EFFICIENT-ADAM: COMMUNICATION-EFFICIENT DISTRIBUTED ADAM WITH COMPLEXITY ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Distributed adaptive stochastic gradient methods have been widely used for large scale nonconvex optimization, such as training deep learning models. However, their iteration complexity on finding ε-stationary points has rarely been analyzed in the nonconvex setting. In this work, we present a novel communication-efficient distributed Adam in the parameter-server model for stochastic nonconvex optimization, dubbed *Efficient-Adam*. Specifically, we incorporate a two-way quantization scheme into Efficient-Adam to reduce the communication cost between the workers and the server. Simultaneously, we adopt a two-way error feedback strategy to reduce the biases caused by the two-way quantization on both the server and workers, respectively. In addition, we establish the convergence rate for Efficient-Adam with a class of quantization operators and further characterize its communication complexity between the server and workers when a ε-stationary point is achieved. Finally, we solve a toy stochastic convex optimization problem and train deep learning models on real-world vision tasks. Extensive experiments together with a theoretical guarantee justify the merits of Efficient Adam.

1 INTRODUCTION

Let $\mathbb{X}$ be a finite-dimensional linear vector space. We focus on a stochastic optimization problem:

$$\min_{x\in\mathbb{X}} F(x) = \mathbb{E}_{\xi\sim\mathbb{P}}[f(x,\xi)], \tag{1}$$

where $F\colon \mathbb{X}\to\mathbb{R}$ is a proper, lower semi-continuous smooth function that could be nonconvex, and ξ is a random variable with an unknown distribution $\mathbb{P}$. Stochastic optimizations commonly appear in the fields of statistical machine learning (Bishop, 2006), deep learning (Goodfellow et al., 2016), and reinforcement learning (Sutton & Barto, 2018), which include sparse learning (Hastie et al., 2015), representation learning (Bengio et al., 2013), classification (Deng et al., 2009), regression, etc.

Due to the existence of expectation over an unknown distribution, the population risk $\mathbb{E}_{\xi\sim\mathbb{P}}[f(x,\xi)]$ with a complicated function F is approximated via an empirical risk function $\frac{1}{n}\sum_{i=1}^{n}[f_i(x,\xi_i)]$ via a large number of samples $\{\xi_i\}$. For example, we will use the ImangeNet dataset for the image classification task (Deng et al., 2009), the COCO dataset for the object detection task (Lin et al., 2014), and the GLUE dataset for the natural language understanding task (Wang et al., 2018), respectively. Meanwhile, to learn appropriate distributions for different tasks, it is hard to design the exact mathematical formulation of f. Thus, an effective way is to set a complicated deep neural network as a surrogate function with a large number of parameters/FLOPs, making problem equation 1 an ultra-large-scale nonconvex stochastic optimization.

However, it could be impossible to train a deep neural network with a large number of parameters over a large-scale dataset within a single machine. Fortunately, we have some promising approaches to tackle this problem. One of them is to extend the stochastic gradient descent (SGD) method to one distributed version (Li et al., 2014b). Then using the distributed SGD method, we train a deep learning model with multiple machines in a distributed mode (Goyal et al., 2017; You et al., 2019). However, for the vanilla SGD method, how to tune a suitable learning rate for different tasks remains challenging. This dilemma is more serious for distributed SGD methods since there are multiple learning rates needed to be tuned for multiple machines. Moreover, the communication overhead is another issue in distributed methods. How to reduce the communication cost among

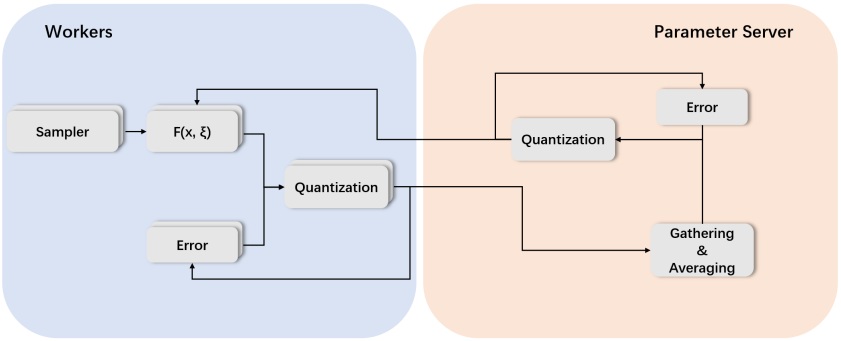

Figure 1: The workflow of Efficient-Adam in the Parameter-Server model. For each worker, parameters are updated via Adam. For the server, information is gathered and averaged to update parameters. There exists a two-way quantization scheme to reduce the communication cost between each worker and server. Moreover, a two-way error feedback strategy compensates for quantization errors for both the workers and server.

