Communication-Compressed Adaptive Gradient Method for Distributed Nonconvex Optimization

Yujia Wang[1], **Lu Lin**[2], **Jinghui Chen**[1]

[1] College of Information Sciences and Technology, Pennsylvania State University
[2] Department of Computer Science, University of Virginia
yjw5427@psu.edu, ll5fy@virginia.edu, jzc5917@psu.edu

Abstract

Due to the explosion in the size of the training datasets, distributed learning has received growing interest in recent years. One of the major bottlenecks is the large communication cost between the central server and the local workers. While error feedback compression has been proven to be successful in reducing communication costs with stochastic gradient descent (SGD), there are much fewer attempts in building communication-efficient adaptive gradient methods with provable guarantees, which are widely used in training large-scale machine learning models. In this paper, we propose a new communication-compressed AMSGrad for distributed nonconvex optimization problem, which is provably efficient. Our proposed distributed learning framework features an effective gradient compression strategy and a worker-side model update design. We prove that the proposed communication-efficient distributed adaptive gradient method converges to the first-order stationary point with the same iteration complexity as uncompressed vanilla AMSGrad in the stochastic nonconvex optimization setting. Experiments on various benchmarks back up our theory.

1 Introduction

The recent success of deep learning and large-scale training made it possible to use machine learning to solve complicated real-world tasks, such as computer vision (He et al. 2016), speech recognition (Hinton et al. 2012), natural language processing (Devlin et al. 2018), etc. Due to the explosion in the size of training datasets in recent years, single GPU training on such models can easily take up to weeks or even months to finish. As a consequence, distributed training algorithms have attracted growing interest over the years. In standard distributed settings with one central server and n workers, local workers parallelly compute local gradients, while the server aggregates the gradients from the workers, updates the model, and sends the new model back to the workers. However, data transmissions across the machines can be quite expansive in terms of both communication costs (especially for cellular networks) and latency. Therefore, various methods were studied in order to achieve communication-efficient distributed learning by either reducing communication bits (Stich, Cordonnier, and Jaggi 2018; Basu et al. 2019; Karimireddy et al. 2019) or saving the number of communication rounds (Zheng et al. 2017; Chen et al. 2021b).

One of the principled approaches for effectively reducing the communication cost is to perform gradient compression before transmissions. It directly compresses the local fresh gradient on each worker before uploading the gradient to the server. However, the compression would slow down the convergence or even diverge (Beznosikov et al. 2020) due to the loss of information at each compression step. Later on, error feedback strategy (Stich, Cordonnier, and Jaggi 2018; Karimireddy et al. 2019) was proposed to alleviate this problem and reduce the information loss by proposing a compensating error sequence. Each worker compresses and uploads the combination of the last step compensating error and the local fresh gradient instead of compressing the gradient directly. Recent studies (Seide et al. 2014; Karimireddy et al. 2019; Stich, Cordonnier, and Jaggi 2018) show that error feedback has been widely used in distributed SGD to communicate efficiently and ensure the same convergence rate as vanilla SGD.

While communication-efficient distributed SGD has been widely studied, there are much fewer attempts in building communication-efficient distributed adaptive gradient methods (Kingma and Ba 2014; Reddi, Kale, and Kumar 2019). It has been shown that SGD works less efficiently compared with adaptive gradient methods when training large-scale models such as BERT (Devlin et al. 2018), GPT-3 (Brown et al. 2020) and GAN (Goodfellow et al. 2014). One of the major challenges is that the traditional error feedback mechanism is not compatible with adaptive gradient methods since the variance term (i.e., second-order moments of the historical gradients) in the adapted gradient method can be unstable due to the accumulated compression error (Tang et al. 2021). 1-bit Adam (Tang et al. 2021) partially solve this problem by first using vanilla Adam at the beginning of training to get an estimate of variance term, then freezing this variance term and performing distributed Adam with the fixed variance. While this solution indeed gets rid of the unstable variance issue, the adaptivity is no longer changing and the major part of the algorithm is actually more similar to SGD with momentum. Due to the same reason, the convergence guarantee provided in (Tang et al. 2021) is not related to distributed Adam but is an extension to distributed

SGD with momentum.

In this paper, we develop a new compression method which is provably efficient, namely **C**ommunication-compressed **D**istributed **Ada**ptive gradient **M**ethod (CD-Adam). We revisit the unstable variance issue in distributed AMSGrad (Reddi, Kale, and Kumar 2019) and identify the need for an improved gradient compression strategy. Specifically, we adopt the Markov compression sequence recently proposed by Richtárik, Sokolov, and Fatkhullin (2021), which ideally could lead to contractive compression error for gradient descent if the gradient sequence (to be compressed) is convergent. Note that this is not simply applying the conclusion of Richtárik, Sokolov, and Fatkhullin (2021) to the distributed AMSGrad setting, as Richtárik, Sokolov, and Fatkhullin (2021) only deals with worker-to-server compression. While extending Richtárik, Sokolov, and Fatkhullin (2021) to both worker-to-server and server-to-worker compression may seem straightforward for standard gradient descent, it is particularly tricky for the adaptive gradient method (see Section 5 for details). Therefore, it requires carefully designed algorithms and analyses to build a provably efficient distributed adaptive gradient method.

We summarize our contribution as follows:

- We propose a new communication-compressed distributed AMSGrad approach which solves the bottleneck of applying communication compression strategies for fully-functional[1] adaptive gradient methods in the distributed setting. The proposed method largely reduces the communication cost by enforcing both worker-to-server and server-to-worker communication compression without any warm-up stages.
- We theoretically prove the convergence of our proposed algorithm in the nonconvex stochastic optimization setting. Specifically, we show that our proposed fully compressed CD-Adam can reach its first-order ϵ-stationary point within $O(1/\epsilon^2)$ iterations, which is the same iteration complexity as the uncompressed vanilla AMSGrad. This suggests that without applying any variance-freezing tricks, the fully compressed distributed adaptive gradient method can still provably converge to its ϵ-stationary point.
- Thorough experiments on various real-world benchmarks show that our proposed CD-Adam reduces the communication cost by around $32\times$ over the original AMSGrad and around $5\times$ over 1-bit Adam.

