Step-Ahead Error Feedback for Distributed Training with Compressed Gradient

An Xu,[1] **Zhouyuan Huo,**[2] **Heng Huang**[1,3]
[1] Electrical and Computer Engineering Department, University of Pittsburgh, PA, USA
[2] Google, Mountain View, CA, USA
[3] JD Finance America Corporation, Mountain View, CA, USA
{an.xu, heng.huang}@pitt.edu, zhouyuan.huo@gmail.com

Abstract

Although the distributed machine learning methods can speed up the training of large deep neural networks, the communication cost has become the non-negligible bottleneck to constrain the performance. To address this challenge, the gradient compression based communication-efficient distributed learning methods were designed to reduce the communication cost, and more recently the local error feedback was incorporated to compensate for the corresponding performance loss. However, in this paper, we will show that a new "gradient mismatch" problem is raised by the local error feedback in centralized distributed training and can lead to degraded performance compared with full-precision training. To solve this critical problem, we propose two novel techniques, 1) step ahead and 2) error averaging, with rigorous theoretical analysis. Both our theoretical and empirical results show that our new methods can handle the "gradient mismatch" problem. The experimental results show that we can even **train faster with common gradient compression** schemes than both the full-precision training and local error feedback **regarding the training epochs and without performance loss**.

Introduction

Distributed training is a common practice in training large models with big datasets. The master-slave architecture is the most common paradigm in centralized learning, where the worker nodes compute gradients based on the local dataset and communicate with the server node. While in decentralized learning (Lian et al. 2018, 2017; Tang et al. 2018a,b), no server node is needed and each worker node only communicates with its neighbors to avoid the heavy traffic of the server node as in centralized training. In recent research, the gradient compression techniques have been widely used to reduce the communication cost in both centralized and decentralized training.

Mild gradient compression techniques (Alistarh et al. 2017; Xu, Huo, and Huang 2020b) offer a mild compression ratio at the cost of negligible performance loss. However, it is more attractive to use an aggressive compression technique such as SignSGD (Bernstein et al. 2018) which is even more favorable for scaling up the number of worker nodes. More recently, the local error feedback method (Karimireddy et al. 2019) was introduced to fix the corresponding non-negligible performance loss resulting from SignSGD via adding the compression error at the current iteration to the next iteration. Note that for SignSGD, we need to scale it by a factor before applying the local error feedback method because it does not satisfy the Assumption 4 described at later section, which is crucial in the theoretical analysis of local error feedback.

Methods other than gradient compression to accelerate distributed training include asynchronous methods (Lian et al. 2015; Ho et al. 2013; Huang et al. 2019; Xu, Huo, and Huang 2020a), local SGD (Stich 2019) which is also a natural fit for solving federated learning (Konečný et al. 2016) problem, and communication scheduling such as the lazy aggregation of gradients (Sun et al. 2019; Hashemi, Jyothi, and Campbell 2018; Chen et al. 2018). Specifically, asynchronous distributed training avoids the synchronization barrier and the worker node does not wait for each other. The performance loss is related to the inconsistency between the worker and server (staleness) allowed during training. In local SGD, each worker node stores a copy of the model and does several iterations of updating before communicating with all other nodes to average the updated model. The more the number of local updating iterations is, the more the model in different worker nodes will diverge, leading to a larger performance loss.

We focus on the line of works with gradient compression. Plain gradient compression has been well studied both in centralized (Alistarh et al. 2017; Wen et al. 2017; Stich, Cordonnier, and Jaggi 2018) and decentralized training (Tang et al. 2018a; Koloskova, Stich, and Jaggi 2019). Later works incorporating local error feedback theoretically and empirically achieve superior performance in centralized (Basu et al. 2019; Wu et al. 2018; Zheng, Huang, and Kwok 2019) and decentralized training (Tang et al. 2019) than plain gradient compression. In this paper, we improve the local error feedback with theoretical analysis and empirical validation in centralized distributed training. When we studied the coarse idea of adding the current compression error to the next iteration as local error feedback does, we found that this strategy could lead to a one-iteration outdated gradient. This staleness may seem trivial at first glance, but theory and practice show that it can be the reason why local error feedback is not always lossless. We summarize the main contributions of our paper as follows:

- We introduce and discuss the new "gradient mismatch" problem caused by the local error feedback with the potential to lead to stale gradients. We show that the local error feedback may not be able to achieve lossless performance all the time in experiments. To the best of our knowledge, this is the first paper to systematically investigate this problem.
- We propose two novel techniques, 1) step ahead and 2) error averaging, to correct the "gradient mismatch" issue. Error averaging can be conducted in a much more infrequent way than the communication of the compressed gradient.
- Theoretical analysis shows a better error bound of our proposed method than local error feedback. Experimental results verify that our method converges even *faster with common gradient compression regarding training epochs without performance loss* compared with both the full-precision training and local error feedback.

Local Error Feedback

We consider the following learning problem:

$$\min_{\mathbf{x}} F(\mathbf{x}) := \mathbb{E}_{\xi\sim\mathcal{D}} f(\mathbf{x};\xi)\,, \tag{1}$$

where $\mathbf{x}$ is the parameters, $F(\cdot)$ is the full loss function, $\mathcal{D}$ is the data distribution, ξ is the random variable associated with stochastic sampling and $f(\cdot)$ is the loss function associated with certain data sample. Stochastic optimization methods compute the stochastic gradient $\nabla f(\mathbf{x};\xi)$ to update $\mathbf{x}$. We assume the data distributions across different workers are identical.

