, then we will continue use the current learning rate until $\tau = 1$.

4.4 Theoretical Guarantees for the Convergence of AdaComm

In this subsection, we are going to provide a convergence guarantee for the proposed adaptive communication scheme by extending the error analysis for PASGD. Without loss of generality, we will analyze an arbitrary communication period sequence $\{\tau_0, \ldots, \tau_R\}$, where R represents the total communication rounds[2]. It will be shown that a decreasing sequence of τ is beneficial to the error-convergence rate.

Theorem 4 (Convergence of adaptive communication scheme). *For PASGD with adaptive communication period and adaptive learning rate, suppose the learning rate remains same in each local update period. If the following conditions are satisfied as $R \to \infty$,*

$$\sum_{r=0}^{R} \eta_r\tau_r \to \infty, \; \sum_{r=0}^{R} \eta_r^2\tau_r \to 0, \; \sum_{r=0}^{R} \eta_r^3\tau_r^2 \to 0, \tag{22}$$

[2]Note that in the error analysis, the subscripts of communication period and learning rate represent the index of local update periods rather than the index of the T_0-length wall-clock time intervals as considered in Sections 4.1-4.3.

then the averaged model $\overline{\mathbf{x}}$ is guaranteed to converge to a stationary point:

$$\mathbb{E}\left[\frac{\sum_{r=0}^{R-1}\eta_r\sum_{k=1}^{\tau_r}\|\nabla F(\overline{\mathbf{x}}_{s_r+k})\|^2}{\sum_{r=0}^{R-1}\eta_r\tau_r}\right]\to 0 \qquad (23)$$

where $s_r=\sum_{j=0}^{r-1}\tau_j$.

The proof details and a non-asymptotic result (similar to Theorem 2 but with variable τ) are provided in Appendix. In order to understand the meaning of condition (22), let us first consider the case when $\tau_0=\cdots=\tau_R$ is a constant. In this case, the convergence condition is identical to mini-batch SGD (Bottou et al., 2018):

$$\sum_{r=0}^{R}\eta_r\to\infty,\ \sum_{r=0}^{R}\eta_r^2\to 0. \qquad (24)$$

As long as the communication period sequence is bounded, it is trivial to adapt the learning rate scheme in mini-batch SGD (24) to satisfy (22). In particular, when the sequence is decreasing, the last two terms in (22) will have faster rate to converge to zero. As a result, (23) also benefits from the faster rate and the local-update SGD algorithm has better convergence guarantee.

5 Experimental Results

5.1 Experimental Setting

Platform. The proposed adaptive communication scheme was implemented in `Pytorch` (Paszke et al., 2017) with `Mpi4Py` (Dalcín et al., 2005). All experiments were conducted on a local cluster where each worker node has an NVIDIA TitanX GPU and 16-core Intel Xeon CPU.

Dataset. We evaluate our method for image classification tasks on CIFAR10 and CIFAR100 dataset (Krizhevsky, 2009), which consists of 50,000 training images and 10,000 validation images in 10 and 100 classes respectively. Each worker machine is assigned with a partition which will be randomly shuffled after every epoch.

Model. We choose to train deep neural networks VGG-16 (Simonyan & Zisserman, 2014) and ResNet-50 (He et al., 2016) from scratch. These two neural networks have different architectures and parameter sizes, thus resulting in different performance of periodic averaging. As shown in Figure 8, for VGG-16, the communication time is about 4 times higher than the computation time. Thus, compared to ResNet-50, it requires a larger τ in order to reduce the runtime-per-iteration and achieve fast convergence.

Moreover, unless otherwise stated, we used 4 worker nodes and mini-batch size on each worker is 128. Therefore, the total mini-batch size per iteration is 512. The initial learning

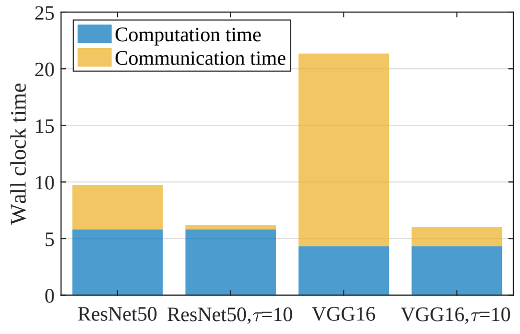


Figure 8. Wall-clock time to finish 100 iterations in a cluster with 4 worker nodes. To achieve the same level communication/computation ratio, VGG-16 requires larger communication period than ResNet-50.

rates for VGG-16 and ResNet-50 are 0.2 and 0.4 respectively. The weight decay for both networks is 0.0005. In the variable learning rate setting, we decay the learning rate by 10 after 80th/120th/160th/200th epochs. We set the time interval length T_0 as 60 seconds (about 10 epochs for the initial communication period).

