Asynchronous Stochastic Gradient Descent with Delay Compensation

Shuxin Zheng [1] **Qi Meng** [2] **Taifeng Wang** [3] **Wei Chen** [3] **Nenghai Yu** [2] **Zhi-Ming Ma** [4] **Tie-Yan Liu** [3]

Abstract

With the fast development of deep learning, it has become common to learn big neural networks using massive training data. Asynchronous Stochastic Gradient Descent (ASGD) is widely adopted to fulfill this task for its efficiency, which is, however, known to suffer from the problem of delayed gradients. That is, when a local worker adds its gradient to the global model, the global model may have been updated by other workers and this gradient becomes "delayed". We propose a novel technology to compensate this delay, so as to make the optimization behavior of ASGD closer to that of sequential SGD. This is achieved by leveraging Taylor expansion of the gradient function and efficient approximation to the Hessian matrix of the loss function. We call the new algorithm Delay Compensated ASGD (DC-ASGD). We evaluated the proposed algorithm on CIFAR-10 and ImageNet datasets, and the experimental results demonstrate that DC-ASGD outperforms both synchronous SGD and asynchronous SGD, and nearly approaches the performance of sequential SGD.

1. Introduction

Deep Neural Networks (DNN) have pushed the frontiers of many applications, such as speech recognition (Sak et al., 2014; Sercu et al., 2016), computer vision (Krizhevsky et al., 2012; He et al., 2016; Szegedy et al., 2016), and natural language processing (Mikolov et al., 2013; Bahdanau et al., 2014; Gehring et al., 2017). Part of the success of DNN should be attributed to the availability of big training data and powerful computational resources, which allow people to learn very deep and big DNN models in parallel (Zhang et al., 2015; Chen & Huo, 2016; Chen et al., 2016).

Stochastic Gradient Descent (SGD) is a popular optimization algorithm to train neural networks (Bottou, 2012; Dean et al., 2012; Kingma & Ba, 2014). As for the parallelization of SGD algorithms (suppose we use M machines for the parallelization), one can choose to do it in either a synchronous or asynchronous way. In synchronous SGD (SSGD), local workers compute the gradients over their own mini-batches of data, and then add the gradients to the global model. By using a barrier, these workers wait for each other, and will not continue their local training until the gradients from all the M workers have been added to the global model. It is clear that the training speed will be dragged by the slowest worker[1]. To improve the training efficiency, asynchronous SGD (ASGD) (Dean et al., 2012) has been adopted, with which no barrier is imposed, and each local worker continues its training process right after its gradient is added to the global model. Although ASGD can achieve faster speed due to no waiting overhead, it suffers from another problem which we call *delayed gradient*. That is, before a worker wants to add its gradient $g(\mathbf{w}_t)$ (calculated based on the model snapshot $\mathbf{w}_t$) to the global model, several other workers may have already added their gradients and the global model has been updated to $\mathbf{w}_{t+\tau}$ (here τ is called the delay factor). Adding gradient of model $\mathbf{w}_t$ to another model $\mathbf{w}_{t+\tau}$ does not make a mathematical sense, and the training trajectory may suffer from unexpected turbulence. This problem has been well known, and some researchers have analyzed its negative effect on the convergence speed (Lian et al., 2015; Avron et al., 2015).

In this paper, we propose a novel method, called Delay Compensated ASGD (or DC-ASGD for short), to tackle the problem of delayed gradients. For this purpose, we study the Taylor expansion of the gradient function $g(\mathbf{w}_{t+\tau})$ at $\mathbf{w}_t$. We find that the delayed gradient $g(\mathbf{w}_t)$ is just the zero-order approximator of the correct gradient $g(\mathbf{w}_{t+\tau})$, and we can leverage more items in the Taylor expansion to achieve more accurate approximation of $g(\mathbf{w}_{t+\tau})$. However, this straightforward idea is practically non-trivial, be-

[1]University of Science and Technology of China [2]School of Mathematical Sciences, Peking University [3]Microsoft Research [4]Academy of Mathematics and Systems Science, Chinese Academy of Sciences. Correspondence to: Taifeng Wang, Wei Chen <taifengw, wche@microsoft.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

[1]Recently, people proposed to use additional backup workers (Chen et al., 2016) to tackle this problem. However, this solution requires redundant computation resources and relies on the assumption that the majority of workers train almost equally fast.

cause even including the first-order derivative of the gradient $g(\mathbf{w}_{t+\tau})$ will require the computation of the second-order derivative of the original loss function (i.e., the Hessian matrix), which will introduce high computation and space complexity. To overcome this challenge, we propose a cheap yet effective approximator of the Hessian matrix, which can achieve a good trade-off between bias and variance of approximation, only based on previously available gradients (without the necessity of directly computing the Hessian matrix).

DC-ASGD is similar to ASGD in the sense that no worker needs to wait for others. It differs from ASGD in that it does not directly add the local gradient to the global model, but compensates the delay in the local gradient by using the approximate Taylor expansion. By doing so, it maintains almost the same efficiency as ASGD and achieves much higher accuracy. Theoretically, we proved that DC-ASGD can converge at a rate of the same order with sequential SGD for non-convex neural networks, if the delay is upper bounded; and it is more tolerant on the delay than ASGD[2]. Empirically, we conducted experiments on both CIFAR-10 and ImageNet datasets. The results show that (1) as compared to SSGD and ASGD, DC-ASGD accelerated the convergence of the training process; (2) the accuracy of the model obtained by DC-ASGD within the same time period is very close to the accuracy obtained by sequential SGD.

2. Problem Setting

In this section, we introduce DNN and its parallel training through ASGD.

Given a multi-class classification problem, we denote $\mathcal{X} = \mathbb{R}^d$ as the input space, $\mathcal{Y} = \{1, ..., K\}$ as the output space, and $\mathbb{P}$ as the joint distribution over $\mathcal{X} \times \mathcal{Y}$. Here d denotes the dimension of the input space, and K denotes the number of categories in the output space.