In local error feedback, the compression error is added into the next iteration of training. We illustrate it in Algorithm 1 (line 14 $\sim$ 16). In the first work (Karimireddy et al. 2019) using local error feedback to fix the performance loss resulting from scaled SignSGD (Bernstein et al. 2018) compression, only SGD rather than momentum SGD is considered (momentum constant $\mu = 0$). (Zheng, Huang, and Kwok 2019) proposed to use local error feedback to fix momentum SGD with block-wise scaled SignSGD compression. In (Zheng, Huang, and Kwok 2019), the feedbacked error $\mathbf{e}_t^{(k)}$ is scaled according to learning rate as $\frac{\eta_{t-1}}{\eta_t}\mathbf{e}_t^{(k)}$. Typically, scaled SignSGD compresses a vector $\mathbf{v}\in\mathbb{R}^d$ to

$$\mathcal{C}(\mathbf{v}) = \frac{\|\mathbf{v}\|_1}{d}\,\mathrm{sign}(\mathbf{v})\,. \tag{2}$$

For simplicity, we refer to scaled SignSGD as SignSGD from now on. To put it in a more general and clearer way as in Algorithm 1, we compress the local model difference $\Delta_{t+1}^{(k)}$ after updating and re-update the local model with the information $\mathcal{C}(\Delta_{t+1})$ that the server has gathered from all the workers and compressed. There are two advantages: 1) it can be easily extended to local SGD, where the local model difference will be communicated every several (> 1) iterations; 2) the local error needn't be scaled when using a decaying learning rate. Note that in Algorithm 1 where the local model difference is communicated every iteration, **the local model $\mathbf{x}_t^{(k)} = \mathbf{x}_t$ is identical across all workers**. We do not need to synchronize the local model at every iteration.

Gradient Mismatch. The effectiveness of local error feedback comes from an auxiliary variable $\tilde{\mathbf{x}}_t := \mathbf{x}_t - (\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)})$ in its theoretical analysis. Although a very aggressive gradient compression scheme may be applied, the update of the auxiliary variable in local error feedback still satisfies:

$$\tilde{\mathbf{x}}_{t+1} = \tilde{\mathbf{x}}_t - \frac{\eta_t}{K}\sum_{k=1}^K \mathbf{m}_{t+1}^{(k)}\,. \tag{3}$$

For vanilla momentum SGD, we update the parameters $\mathbf{x} \leftarrow \mathbf{x} - \frac{\eta_t}{K}\sum_{k=1}^K \mathbf{m}_{t+1}^{(k)}$ at iteration t. While in local error feedback, the auxiliary variable $\tilde{\mathbf{x}}_t$ is updated in a similar way shown by Eq. (3). We refer to $(\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)})$ as the compression error term which is usually trivial at the end of training. The small compression error makes the output parameters $\mathbf{x}_T$ similar to the auxiliary variable $\tilde{\mathbf{x}}_T$. However, is the auxiliary variable $\tilde{\mathbf{x}}_T$ the same as the training results of vanilla momentum SGD? The answer is "no" due to a slight but important difference: **the momentum term $\mathbf{m}_{t+1}^{(k)}$ is computed based on the gradient of $\mathbf{x}_t$** ($\nabla f(\mathbf{x}_t^{(k)};\xi_t^{(k)})$ in Algorithm 1 line 14) **but used to update $\tilde{\mathbf{x}}_t$**. We name it as the "gradient mismatch" problem, which can jeopardize the scalability of local error feedback in various tasks and models.

Resolving Gradient Mismatch Problem

To alleviate the effect of gradient mismatch, we propose a new step-ahead local error-feedback (SAEF) algorithm as summarized in Algorithm 1 (line 6 $\sim$ 12). According to the update of the auxiliary variable Eq. (3), for momentum SGD with local error feedback we have

$$\tilde{\mathbf{x}}_{t+1} = \tilde{\mathbf{x}}_t - \frac{\eta_t}{K}\sum_{k=1}^K(\mu\mathbf{m}_t^{(k)} + \nabla f(\mathbf{x}_t^{(k)};\xi_t^{(k)})) =$$
$$\tilde{\mathbf{x}}_t - \frac{\eta_t}{K}\sum_{k=1}^K(\mu\mathbf{m}_t^{(k)} + \nabla f(\tilde{\mathbf{x}}_t + (\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)});\xi_t^{(k)}))\,. \tag{4}$$

Relationship with Staleness. Asynchronous distributed training behaves in a similar pattern as the above equation. Let the staleness of the gradient computed at worker k be $\tau_t^{(k)}$ and one worker is selected to update the model at the server node in each iteration. Then we have the following update rule in asynchronous SGD (Lian et al. 2015):

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t\nabla f(\mathbf{x}_{t-\tau_t^{(k)}};\xi_t^{(k)})\,, \tag{5}$$

where the gradient mismatch also exists as parameters $\mathbf{x}_t$ is updated by the gradient computed at stale and different parameters $\mathbf{x}_{t-\tau_t^{(k)}}$. Consequently we regard the staleness of local error feedback as one because the compression error $(\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)})$ in Eq. (4) is computed at iteration $t-1$, while $\tau_t^{(k)}$ in Eq. (5) can be larger than one. Addressing

Algorithm 1 Distributed Momentum SGD with Double-Way Compression.

```
1:  Input: averaging period $p > 1$, number of iterations $T$, number of workers $K$, learning rate $\{\eta_t\}_{t=0}^{T-1}$, parameters $\mathbf{x}_0$, compression scheme $\mathcal{C}(\cdot)$ and the momentum constant $0 \leq \mu < 1$.
2:  Initialize: $\forall 1 \leq k \leq K$, initial local parameters $\mathbf{x}_0^{(k)} = \mathbf{x}_0$ and local error $\mathbf{e}_0^{(k)} = \mathbf{0}$ and local momentum buffer $\mathbf{m}_0^{(k)} = \mathbf{0}$. $\mathbf{x}_t^{(k)} = \mathbf{x}_t$ for all $t = 0, \cdots, T$.
3:  for $t = 0, \cdots, T-1$ do
4:      Worker-$k$:
5:      if Step Ahead Error Feedback (SAEF) then
6:          if mod $(t+1, p) = 0$ then
7:              Average local error $\mathbf{e}_t^{(k)} \leftarrow \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)}$
8:          end if
9:          $\mathbf{x}_{t+\frac{1}{2}}^{(k)} = \mathbf{x}_t^{(k)} - \mathbf{e}_t^{(k)}$        // One step ahead.
10:         $\mathbf{m}_{t+1}^{(k)} = \mu\mathbf{m}_t^{(k)} + \nabla f(\mathbf{x}_{t+\frac{1}{2}}^{(k)}; \xi_t^{(k)})$
11:         $\mathbf{x}_{t+1}^{(k)} = \mathbf{x}_{t+\frac{1}{2}}^{(k)} - \eta_t\mathbf{m}_{t+1}^{(k)}$ // Momentum SGD update.
12:         $\Delta_{t+1}^{(k)} = \mathbf{e}_t^{(k)} + \mathbf{x}_{t+\frac{1}{2}}^{(k)} - \mathbf{x}_{t+1}^{(k)}$
13:     else if Local Error Feedback (EF) then
14:         $\mathbf{m}_{t+1}^{(k)} = \mu\mathbf{m}_t^{(k)} + \nabla f(\mathbf{x}_t^{(k)}; \xi_t^{(k)})$
15:         $\mathbf{x}_{t+1}^{(k)} = \mathbf{x}_t^{(k)} - \eta_t\mathbf{m}_{t+1}^{(k)}$ // Momentum SGD update.
16:         $\Delta_{t+1}^{(k)} = \mathbf{e}_t^{(k)} + \mathbf{x}_t^{(k)} - \mathbf{x}_{t+1}^{(k)}$
17:     end if
18:     $\mathbf{e}_{t+1}^{(k)} = \Delta_{t+1}^{(k)} - \mathcal{C}(\Delta_{t+1}^{(k)})$
19:     Send $\mathcal{C}(\Delta_{t+1}^{(k)})$ to the server node.

20:     Server:
21:     $\Delta_{t+1} = \mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathcal{C}(\Delta_{t+1}^{(k)})$
22:     $\mathbf{e}_{t+1} = \Delta_{t+1} - \mathcal{C}(\Delta_{t+1})$
23:     Broadcast $\mathcal{C}(\Delta_{t+1})$ to all the worker nodes.

24:     Worker-$k$:
25:     $\mathbf{x}_{t+1}^{(k)} = \mathbf{x}_t^{(k)} - \mathcal{C}(\Delta_{t+1})$        // Re-update.
26: end for
27: Output: parameters $\mathbf{x}_T = \mathbf{x}_T^{(k)}$
```