2 Related Work

2.1 Stochastic gradient descent and adaptive gradient methods

Stochastic gradient descent (SGD) (Robbins and Monro 1951) is broadly used in training large-scale machine learning problems. Despite being simple to implement, SGD can be sensitive to parameters such as learning rate and slow to converge. Adaptive gradient methods were proposed to further improve over SGD. Adam (Kingma and Ba 2014), one of the most popular adaptive gradient methods, has shown to be fast convergent and also robust to hyper-parameters like learning rate. It designs an adaptive learning rate for each different coordinate using the past gradient. Aside from Adam, AdaGrad (Duchi, Hazan, and Singer 2011) applied the second moment of the gradient to adaptive the learning rate, RMSProp (Tieleman, Hinton et al. 2012) further used the second moment of the gradient with a decay rate, AdaDelta (Zeiler 2012) is an extension of AdaGrad with a non-increasing learning rate. Later on, Reddi, Kale, and Kumar (2019) pointed out a non-convergence issue of Adam, and proposed a new AMSGrad algorithm for ensuring the convergence. Zaheer et al. (2018) studied the effect of adaptive denominator constant ν and mini-batch size in the convergence of adaptive gradient methods. AdaBound (Luo et al. 2019) proposed with both upper and lower bound for the variance term of Adam. AdamW (Loshchilov and Hutter 2019) proposed to fix the weight decay regularization in Adam by decoupling the weight decay from the gradient update. Chen et al. (2020) proposed a partially adaptive gradient method and proved its convergence in nonconvex settings. Chen et al. (2019); Zhou et al. (2018) showed the convergence rate of a class of adaptive gradient methods under the nonconvex stochastic optimization setting. Alacaoglu et al. (2020) proposed a new framework to derive data-dependent regret bounds with a constant momentum parameter in various settings.

2.2 Distributed SGD and error feedback compression

For communication-efficient distributed SGD, one of the most common strategies is to compress the gradients before uploading. Alistarh et al. (2017) provided a theoretical analysis of the centralized compressed distributed SGD. Seide et al. (2014) compressed the coordinates of the gradient into ± 1 by its sign. Bernstein et al. (2018) proposed signSGD and proved its convergence in the nonconvex setting. Variant works has applied kinds of compression methods such as sparsification (Stich, Cordonnier, and Jaggi 2018; Basu et al. 2019), quantization (Karimireddy et al. 2019), and sketching (Ivkin et al. 2019).

Error feedback largely improves the compression error bound and has been shown to be critical for ensuring fast convergence of the compression mentioned above. Seide et al. (2014) showed that with error feedback, even 1-bit gradient communication under SGD still obtains the convergence rate of vanilla SGD. Karimireddy et al. (2019) applied error feedback to signSGD (Bernstein et al. 2018) under nonconvex setting, Stich, Cordonnier, and Jaggi (2018) performed error feedback in sparsity strong convex settings, and Stich and Karimireddy (2019) proposed a error-feedback framework also works on nonconvex but single node setting. There are also more works on the distributed SGD under nonconvex optimization (Tang et al. 2019; Koloskova et al. 2019; Basu et al. 2019).

[1]Here we use "fully-functional" to differentiate with the variance-freezed Adam used in Tang et al. (2021).

2.3 Communication-efficient distributed adaptive gradient method

There are only fewer attempts in developing communication-efficient distributed adaptive gradient methods. 1-bit Adam (Tang et al. 2021) adopted a variance-freezed Adam by pointing out that the variance term of Adam becomes stable in later training stages. Combined with error feedback, 1-bit Adam achieves the same convergence rate as distributed SGD. Chen et al. (2021a) developed a distributed quantized Adam with error feedback. However, the proposed algorithm in Chen et al. (2021a) can only converge to its ϵ-stationary point with worker-to-server compression alone but not fully compressed Adam with both worker-to-server and server-to-worker compression. Another related work is CADA (Chen et al. 2021b), which reduced the communication rounds instead of performing communication compression. CADA adaptively reused the stale update parameters, and it achieved the same convergence rate as vanilla AMSGrad for nonconvex optimization. Note that CADA's strategy of reducing communication rounds is orthogonal to ours can possibly be combined with ours for further improvements.

3 Problem Formulation

In this paper, we aim to solve the following distributed optimization problem under nonconvex stochastic optimization setting:

$$\min_{x\in\mathbb{R}^d} f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n}\underbrace{\mathbb{E}_{\xi^{(i)}} f_i(\mathbf{x};\xi^{(i)})}_{:=f_i(\mathbf{x})}, \tag{3.1}$$

where d denotes the dimension of the model, $\xi^{(i)}$ denotes the stochastic noise variable, and $f_i(\mathbf{x})$ denotes the loss function on the i-th worker. In the stochastic setting, we cannot obtain the full gradient of loss function $f_i(\mathbf{x})$. Instead, we can only obtain the unbiased estimators of $\nabla f_i(\mathbf{x})$, i.e., $\nabla f_i(\mathbf{x};\xi^{(i)})$.

First, let us revisit the vanilla distributed setting of AMSGrad (Reddi, Kale, and Kumar 2019). Consider a distributed learning system that contains a parameter server and n workers, with each worker i owns its local data from distribution $\mathcal{D}_i$. Let $\mathbf{x}_t$ denotes the current model at t-th iteration. The server will first broadcast $\mathbf{x}_t$ to all the workers in each iteration. Each worker i computes the stochastic gradient $\mathbf{g}_t^{(i)} = \nabla f_i(\mathbf{x}_t;\xi_t^{(i)})$ using the local samples, and then uploads $\mathbf{g}_t^{(i)}$ to the server. The server then aggregates the stochastic gradients and obtains $\mathbf{g}_t = \frac{1}{n}\sum_{i=1}^{n}\mathbf{g}_t^{(i)}$. The model uses $\mathbf{g}_t$ to update parameter via vanilla single node AMSGrad:

$$\mathbf{m}_t = \beta_1\mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t, \mathbf{v}_t = \beta_2\mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2,$$
$$\widehat{\mathbf{v}}_t = \max(\widehat{\mathbf{v}}_{t-1}, \mathbf{v}_t), \mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t\widehat{\mathbf{V}}_t^{-1/2}\mathbf{m}_t,$$

where $\widehat{\mathbf{V}}_t = \mathrm{diag}(\widehat{\mathbf{v}}_t + \nu)$. This vanilla distributed AMSGrad achieves the same convergence rate as its centralized version (Reddi, Kale, and Kumar 2019). However, the worker-to-server and server-to-worker communications in each iteration can be extremely expansive especially for cellular networks, and the communication latency also makes the overall system less efficient. Therefore, attempts have been made to find novel approaches that can further reduce the communication cost while maintaining a similar convergence rate as its uncompressed counterpart. In the following, let's first revisit some traditional communication compression strategies for distributed SGD.