We have a training set $\{(x_1, y_1), ..., (x_S, y_S)\}$, whose elements are i.i.d. sampled from $\mathcal{X} \times \mathcal{Y}$ according to distribution $\mathbb{P}$. Our goal is to learn a neural network model $O \in \mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ parameterized by $\mathbf{w} \in \mathbb{R}^n$ based on the training set. Specifically, the neural network models have hierarchical structures, in which each node conducts linear combination and non-linear activation over its connected nodes in the lower layer. The parameters are the weights on the edges between two layers. The neural network model produces an output vector, i.e., $(O(x, k; \mathbf{w}); k \in \mathcal{Y})$ for each input $x \in \mathcal{X}$, indicating its likelihoods of belonging to different categories. Because the underlying distribution $\mathbb{P}$ is unknown, a common way of learning the model is to minimize the empirical loss function. A widely-used loss function for deep neural networks is the cross-entropy loss, which is defined as follows,

$$f(x, y; \mathbf{w}) = -\sum_{k=1}^{K} (I_{[y=k]} \log \sigma_k(x; \mathbf{w})). \tag{1}$$

Here $\sigma_k(x; \mathbf{w}) = \frac{e^{O(x,k;\mathbf{w})}}{\sum_{k'=1}^{K} e^{O(x,k';\mathbf{w})}}$ is the *Softmax* operator. The objective is to optimize the empirical risk, defined as below,

$$F(\mathbf{w}) = \frac{1}{S} \sum_{s=1}^{S} f_s(\mathbf{w}) := \frac{1}{S} \sum_{s=1}^{S} f(x_s, y_s; \mathbf{w}). \tag{2}$$

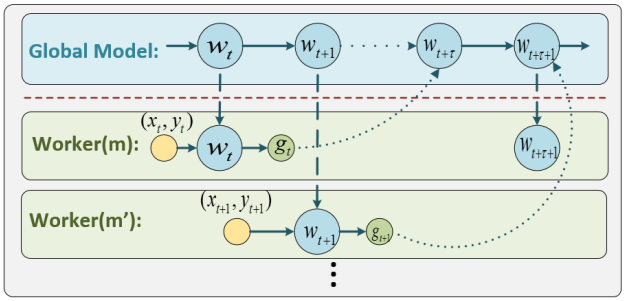

Figure 1. ASGD training process.

As mentioned in the introduction, ASGD is a widely-used approach to perform parallel training of neural networks. Although ASGD is highly efficient, it is well known to suffer from the problem of *delayed gradient*. To better illustrate this problem, let us have a close look at the training process of ASGD as shown in Figure 1. According to the figure, local worker m starts from $\mathbf{w}_t$, the snapshot of the global model at time t, calculates the local gradient $g(\mathbf{w}_t)$, and then add this gradient back to the global model[3]. However, before this happens, some other τ workers may have already added their local gradients to the global model, the global model has been updated τ times and becomes $\mathbf{w}_{t+\tau}$. The ASGD algorithm is blind to this situation, and simply adds the gradient $g(\mathbf{w}_t)$ to the global model $\mathbf{w}_{t+\tau}$, as follows.

$$\mathbf{w}_{t+\tau+1} = \mathbf{w}_{t+\tau} - \eta g(\mathbf{w}_t), \tag{3}$$

where η is the learning rate.

It is clear that the above update rule of ASGD is problematic (and inequivalent to that of sequential SGD): one actually adds a "delayed" gradient $g(\mathbf{w}_t)$ to the current global model $\mathbf{w}_{t+\tau}$. In contrast, the correct way is to update the global model $\mathbf{w}_{t+\tau}$ based on the gradient w.r.t. $\mathbf{w}_{t+\tau}$. This

[2]We also obtained similar results for the convex cases. Due to space restrictions, we put the corresponding theorems and proofs in the appendix.

[3]Actually, the local gradient is also related to the randomly sampled data (x_{i_t}, y_{i_t}). For simplicity, when there is no confusion, we will omit x_{i_t}, y_{i_t} in the notations.

problem of delayed gradient has been well known (Agarwal & Duchi, 2011; Recht et al., 2011; Lian et al., 2015; Avron et al., 2015), and many practical observations indicate that it usually costs ASGD more iterations to converge than sequential SGD, and sometimes, the converged model of ASGD cannot reach accuracy parity of sequential SGD, especially when the number of workers is large (Dean et al., 2012; Ho et al., 2013; Zhang et al., 2015). Researchers have tried to improve ASGD from different perspectives (Ho et al., 2013; McMahan & Streeter, 2014; Zhang et al., 2015; Sra et al., 2015; Mitliagkas et al., 2016), however, to the best of our knowledge, there is still no solution that can compensate the delayed gradient while keeping the high efficiency of ASGD. This is exactly the motivation of our paper.

3. Delay Compensation using Taylor Expansion and Hessian Approximation

As explained in the previous sections, ideally, the optimization algorithm should add gradient $g(\mathbf{w}_{t+\tau})$ to the global model $\mathbf{w}_{t+\tau}$, however, ASGD adds a delayed version $g(\mathbf{w}_t)$. In this section, we propose a novel method to bridge this gap by using Taylor expansion and Hessian approximation.

3.1. Gradient Decomposition using Taylor Expansion

The Taylor expansion of the gradient function $g(\mathbf{w}_{t+\tau})$ at $\mathbf{w}_t$ can be written as follows (Folland, 2005),

$$g(\mathbf{w}_{t+\tau}) = g(\mathbf{w}_t) + \nabla g(\mathbf{w}_t)(\mathbf{w}_{t+\tau} - \mathbf{w}_t) + \mathcal{O}((\mathbf{w}_{t+\tau} - \mathbf{w}_t)^2) I_n, \tag{4}$$

where ∇g denotes the matrix with the element $g_{ij} = \frac{\partial^2 f}{\partial w_i \partial w_j}$ for $i \in [n]$ and $j \in [n]$, $(\mathbf{w}_{t+\tau} - \mathbf{w}_t)^2 = (w_{t+\tau,1} - w_{t,1})^{\alpha_1} \cdots (w_{t+\tau,n} - w_{t,n})^{\alpha_n}$ with $\sum_{i=1}^n \alpha_i = 2$ and $\alpha_i \in \mathbb{N}$ and I_n is a n-dimension vector with all the elements equal to 1.

By comparing the above formula with Eqn. (3), we can immediately find that ASGD actually uses the zero-order item in Taylor expansion as its approximation to $g(\mathbf{w}_{t+\tau})$, and totally ignores all the higher-order terms $\nabla g(\mathbf{w}_t)(\mathbf{w}_{t+\tau} - \mathbf{w}_t) + \mathcal{O}((\mathbf{w}_{t+\tau} - \mathbf{w}_t)^2) I_n$. This is exactly the root cause of the problem of *delayed gradient*. With this insight, a straightforward and ideal method is to use the full Taylor expansion to compensate the delay. However, this is practically intractable, since it involves the sum of an infinite number of items. And even the simplest delay compensation, i.e., additionally keeping the first-order item in the Taylor expansion (which is shown below), is highly non-trivial,

$$g(\mathbf{w}_{t+\tau}) \approx g(\mathbf{w}_t) + \nabla g(\mathbf{w}_t)(\mathbf{w}_{t+\tau} - \mathbf{w}_t). \tag{5}$$

This is because the first-order derivative of the gradient function g corresponds to the Hessian matrix of the original loss function f (e.g., cross entropy for neural networks), which is defined as $\mathbf{H}f(\mathbf{w}) = [h_{ij}]_{i,j=1,\cdots,n}$ where $h_{ij} = \frac{\partial^2 f}{\partial w_i \partial w_j}(\mathbf{w})$.