multiple machines is also challenging. Recently, Hou et al. (2019) proposed a distributed Adam (Kingma & Ba, 2014) with weights and gradients being quantized to reduce the communication cost between the workers and the server. However, their theoretical analysis is merely restricted to the convex setting, and they didn't provide the bit-communication complexity either, which hampers the potential applications. In addition, both the weights and gradients quantization techniques will introduce additional errors, which may degrade the performance of the vanilla Adam optimizer. On the other hand, distributed Adam has already been built in several deep learning platforms, such as PyTorch (Paszke et al., 2019), TensorFlow (Abadi et al., 2016), and MXNet (Chen et al., 2015), and it has since been broadly used for training deep learning models. However, its iteration complexity and communication complexity under the distributed mode have rarely been analyzed in both convex and nonconvex settings.

In this work, we propose a communication-efficient distributed adaptive stochastic gradient descent method by incorporating a two-way quantization of updates and a two-way error feedback strategy, dubbed Efficient-Adam, to solve the stochastic problem equation 1. As it is illustrated in Figure 1, the architecture of Efficient-Adam belongs to a Parameter-Server (Smola & Narayanamurthy, 2010; Li et al., 2014a) distributed system. For each worker, in each iteration, we first sample a stochastic gradient of problem equation 1. Then, we calculate the updates with Adam optimizer. Next, we quantize the updates based on a specifically designed quantization mapping and an error term and send the quantized updates to the server. After that, we update local error terms with the updates, quantization mapping, and error term. At last, we receive the averaged updates from the server and update local iterates. On the other hand, in the server, we first gather and average the compressed updates from each worker. Then, we quantize the average updates and broadcast them to each worker. Next, we update iterate and error feedback terms. From the workflow of Efficient-Adam in Figure 1, it can be seen that communicated bi-directional information is all quantized in advance. With the proper quantization mapping, communication costs can be reduced largely. In addition, error terms in workers and the server can compensate for errors that are introduced by the two-way quantization steps between all workers and the server, which help accelerate the convergence of Efficient-Adam for problem equation 1.

Moreover, we explore the iteration complexity for Efficient-Adam with a class of quantization operators when an ε-stationary point is achieved. For a carefully designed quantization mapping, we further characterize its overall communication complexity in terms of bits between the server and workers. On the other hand, when the quantization mapping is generalized to a compressor as (Stich & Karimireddy, 2019; Stich et al., 2018; Zheng et al., 2019), Efficient-Adam can enjoy the same convergence rate as full-precision Adam in (Zou et al., 2019). Experimentally, we apply Efficient-Adam to train deep learning models on computer vision and natural language processing tasks to demonstrate its efficacy. To the end, we summarize our contribution in four-fold:

- We propose a communication-efficient distributed Adam to the solve stochastic nonconvex optimization problem equation 1. We dub it *Efficient-Adam* which utilizes a two-way

quantization scheme to reduce the communication cost and a two-way error feedback strategy to compensate for quantization errors.

- We characterize the iteration complexity of Efficient-Adam in the non-convex setting. Under proper assumptions, we further characterize its overall communication complexity in terms of bits between the server and workers via a specifically designed quantization mapping.
- We show that the convergence rate of Efficient-Adam can further be improved to the same order of vanilla Adam, i.e., $\mathcal{O}(1/\sqrt{T})$, once we generalize the quantization mapping in Efficient-Adam to a compressor mapping as (Stich & Karimireddy, 2019; Stich et al., 2018).
- We conduct several experiments on computer vision tasks to demonstrate the efficacy of the proposed Efficient-Adam, as well as the related two-way quantization and error feedback techniques.