4 Existing Solutions and Drawbacks

Naive compression for SGD: The simplest strategy to reduce communication cost is to directly compress the local gradient $\mathbf{g}_t^{(i)}$ with a compressor $\mathcal{C}(\cdot)$ before sending to the server. The server aggregates the compressed gradient $\widehat{\mathbf{g}}_t^{(i)} = \mathcal{C}(\mathbf{g}_t^{(i)})$ and updates model by

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\alpha}{n}\sum_{i=1}^{n}\widehat{\mathbf{g}}_t^{(i)},$$

where α denotes the learning rate. The common choice of $\mathcal{C}(\cdot)$ can be top-k (Basu et al. 2019) or sign operation (leads to signSGD (Bernstein et al. 2018)). Although this naive compression method is intuitive, it can diverge in practice, even in simple quadratic problems (Beznosikov et al. 2020) or constraint linear problems (Karimireddy et al. 2019). Intuitively speaking, one of the major drawbacks of naive compression is that the compression error is accumulating during the training process. Each step will introduce new errors that cannot be canceled later, and the accumulation of compression error leads the divergence.

Error feedback for SGD: Error feedback (EF), or error compensation (Karimireddy et al. 2019; Stich, Cordonnier, and Jaggi 2018) is widely used for correcting the bias generated by compression errors. Distributed SGD with error feedback effectively reduces the communication bits by introducing a compensating error sequence to cancel the compression error in previous iterations and obtains the same convergence rate as vanilla SGD. Specifically, error feedback introduces a new sequence $\boldsymbol{\delta}_t^{(i)}$, which denotes the accumulated compression error at iteration t. At t-th iteration, the i-th worker computes the compressed gradient $\widehat{\mathbf{g}}_t^{(i)}$ based on the previous iteration's error $\boldsymbol{\delta}_{t-1}^{(i)}$ and the current local gradient $\mathbf{g}_t^{(i)}$, i.e., $\widehat{\mathbf{g}}_t^{(i)} = \mathcal{C}(\mathbf{g}_t^{(i)} + \boldsymbol{\delta}_{t-1}^{(i)})$. And the new compensating error is updated by $\boldsymbol{\delta}_t^{(i)} = \mathbf{g}_t^{(i)} + \boldsymbol{\delta}_{t-1}^{(i)} - \widehat{\mathbf{g}}_t^{(i)}$. Upon compression, the server collects the compressed gradients $\widehat{\mathbf{g}}_t^{(i)}$ from all workers and update model parameters via vanilla SGD. Karimireddy et al. (2019) showed that the compression error of error feedback is bounded by constant if the following assumption hold for the biased compressor.

Assumption 4.1 (Biased compressor)**.** Consider a biased operator $\mathcal{C}:\mathbb{R}^d\to\mathbb{R}^d$, there exists a constant $0<\pi\le 1$ such that

$$\mathbb{E}\big\|\mathcal{C}(\mathbf{x})-\mathbf{x}\big\|_2^2 \le (1-\pi)\|\mathbf{x}\|_2^2, \quad \forall\mathbf{x}\in\mathbb{R}^d. \tag{4.1}$$

Note that $\pi = 1$ leads to $\mathcal{C}(\mathbf{x}) = \mathbf{x}$. Assumption 4.1 is a common assumption for biased compressor (Richtárik,

Sokolov, and Fatkhullin 2021; Karimireddy et al. 2019; Stich, Cordonnier, and Jaggi 2018). Canonical examples of the compressor satisfying Assumption 4.1 include top-k or random-k as well as scaled sign compressor[2]. With Assumption 4.1, the distributed SGD with error feedback (Karimireddy et al. 2019) in nonconvex setting achieves the same convergence rate as vanilla SGD.

Drawbacks of error feedback on adaptive gradient methods: While error feedback guarantees bounded accumulated gradient compression error, in adaptive gradient methods (Kingma and Ba 2014; Reddi, Kale, and Kumar 2019), the variance term $\mathbf{v}_t$, which is the moving average of the quadratic of gradient, can be unstable due to accumulating gradient compression error (Tang et al. 2021). Specifically, let's denote $\mathbf{g}_t$ as the averaged fresh gradient without compression (average of all $\mathbf{g}_t^{(i)}$), denote $\widehat{\mathbf{g}}_t$ as the averaged compressed gradient (average of all $\widehat{\mathbf{g}}_t^{(i)}$). The updating rule for $\mathbf{v}_{t+1}$ follows:

$$\begin{aligned}\mathbf{v}_{t+1} &= \beta_2\mathbf{v}_t + (1-\beta_2)\widehat{\mathbf{g}}_t^2 \\ &= \beta_2\mathbf{v}_t + (1-\beta_2)[\widehat{\mathbf{g}}_t - \mathbf{g}_t + \mathbf{g}_t]^2 \\ &= \beta_2\mathbf{v}_t + (1-\beta_2)\mathbf{g}_t^2 + \underbrace{(1-\beta_2)(\widehat{\mathbf{g}}_t - \mathbf{g}_t)^2}_{\text{accumulating error term}} \\ &\quad + 2(1-\beta_2)\langle \mathbf{g}_t, \widehat{\mathbf{g}}_t - \mathbf{g}_t\rangle. \end{aligned} \tag{4.2}$$

Tang et al. (2021) claims that the inner product term in (4.2) can possibly be canceled out during training, while the quadratic term will certainly accumulate. Since the traditional error feedback can only guarantee constant compression error, the accumulation of the quadratic error after T-steps will make the variance diverge. Therefore, the communication-efficient adaptive gradient method actually requires a stronger gradient compression error bound to obtain a stable variance term.