For a neural network model with millions of parameters (which is very common and may only be regarded as a medium-size network today), the corresponding Hessian matrix will contain trillions of elements. It is clearly very computationally and spatially expensive to obtain such a large matrix[4]. Fortunately, as shown in the next subsection, we find an easy-to-compute/store approximator to the Hessian matrix, which makes our proposal of delay compensation technically feasible.

3.2. Approximation of Hessian Matrix

Computing the exact Hessian matrix is computationally and spatially expensive, especially for large models. Alternatively, we want to find some approximators that are theoretically close to the Hessian matrix, but can be easily stored and computed without introducing additional complexity (i.e., just using what we already have during the previous training process).

First, we show that the outer product of the gradients is an asymptotically unbiased estimation of the Hessian matrix. Let us use $G(\mathbf{w}_t)$ to denote the outer product matrix of the gradient at $\mathbf{w}_t$, i.e.,

$$G(\mathbf{w}_t) = \left(\frac{\partial}{\partial \mathbf{w}} f(x, y, \mathbf{w}_t)\right) \left(\frac{\partial}{\partial \mathbf{w}} f(x, y, \mathbf{w}_t)\right)^T. \tag{6}$$

Because the cross entropy loss is a negative log-likelihood with respect to the Softmax distribution of the model, i.e., $\mathbb{P}(Y = k|x, \mathbf{w}_t) \triangleq \sigma_k(x; \mathbf{w}_t)$, it is not difficult to obtain that the outer product of the gradient is an asymptotically unbiased estimation of Hessian, according to the two equivalent methods to calculate the fisher information matrix (Friedman et al., 2001)[5]:

$$\epsilon_t \triangleq \mathbb{E}_{(y|x,\mathbf{w}^*)} ||G(\mathbf{w}_t) - H(\mathbf{w}_t)|| \to 0, t \to \infty. \tag{7}$$

The assumption behind the above equivalence is that the underlying distribution equals the model distribution with parameter $\mathbf{w}^*$ (or there is no approximation error of the NN hypothesis space) and the training model $\mathbf{w}_t$ gradually converges to the optimal model $\mathbf{w}^*$ along with the training process. This assumption is reasonable considering the universal approximation property of DNN (Hornik, 1991) and the recent results on the optimality of the local optima of DNN (Choromanska et al., 2015; Kawaguchi, 2016).

Second, we show that by further introducing a well-designed weight to the outer product of the gradients, we

[4] Although Hessian-free methods were used in some previous works (Martens, 2010), they double the computation and communication for each local worker and are therefore not very feasible in practice.

[5] In this paper, the norm of the matrix is Frobenius norm.

can achieve a better trade-off between bias and variance for the approximation.

Although the outer product of the gradients can achieve unbiased estimation to the Hessian matrix, it may induce high approximation error due to potentially large variance. To further control the variance, we use mean square error (MSE) to measure the quality of an approximator, which is defined as follows,

$$mse^t(G) = \mathbb{E}_{(y|x,\mathbf{w}^*)}\|\big(G(\mathbf{w}_t) - H(\mathbf{w}_t)\big)\|^2. \tag{8}$$

We consider the following new approximator $\lambda G(\mathbf{w}_t) \triangleq \left[\lambda g_{ij}^t\right]$, and prove that with appropriately set λ, $\lambda G(\mathbf{w}_t)$ can lead to smaller MSE than $G(\mathbf{w}_t)$, for arbitrary model $\mathbf{w}_t$ during the training.

Theorem 3.1 *Assume that the loss function is L_1-Lipschitz, and for arbitrary $k \in [K]$, $\left|\frac{\partial \sigma_k}{\partial w_i}\right| \in [l_i, u_i]$, $|\frac{\sigma_k(x,\mathbf{w}^*)}{\sigma_k(x,\mathbf{w}_t)}| \in [\alpha, \beta]$. If $\lambda \in [0, 1]$ makes the following inequality holds,*

$$\sum_{k=1}^{K} \frac{1}{\sigma_k^3(x, \boldsymbol{w}_t)} \geq 2C \left[\left(\sum_{k=1}^{K} \frac{1}{\sigma_k(x, \boldsymbol{w}_t)} \right)^2 + 2L_1^2 \epsilon_t \right], \tag{9}$$

where $C = \max_{i,j} \frac{1}{1+\lambda}(\frac{u_i u_j \beta}{l_i l_j \alpha})^2$, and the model $\boldsymbol{w}_t$ converges to the optimal model $\boldsymbol{w}^*$, then $mse^t(\lambda G) \leq mse^t(G)$.

The following corollary gives simpler sufficient conditions for Theorem 3.1.

Corollary 3.2 *A sufficient condition for inequality (9) is $\exists k_0 \in [K]$ such that $\sigma_{k_0} \in \left[1 - \frac{K-1}{2C(K^2+L_1^2\epsilon_t)}, 1\right]$.*

According to Corollary 3.2, we have the following discussions. Please note that, if $\mathbf{w}_t$ converges to $\mathbf{w}^*$, ϵ_t is a decreasing term and approaches 0. Thus, ϵ_t can be upper bounded by a very small constant for large t. Therefore, the condition on $\sigma_k(x, \mathbf{w}_t)$ is more likely to be satisfied when $\sigma_k(x, \mathbf{w}_t)$ ($\exists k \in [K]$) is close to 1. Please note that this is not a strong condition, since if $\sigma_k(x, \mathbf{w}_t)$ ($\forall k \in [K]$) is very small, the classification power of the corresponding neural network model will be very weak and not useful in practice.