2 Related Work

Optimizing a large-scale stochastic non-convex function has been studied for many years, in which the distributed SGD method has been widely explored. For distributed SGD, its convergence rate has been established on both the Parameter-Server model (Agarwal & Duchi, 2011) and decentralized model (Lian et al., 2017). For the variant method of SGD, distributed SGD with momentum (Liu et al., 2020), and distributed Adam (Reddi et al., 2020) are proposed and establish their convergence results. To reduce the communication cost, compression on the communication has been added into the distributed stochastic methods, such as QSGD (Alistarh et al., 2017) and Sign SGD (Bernstein et al., 2018). However, due to the error introduced by compression, most of the distributed methods will fail to converge with compressed communication. Several works (Jiang & Agrawal, 2018; Wangni et al., 2018; Wen et al., 2017) have tried to use unbiased compressors to remove errors brought by compression and provide related convergence analyses. However, with some biased compressors (e.g. SignSGD), the SGD method empirically achieves a good solution, which cannot be explained by those works. To deal with biased compression, Karimireddy et al. (2019) firstly introduced error feedback into the SignSGD method, and established its convergence. However, their analysis only focuses on the setting in a single machine. In addition, there exist several decentralized SGDs that try to utilize error feedback to reduce compression errors, such as ChocoSGD (Koloskova et al., 2019) and DeepSqueeze (Tang et al., 2019).

Further, for the parameter-server model, Zheng et al. (2019) introduced a two-way error feedback technique into SGD with momentum and showed its convergence in the nonconvex setting. Moreover, they use a block-wise compressor to meet the compressor assumption. However, even with the block-wise compression, their assumption still does not hold when the gradient goes to 0. Besides, the learning rate for each worker may be hard to tune when the number of works is large. On the other hand, Hou et al. (2019) adapted the distributed Adam with quantized gradients and weights to reduce the communication overhead, and gave the convergence bound for the convex case. However, in their work, they considered an unbiased quantization function and did not consider error feedback terms, which may limit the use of the analysis.

Different from these existing works, we propose an adaptive distributed stochastic algorithm with a two-way quantization/compressor and a two-way error feedback strategy, dubbed Efficient Adam. We also establish its iteration complexity and communication complexity in terms of bits under some specialized designed quantization mappings in the non-convex setting. In the following Table 1, we summarize the differences between our proposed Efficient-Adam and several existing communication-efficient distributed SGD methods in the parameter-server model.

3 Preliminaries

For presenting and analyzing the proposed Efficient-Adam in the following sections, we first give several basic definitions and assumptions. First, we denote the stochastic estimation of $F(x) = \mathbb{E}_{\xi}[f(x,\xi)]$ as $g = \nabla_x f(x,\xi)$ with a given sample ξ. In addition, we summarize a few necessary assumptions on function F and stochastic estimate g, and define ε-stationary point of problem equation 1.

Table 1: Comparison against existing methods. Notably, Zheng et al. (2019) merely supported compressor mapping while Efficient-Adam can support both the compressor and quantization mappings.

Method	non-convex	adaptive lr	compression	error feedback	momentum
Distributed SGD	$\surd$	$\times$	$\times$	$\times$	$\times$
Alistarh et al. (2017)	$\surd$	$\times$	$\surd$	$\times$	$\times$
Karimireddy et al. (2019)	$\surd$	$\times$	one side	$\surd$	$\times$
Hou et al. (2019)	$\times$	$\surd$	$\surd$	$\times$	$\times$
Zheng et al. (2019)	$\surd$	$\times$	$\surd$	$\surd$	$\surd$
Efficient-Adam	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Assumption 1. *Function $F(x)$ is assumed to be lower bounded, i.e., $F(x) \geq F^* > -\infty$ for some give constant F^*. Gradient ∇F is assumed to be L-Lipschitz continuous, i.e., $\|\nabla F(x) - \nabla F(y)\| \leq L\|x-y\|$. Besides, we assume that for given x, g is an unbiased estimator of gradient with bounded second moment, i.e., $\mathbb{E}_\xi(g) = \nabla F(x) = \nabla_x \mathbb{E}_\xi(f(x,\xi))$ and $\|g\| \leq G$.*

Definition 1. *We denote $x^* \in \mathbb{X}$ as ε-stationary point of problem equation 1, if it satisfies $\mathbb{E}\|\nabla F(x^*)\|^2 \leq \varepsilon$.*

Assumption 1 is widely used for establishing convergence rates of adaptive SGD methods towards ε-stationary points, such as Adam/RMSProp (Zou et al., 2019), AdaGrad (Ward et al., 2018; Zou et al., 2018; Défossez et al., 2020), and AMSGrad (Reddi et al., 2019; Chen et al., 2018).