gradient mismatch can be equivalent to reducing the effect of staleness.

The motivation for us to resolve gradient mismatch problem is that since $\mathbf{x}_t$ differs from $\tilde{\mathbf{x}}_t$ in the compression error term $(\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)})$, we can **improve the training of $\mathbf{x}_t$ by improving the training of $\tilde{\mathbf{x}}_t$**. To free the training of $\tilde{\mathbf{x}}_t$ from the effect of gradient mismatch, we propose to approximate the following update rules:

$$\tilde{\mathbf{x}}_{t+1} \approx \tilde{\mathbf{x}}_t - \frac{\eta_t}{K}\sum_{k=1}^K (\mu\mathbf{m}_t^{(k)} + \nabla f(\tilde{\mathbf{x}}_t; \xi_t^{(k)}))\,. \tag{6}$$

Before to quantitatively define the amount of the gradient mismatch, we first made some common assumptions in non-convex optimization.

Assumption 1 *(L-Lipschitz gradient) Assume the full loss function $F(\cdot)$ is L-smooth, that is, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ we have:*

$$\|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_2\,. \tag{7}$$

Assumption 2 *(Bounded variance) The stochastic gradient $\nabla f(\boldsymbol{x}_t^{(k)}; \xi_t^{(k)})$ has bounded variance:*

$$\mathbb{E}\|\nabla f(\boldsymbol{x}_t^{(k)}; \xi_t^{(k)}) - \nabla F(\boldsymbol{x}_t^{(k)})\|_2^2 \leq \sigma^2\,. \tag{8}$$

With the Assumptions 1 and 2, we define the amount of the gradient mismatch ϵ_t of local error feedback as:

$$\begin{aligned}\epsilon_t &:= \frac{1}{K}\sum_{k=1}^K \mathbb{E}\|\nabla f(\tilde{\mathbf{x}}_t; \xi_t^{(k)}) - \nabla f(\mathbf{x}_t^{(k)}; \xi_t^{(k)})\|_2^2 \\ &\leq L^2\mathbb{E}\|\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)}\|_2^2 + 4\sigma^2\,.\end{aligned} \tag{9}$$

Step Ahead

Although local error feedback is proved to have the same convergence rate $\mathcal{O}(\frac{1}{\sqrt{T}})$ as SGD, the gradient mismatch ϵ_t leads to an additional error term in the convergence bound. In stead of computing stochastic gradient $\nabla f(\mathbf{x}_t^{(k)}; \xi_t^{(k)})$ at $\mathbf{x}_t^{(k)}$, we propose to compute stochastic gradient $\nabla f(\mathbf{x}_{t+\frac{1}{2}}^{(k)}; \xi_t^{(k)})$ at $\mathbf{x}_{t+\frac{1}{2}}^{(k)} := \mathbf{x}_t^{(k)} - \mathbf{e}_t^{(k)}$ as in Algorithm 1 (line 6 $\sim$ 12). Note that the local error $\mathbf{e}_t^{(k)}$ is locally accessible without additional communication costs. By replacing $\mathbf{x}_t^{(k)}$ with $\mathbf{x}_{t+\frac{1}{2}}^{(k)}$, the gradient mismatch ϵ_t of our proposed SAEF becomes

$$\begin{aligned}\epsilon_t &:= \frac{1}{K}\sum_{k=1}^K \mathbb{E}\|\nabla f(\tilde{\mathbf{x}}_t; \xi_t^{(k)}) - \nabla f(\mathbf{x}_{t+\frac{1}{2}}^{(k)}; \xi_t^{(k)})\|_2^2 \\ &\leq \frac{L^2}{K}\sum_{k=1}^K \mathbb{E}\|\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)} - \mathbf{e}_t^{(k)}\|_2^2 + 4\sigma^2\,.\end{aligned} \tag{10}$$

When to step ahead? Our goal is to fix gradient mismatch (reduce staleness) to improve the training of $\tilde{\mathbf{x}}_t$. As we want a smaller upper bound of ϵ_t, it will be better to step ahead if $\frac{1}{K}\sum_{k=1}^K \mathbb{E}\|\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)} - \mathbf{e}_t^{(k)}\|_2^2 < \mathbb{E}\|\mathbf{e}_t + \frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)}\|_2^2$. This is intuitively true when we have a small variance because the effect of $(\frac{1}{K}\sum_{k=1}^K \mathbf{e}_t^{(k)} - \mathbf{e}_t^{(k)})$ is cancelled in expectation. The following proposition illustrates it when the variance is smaller than the square of expectation. Note it only gives a motivation of our proposed method as the local error is usually not identical in SAEF-SGD and EF-SGD.