5 Proposed Method

In this section, we formally develop our proposed **C**ommunication-compressed **D**istributed **Ada**ptive gradient **M**ethod (CD-Adam). Our proposed method consists of two key components: Markov compression sequence and worker-side model update design, which jointly provide a better compression error bound. We first investigate the definition and property of Markov compression sequence (Richtárik, Sokolov, and Fatkhullin 2021).

Markov compression sequence: Markov compression sequence is firstly introduced in Richtárik, Sokolov, and Fatkhullin (2021). Given a biased compressor $\mathcal{C}(\cdot)$ and a sequence of vectors $\{\mathbf{w}_t\}$, Markov compression sequence $\{\widehat{\mathbf{w}}_t\}$ can be recursively defined as: $\widehat{\mathbf{w}}_0 = \mathcal{C}(\mathbf{w}_0)$, $\widehat{\mathbf{w}}_{t+1} = \widehat{\mathbf{w}}_t + \mathcal{C}(\mathbf{w}_{t+1} - \widehat{\mathbf{w}}_t)$. The main advantage of Markov compression sequence lies in that the compression error can be largely reduced if the underlying sequence $\{\mathbf{w}_t\}$ is convergent:

$$\begin{aligned}\big\|\widehat{\mathbf{w}}_{t+1} - \mathbf{w}_{t+1}\big\|^2 &= \big\|\widehat{\mathbf{w}}_t + \mathcal{C}(\mathbf{w}_{t+1} - \widehat{\mathbf{w}}_t) - \mathbf{w}_{t+1}\big\|^2 \\ &\le (1-\pi)\big\|\mathbf{w}_{t+1} - \widehat{\mathbf{w}}_t\big\|^2 \\ &\le (1-\pi)(1+\gamma)\big\|\widehat{\mathbf{w}}_t - \mathbf{w}_t\big\|^2 \\ &\quad + (1-\pi)(1+\gamma^{-1})\big\|\mathbf{w}_{t+1} - \mathbf{w}_t\big\|^2, \end{aligned} \tag{5.1}$$

where the last inequality holds due to Young's inequality. As can be seen from (5.1), the compression error $\big\|\widehat{\mathbf{w}}_{t+1} - \mathbf{w}_{t+1}\big\|$ is directly controlled by $\|\mathbf{w}_{t+1} - \mathbf{w}_t\|$. If the sequence of $\{\mathbf{w}_t\}$ is convergent, Markov compression error at t-step will be much smaller than the constant error bound obtained by plain error feedback. In particular, Richtárik, Sokolov, and Fatkhullin (2021) showed that if the sequence of $\{\mathbf{w}_t\}$ achieves a linear rate of convergence, the compression error of the Markov compression can converge to 0 as $t \to \infty$.

Worker-side model update: We emphasize that Richtárik, Sokolov, and Fatkhullin (2021) only dealt with one-way compression from workers to the server, while in practice, the server needs to send back the updated model $\mathbf{x}_t$ to workers. To further reduce the communication cost of the distributed method and take advantage of the Markov compression sequence, we perform a two-way compression strategy and adjust the model update schedule to the worker-side model update. Specifically, at t-th iteration, each worker locally updates model $\mathbf{x}_{t+1}$ while the server simply aggregates the compressed gradients from the workers, performs another compression and sends them back to the workers. This may sound counter-intuitive as a more common strategy is to update $\mathbf{x}_{t+1}$ on the server and send the update to all workers. For example, the server may compress the model update parameter $\widehat{\mathbf{V}}_t^{-1/2}\mathbf{m}_t$ and send it to all workers for updating. While this plan can also efficiently reduce the communication cost, in such case, the Markov compression sequence cannot provide us a better compression error bound for the model update parameter since it is difficult to obtain the convergence of the underlying sequence $\widehat{\mathbf{V}}_t^{-1/2}\mathbf{m}_t$. On the other hand, in our current worker-side model update design, the compression errors are all about the fresh stochastic gradients, or aggregated gradients, whose convergences are much easier to obtain. This adjustment makes it much easier to establish the theoretical guarantees of our proposed method.

Algorithm overview: Combining Markov compression sequence and worker-side model update design, we propose a new communication-compressed distributive gradient method, which is summarized in Algorithm 1. Specifically, at t-th iteration, each worker first computes the stochastic gradient $\mathbf{g}_t^{(i)}$ with a mini-batch size τ, then builds the Markov compression sequence $\widehat{\mathbf{g}}_t^{(i)}$ with $\mathbf{c}_t^{(i)} = \mathcal{C}(\mathbf{g}_t^{(i)} - \widehat{\mathbf{g}}_{t-1}^{(i)})$ and $\widehat{\mathbf{g}}_{t-1}^{(i)}$. Note that the Markov compression sequence $\widehat{\mathbf{g}}_t^{(i)}$ is no longer bits-compressed, therefore, only $\mathbf{c}_t^{(i)}$ is being uploaded to the server. On the other hand, the server updates the aggregated compressed gradient by

[2]See the supplemental materials for more details about the above three compressors as well as other compressors satisfying Assumption 4.1.

$\widehat{\mathbf{g}}_t = \widehat{\mathbf{g}}_{t-1} + \frac{1}{n}\sum_{i=1}^n \mathbf{c}_t^{(i)}$ as the last iterate $\widehat{\mathbf{g}}_{t-1}$ has been stored locally in the server. The server then builds another Markov compression sequence $\widetilde{\mathbf{g}}_t$ based on $\widetilde{\mathbf{g}}_{t-1}$ and $\mathbf{c}_t = \mathcal{C}(\widehat{\mathbf{g}}_t - \widetilde{\mathbf{g}}_{t-1})$ and sends $\mathbf{c}_t$ to all workers. Similarly, each worker can recover the Markov compression sequence $\widetilde{\mathbf{g}}_t$ upon receiving $\mathbf{c}_t$, since the last iterate $\widetilde{\mathbf{g}}_{t-1}$ has been locally stored in each worker. Then each worker uses the double-compressed gradient $\widetilde{\mathbf{g}}_t$ for updating model $\mathbf{x}_{t+1}$ by following standard AMSGrad.