Below, we define a class of quantization mappings that are used to quantize the communicated information between the workers and server in Figure 1 to reduce the communication cost.

Definition 2. *We call $\mathcal{Q} : \mathbb{X} \to \mathbb{X}$ a quantization mapping, if there exist constants $\delta > 0$ and $\delta' \geq 0$ such that the two inequalities hold: $\|x - \mathcal{Q}(x)\| \leq (1-\delta)\|x\| + \delta'$ and $\|\mathcal{Q}(x)\| \leq \|x\|$.*

We give a few quantization mappings that satisfy the above definition. In addition, we give the bit analysis that the quantization function needs in one communication round.

Example 1. *Let $M = \{0, \frac{1}{B}, \cdots, \frac{GB-1}{B}, \frac{GB}{B}\}$ and $Q(x_i) = sign(x_i) \max_{y_i \in M,\ y_i \leq |x_i|}\{y_i\}$. Given $x \in \mathbb{X}$, we define $\mathcal{Q}(x)_i = Q(x_i)$. Besides, to transmit $\mathcal{Q}(x)$, $d(\log(GB+1)+1)$ bits are needed.*

Example 2. *Let $M_1 = \big\{0, 2^{-K}, 2^{-K+1}, \cdots, 2^0, 2^1, \cdots, 2^{\lfloor \log(G) \rfloor}\big\}$, $M_2 = \big\{0, \frac{1}{2^m-1}, \frac{2}{2^m-1}, \cdots, \frac{2^m-2}{2^m-1}, 1\big\}$, $Q_1(x_i) = \max_{y_i \in M_1, y_i \leq |x_i|}\{y_i\}$, and $Q_2(x_i) = \max_{y_i \in M_2, y_i \leq |x_i|} y_i$. Given $x \in \mathbb{X}$, we define $\mathcal{Q}(x)_i = sgn(x_i) Q_1(\|x\|_\infty) Q_2(x_i/\|x\|_\infty)$. Besides,to transmit $\mathcal{Q}(\mathbf{x})$, $\big(\log(\log(G)+K+1)+dm\big)$ bits are needed.*

Example 3. *Let $M = \{0, 1, 2, \cdots, 2^m - 1\}$, $E = \{-K, -K+1, \cdots, -1, 0, 1, 2, \cdots, \log(G)\}$, and $e(x) = max(\lfloor \log(x) \rfloor, -K)$. Given $x \in \mathbb{X}$, we define*

$$Q(x_i) = sgn(x_i) \times 2^{e(x_i)} \times \begin{cases} \max_{y_i \in M, y_i * 2^{e(x_i)-m} \leq |x_i|}\{2^{-m} y_i\}, & \text{if } |x_i| < 2^{e(x_i)} \\ \max_{y_i \in M, y_i * 2^{e(x_i)-m} \leq |x_i| - 2^{e(x_i)}}\{2^{-m} y_i + 1\}, & \text{otherwise.} \end{cases}$$

We define $\mathcal{Q}(x)_i = Q(x_i)$. Besides, to transmit $Q(x)$, $d(m + \log(\log(G)+K+1))$ bits are needed.

Proposition 1. *All of the quantization mappings defined via Examples 1-3 satisfy Definition 2.*

There exist several works that utilize compressor mappings (Stich et al., 2018; Stich & Karimireddy, 2019) to reduce the communication cost for distributed SGD in the Parameter-Server model. Formally, compressor mapping is defined as follows.

Definition 3. *We call $\mathcal{Q} : \mathbb{X} \to \mathbb{X}$ as Compressor mapping (Stich et al., 2018; Stich & Karimireddy, 2019; Zheng et al., 2019), if there exists a positive value δ such that the inequality holds: $\|x - \mathcal{Q}(x)\| \leq (1-\delta)\|x\|$.*

Remark 1. *Most of the quantization mappings do not belong to the class of compressors, since, with a finite set as range, quantization mapping cannot achieve the condition of compressors for arbitrarily small parameter x. Therefore, we cannot find any quantization function that satisfies the definition of the compressor in Definition 3. Moreover, when δ' is much smaller than the precision which we want to achieve, we can ignore it and reduce the condition to the compressor condition. Meanwhile, we have an additional condition $\|Q(x)\| \leq \|x\|$, which can be easily achieved by replacing rounding to flooring. Therefore, we give a more practical condition for analyzing communication complexity. Moreover, the convergence of Efficient-Adam with quantization can be easily extended to compressors.*