Metrics. We compare the performance of proposed adaptive communication scheme with following methods with a fixed communication period: (1) Baseline: fully synchronous SGD ($\tau=1$); (2) Extreme high throughput case where $\tau=100$; (3) Manually tuned case where a moderate value of τ is selected after trial runs with different communication periods. Instead of training for a fixed number of epochs, we train all methods for sufficiently long time to convergence and compare the training loss and test accuracy, both of which are recorded after every 100 iterations.

5.2 Adaptive Communication in PASGD

We first validate the effectiveness of AdaComm which uses the communication period update rule (19) combined with (21) on original PASGD without momentum.

Figure 9 presents the results for VGG-16 for both fixed and variable learning rates. A large communication period τ initially results in a rapid drop in the error, but the error finally converges to higher floor. By adapting τ, the proposed AdaComm scheme strikes the best error-runtime trade-off in all settings. In Figure 9a, while fully synchronous SGD takes 33.5 minutes to reach 3×10^{-3} training loss, AdaComm costs 15.5 minutes achieving more than $2\times$ speedup. Similarly, in Figure 9b, AdaComm takes 11.5 minutes to reach 4.5×10^{-2} training loss achieving $3.3\times$ speedup over fully synchronous SGD (38.0 minutes).

However, for ResNet-50, the communication overhead is no longer the bottleneck. For fixed communication period, the negative effect of performing local updates becomes more obvious and cancels the benefit of low communication delay (see Figures 10b and 10c). It is not surprising to see fully synchronous SGD is nearly the best one in the error-runtime plot among all fixed-τ methods. Even in this extreme case, adaptive communication can still have a competitive performance. When combined with learning rate decay, the adaptive scheme is about 1.3 times faster than fully synchronous SGD (see Figure 10a).

Table 1 lists the test accuracies in different settings; we report the best accuracy within a time budget for each setting. The results show that adaptive communication method have better generalization than fully synchronous SGD. In the variable learning rate case, the adaptive method even gives the better test accuracy than PASGD with the best fixed τ.

Table 1. Best test accuracies on CIFAR10 in different settings (SGD without momentum).

MODEL	METHODS	FIXED LR	VARIABLE LR
VGG-16	$\tau = 1$	90.5	92.75
	$\tau = 20$	**92.25**	92.5
	$\tau = 100$	92.0	92.4
	ADACOMM	91.1	**92.85**
RESNET-50	$\tau = 1$	88.76	92.26
	$\tau = 5$	**90.42**	92.26
	$\tau = 100$	88.66	91.8
	ADACOMM	89.57	**92.42**

5.3 Adaptive Communication in Momentum SGD

The adaptive communication scheme is proposed based on the joint error-runtime analysis for PASGD without momentum. However, it can also be extended to other SGD variants, and in this subsection, we show that the proposed method works well for SGD with momentum.

5.3.1 *Block Momentum in Periodic Averaging*

Before presenting the empirical results, it is worth describing how to introduce momentum in PASGD. The most straightforward way is to apply the momentum independently to each local model, where each worker maintains an independent momentum buffer, which is the latest change in the parameter vector $\mathbf{x}$. However, this does not account for the potential dramatic change in $\mathbf{x}$ at each averaging step. When local models are synchronized, the local momentum buffer will contain the update steps before averaging, resulting in a large momentum term in the first SGD step of the next local update period. When the communication period is large, this large momentum term can sidetrack the SGD descent direction resulting in slower convergence.

To address this issue, a *block momentum scheme* was proposed in (Chen & Huo, 2016) and applied to speech recognition tasks. The basic idea is treating the accumulated local updates in one period as one big gradient step between two synchronized models and introducing a global momentum for this big accumulated step. The update rule can be written as follows in terms of the momentum $\mathbf{u}_j$:

$$\mathbf{u}_j = \beta_{\text{glob}}\mathbf{u}_{j-1} + \mathcal{G}_j \quad (25)$$

$$\mathbf{x}_{(j+1)\tau+1} = \mathbf{x}_{j\tau+1} - \eta_j \mathbf{u}_j \quad (26)$$