Algorithm 1 is indeed communication-efficient for distributed learning. For both server-to-worker and worker-to-server communications, each worker and the server maintain the previous step compressed gradients; only the compressed vectors are transferred instead of full precision vectors. Specifically, for the scaled sign compressor, CD-Adam only takes 1-bit[3] instead of 32-bits per dimension in each communication round. This greatly reduces the communication costs for distributed implementation of adaptive methods. Note that compared with 1-bit Adam (Tang et al. 2021), which performs a few epochs of uncompressed Adam at the beginning of training, our proposed CD-Adam method is much more communication-efficient (see Figure 1) as we start the compression from the very first iteration.

Algorithm 1: **C**ommunication-Compressed **D**istributed **Ada**ptive Gradient **M**ethod (CD-Adam)

Input: initial point $\mathbf{x}_1$, step size $\{\alpha_t\}_{t=1}^T$, β_1, β_2, ν, batch size τ. Sacled sign compressor $\mathcal{C}(\mathbf{x}) = \|\mathbf{x}\|_1 \cdot \mathrm{sign}(\mathbf{x})/d$.

$\mathbf{g}_0^{(i)} \leftarrow 0, \mathbf{m}_0 \leftarrow 0$, $\mathbf{v}_0 \leftarrow 0, \widehat{\mathbf{g}}_0^{(i)} = \mathcal{C}(\mathbf{g}_0^{(i)}), \widetilde{\mathbf{g}}_0 = \mathcal{C}(\frac{1}{n}\sum_{i=1}^n \widehat{\mathbf{g}}_0^{(i)})$
for $t = 1$ to T **do**
  (On i-th worker)
  Compute local stochastic gradient with batch size τ: $\mathbf{g}_t^{(i)} = \frac{1}{\tau}\sum_{j=1}^{\tau} \nabla f_i(\mathbf{x}_t; \xi_t^{(i,j)})$
  Compress $\mathbf{c}_t^{(i)} = \mathcal{C}(\mathbf{g}_t^{(i)} - \widehat{\mathbf{g}}_{t-1}^{(i)})$
  Send $\mathbf{c}_t^{(i)}$ to the server and update local state $\widehat{\mathbf{g}}_t^{(i)} = \widehat{\mathbf{g}}_{t-1}^{(i)} + \mathbf{c}_t^{(i)}$
  (On Server)
  Update $\widehat{\mathbf{g}}_t = \widehat{\mathbf{g}}_{t-1} + \frac{1}{n}\sum_{i=1}^n \mathbf{c}_t^{(i)}$
  Compress $\mathbf{c}_t = \mathcal{C}(\widehat{\mathbf{g}}_t - \widetilde{\mathbf{g}}_{t-1})$
  Send $\mathbf{c}_t$ to all the workers and update local state $\widetilde{\mathbf{g}}_t$: $\widetilde{\mathbf{g}}_t = \widetilde{\mathbf{g}}_{t-1} + \mathbf{c}_t$
  (On i-th worker)
  Update $\widetilde{\mathbf{g}}_t = \widetilde{\mathbf{g}}_{t-1} + \mathbf{c}_t$
  $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\widetilde{\mathbf{g}}_t$
  $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\widetilde{\mathbf{g}}_t^2$
  $\widehat{\mathbf{v}}_t = \max(\widehat{\mathbf{v}}_{t-1}, \mathbf{v}_t)$
  Update $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t \widehat{\mathbf{V}}_t^{-1/2}\mathbf{m}_t$ with $\widehat{\mathbf{V}}_t = \mathrm{diag}(\widehat{\mathbf{v}}_t + \nu)$
end for

[3]Rigorously, the scaling number will also take 32-bits for communication, so the overall cost for compressing a d-dimensional vector should be $32 + d$ bits.

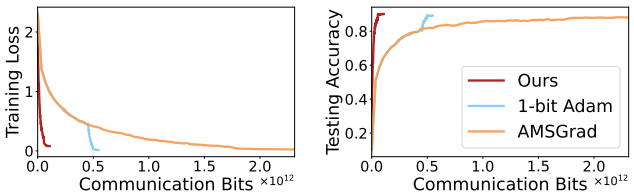

Figure 1: Comparison of training loss and testing accuracy against communication bits of training ResNet-18 on CIFAR-10. Our proposed method achieves around $32\times$ communication cost improvement over the original AMSGrad and around $5\times$ over 1-bit Adam.

6 Convergence Analysis

In this section, we present the convergence results of our proposed Communication-Compressed Distributed Adaptive Gradient Method (CD-Adam) in Algorithm 1. Before we jump into the main theorem, let us first introduce more assumptions needed for the proof.

Assumption 6.1 (Smoothness)**.** Each component loss function on the i-th worker $f_i(\mathbf{x}) = \mathbb{E}[f(\mathbf{x}; \xi^{(i)})]$ is L-smooth, i.e., $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$\big|f_i(\mathbf{x}) - f_i(\mathbf{y}) - \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle\big| \le \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|_2^2.$$

Assumption 6.1 is a standard assumption for nonconvex stochastic optimization, which has been also adopted in (Koloskova et al. 2019; Basu et al. 2019; Richtárik, Sokolov, and Fatkhullin 2021). Note that the L-smoothness assumption on each worker's loss $f_i(\cdot)$ implies the L-smoothness condition on $f(\cdot)$. Assumption 6.1 also implies the L-gradient Lipschitz condition, i.e., $\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\|_2 \le L\|\boldsymbol{x} - \boldsymbol{y}\|_2$.