4 EFFICIENT ADAM

In this section, we describe the proposed Efficient-Adam, whose workflow has already been displayed in Figure 1. To make the presentation clear, we split Efficient-Adam into two parts: the parameter server part (Algorithm 1) and N-workers part (Algorithm 2). To reduce the communication overhead among the server and workers, we introduce a two-way quantization mapping. We denote the quantization function in the parameter server as $\mathcal{Q}_s(\cdot)$ and the quantization function in the workers as $\mathcal{Q}_w(\cdot)$, respectively. The detailed iterations of Efficient-Adam are presented in the following.

In the parameter server (see Algorithm 1), the initial value of x will be broadcast to each worker at the initialization phase. Then in each iteration, the server gathers the update from each worker, calculates the average of these updates, and then broadcasts this average after a quantization function.

In each worker (see Algorithm 2), at the initial phase, the initial value of x will be received. In each update iteration, a worker will sample a stochastic gradient g_t and calculate the update vector δ_t. Then it will send the update vector to the server with a quantization mapping and receive the average update vector from the server. Finally, the worker will update its local parameters.

Algorithm 1 Efficient-Adam: server

Parameters: Choose quantization mapping $\mathcal{Q}_s(\cdot)$ via Definition 2. Initialize $x_1 \in \mathbb{X}$ and error term $e_1 = 0$.

1: Broadcasting x_1 to all workers;
2: **for** $t = 1, 2, \cdots, T$ **do**
3: Gathering and averaging all updates from workers $\hat{\delta}_t = \frac{1}{N}\sum_{i=1}^{N} \delta_t^{(i)}$;
4: Broadcasting $\tilde{\delta}_t = \mathcal{Q}_s(\hat{\delta}_t + e_t)$;
5: $e_{t+1} = \hat{\delta}_t + e_t - \tilde{\delta}_t$;
6: $x_{t+1} = x_t - \tilde{\delta}_t$;
7: **end for**

Output: x_{T+1}.

4.1 COMPLEXITY ANALYSIS

In this subsection we give iteration complexity analysis and bit communication complexity analysis in order to obtain an ε-stationary solution of problem equation 1 in Definition 1, i.e., $\mathbb{E}\|\nabla F(x)\|^2 \le \varepsilon$. We denote δ_s and δ_s' as the constants defined in Definition 1 for $\mathcal{Q}_s()$, and δ_w, δ_w' for $\mathcal{Q}_w()$. In addition, we further make another assumption on hyperparameters θ_t and α_t.

Assumption 2. *For a given maximum number of iterations T, we define the exponential moving average parameter $\theta_t = 1 - \frac{\theta}{T}$, and base learning rate $\alpha_t = \frac{\alpha}{\sqrt{T}}$. Besides, we define $\gamma = \beta/(1 - \frac{\theta}{T})$.*

Below, we characterize the iteration complexity of Efficient-Adam to achieve an ε-stationary solution of the problem equation 1. In the corollary, we characterize the overall bit communication complexity of Efficient-Adam with the quantization mapping defined in Example 3.

Algorithm 2 Efficient-Adam: i-th worker

Parameters: Choose hyperparameters $\{\alpha_t\}, \beta, \{\theta_t\}$, and quantization mapping $\mathcal{Q}_w(\cdot)$ satisfying Definition 2. Set initial values $m_0^{(i)} = 0$, $v_0^{(i)} = \epsilon$, and error term $e_1^{(i)} = 0$.

1: Receiving x_1 from the server;
2: **for** $t = 1, 2, \cdots, T$ **do**
3: Sample a stochastic gradient of $f(x_t)$ as $g_t^{(i)}$;
4: $v_t^{(i)} = \theta_t v_{t-1}^{(i)} + (1-\theta_t)\big[g_t^{(i)}\big]^2$;
5: $m_t^{(i)} = \beta m_{t-1}^{(i)} + (1-\beta) g_t^{(i)}$;
6: Sending $\delta_t^{(i)} = \mathcal{Q}_w\left(\alpha_t m_t^{(i)}/\sqrt{v_t^{(i)}} + e_t^{(i)}\right)$;
7: $e_{t+1}^{(i)} = \alpha_t m_t^{(i)}/\sqrt{v_t^{(i)}} + e_t^{(i)} - \delta_t^{(i)}$;
8: Receiving $\tilde{\delta}_t$ from the server;
9: $x_{t+1} = x_t - \tilde{\delta}_t$;
10: **end for**