Third, to reduce the storage of the approximator $\lambda G(\mathbf{w})$, we adopt a widely-used diagonalization trick (Becker et al., 1988), which has shown promising empirical results. To be specific, we only store the diagonal elements of the approximator $\lambda G(\mathbf{w})$ and make all the other elements to be zero. We denote the refined approximator as $Diag(\lambda G(\mathbf{w}))$ and assume that the diagonalization error is upper bounded by ϵ_D, i.e., $||Diag(H(\mathbf{w}_t)) - H(\mathbf{w}_t)|| \leq \epsilon_D$. We give a uniform upper bound of its MSE in the supplementary materials, from which we can see that λ plays a role of trading off variance and Lipschitz[6].

4. *Delay Compensated ASGD*: Algorithm Description

In Section 3, we have shown that $Diag(\lambda G(\mathbf{w}))$ is a cheap approximator of the Hessian matrix, with guaranteed approximation accuracy. In this section, we will use this approximator to compensate the gradient delay, and call the corresponding algorithm Delay-Compensated ASGD (DC-ASGD). Since $Diag(\lambda G(\mathbf{w})) = \lambda g(\mathbf{w}_t) \odot g(\mathbf{w}_t)$, where $\odot$ indicates the element-wise product, the update rule for DC-ASGD can be written as follows:

$$\mathbf{w}_{t+\tau+1} = \mathbf{w}_{t+\tau} - \eta \left(g(\mathbf{w}_t) + \lambda g(\mathbf{w}_t) \odot g(\mathbf{w}_t) \odot (\mathbf{w}_{t+\tau} - \mathbf{w}_t) \right), \tag{10}$$

We call $g(\mathbf{w}_t) + \lambda g(\mathbf{w}_t) \odot g(\mathbf{w}_t) \odot (\mathbf{w}_{t+\tau} - \mathbf{w}_t)$ the *delay-compensated gradient* for ease of reference.

The flow of DC-ASGD is shown in Algorithms 1 and 2. Here we assume that DC-ASGD is implemented by using the parameter server framework (although it can also be implemented in other frameworks). According to Algorithm 1, local worker m pulls the latest global model $\mathbf{w}_t$ from the parameter server, computes its gradient g_m and sends it back to the server. According to Algorithm 2, the parameter server will store a backup model $\mathbf{w}_{bak}(m)$ when worker m pulls $\mathbf{w}_t$. When the delayed gradient g_m calculated by worker m is received at time t, the parameter server updates the global model according to Eqn (10).

Please note that as compared to ASGD, DC-ASGD has no extra communication cost and no extra computational requirement on the local workers. And the additional computations regarding Eqn(10) only introduce a lightweight overhead to the parameter server. As for the space requirement, for each worker $m \in \{1, 2, \cdots, M\}$, the parameter server needs to additionally store a backup model $\mathbf{w}_{bak}(m)$. This is not a critical issue since the parameter server is usually implemented in a distributed manner, and the parameters and its backup version are stored in CPU-side memory which is usually far beyond the total parameter size. In this case, the cost of DC-ASGD is quite similar to ASGD, which is also reflected by our experiments.

The Delay Compensation is not only applicable to ASGD but SSGD. Recently a study on SSGD(Goyal et al., 2017) assumes

Algorithm 1 DC-ASGD: worker m

repeat
  Pull $\mathbf{w}_t$ from the parameter server.
  Compute gradient $g_m = \nabla f_m(\mathbf{w}_t)$.
  Push g_m to the parameter server.
until $forever$

Algorithm 2 DC-ASGD: parameter server

Input: learning rate η, variance control parameter λ_t.
Initialize: $t = 0$, $\mathbf{w}_0$ is initialized randomly, $\mathbf{w}_{bak}(m) = \mathbf{w}_0$, $m \in \{1, 2, \cdots, M\}$
repeat
  if receive "g_m" **then**
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \cdot \big(g_m + \lambda_t g_m \odot g_m \odot (\mathbf{w}_t - \mathbf{w}_{bak}(m))\big)$
    $t \leftarrow t + 1$
  else if receive "pull request" **then**
    $\mathbf{w}_{bak}(m) \leftarrow \mathbf{w}_t$
    Send $\mathbf{w}_t$ back to worker m.
  end if
until $forever$

[6] See **Lemma 3.1** in Supplementary.

$g(\mathbf{w}_{t+j}) \approx g(\mathbf{w}_t)$ for $j < M$ to make the updates from small and large mini-batch SGD similar, which can be immediately improved by applying delay-compensated gradient. Please check the detailed discussion in Supplementary.

5. Convergence Analysis

In this section, we prove the convergence rate of DC-ASGD. Due to space restrictions, we only give the results for the non-convex case, and leave the results for the convex case (which is much easier) to the supplementary.

In order to present our main theorem, we need to introduce the following mild assumptions.

Assumption 1 (Smoothness): (Lian et al., 2015)(Recht et al., 2011) The loss function is smooth w.r.t. the model parameter, and we use L_1, L_2, L_3 to denote the upper bounds of the first, second, and third-order derivatives of the loss function. The activation function $\sigma_k(\mathbf{w})$ is L-Lipschitz continuous.

Assumption 2 (Non-convexity): (Lee et al., 2016) The loss function is μ-strongly convex in a ball centered at each local optimum which is denoted as $d(\mathbf{w}_{loc}, r)$ with radius r, and twice differential about $\mathbf{w}$.

We also introduce some notations to simplify the presentation of our results, i.e.,

$$M = \max_{k,\mathbf{w}_{loc}} \left|\mathbb{P}(Y=k|x,\mathbf{w}_{loc}) - \mathbb{P}(Y=k|x,\mathbf{w}^*)\right|,$$

$$H = \max_{k,x,\mathbf{w}} \left| \frac{\partial^2 \mathbb{P}(Y=k|x,\mathbf{w})}{\partial^2 \mathbf{w}} \times \frac{1}{\mathbb{P}(Y=k|x,\mathbf{w})} \right|, \quad \forall k \in [K], x, w.$$

Actually, the non-convexity error $\epsilon_{nc} = HKM$, which is defined as the upper bound of the difference between the prediction outputs of the local optima and the global optimum (Please see Lemma 5.1 in the supplementary materials). We assume that the DC-ASGD search in the set $\|\mathbf{w}-\mathbf{w}'\|_2^2 \le \pi^2, \forall \mathbf{w}, \mathbf{w}'$ and denote $D_0 = F(\mathbf{w}_1) - F(\mathbf{w}^*)$, $C_\lambda^2 = (L_3^2\pi^2/2 + 2((1-\lambda)L_1^2 + \epsilon_D)^2 + 2\epsilon_{nc}^2)$, $\tilde{C}_\lambda^2 = 4T_0 \max_{s=1,\cdots,T_0} {\epsilon_s}^2 + 4\theta^2 \log(T-T_0)$ where $T_0 \ge \mathcal{O}(1/r^4)$, $\theta = \frac{2HKLV L_2}{\mu^2}\sqrt{\frac{1}{\mu}\left(1 + \frac{L_2+\lambda L_1^2}{L_2}\tau\right)}$.