Proposition 1 *If Assumptions 1 and 2 exist, for the the same error $\boldsymbol{e}_t^{(k)}$ $(k = 1, \cdots, K)$ and $\boldsymbol{e}_t$, the upper bound of ϵ_t we can prove in SAEF-SGD is better than that in EF-SGD if $Var(\boldsymbol{e}_t^{(k)}) \leq \|\mathbb{E}\boldsymbol{e}_t^{(k)}\|_2^2$.*

How related to compression? Take the flexible Top-K gradient compression as an example, where only large gradient components are sent with the rest set to zero. We regard $\epsilon_t = \|(\epsilon_{t,1}, \cdots, \epsilon_{t,d})\|_2^2$ in an element-wise way, which means that the improvement of ϵ_t related to $\mathbf{e}_t^{(k)}$ of SAEF-SGD over EF-SGD is proportional to the number of non-zero components in $\mathbf{e}_t^{(k)}$. When the Top-K compression is more aggressive and fewer gradient components are sent, there are more non-zero components in $\mathbf{e}_t^{(k)}$. In other words, the improvement of SAEF-SGD over EF-SGD favors more aggressive Top-K compression, which is desirable due to lower communication costs. The less aggressive compression incurs smaller performance loss, but the improvement of local error feedback is not as essential.

Error Averaging

When the gradient mismatch is too hard to resolve only by step ahead, we propose to average the local error $\mathbf{e}_t^{(k)} \leftarrow \frac{1}{K}\sum_{k=1}^{K}\mathbf{e}_t^{(k)}$. It cancels the effect of local compression error $\mathbf{e}_t^{(k)}$ with a brutal force at the averaging iteration t:

$$\begin{aligned}\epsilon_t &:= \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(\tilde{\mathbf{x}}_t;\xi_t^{(k)}) - \nabla f(\mathbf{x}_{t+\frac{1}{2}}^{(k)};\xi_t^{(k)})\|_2^2 \\ &\leq \frac{L^2}{K}\sum_{k=1}^{K}\mathbb{E}\|\mathbf{e}_t\|_2^2 + 4\sigma^2 .\end{aligned} \tag{11}$$

The error averaging operation can be conducted either in a master-slave way or the ring-based all-reduce way to avoid the traffic jam in the master-slave framework. However, this is still a costly operation and we do not want to conduct it frequently. In fact, we should average the local error every $p(>1)$ iteration depending on how fast the local error diverges in different nodes. When $p = \infty$ we do not perform error averaging. To make a fair comparison in experiments, for SAEF with error averaging we apply the less aggressive gradient compression to balance the communication cost.

How much contribution? Error averaging set $\mathbb{E}\|\frac{1}{K}\sum_{k=1}^{K}\mathbf{e}_t^{(k)} - \mathbf{e}_t^{(k)}\|_2^2$ to zero every p iteration. It reduces the upper bound of ϵ_t related to $\mathbf{e}_t^{(k)}$ at least by a factor of $\frac{1}{p}$. Moreover, it prevents the local error $\mathbf{e}_t^{(k)}$ in different worker k from further diverging. Consequently averaging error every p iteration reduce the effect of local error $\mathbf{e}_t^{(k)}$ at worker nodes by a factor larger than $\frac{1}{p}$.

Theoretical Analysis

We further make Assumption 3 which is common in non-convex optimization, and Assumption 4 which has been leveraged in previous works (Stich, Cordonnier, and Jaggi 2018; Karimireddy et al. 2019; Zheng, Huang, and Kwok 2019; Basu et al. 2019). For simplicity, we denote $\min_{t=0,1,\cdots,T-1}$ as min. The theoretical results do not include error averaging.

Assumption 3 *(Bounded second moment) The full gradient is bounded:*

$$\|\nabla F(\boldsymbol{x}_t^{(k)})\|_2^2 \leq M^2 . \tag{12}$$

It implies the second moment of the stochastic gradient is bounded if Assumption 2 exists at the same time:

$$\mathbb{E}\|\nabla f(\boldsymbol{x}_t^{(k)};\xi_t^{(k)})\|_2^2 \leq \sigma^2 + M^2 . \tag{13}$$

Assumption 4 *(δ-approximate compressor) The compression function $\mathcal{C}(\cdot):\mathbb{R}^d \to \mathbb{R}$ is a δ-approximate compressor for $0 < \delta \leq 1$ if for all $\boldsymbol{v} \in \mathbb{R}^d$,*

$$\|\mathcal{C}(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 \leq (1-\delta)\|\boldsymbol{v}\|_2^2 . \tag{14}$$

Lemma 1 *With Assumptions 2, 3 and 4, we have*

$$\begin{aligned}&\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\boldsymbol{e}_t + \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{e}_t^{(k)} - \boldsymbol{e}_t^{(k)}\|_2^2 \\ &\leq C\cdot\frac{1-\delta}{1-\sqrt{1-\delta}}\frac{\eta_{max}^2(M^2+\sigma^2)}{(1-\mu)^2} ,\end{aligned} \tag{15}$$

where the constant $C = \frac{2-\delta}{1-\sqrt{1-\delta}} + \frac{K-1}{K}$.

Lemma 1 is an essential intermediate result for the convergence analysis of SAEF both with or without momentum as it helps to bound the gradient mismatch ϵ_t.