Theorem 1. *Let $\{x_t\}$ be the point generated by Algorithm 1 and Algorithm 2. In addition, assume that all workers work identically and independently. Let x_τ^T be the random variable x_τ with τ taking from $\{1, 2, \cdots, T\}$ with the same probability. For given ε, when $T \ge \frac{4(G^2+\epsilon d)}{(1-\beta)^2\alpha^2\varepsilon^2}\left(F(x_1) - F^* + C_2\right)^2$, it always holds that*

$$\mathbb{E}[\|\nabla F(x_\tau^T)\|^2] \le \varepsilon + \left(\frac{\delta_s'}{\delta_s} + \frac{\delta_w'}{\delta_s\delta_w}\right)\frac{2L\sqrt{G^2+\epsilon d}C_3}{(1-\beta)},$$

where $C_1 = \left(\frac{\beta/(1-\beta)}{\sqrt{(1-\gamma)\theta_1}}+1\right)^2$, $C_2 = \frac{d}{1-\sqrt{\gamma}}\left(\frac{\alpha^2 L}{\theta(1-\sqrt{\gamma})^2\delta_w\delta_s}+\frac{2C_1G\alpha}{\sqrt{\theta}}\right)\left[\log\left(1+\frac{G^2}{\epsilon d}\right)+\frac{\theta}{1-\theta}\right]$, *and* $C_3 = \sqrt{\frac{d}{\theta(1-\sqrt{\gamma})^4}\left[\log\left(1+\frac{G^2}{\epsilon d}\right)+\frac{\theta}{1-\theta}\right]}$.

Proof Sketch

First, we establish the bound of error term $\|e_t\|$ by $\|\alpha_t m_t/\sqrt{v_t}\|$.

Then, by the Lipschitz of f, $f(x_{t+1}) \le f(x_t) + \langle \nabla f(x_t), -\alpha_t m_t/\sqrt{v_t}\rangle + \frac{L}{2}\|\alpha_t m_t/\sqrt{v_t}\|^2$.

By bounding terms $\sum_{t=1}^T \langle \nabla f(x_t), -\alpha_t m_t/\sqrt{v_t}\rangle + \frac{L}{2}\|\alpha_t m_t/\sqrt{v_t}\|^2 \le -\mathcal{O}(1)\sum_{t=1}^T \|\nabla f(x_t)\|_2^2 + \mathcal{O}(\sqrt{T})$. And we can get the desired result.

Corollary 1. *With the quantization function in Example 3, when we want to get an ε-stationary solution we need at most C_5 bits per-iteration on the workers and the server. Besides, to achieve an ε-stationary point, we need $\frac{16(G^2+\epsilon d)}{(1-\beta)^2\alpha^2\varepsilon^2}\left(F(x_1)-F^*+C_4\right)^2$ iterations, where $C_4 = \frac{d}{1-\sqrt{\gamma}}\left(\frac{4\alpha^2 L}{\theta(1-\sqrt{\gamma})^2}+\frac{2C_1G\alpha}{\sqrt{\theta}}\right)\left[\log\left(1+\frac{G^2}{\epsilon d}\right)+\frac{\theta}{1-\theta}\right]$, $C_5 = d\left(1+\log\left(\log(G)+\log\left(\frac{32Ld^2\sqrt{G^2+\epsilon d}C_3}{\varepsilon(1-\beta)}\right)+1\right)\right)$, and C_1, C_3 are defined in Theorem 1.*

From the above theorem and corollary, we can reduce the bits of communication by two-way quantization and error feedback. In addition, the bit communication complexity and iteration complexity are $O(d\log\log\frac{d^3}{\varepsilon})$ and $O(\frac{d^2}{\varepsilon^2})$, respectively. Moreover, it can be seen that adding compression in both sides can converge to an arbitrary ε-stationary point, when we have large enough communication bandwidth, and sufficient iterations. There is a limited influence when we add quantization into communication if we can have a small additional error δ_w'. Below, we show that if we replace the quantization mapping in Definition 2 as Compressor in Definition 3 in Algorithms 1-2, Efficient-Adam can attain the same order of iteration complexity as original Adam without introducing additional error terms.