Assumption 6.2 (Bounded gradient)**.** Each component loss function on the i-th worker $f_i(\mathbf{x})$ has G-bounded stochastic gradient on ℓ_2 and has G_∞-bounded stochastic gradient on ℓ_∞, i.e., for all ξ,

$$\|\nabla f_i(\mathbf{x}; \xi)\|_2 \le G, \qquad \|\nabla f_i(\mathbf{x}; \xi)\|_\infty \le G_\infty.$$

Assumption 6.2 also implies the bounded gradient for $f(\mathbf{x})$, i.e., $\|\nabla f(\mathbf{x}; \xi)\|_2 \le G$, $\|\nabla f(\mathbf{x}; \xi)\|_\infty \le G_\infty$. The bounded gradient assumption has also been used in Kingma and Ba (2014); Chen et al. (2019); Zhou et al. (2018); Chen et al. (2018).

Assumption 6.3 (Bounded variance)**.** Each stochastic gradient on the i-th worker $\mathbf{g}^{(i,j)} = f_i(\mathbf{x}, \xi^{(i,j)})$ has a bounded variance, i.e., for all $\mathbf{x}$ and all i, j,

$$\mathbb{E}_{\xi^{(i,j)} \sim \mathcal{D}_i}\big\|\nabla f_i(\mathbf{x}, \xi^{(i,j)}) - \nabla f_i(\mathbf{x})\big\|^2 \le \sigma^2.$$

Assumption 6.3 implies that the variance of stochastic gradients is bounded on each worker. The assumption of bounded variance for stochastic gradient has also been used in (Basu et al. 2019; Koloskova et al. 2019; Tang et al. 2021).

Now we are ready to present our main theorem.

Theorem 6.4. Under Assumptions 6.1, 6.2 and 6.3, suppose $\beta_1 < \beta_2^{1/2}$, then Algorithm 1 achieves an ϵ-stationary point of (3.1), i.e., $\min_{t\in[T]} \mathbb{E}\big[\|\nabla f(\mathbf{x}_t)\|_2^2\big] \le \epsilon$, with

$$\alpha_t = \alpha = \frac{\epsilon}{3M_3}, \tau = \left\lceil \max\left\{1, \left(\frac{3M_4\sigma}{\epsilon}\right)^2\right\}\right\rceil,$$
$$T = \left\lceil \frac{9M_1M_3}{\epsilon^2} + \frac{3M_2}{\epsilon}\right\rceil,$$

where

$$M_1 = (\widetilde{G}_\infty^2 + \nu)^{1/2}\Delta f,$$
$$M_2 = \frac{1}{1-\beta_1}(\widetilde{G}_\infty^2 + \nu)^{1/2} G\widetilde{G}\nu^{-1/2},$$
$$M_3 = \frac{L(\widetilde{G}_\infty^2 + \nu)^{1/2}\widetilde{G}_\infty d}{\nu^{1/2}(1-\beta_2)^{1/2}(1-\beta_1/\beta_2^{1/2})}\Big(1 + \frac{3\beta_1^2}{2(1-\beta_1)}\Big) + \frac{2\theta L(\widetilde{G}_\infty^2 + \nu)^{1/2} G\widetilde{G}\nu^{-1}\sqrt{d}}{(1-\theta)^2},$$
$$M_4 = \frac{4\theta}{(1-\theta)^2}(\widetilde{G}_\infty^2 + \nu)^{1/2} G\nu^{-1/2},$$
$$\widetilde{G}_\infty = \frac{2-\pi+2\sqrt{1-\pi}}{2-\pi-2\sqrt{1-\pi}} G_\infty, \theta = \sqrt{1-\pi}$$
$$\Delta f = f(\mathbf{x}_1) - \inf_x f(\mathbf{x}).$$

Remark 6.5. Theorem 6.4 suggests that Algorithm 1 reaches the ϵ-stationary point of (3.1) with $O(1/\epsilon^2)$ iterations, which matches the iteration complexity of uncompressed vanilla AMSGrad. This suggests that our proposed communication-compressed distributed AMSGrad algorithm is indeed provably efficient, and the fully compressed distributed adaptive gradient method can reach ϵ-stationary point with the same iteration complexity as the uncompressed counterpart without applying any variance-freezing tricks.

7 Experiments

In this section, we present empirical results of our proposed communication-compressed distributed adaptive gradient method and compare it with other communication-compressed distributed baselines. Specifically, we first present the experimental results on a nonconvex logistic regression problem as an illustrative case study. Then we present the deep learning experiments of image classification on standard benchmarks.

7.1 Illustrative case study: nonconvex logistic regression

We consider minimizing the following logistic regression problem with a nonconvex regularizer similar to Richtárik, Sokolov, and Fatkhullin (2021),

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^n \log\Big(1 + \exp\big(-y_i a_i^\top x\big)\Big) + \lambda\sum_{j=1}^d \frac{x_j^2}{1+x_j^2}, \tag{7.1}$$

where $a_i \in \mathbb{R}^d$, y_i are training data with value ± 1, and λ is the regularizer parameter. We set $\lambda = 0.1$ in the following nonconvex logistic regression experiments.

We use datasets `phishing`, `mushrooms`, `a9a` and `w8a` from LibSVM (Chang and Lin 2011). We equally separate each dataset to $n = 20$ parts and each worker keeps the corresponding subset. For each method, we choose the best stepsize starting from 0.001 and increase it by adding 0.002 till achieving 0.01.

Let's first study the effect of using different compression strategies on AMSGrad, though the other strategies do not enjoy any theoretical guarantees. Figure 2 compares the gradient norm convergence of our proposed CD-Adam with AMSGrad using error feedback, naive compression and without compression at all. For all compressed methods, we use the scaled sign as a canonical example of biased compressor[4] $\mathcal{C}(\cdot)$. We observe that our proposed CD-Adam achieves the best performances on all four datasets against the other three compression strategies. Specifically, if we compare the communication cost, CD-Adam achieves much better communication efficiency against uncompressed AMSGrad. Compared with the error feedback and the naive compression, CD-Adam still achieves a much smaller gradient norm under the same communication budgets. If we compare the performance of different compression strategies under the same iteration complexity, our proposed algorithm achieves roughly the same final gradient norm as the uncompressed AMSGrad across all four datasets. In fact, it achieves a much better convergence result comparing with the error feedback and the naive compression strategies, whose gradient norm stops decreasing at the early stages.