With all the above, we have the following theorem.

Theorem 5.1 *Assume that Assumptions 1-2 hold. Set the learning rate* $\eta = \sqrt{\frac{2D_0}{bTL_2V^2}}$*,where b is the mini-batch size, and V is the upper bound of the variance of the delay-compensated gradient. If* $T \ge \max\{\mathcal{O}(1/r^4), 2D_0 b L_2/V^2\}$ *and delay* τ *is upper-bounded as below,*

$$\tau \le \min\left\{\frac{L_2\gamma}{C_\lambda}, \frac{\gamma}{C_\lambda}, \frac{\sqrt{T}\gamma}{\tilde{C}}, \frac{L_2 T\gamma}{4\tilde{C}}\right\}, \tag{11}$$

where $\gamma = \sqrt{\frac{L_2TV^2}{2D_0 b}}$*, then DC-ASGD has the following ergodic convergence rate,*

$$\min_{t=\{1,\cdots,T\}} \mathbb{E}(\|\nabla F(\boldsymbol{w}_t)\|^2) \le V\sqrt{\frac{2D_0L_2}{bT}}, \tag{12}$$

where T is the number of iteration, the expectation is taken with respect to the random sampling in SGD and the data distribution $P(Y|x,\boldsymbol{w}^*)$.

Proof Sketch[7]*:*

Step 1: We denote the delay-compensated gradient as $g_m^{dc}(w_t)$ where $m \in \{1,\cdots,b\}$ is the index of instances in the mini-batch and $\nabla F^h(w_t) = \nabla F(w_t) + \mathbb{E}H(w_t)(w_{t+\tau} - w_t)$. According to Assumption 1, we have

$$\begin{aligned}
&\mathbb{E}F(w_{t+\tau+1}) - F(w_{t+\tau}) \\
\le\ & -\frac{b\eta_{t+\tau}}{2}\left(\|\nabla F(w_{t+\tau})\|^2 + \left\|\sum_{m=1}^b \mathbb{E}g_m^{dc}(w_t)\right\|^2\right) \\
&+b\eta_{t+\tau}\left\|\nabla F(w_{t+\tau}) - \sum_{m=1}^b \nabla F^h(w_t)\right\|^2 \\
&+b\eta_{t+\tau}\left\|\sum_{m=1}^b \mathbb{E}g_m^{dc}(w_t) - \sum_{m=1}^b F^h(w_t)\right\|^2 \\
&+\frac{\eta_{t+\tau}^2 L_2}{2}\mathbb{E}\left(\left\|\sum_{m=1}^b g_m^{dc}(w_t)\right\|^2\right).
\end{aligned} \tag{13}$$

The term $\left\|\sum_{m=1}^b \mathbb{E}g_m^{dc}(w_t) - \sum_{m=1}^b F^h(w_t)\right\|^2$, measured by the expectation with respect to $\mathbb{P}(Y|x,w^*)$, is bounded by $C_\lambda^2 \cdot \|w_{t+\tau} - w_t\|^2$. The term $\left\|\nabla F(w_{t+\tau}) - \sum_{m=1}^b \nabla F^h(w_t)\right\|^2$ can be bounded by $\frac{L_3^2}{4}\|w_{t+\tau} - w_t\|^4$, which will be smaller than $\|w_{t+\tau} - w_t\|^2$ when $\|w_{t+\tau} - w_t\|$ is small. Other terms which are related to the gradients can be further upper bounded by the smoothness property of the loss function.

Step 2: We proved that, under the non-convexity assumption, if $\|\lambda g(w_t) \odot g(w_t)\| \le \lambda L_1^2$, then when $t > \mathcal{O}(1/r^4)$, $\epsilon_t \le \theta\sqrt{\frac{1}{t-T_0}} + \epsilon_{nc}$, where $T_0 = \mathcal{O}(1/r^4)$. That is, we can find a weaker condition for the decreasing of ϵ_t than that for $\mathbf{w}_t \to \mathbf{w}^*$.

Step 3: By plugging in the decreasing rate of ϵ_t in *Step 1* and following a similar proof of the convergence rate of ASGD (Lian et al., 2015), we can get the result in the theorem.

Discussions:

(1) The above theorem shows that the convergence rate of DC-ASGD is in the order of $O(\frac{V}{\sqrt{T}})$. Recall that the convergence rate of ASGD is $O(\frac{V_1}{\sqrt{T}})$, where V_1 is the variance for the delayed gradient $g(w_t)$. By simple calculation, V can be upper bounded by $V_1 + \lambda V_2$, where V_2 is the extra moments of the noise introduced by the delay compensation term. Thus if we set $\lambda \in [0, V_1/V_2]$, DC-ASGD and ASGD will converge at the same rate. As the training process goes on, $g(w)$ will become smaller. Compared with V_1, V_2 (composed by variance of $g \odot g$) will not be the dominant order and can be gradually neglected. As a result, the feasible range for λ is actually very large.

(2) Although DC-ASGD converges at the same rate with ASGD, its tolerance on the delay is much better if $T \ge \max\{\tilde{C}^2, 4\tilde{C}/L_2\}$ and $C_\lambda < \min\{L_2, 1\}$. The intuition for the condition on T is that larger T induces smaller step size η. A small step size means that w_t and $w_{t+\tau}$ are close to each other. According to the upper bound of Taylor expansion series (Folland, 2005), we can see that *delay compensated gradient* will be more

[7] Please check the complete proof in the supplementary material.