SAEF-SGD

Theorem 1 *If Assumptions 1, 2, 3 and 4 exist, and the learning rate $0 < \eta_t = \eta < \frac{3}{4L}$ for all $t = 0,\cdots,T-1$, for SAEF-SGD we have*

$$\begin{aligned}\min\mathbb{E}\|\nabla F(\tilde{\boldsymbol{x}}_t)\|_2^2 &\leq \frac{4[F(\tilde{\boldsymbol{x}}_0) - F(\tilde{\boldsymbol{x}}^*)]}{\eta(3-4\eta L)T} + \frac{2\eta L\sigma^2}{(3-4\eta L)K} \\ &+ \frac{C(1-\delta)}{1-\sqrt{1-\delta}}\frac{4(\eta L+1)\eta^2L^2}{3-4\eta L}(M^2+\sigma^2) .\end{aligned} \tag{16}$$

Theorem 2 *If Assumptions 1, 2, 3 and 4 exist, and the learning rate $0 < \eta_t = \eta < \frac{3}{2L}$ for all $t = 0,\cdots,T-1$, for SAEF-SGD we have*

$$\begin{aligned}\min\mathbb{E}\|\nabla F(\boldsymbol{x}_t^{(k)})\|_2^2 &\leq \frac{4[F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)]}{\eta(3-2\eta L)T} + \frac{4\eta L\sigma^2}{(3-2\eta L)K} \\ &+ \left(1+\frac{8C}{3-2\eta L}\right)\frac{1-\delta}{1-\sqrt{1-\delta}}\eta^2L^2(M^2+\sigma^2) .\end{aligned} \tag{17}$$

Corollary 1 *Under the same conditions of Theorem 2, the compression error term*

$$\left(1+\frac{8C}{3-2\eta L}\right)\frac{1-\delta}{1-\sqrt{1-\delta}}\eta^2L^2(M^2+\sigma^2) \tag{18}$$

in the upper bound of Theorem 2 is tighter than the corresponding EF-SGD compression error term in (Zheng, Huang, and Kwok 2019):

$$\frac{32L^2(1-\delta)(M^2+\sigma^2)}{\delta^2}(1+\frac{16}{\delta^2})\frac{\eta^2}{3-2\eta L} . \tag{19}$$

The effect of a tighter bound achieved above by SAEF will gradually vanish with a decaying learning rate as the total training steps T goes to infinity. However, in practical and common training of deep neural networks, the learning rate is usually chosen to be large in the beginning and seldom goes to zero in the end, which contributes to the faster training of SAEF than local error feedback.

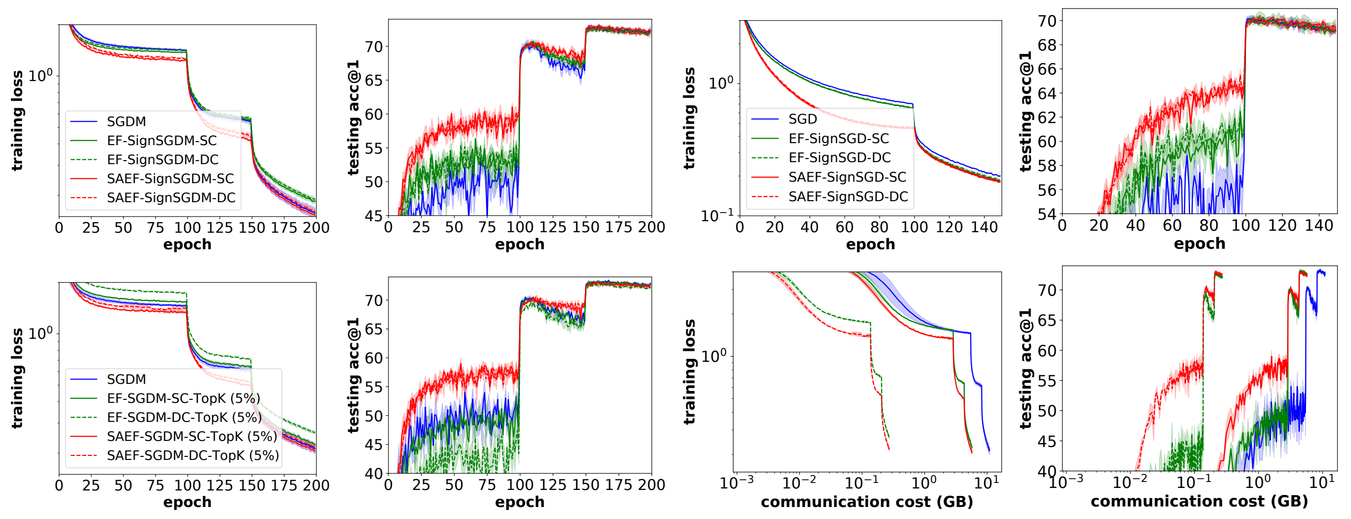

Figure 1: Train ResNet-56 on CIFAR-100. Mean metrics are plotted with standard deviation (shaded area). The top row employs 4 workers and SignSGD compression with momentum SGD applied in the left two figures and SGD applied in the right two figures. The bottom row employs 8 workers, Top-K compression and momentum SGD, where training curves regarding epochs are shown in the left two figures and training curves regarding communication costs are shown in the right two figures.

Corollary 2 *Under the same conditions of Theorem 2, let the learning rate $\eta < \frac{c\sqrt{K}}{\sqrt{T}}$, where $c > 0$ is some constant. Then the convergence rate of $\boldsymbol{x}_t^{(k)}$ in SAEF-SGD satisfies*

$$\min_{t=0,\cdots,T-1} \mathbb{E}\|\nabla F(\boldsymbol{x}_t^{(k)})\|_2^2 = \mathcal{O}(\frac{1}{\sqrt{KT}})\,. \tag{20}$$

Please see Section 5 of the Supplement for the proof.

SAEF-SGD with Momentum

Theorem 3 *If Assumption 1, 2, 3 and 4 exist, and the learning rate $0 < \eta_t = \eta$ satisfies $\alpha := 1 - \frac{\eta L}{1-\mu} - \frac{2\mu^2\eta^2L^2}{(1-\mu)^4} > 0$ for all $t = 0, \cdots, T-1$, for SAEF-SGD with momentum we have*

$$\begin{aligned}\min \mathbb{E}\|\nabla F(\boldsymbol{x}_t^{(k)})\|_2^2 \le{}& \frac{4(1-\mu)[F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)]}{\alpha\eta T}\\ &+ \frac{2\left(1+\frac{2\mu^2\eta L}{(1-\mu)^3}\right)\eta L\sigma^2}{\alpha(1-\mu)K}\\ &+ (4C+\alpha)\,\frac{1-\delta}{1-\sqrt{1-\delta}}\,\frac{\eta^2L^2(M^2+\sigma^2)}{\alpha(1-\mu)^2}\,,\end{aligned} \tag{21}$$

$$\begin{aligned}\min \mathbb{E}\|\nabla F(\tilde{\boldsymbol{x}}_t)\|_2^2 \le{}& \frac{4(1-\mu)[F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)]}{\alpha\eta T}\\ &+ \frac{2\left(1+\frac{2\mu^2\eta L}{(1-\mu)^3}\right)\eta L\sigma^2}{\alpha(1-\mu)K}\\ &+ (\frac{4}{\alpha}+2)\frac{C(1-\delta)}{1-\sqrt{1-\delta}}\frac{\eta^2L^2(M^2+\sigma^2)}{(1-\mu)^2}\,.\end{aligned} \tag{22}$$