7.2 Deep learning experiments on image classification

We compare CD-Adam with several state-of-the-art communication-compressed distributed learning algorithms that are provably efficient, including: (1) EF21 (Richtárik, Sokolov, and Fatkhullin 2021) (2) 1-bit Adam (Tang et al. 2021). Note that the original EF21 paper (Richtárik, Sokolov, and Fatkhullin 2021) adopts the top-K compressor as the base compressor $\mathcal{C}(\cdot)$ with only worker-to-server compression applied. For a fair comparison, we further extend EF21 to allow both worker-to-server and server-to-worker compression and choose K$= 0.016d$ such that the communication compression ratio keeps roughly the same as the scaled-sign based compressor.

We train `CIFAR10` (Krizhevsky, Hinton et al. 2009) using three popular models: `ResNet-18` (He et al. 2016), `VGG-16` (Simonyan and Zisserman 2014) and `WideResNet-16-4` (Zagoruyko and Komodakis 2016). The image classification dataset `CIFAR10` includes a training set of 50000 images and a test set of 10000 images, where each image is assigned one of the 10 labels. The dataset is split into $n = 8$ equal parts, which are distributed to 8 workers. The mini-batch size for each worker is set to

[4] Additional experiments based on the Top-k biased compressor can be found in the supplemental materials.

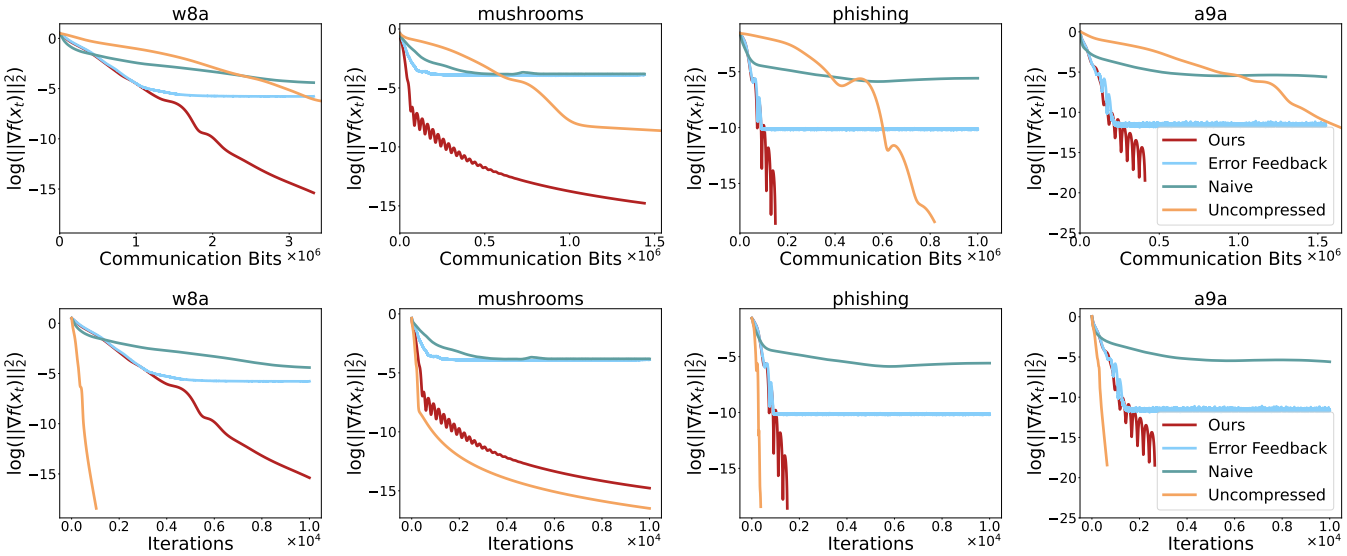

Figure 2: Gradient norm comparison of different compressing strategies on nonconvex logistic regression trained by AMSGrad with scaled sign compressor. The upper row shows the norm convergence with respect to the communication cost, and the lower row is with respect to the training iteration.

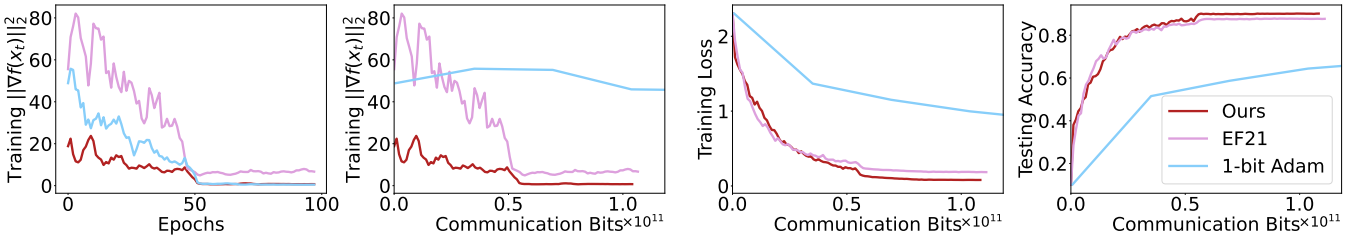

Figure 3: Comparison on gradient norm, training loss and testing accuracy among our proposed method and the baselines when training `ResNet-18` model on `CIFAR-10`.

be 128. We set the learning rate as 1×10^{-1} for SGD, which is used in EF21, and 1×10^{-4} for 1-bit Adam and our proposed CD-Adam. We also set $\beta_1 = 0.9$ and $\beta_2 = 0.99$ for 1-bit Adam and our CD-Adam. For 1-bit Adam, its warm-up epochs are set as 13 according to its original paper (Tang et al. 2021). The normalized weight decay is set to 5×10^{-4} for all methods. We test all methods for 100 epochs and decay the learning rate by 0.1 at the 50th and 75th epoch. Due to the space limit, we leave the experimental results on `VGG-16` and `WideResNet-16-4` models in the supplemental materials.