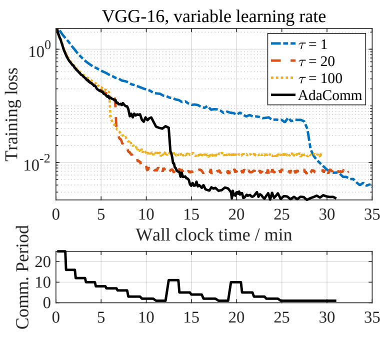
where $\mathcal{G}_j = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{\tau} g(\mathbf{x}_{j\tau+k}^{(i)})$ represents the accumulated gradients in the j^{th} local update period and β_{glob} denotes the global momentum factor. Moreover, workers can also conduct momentum SGD on local models, but their local momentum buffer will be cleared at the beginning of each local update period. That is, we restart momentum SGD on local models after every averaging step. The same strategy was also suggested in Microsoft's CNTK framework (Seide & Agarwal, 2016). In our experiments, we set the global momentum factor as 0.3 and local momentum factor as 0.9 following (Lin et al., 2018). In the fully synchronous case, there is no need to introduce the block momentum and we simply follow the common practice setting the momentum factor as 0.9.

5.3.2 ADACOMM *plus Block Momentum*

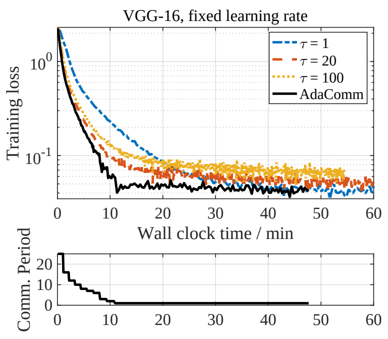
We applied our adaptive communication strategy in PASGD with block momentum and observed significant performance gain on CIFAR10/100 (see Figure 11). In particular, the adaptive communication scheme has the fastest convergence rate with respect to wall-clock time in the whole training process. While fully synchronous SGD gets stuck with a plateau before the first learning rate decay, the training loss of adaptive method continuously decreases until converging. For VGG-16 in Figure 11b, ADACOMM is $3.5\times$ faster (in terms of wall-clock time) than fully synchronous SGD in reaching a 3×10^{-3} training loss. For ResNet-50 in Figure 11a, ADACOMM takes 15.8 minutes to get 2×10^{-2} training loss which is 2 times faster than fully synchronous SGD (32.6 minutes).

6 CONCLUDING REMARKS

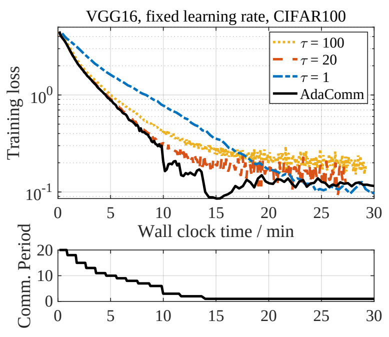
The design of fast communication-efficient distributed SGD algorithms that are robust to system variability is vital to enable machine learning training to scale to resource-limited computing nodes. This paper is one of the first to analyze the convergence of error with respect to wall-clock time instead of number of iterations by accounting for the dependence of runtime per iteration on systems aspects such as computation and communication delays. We present a theoretical analysis of the error-runtime trade-off for periodic averaging SGD (PASGD), where each worker node performs local updates and their models are averaged after



(a) Variable learning rate on CIFAR10.

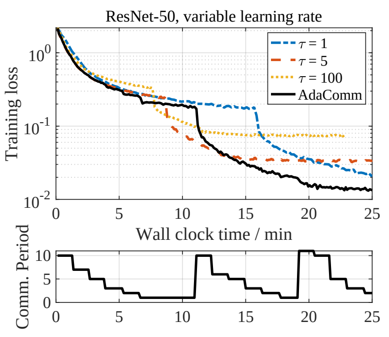


(b) Fixed learning rate on CIFAR10.

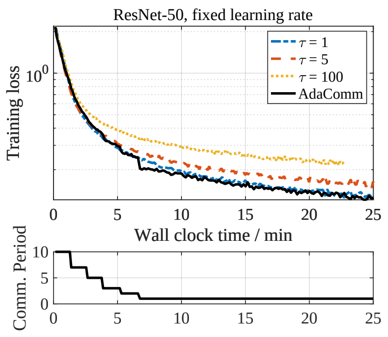


(c) Fixed learning rate on CIFAR100.

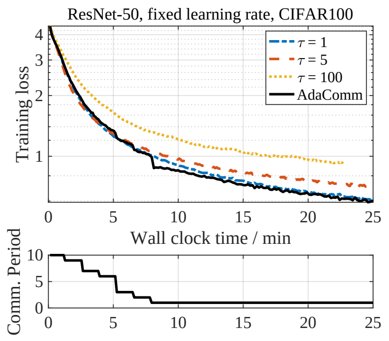
Figure 9. AdaComm on VGG-16: Achieves 3.3× speedup over fully synchronous SGD (in (b), 11.5 versus 38.0 minutes to achieve 4.5×10^{-2} training loss).