Corollary 3 *Under the same conditions of Theorem 3, let the learning rate $\eta < \frac{c\sqrt{K}}{\sqrt{T}}$, where $c > 0$ is some constant. Then the convergence rate of $\boldsymbol{x}_t^{(k)}$ in SAEF-SGD with momentum satisfies*

$$\min_{t=0,\cdots,T-1} \mathbb{E}\|\nabla F(\boldsymbol{x}_t^{(k)})\|_2^2 = \mathcal{O}(\frac{1}{\sqrt{KT}})\,. \tag{23}$$

Please see Section 6 of the Supplement for the proof.

Experiments

All experiments are implemented with PyTorch (Paszke et al. 2019). We first explain the notations of different methods "(EF, SAEF)-(SGD, SGDM, SignSGD, SignSGDM)-(SC, DC)-(TopK)" as used in Figures 1, 2, and 3:

- Local error feedback (EF) or our proposed step ahead error feedback (SAEF).
- SGD, momentum SGD (SGDM), SGD with SignSGD compression (SignSGD), momentum SGD with SignSGD compression (SignSGDM).
- Single-way compression (SC), that is, no compression of what the server sends back to the workers, or double-way compression (DC).
- Whether to use Top-K gradient sparsification. If Top-K is employed, we specify the sparsity in percentage. $p = \infty$ (no error averaging) by default unless specified otherwise. All compression are performed in a layer-wise way.

CIFAR Settings

We train the ResNet-56 (He et al. 2016) model with multiple workers (GPUs) on CIFAR-100 (Krizhevsky, Hinton et al. 2009) image classification task. We report the mean and standard deviation metrics over 5 runs. The base learning rate is 0.1 and the total batch size is 128. The momentum constant is 0.9 and the weight decay is 5×10^{-4}. For momentum SGD the model is trained for 200 epochs with a

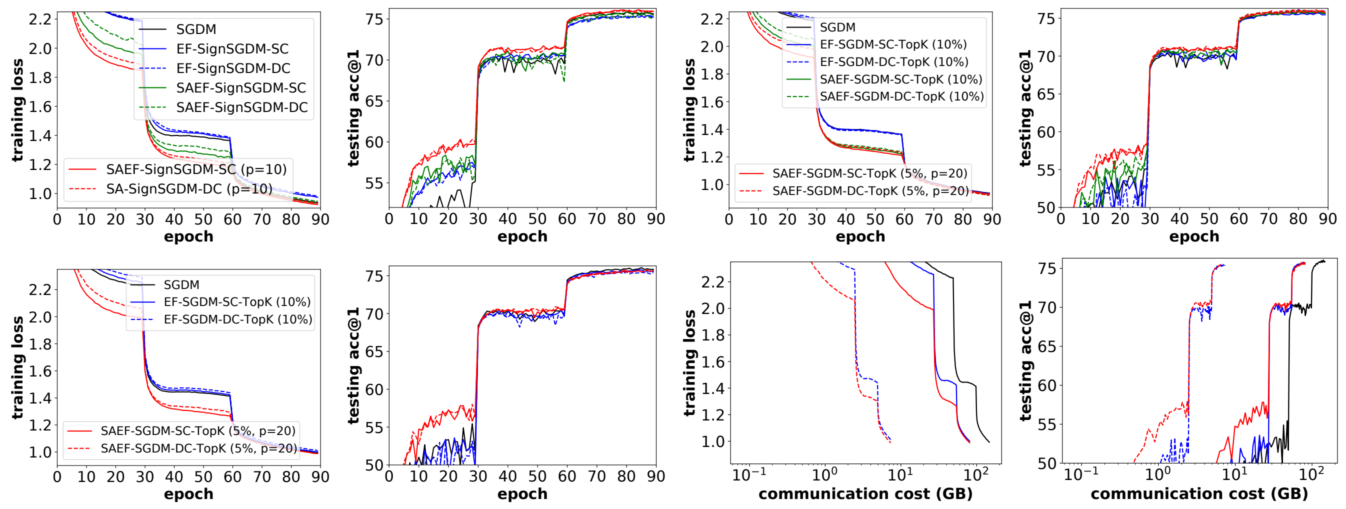

Figure 2: Train ResNet-50 on ImageNet. Error averaging is compared. The top row employs 4 workers with SignSGD compression applied in the left two figures and Top-K compression applied in the right two figures. The bottom row employs 8 workers and Top-K compression, where training curves regarding epochs are shown in the left two figures and training curves regarding communication costs are shown in the right two figures.

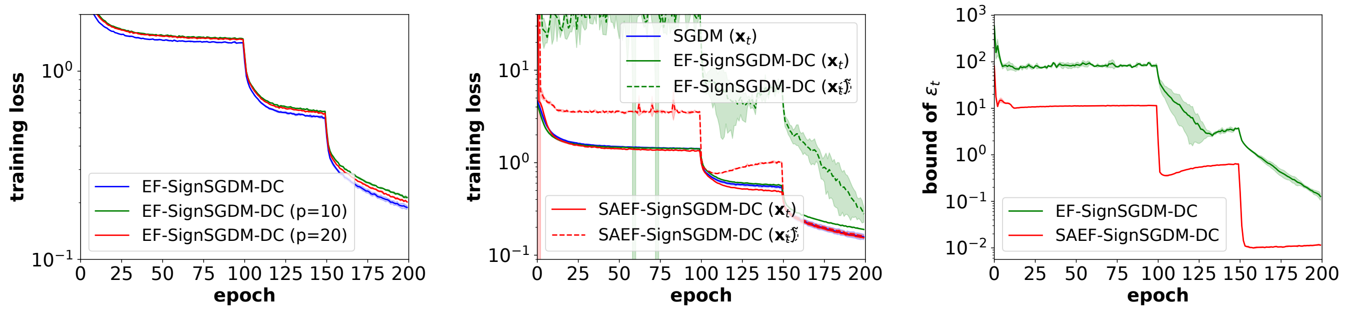

Figure 3: Train ResNet-56 on CIFAR-100 with 4 workers, SignSGD compression and momentum SGD related methods. Left: error averaging in local error feedback. Middle: $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$ in SAEF and local error feedback. Right: the bound of gradient mismatch (L and σ^2 are ignored).

learning rate decay of 0.1 at epoch 100 and 150. For SGD the model is trained for 150 epochs with a learning rate decay of 0.1 at epoch 100 because there is barely any further improvement of the testing performance if we do a second learning rate decay. Random cropping, random flipping, and standardization are applied as data augmentation techniques.