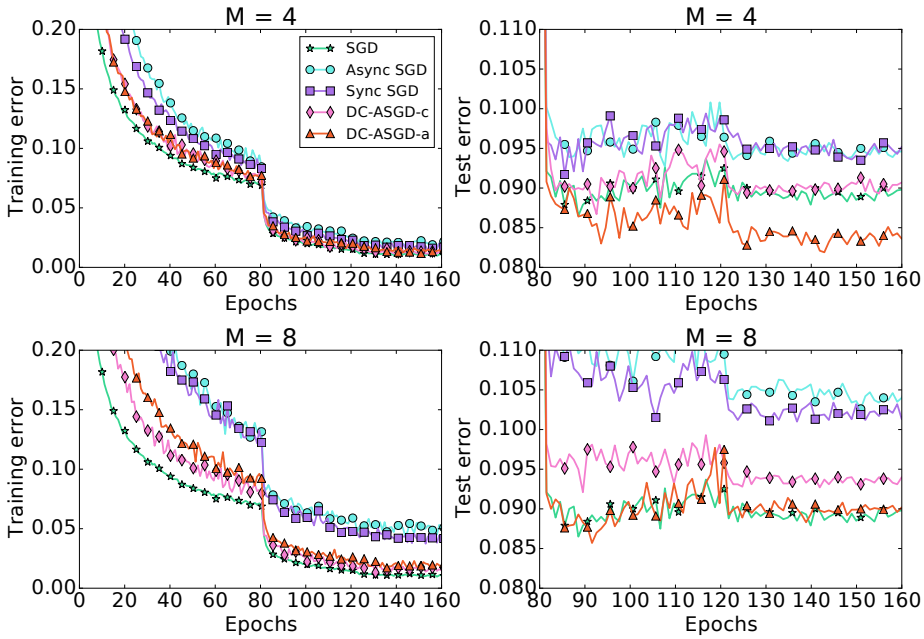

Figure 2. Error rates of the global model w.r.t. number of effective passes of data on CIFAR-10

accurate than the delayed gradient used in ASGD. Since C_λ is related to the diagonalization error ϵ_D and the non-convexity error ϵ_{nc}, smaller ϵ_D and ϵ_{nc} will lead to looser conditions for the convergence. If these two error are sufficiently small (which is usually the case according to (Choromanska et al., 2015; Kawaguchi, 2016; LeCun, 1987)), the condition $L_2 > C_\lambda$ can be simplified as $L_2 > (1-\lambda)L_1^2 + L_3\pi$, which is easy to be satisfied with a small $1-\lambda$. Assume that $L_2 - L_3\pi > 0$, which is easily to be satisfied if the gradient is small (e.g. at the later stage of the training progress). Accordingly, we can obtain the feasible range for λ as $\lambda \in [1-(L_2-L_3\pi)/2L_1^2, 1]$. λ can be regarded as a trade-off between the extra variance introduced by the delay-compensate term $\lambda g(w_t) \odot g(w_t)$ and the bias in Hessian approximation.

(3) Actually ASGD is an extreme case for DC-ASGD, with $\lambda = 0$. Another extreme case is with $\lambda = 1$. DC-ASGD prefers larger T and smaller π, which can lead to a faster speed-up and larger tolerant for delay.

Based on the above discussions, we have the following corollary, which indicates that DC-ASGD is superior to ASGD in most cases.

Corollary 5.2 *Let $C_0 = \max\{\tilde{C}^2, 4\tilde{C}/L_2\}$, which is a constant. If we choose $\lambda \in \left[1 - \frac{L_2 - L_3\pi}{L_1^2}, 1\right] \cap [0, V_1/V_2] \cap [0,1]$ and the number of total iterations $T \geq C_0$, DC-ASGD will outperform ASGD by a factor of T/C_0.*

6. Experiments

In this section, we evaluate our proposed DC-ASGD algorithm. We used two datasets: CIFAR-10 (Hinton, 2007) and ImageNet ILSVRC 2013 (Russakovsky et al., 2015). The experiments were conducted on a GPU cluster interconnected with InfiniBand. Each node has four K40 Tesla GPU processors. We treat each GPU as a separate local worker. For the DNN algorithm running on each worker, we chose ResNet (He et al., 2016) since it produces the state-of-the-art accuracy in many image related tasks and its implementation is available through open-source projects[8]. For the parallelization of ResNet across machines, we leveraged an open-source parameter server[9].

We implemented DC-ASGD on this experimental platform. We have two versions of implementations, one sets $\lambda_t = \lambda_0$ as a constant, and the other adaptively tunes λ_t using a moving average method proposed by (Tieleman & Hinton, 2012). Specifically, we first define a quantity called MeanSquare as follows,

$$MeanSquare(t) = m \cdot MeanSquare(t-1) + (1-m) \cdot g(\mathbf{w}_t)^2, \tag{14}$$

where m is a constant taking value from $[0, 1)$. And then we divide the initial λ_0 by $\sqrt{MeanSquare(t) + \epsilon}$, where $\epsilon = 10^{-7}$ for all our experiments. This adaptive method is adopted to reduce the variance among coordinates with historical gradient values. For ease of reference, we denote the first implementation as DC-ASGD-c (constant) and the second as DC-ASGD-a (adaptive).

In addition to DC-ASGD, we also implemented ASGD and SSGD, which have been used in many previous works as baselines (Dean et al., 2012; Chen et al., 2016; Das et al., 2016). Furthermore, for the experiments on CIFAR-10, we used the sequential SGD algorithm as a reference model to examine the accuracy of parallel algorithms. However, for the experiments on ImageNet, we were not able to show this reference because it simply took too long time for a single machine to finish the training[10]. For sake of fairness, all experiments started from the same randomly initial-

[8] https://github.com/KaimingHe/deep-residual-networks

[9] http://www.dmtk.io/

[10] We also implemented the momentum variants of these algorithms. The corresponding comparisons are very similar to those without momentum.

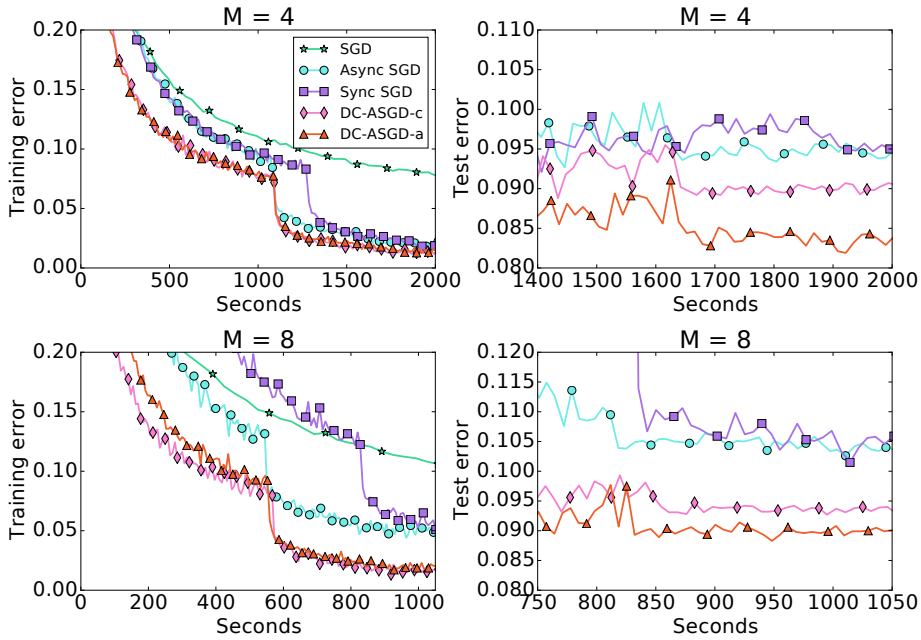

Figure 3. Error rates of the global model w.r.t. wallclock time on CIFAR-10

ized model, and used the same strategy for learning rate scheduling. The data were repartitioned randomly onto the local workers every epoch.