We show that our proposed algorithm enjoys a fast convergence speed with high accuracy and less communication cost comparing with EF21 and 1-bit Adam. The first and second plots in Figure 3 demonstrate the training gradient norm in terms of epochs and communication bits respectively. We can observe that our proposed CD-Adam obtains a smaller gradient norm than EF21 and 1-bit Adam under the same epoch or communication budget. Note that since 1-bit Adam will need to run uncompressed Adam for a few epochs as a warm-up, its per communication bits performance is actually much worse (has been shown in Figure 1). The third plot in Figure 3 shows the training loss against communication bits on `CIFAR-10` dataset. At the early stage of the training process, CD-Adam and EF21 obtain a similar speed of reducing the training loss, while 1-bit Adam is much slower due to the warm-up process. At later stages, CD-Adam shows a clear advantage compared with EF21. The last plot in Figure 3 shows the test accuracy against communication bits. Similar phenomenon can be observed that CD-Adam achieves the overall best test accuracy compared with EF21 and 1-bit Adam, under the same communication budget. These results suggest that our proposed CD-Adam is indeed much more communication-efficient while maintaining a high accuracy compared with other communication compression baselines.

8 Conclusion

In this paper, we propose a communication-compressed distributed adaptive gradient method which solves the bottleneck of applying communication compression strategies for adaptive gradient methods in distributed settings. We provide theoretical convergence analysis in the noncon-

vex stochastic optimization setting and show that our proposed algorithm converges to an ϵ-stationary point with the same iteration complexity as the uncompressed vanilla AMSGrad. Furthermore, compared with prior work which adopts variance-freezed Adam, our proposed algorithm is fully adaptive during the entire training process. It does not require any warm-up procedure with uncompressed communication, leading to better empirical results in terms of both performance and communication cost.

References

Alacaoglu, A.; Malitsky, Y.; Mertikopoulos, P.; and Cevher, V. 2020. A new regret analysis for Adam-type algorithms. In *International Conference on Machine Learning*, 202–210. PMLR.

Alistarh, D.; Grubic, D.; Li, J.; Tomioka, R.; and Vojnovic, M. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 30: 1709–1720.

Basu, D.; Data, D.; Karakus, C.; and Diggavi, S. 2019. Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations. *arXiv preprint arXiv:1906.02367*.

Bernstein, J.; Wang, Y.-X.; Azizzadenesheli, K.; and Anandkumar, A. 2018. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, 560–569. PMLR.

Beznosikov, A.; Horváth, S.; Richtárik, P.; and Safaryan, M. 2020. On biased compression for distributed learning. *arXiv preprint arXiv:2002.12410*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3).

Chen, C.; Shen, L.; Huang, H.; and Liu, W. 2021a. Quantized Adam with Error Feedback. *ACM Trans. Intell. Syst. Technol.*, 12(5).

Chen, J.; Zhou, D.; Tang, Y.; Yang, Z.; Cao, Y.; and Gu, Q. 2020. Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks. In *International Joint Conferences on Artificial Intelligence*.

Chen, T.; Guo, Z.; Sun, Y.; and Yin, W. 2021b. Cada: Communication-adaptive distributed adam. In *International Conference on Artificial Intelligence and Statistics*, 613–621. PMLR.

Chen, X.; Liu, S.; Sun, R.; and Hong, M. 2018. On the convergence of a class of adam-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*.

Chen, X.; Liu, S.; Sun, R.; and Hong, M. 2019. On the Convergence of A Class of Adam-Type Algorithms for Non-Convex Optimization. In *International Conference on Learning Representations*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Duchi, J. C.; Hazan, E.; and Singer, Y. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In *J. Mach. Learn. Res.*

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

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.

Hinton, G.; Deng, L.; Yu, D.; Dahl, G.; Mohamed, A.-r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Kingsbury, B.; et al. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29.

Ivkin, N.; Rothchild, D.; Ullah, E.; Braverman, V.; Stoica, I.; and Arora, R. 2019. Communication-efficient distributed SGD with sketching. *arXiv preprint arXiv:1903.04488*.

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. PMLR.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koloskova, A.; Lin, T.; Stich, S. U.; and Jaggi, M. 2019. Decentralized deep learning with arbitrary communication compression. *arXiv preprint arXiv:1907.09356*.

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

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Luo, L.; Xiong, Y.; Liu, Y.; and Sun, X. 2019. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*.

Reddi, S. J.; Kale, S.; and Kumar, S. 2019. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

Richtárik, P.; Sokolov, I.; and Fatkhullin, I. 2021. EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback. *arXiv preprint arXiv:2106.05203*.

Robbins, H.; and Monro, S. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3): 400 – 407.

Seide, F.; Fu, H.; Droppo, J.; Li, G.; and Yu, D. 2014. 1-Bit Stochastic Gradient Descent and Application to Data-Parallel Distributed Training of Speech DNNs. In *Interspeech 2014*.

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

Stich, S. U.; Cordonnier, J.-B.; and Jaggi, M. 2018. Sparsified SGD with Memory. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*, 4447–4458. Curran Associates, Inc.

Stich, S. U.; and Karimireddy, S. P. 2019. The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*.

Tang, H.; Gan, S.; Awan, A. A.; Rajbhandari, S.; Li, C.; Lian, X.; Liu, J.; Zhang, C.; and He, Y. 2021. 1-bit Adam: Communication Efficient Large-Scale Training with Adam's Convergence Speed. *arXiv preprint arXiv:2102.02888*.

Tang, H.; Yu, C.; Lian, X.; Zhang, T.; and Liu, J. 2019. `DoubleSqueeze`: Parallel Stochastic Gradient Descent with Double-pass Error-Compensated Compression. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 6155–6165. Long Beach, California, USA: PMLR.

Tieleman, T.; Hinton, G.; et al. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2): 26–31.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zaheer, M.; Reddi, S.; Sachan, D.; Kale, S.; and Kumar, S. 2018. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, 9793–9803.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zheng, S.; Meng, Q.; Wang, T.; Chen, W.; Yu, N.; Ma, Z.-M.; and Liu, T.-Y. 2017. Asynchronous stochastic gradient descent with delay compensation. In *International Conference on Machine Learning*, 4120–4129. PMLR.

Zhou, D.; Chen, J.; Cao, Y.; Tang, Y.; Yang, Z.; and Gu, Q. 2018. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*.