ImageNet Settings

We train the ResNet-50 model with multiple workers (GPUs) on ImageNet (Russakovsky et al. 2015) image classification tasks. The model is trained for 90 epochs with a learning rate decay of 0.1 at epoch 30 and 60. The base learning rate is 0.1 and the total batch size is 256. The momentum constant is 0.9 and the weight decay is 1×10^{-4}. Similar data augmentation techniques as in CIFAR-100 experiments are applied.

Performance Comparison

Faster Convergence. The training curves in Figures 1 and 2 show that employing our proposed SAEF in SGD/momentum SGD, with SignSGD/Top-K compression and single-way/double-way compression all lead to significantly faster convergence of the training loss. It is **not only faster than local error feedback but also vanilla SGD/momentum SGD with full precision gradient**. For local error feedback, we observe that its training loss is very similar to that of SGD/momentum SGD. But sometimes it may perform worse in CIFAR-100 experiments as shown in the bottom left of Figure 1, and the final training loss as shown in the top left of Figure 1. Its initial training performance can also perform worse in ImageNet experiments as shown in the bottom left of Figure 2.

Better Initial Generalization. Although we observe a very similar final testing performance for SAEF, local error feedback, and vanilla methods, SAEF always enjoys a better testing performance before the second learning rate decay in momentum SGD experiments. The improvement is very significant, especially during the initial training. This can be crucial in the communication-constraint scenario where we need gradient compression to reduce the cost. As shown in

CIFAR-100	EF	$p=\infty$	$p=40$	$p=20$	$p=10$	$p=5$	$p=1$
Top-1%	50.00 ± 0.70	60.59 ± 0.28	62.83 ± 0.27	63.96 ± 0.35	64.55 ± 0.20	65.62 ± 0.24	65.89 ± 0.32
Top-5%	50.06 ± 1.17	60.21 ± 0.64	60.00 ± 0.64	60.94 ± 0.34	61.72 ± 0.13	61.81 ± 0.08	62.10 ± 0.19
Top-10%	51.11 ± 0.24	57.78 ± 0.21	57.85 ± 0.22	58.22 ± 0.29	58.34 ± 0.38	58.19 ± 0.16	58.76 ± 0.31

Table 1: Best Top-1 Testing Accuracy (%) at epoch 100 of training ResNet-56 on CIFAR-100 using SAEF to demonstrate its faster convergence. We use Top-K sparsification (the first column shows the sparsity) and double-way compression. The second column is the result of EF (local error feedback) for comparison. We report the result in the form of (mean $\pm$ standard deviation) over 5 runs.

the bottom right two figures of Figure 1 and Figure 2, SAEF achieves **much better training and testing performance under the same communication budget than both local error feedback and vanilla methods**. Note that in Figure 2, we employ a more aggressive compression scheme for SAEF with error averaging to maintain the same communication budget. Error averaging improves SAEF (the top right two figures of Figure 2) but slightly *degrades local error feedback's convergence* (the left of Figure 3).

Effect of Gradient Mismatch. In the right of Figure 3, local error feedback features a much larger bound of gradient mismatch ϵ_t during the whole training, contributing to a worse $\tilde{\mathbf{x}}_t$ and a larger gap between the training of $\mathbf{x}_t$ and the auxiliary variable $\tilde{\mathbf{x}}_t$ as shown in the middle of Figure 3. The gap is even more obvious during the initial training. By reducing this gap with SAEF we achieve faster training using compressed gradients. Note that the training curves of $\tilde{\mathbf{x}}_t$ may seem poor in the initial training because we have tuned the best hyperparameters for the real trained model $\mathbf{x}_t$.

Effect of Compression Ratio and Averaging Period p. Firstly, we stress that the averaging period p should be *large*, so that the local error $\mathbf{e}_t^{(k)}$ will be communicated much more *infrequently* than the gradient. As a matter of fact, we use $p = \infty$ in all our experiments except the 8-worker distributed training of ResNet-50 on ImageNet, where $p = 20$ with Top-5% gradient sparsification. To explore the effect of the different combinations of the compression ratio and averaging period, we report the top-1 testing accuracy using Top-K compression with different sparsity and vary the averaging period. We summarize it in Table 1. The results confirm our previous analysis that the improvement of SAEF over local error feedback gets enlarged as the compression scheme becomes more aggressive. Decreasing the averaging period usually can further accelerate the training at the cost of a larger communication budget, and it is also more obvious for an aggressive compression scheme. However, even if we do not perform error averaging ($p = \infty$), there is still a considerable improvement by using SAEF.

Related Works

Most existing works employ local error feedback as a standard technique in dealing with the performance loss resulting from aggressive gradient compression. We believe that they may replace local error feedback with our proposed SAEF both theoretically and empirically.

The leverage of local error feedback can be as early as (Seide et al. 2014) for accelerating the training of speech models. Lin et al. (2018) proposed to locally accumulate those small gradient components until they reach a certain threshold before sending. ECQ-SGD (Wu et al. 2018) analyzed local error feedback for quantized gradients on quadratic functions. Deterioration of training performance can be observed in ECQ-SGD experiments. The Top-K compression has been proposed in (Strom 2015; Aji and Heafield 2017; Alistarh et al. 2018; Stich, Cordonnier, and Jaggi 2018). Combine it with local error feedback and we can make each parameter get updated sooner or later. Local error feedback was first utilized to analyze and fix the testing performance loss resulting from SignSGD compression in (Karimireddy et al. 2019). Zheng, Huang, and Kwok (2019) later developed it for distributed momentum SGD with double-way blockwise SignSGD compression. (Basu et al. 2019) combined gradient compression, local error feedback, and local SGD but only considered single-way compression. Asynchronous training is also considered in (Basu et al. 2019). However, all these works *did not* show that we can train faster with compressed gradients without loss of performance.