6.1. Experimental Results on CIFAR-10

The CIFAR-10 dataset consists of a training set of 50k images and a test set of 10k images in 10 classes. We trained a 20-layer ResNet model on this dataset (without data augmentation). For all the algorithms under investigation, we performed training for 160 epochs, with a mini-batch size of 128, and an initial learning rate which was reduced by ten times after 80 and 120 epochs following the practice in (He et al., 2016). We performed grid search for the hyper-parameter and the best test performances are obtained by choosing the initial learning rate $\eta = 0.5$, $\lambda_0 = 0.04$ for DC-ASGD-c, and $\lambda_0 = 2$, $m = 0.95$ for DC-ASGD-a. We tried different numbers of local workers in our experiments: $M = \{1, 4, 8\}$.

Table 1. Classification error on CIFAR-10 test set. The number of † is 8.75 reported in (He et al., 2016). Fig. 2 and 3 show the training procedures.

# workers	algorithm	error(%)
1	SGD	8.65†
4	ASGD	9.27
	SSGD	9.17
	DC-ASGD-c	8.67
	DC-ASGD-a	**8.19**
8	ASGD	10.26
	SSGD	10.10
	DC-ASGD-c	9.27
	DC-ASGD-a	**8.57**

First, we investigate the learning curves with fixed number of effective passes as shown in Figure 2. From the figure, we have the following observations: (1) Sequential SGD achieves the best accuracy, and its final test error is 8.65%. (2) The test errors of ASGD and SSGD increase with respect to the number of local workers. In particular, when $M = 4$, ASGD and SSGD achieve test errors of 9.27% and 9.17% respectively; and when $M = 8$, their test errors become 10.26% and 10.10% respectively. These results are reasonable: ASGD suffers from delayed gradients which becomes more serious for a larger number of workers; SSGD increases the effective mini-batch size by M times, and enlarged mini-batch size usually affects the training performances of DNN. (3) For DC-ASGD, no matter which λ_t is used, its performance is significantly better than ASGD and SSGD, and catches up with sequential SGD. For example, when $M = 4$, the test error of DC-ASGD-c is 8.67%, which is indistinguishable from sequential SGD, and the test error for DC-ASGD-a is 8.19%, which is even better than that achieved by sequential SGD. It is not by design that DC-ASGD can beat sequential SGD. The test performance lift might be attributed to the regularization effect brought by the variance introduced by parallel training. When $M = 8$, DC-ASGD-c can reduce the test error to 9.27%, which is nearly 1% better than ASGD and SSGD, meanwhile the test error is 8.57% for DC-ASGD-a, which again slightly better than sequential SGD.

We further compared the convergence speeds of different algorithms as shown in Figure 3. From this figure, we have the following observations: (1) Although the convergent point is not very good, ASGD runs indeed very fast, and achieves almost linear speed-up as compared to sequential SGD in terms of throughput. (2) SSGD also runs faster than sequential SGD. However, due to the synchronization barrier, it is significantly slower than ASGD. (3) DC-ASGD achieves very good balance between accuracy and speed. On one hand, its converge speed is very similar to that of ASGD (although it involves a little more computational cost and

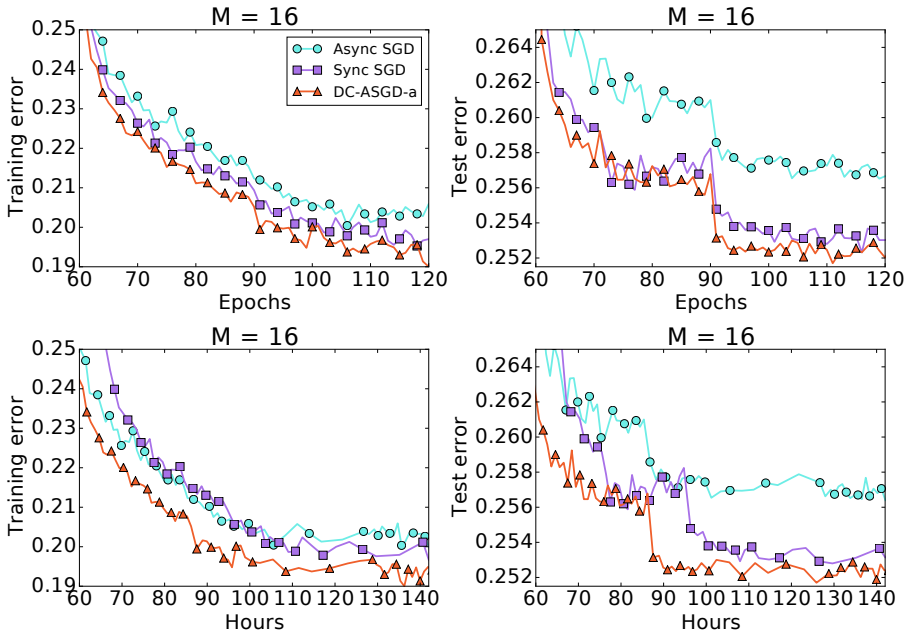

Figure 4. Error rates of the global model w.r.t. both number of effective passes and wallclock time on ImageNet

some memory cost when compensating the delay). On the other hand, its convergent point is as good as, or even better than that of sequential SGD. The experiments results clearly demonstrate the effectiveness of our proposed delay compensation technologies[11].

6.2. Experimental Results on ImageNet

In order to further verify our method on the large-scale setting, we conducted the experiment on the ImageNet dataset, which contains 1.28 million training images and 50k validation images in 1000 categories. We trained a 50-layer ResNet model (He et al., 2016) on this dataset.