(a) Variable learning rate on CIFAR10.

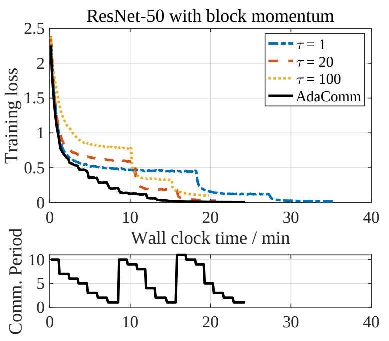


(b) Fixed learning rate on CIFAR10.

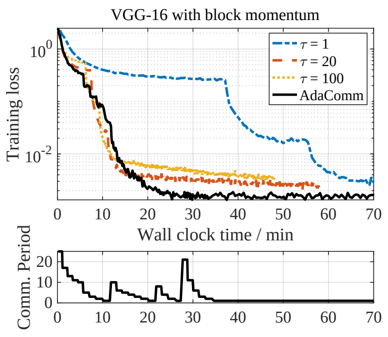


(c) Fixed learning rate on CIFAR100.

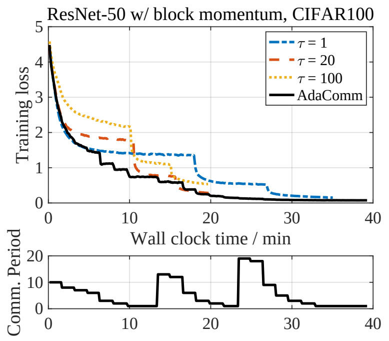
Figure 10. AdaComm on ResNet-50: Achieves 1.4× speedup over Sync SGD (in (a), 18.8 versus 26.6 minutes to achieve 2×10^{-2} training loss).



(a) ResNet-50 on CIFAR10.



(b) VGG-16 on CIFAR10.



(c) ResNet-50 on CIFAR100.

Figure 11. AdaComm with block momentum: Achieves 3.5× speedup over Sync SGD (in (b), 19.0 versus 66.7 minutes to achieve 3×10^{-3} training loss).

every τ iterations. Based on the joint error-runtime analysis, we design the first (to the best of our knowledge) adaptive communication strategy called AdaComm for distributed deep learning. Experimental results using VGGNet and ResNet show that the proposed method can achieve up to a 3× improvement in runtime, while achieving the same error floor as fully synchronous SGD. Going beyond periodic-averaging SGD, our idea of adapting frequency of averaging distributed SGD updates can be easily extended to other SGD frameworks including elastic-averaging (Zhang et al., 2015), decentralized SGD (Lian et al., 2017) and parameter server-based training (Dean et al., 2012).

References

Arnold, B. C. and Groeneveld, R. A. Bounds on expectations of linear systematic statistics based on dependent samples. *The Annals of Statistics*, 7(1):220–223, January 1979. doi: 10.1214/aos/1176344567. URL https://doi.org/10.1214/aos/1176344567.

Balles, L., Romero, J., and Hennig, P. Coupling adaptive batch sizes with learning rates. *arXiv preprint arXiv:1612.05086*, 2016.

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, 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 *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pp. 37–48, 2014.

Dalcín, L., Paz, R., and Storti, M. MPI for python. *Journal of Parallel and Distributed Computing*, 65(9):1108–1115, 2005.

David, H. A. and Nagaraja, H. N. *Order statistics*. John Wiley, Hoboken, N.J., 2003.

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.

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

Dutta, S., Joshi, G., Ghosh, S., Dube, P., and Nagpurkar, P. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, pp. 803–812. PMLR, 2018.

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

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

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

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.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., and Keutzer, K. Firecaffe: near-linear acceleration of deep neural network training on compute clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2592–2600, 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. 5906–5916, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

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

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. 5336–5346, 2017.

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

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-Efficient Learning of Deep Networks from Decentralized Data. *International Conference on Artificial Intelligenece and Statistics (AISTATS)*, April 2017. URL https://arxiv.org/abs/1602.05629.

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

Moritz, P., Nishihara, R., Stoica, I., and Jordan, M. I. SparkNet: Training deep networks in spark. *arXiv preprint arXiv:1511.06051*, 2015.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

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

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Smith, S. L., Kindermans, P.-J., and Le, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017a.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 4424–4434. 2017b. URL http://papers.nips.cc/paper/7029-federated-multi-task-learning.pdf.

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

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.

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 *NIPS'15 Proceedings of the 28th International Conference on 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.