We note that certain gradient compression scheme may accelerate the initial training but lead to performance loss more or less in the end. SignSGD, for example, can be faster than SGD in the beginning but quickly deteriorates in terms of the final performance. In this work, however, we have been focused on improving local error feedback with common gradient compression schemes and without performance loss.

Conclusion

In this paper, we first identified the "gradient mismatch" problem in the local error feedback method (to the best of our knowledge, this is the first paper to systematically discuss this problem) and showed that this issue can cause performance loss in local error feedback. After that, we proposed a new SAEF (Step-Ahead Error Feedback) algorithm to train faster with compressed gradient than local error feedback and vanilla optimization methods with full precision gradient, both in terms of the performance regarding training epochs and communication costs. We theoretically show that our SAEF algorithm achieves a better convergence bound than local error feedback and empirically validate its faster convergence speed via image classification tasks. We also explore different experimental settings to confirm the scalability of SAEF.

Acknowledgements

This work was partially supported by NSF IIS 1845666, 1852606, 1838627, 1837956, 1956002, 2040588.

References

Aji, A. F.; and Heafield, K. 2017. Sparse Communication for Distributed Gradient Descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 440–445. Copenhagen, Denmark: Association for Computational Linguistics. doi:10.18653/v1/D17-1045. URL https://www.aclweb.org/anthology/D17-1045.

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

Alistarh, D.; Hoefler, T.; Johansson, M.; Konstantinov, N.; Khirirat, S.; and Renggli, C. 2018. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, 5973–5983.

Basu, D.; Data, D.; Karakus, C.; and Diggavi, S. 2019. Qsparse-local-SGD: Distributed SGD with Quantization, Sparsification and Local Computations. In *Advances in Neural Information Processing Systems*, 14668–14679.

Bernstein, J.; Wang, Y.-X.; Azizzadenesheli, K.; and Anandkumar, A. 2018. signSGD: Compressed Optimisation for Non-Convex Problems. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 560–569. Stockholmsmässan, Stockholm Sweden: PMLR. URL http://proceedings.mlr.press/v80/bernstein18a.html.

Chen, T.; Giannakis, G.; Sun, T.; and Yin, W. 2018. LAG: Lazily aggregated gradient for communication-efficient distributed learning. In *Advances in Neural Information Processing Systems*, 5050–5060.

Hashemi, S. H.; Jyothi, S. A.; and Campbell, R. H. 2018. TicTac: Accelerating distributed deep learning with communication scheduling. *arXiv preprint arXiv:1803.03288* .

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ho, Q.; Cipar, J.; Cui, H.; Lee, S.; Kim, J. K.; Gibbons, P. B.; Gibson, G. A.; Ganger, G.; and Xing, E. P. 2013. More effective distributed ml via a stale synchronous parallel parameter server. In *Advances in neural information processing systems*, 1223–1231.

Huang, Y.; Yan, X.; Jiang, G.; Jin, T.; Cheng, J.; Xu, A.; Liu, Z.; and Tu, S. 2019. Tangram: bridging immutable and mutable abstractions for distributed data analytics. In *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*, 191–206.

Karimireddy, S. P.; Rebjock, Q.; Stich, S.; and Jaggi, M. 2019. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In *International Conference on Machine Learning*, 3252–3261.

Koloskova, A.; Stich, S. U.; and Jaggi, M. 2019. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340* .

Konečnỳ, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* .

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images .

Lian, X.; Huang, Y.; Li, Y.; and Liu, J. 2015. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, 2737–2745.

Lian, X.; Zhang, C.; Zhang, H.; Hsieh, C.-J.; Zhang, W.; and Liu, J. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, 5330–5340.

Lian, X.; Zhang, W.; Zhang, C.; and Liu, J. 2018. Asynchronous Decentralized Parallel Stochastic Gradient Descent. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 3049–3058. PMLR. URL http://proceedings.mlr.press/v80/lian18a.html.

Lin, Y.; Han, S.; Mao, H.; Wang, Y.; and Dally, B. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=SkhQHMW0W.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 8024–8035.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3): 211–252. doi:10.1007/s11263-015-0816-y.

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

Stich, S. U. 2019. Local SGD Converges Fast and Communicates Little. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=S1g2JnRcFX.

Stich, S. U.; Cordonnier, J.-B.; and Jaggi, M. 2018. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, 4447–4458.

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

Sun, J.; Chen, T.; Giannakis, G.; and Yang, Z. 2019. Communication-efficient distributed learning via lazily aggregated quantized gradients. In *Advances in Neural Information Processing Systems*, 3365–3375.

Tang, H.; Gan, S.; Zhang, C.; Zhang, T.; and Liu, J. 2018a. Communication compression for decentralized training. In *Advances in Neural Information Processing Systems*, 7652–7662.

Tang, H.; Lian, X.; Qiu, S.; Yuan, L.; Zhang, C.; Zhang, T.; and Liu, J. 2019. DeepSqueeze: Decentralized meets error-compensated compression. *arXiv preprint arXiv:1907.07346* .

Tang, H.; Lian, X.; Yan, M.; Zhang, C.; and Liu, J. 2018b. D^2: Decentralized Training over Decentralized Data. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4848–4856. Stockholmsmässan, Stockholm Sweden: PMLR. URL http://proceedings.mlr.press/v80/tang18a.html.

Wen, W.; Xu, C.; Yan, F.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2017. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, 1509–1519.

Wu, J.; Huang, W.; Huang, J.; and Zhang, T. 2018. Error compensated quantized SGD and its applications to large-scale distributed optimization. *arXiv preprint arXiv:1806.08054* .

Xu, A.; Huo, Z.; and Huang, H. 2020a. On the Acceleration of Deep Learning Model Parallelism With Staleness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2088–2097.

Xu, A.; Huo, Z.; and Huang, H. 2020b. Optimal Gradient Quantization Condition for Communication-Efficient Distributed Training. *arXiv preprint arXiv:2002.11082* .

Zheng, S.; Huang, Z.; and Kwok, J. 2019. Communication-efficient distributed blockwise momentum sgd with error-feedback. In *Advances in Neural Information Processing Systems*, 11446–11456.