Theorem 2. *Let $\{x_t\}$ be the point generated by Algorithm 1 and Algorithm 2 with the quantization mapping replaced by compressor in Definition 3. In addition, assume that all workers work identically and independently. Let x_τ^T be the random variable x_τ with τ taking from $\{1,2,\cdots,T\}$ with the same probability. For given ε, when $T \ge \frac{4(G^2+\epsilon d)}{(1-\beta)^2\alpha^2\varepsilon^2}\left(F(x_1)-F^*+C_6\right)^2$, it always holds:*

$$\mathbb{E}[\|\nabla F(x_\tau^T)\|^2] \le \varepsilon,$$

where $C_1 = \left(\frac{\beta/(1-\beta)}{\sqrt{(1-\gamma)\theta_1}}+1\right)^2$, and $C_6 = \frac{d}{1-\sqrt{\gamma}}\left(\frac{(2-\delta_w)(2-\delta_s)\alpha^2 L}{\theta(1-\sqrt{\gamma})^2\delta_w\delta_s}+\frac{2C_1G\alpha}{\sqrt{\theta}}\right)\left[\log\left(1+\frac{G^2}{\epsilon d}\right)+\frac{\theta}{1-\theta}\right]$.

From the above theorem, it can be shown that the convergence rate of Efficient-Adam matches the convergence rate of the original Adam. Thanks to the two-way error feedback techniques employed in Efficient-Adam, the convergence rate will not be hurt by the introduced bi-direction compressor as defined in Definition 3, which merely affects the speed of convergence at the constant level.

5 EXPERIMENTS

In this section, we apply our algorithms to handle several vision and language tasks. First, to accurately test the theorem on $\mathbb{E}\|\nabla f(x_t)\|^2$, we apply the algorithms to a simple stochastic convex case. Then we apply our algorithm for training neural networks on the image classification task.

5.1 STOCHASTIC CONVEX CASE

In this section, we first optimize the following toy stochastic convex optimization problem:

$$\min_x \mathbb{E}_\xi\|\xi - x\|^2,$$

where $e = [1,1,\cdots,1]^T$, x^* randomly takes from $U(0,e)$, and $\xi \sim U(0,e)+x^*$. We start from the initial point $x_1 = 0$. Then, we set $\beta = 0.9$, $T = 200000$, $\theta_t = 1-1/T$, and $\alpha_t = 1e-2/\sqrt{T}$. We

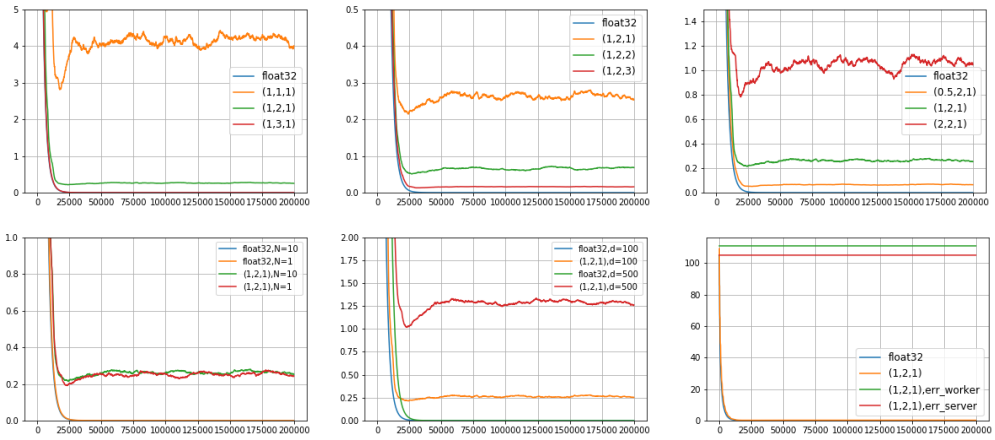

Figure 2: Results of the stochastic convex case for $N = 10$. In the figure the triple represents (G, E, M), and "err_worker" means running the algorithm without error feedback on the workers and "err_server" means running the algorithm without error feedback on the server. The x-axis represents the iterations we do the optimization. The y-axis represents the norm of expected gradient.