According to the previous subsection, DC-ASGD-a seems to be better, therefore in this large-scale experiment, we only implemented DC-ASGD-a. For all algorithms in this experiment, we performed training for 120 epochs , with a mini-batch size of 32, and an initial learning rate reduced by ten times after every 30 epochs following the practice in (He et al., 2016). We did grid search for hyperparameter tuning and set the initial learning rate $\eta = 0.1$, $\lambda_0 = 2$, $m = 0$. Since the training on the ImageNet dataset is very time consuming, we employed $M = 16$ GPU nodes in our experiments. The top-1 accuracies based on **1-crop** testing of different algorithms are given in Figure 4.

Table 2. Top-1 error on **1-crop** ImageNet validation. Fig. 4 shows the training procedures.

# workers	algorithm	error(%)
16	ASGD	25.64
	SSGD	25.30
	DC-ASGD-a	**25.18**

[11]Please refer to the supplementary materials for the experiments on tuning the parameter λ.

According to the figure, we have the following observations: (1) After processing the same amount of training data, DC-ASGD always outperforms SSGD and ASGD. In particular, while the eventual test error achieved by ASGD and SSGD were 25.64% and 25.30% respectively, DC-ASGD achieved a lower error rate of 25.18%. Please note this time the accuracy of SSGD is quite good (which is consistent with a separate observation in (Chen et al., 2016)). An explanation is that the training on ImageNet is less sensitive to the mini-batch size than that on CIFAR-10. (2) If we look at the learning curve with respect to wallclock time, SSGD is slowed down due to the synchronization barrier; ASGD and DC-ASGD have similar efficiency, once again indicating that the extra overhead for delay compensation introduced by DC-ASGD can almost be neglected in practice. Based on all our experiments, we can clearly see that DC-ASGD has outstanding performance in terms of both classification accuracy and convergence speed, which in return verifies the soundness of our proposed delay compensation technologies.

7. Conclusion

In this paper, we have given a theoretical analysis on the problem of delayed gradients in the asynchronous parallelization of stochastic gradient descent (SGD) algorithms, and proposed a novel algorithm called Delay Compensated Asynchronous SGD (DC-ASGD) to tackle the problem. We have evaluated DC-ASGD on CIFAR-10 and ImageNet datasets, and the results demonstrate that it can achieve better accuracy than both synchronous SGD and asynchronous SGD, and nearly approaches the performance of sequential SGD. As for the future work, we plan to test DC-ASGD on larger computer clusters, where with the increasing number of local workers, the delay will become more serious. Furthermore, we will investigate the economical approximation of higher-order items in the Taylor expansion to achieve more effective delay compensation.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (Grant No. 61371192).

References

Agarwal, Alekh and Duchi, John C. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pp. 873–881, 2011.

Avron, Haim, Druinsky, Alex, and Gupta, Anshul. Revisiting asynchronous linear solvers: Provable convergence rate through randomization. *Journal of the ACM (JACM)*, 62(6): 51, 2015.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Becker, Sue, Le Cun, Yann, et al. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pp. 29–37. San Matteo, CA: Morgan Kaufmann, 1988.

Bottou, Léon. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

Chen, Jianmin, Monga, Rajat, Bengio, Samy, and Jozefowicz, Rafal. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

Chen, Kai and Huo, Qiang. 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.

Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, Gérard Ben, and LeCun, Yann. The loss surfaces of multilayer networks. In *AISTATS*, 2015.

Das, Dipankar, Avancha, Sasikanth, Mudigere, Dheevatsa, Vaidynathan, Karthikeyan, Sridharan, Srinivas, Kalamkar, Dhiraj, Kaul, Bharat, and Dubey, Pradeep. Distributed deep learning using synchronous stochastic gradient descent. *arXiv preprint arXiv:1602.06709*, 2016.

Dean, Jeffrey, Corrado, Greg, Monga, Rajat, Chen, Kai, Devin, Matthieu, Mao, Mark, Senior, Andrew, Tucker, Paul, Yang, Ke, Le, Quoc V, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012.

Folland, GB. Higher-order derivatives and taylors formula in several variables, 2005.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Gehring, Jonas, Auli, Michael, Grangier, David, Yarats, Denis, and Dauphin, Yann N. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

Goyal, Priya, Dollar, Piotr, Girshick, Ross, Noordhuis, Pieter, Wesolowski, Lukasz, Kyrola, Aapo, Tulloch, Andrew, Jia, Yangqing, and He, Kaiming. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hinton, Geoffrey E. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.

Ho, Qirong, Cipar, James, Cui, Henggang, Lee, Seunghak, Kim, Jin Kyu, Gibbons, Phillip B, Gibson, Garth A, Ganger, Greg, and Xing, Eric P. More effective distributed ml via a stale synchronous parallel parameter server. In *Advances in neural information processing systems*, pp. 1223–1231, 2013.

Hornik, Kurt. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Kawaguchi, Kenji. Deep learning without poor local minima. *arXiv preprint arXiv:1605.07110*, 2016.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Yann. *Modèles connexionnistes de lapprentissage*. PhD thesis, These de Doctorat, Universite Paris 6, 1987.

Lee, Jason D, Simchowitz, Max, Jordan, Michael I, and Recht, Benjamin. Gradient descent converges to minimizers. *University of California, Berkeley*, 1050:16, 2016.

Lian, Xiangru, Huang, Yijun, Li, Yuncheng, and Liu, Ji. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pp. 2737–2745, 2015.

Martens, James. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 735–742, 2010.

McMahan, Brendan and Streeter, Matthew. Delay-tolerant algorithms for asynchronous distributed online learning. In *Advances in Neural Information Processing Systems*, pp. 2915–2923, 2014.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Mitliagkas, Ioannis, Zhang, Ce, Hadjis, Stefan, and Ré, Christopher. Asynchrony begets momentum, with an application to deep learning. *arXiv preprint arXiv:1605.09774*, 2016.

Recht, Benjamin, Re, Christopher, Wright, Stephen, and Niu, Feng. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 693–701, 2011.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Sak, Haşim, Senior, Andrew, and Beaufays, Françoise. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Sercu, Tom, Puhrsch, Christian, Kingsbury, Brian, and LeCun, Yann. Very deep multilingual convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 4955–4959. IEEE, 2016.

Sra, Suvrit, Yu, Adams Wei, Li, Mu, and Smola, Alexander J. Adadelay: Delay adaptive distributed stochastic convex optimization. *arXiv preprint arXiv:1508.05003*, 2015.

Szegedy, Christian, Ioffe, Sergey, Vanhoucke, Vincent, and Alemi, Alex. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2), 2012.

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