use type float32 as the baseline to compare the results. In this experiment, the quantization mapping is used as Example 3. We use G as the maximum value we can transmit, when the function value is larger than G we will clip it to G. We use E to represent the bit need for represent set E in Example 3, which is $\log(\log(G) + K + 1)$, and M represents the bit needed to represent set N in the Example 3. We use N to denote the number of workers. Besides, we turn off the error feedback from the server to the worker's side or workers to the server's side to see whether error feedback help accelerate the convergence. We run each setting 10 times with 10 workers and give the results of the average expected gradient norm in Figure 2. When E and M go large, we can converge to a better solution. With the smaller G, when we keep the same E, we can get a larger K. That is why we can get a better solution in the end. However, when G goes to zero, which means only a small step we can move, the convergence speed will be affected. Then we change the dimension of x from 100 to 500. As shown in the figure when the dimension grows, the solution algorithms find gets worse, which complies with the theorem. Besides, we find that error feedback on workers is more important than it on the server because without error feedback on the workers the algorithm achieves a worse result than what it does on the server.

Further, We compare our algorithm with Wen et al. (2017), Zheng et al. (2019) and Bernstein et al. (2018). To fairly compare with the other compression methods, we use the Example 2 instead of 3 in the experiment because Wen et al. (2017) and Zheng et al. (2019) are used the Example 2 as compression functions. The results are shown in figure 3. It can be seen that our algorithm converges faster and can converge to a more accurate point than SignSGD and TernGrad. and can converge faster than Zheng et al. (2019).

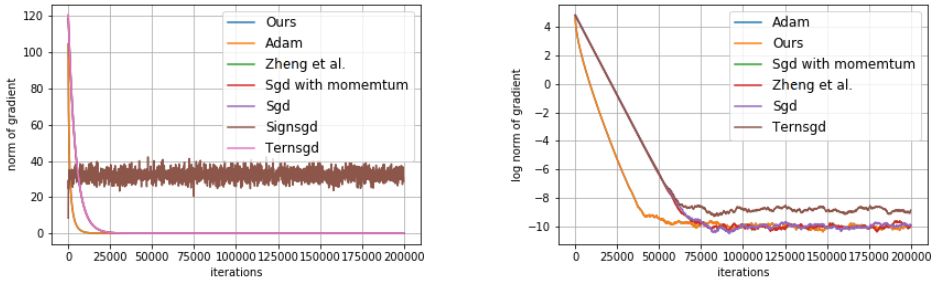

Figure 3: Results of the stochastic convex case for $N = 10$ for different method.

5.2 Image Classification

In this section, we apply our method to an image classification task. We train a ResNet-18(He et al., 2016) on the dataset CIFAR100(Krizhevsky, 2009). The CIFAR100 dataset contains 60,000 32×32

images with RGB channels, and they are labeled into 100 classes. In each class, there are 600 images, of which 500 images are used for training and 100 images are used for the test. ResNet18 contains 17 convolutional layers and one fully-connected layer. Input images will be downscaled into 1/8 size of its original size and then fed into a fully connected layer for classification training and testing. For the convolutional layers, we have 64 channels in the 1-5 convolutional layers, 128 channels in the 6-9 layers, 256 channels in the 10-13 layers, and 512 channels in the 14-17 layers. With the batch size 32, we train a network for 78200 iterations. We use cross-entropy loss with regularization to train the network. The weight of ℓ_2 regularization we used is $5e-4$. For the other hyper-parameter, we use $\beta = 0.9$, $\theta_t = 0.999$. In addition, we set $\alpha_t = 1e-4$ in the beginning and reduce it into $0.2\alpha_t$ after every 19,550 iterations, which is often used in deep learning settings to get better performance. For the communication part, the baseline model is float32, and we test quantization functions with 6 bits and 5 bits. For the 6 bits quantization function, we use $G = 1, E = 4$, and $M = 1$, while for the 5 bits quantization function, $G = 0.0625,\ E = 3,\ M = 1$, where the definitions of $G,\ E,$ and$M$ are the same as them in Section 5.1. The results are shown in Figure 4. It is shown in the results that we can have similar training errors with both 6 bits and 5 bits. However, when we test the network 6-bits version performs better than the 5-bits version, even though the 6-bits version beats the float32 version in the test. This may be because adding the quantization in the communication limits the range of parameters, which can be seen as regularization, and with proper regularization, the generality of the network will become better. Meanwhile, we test whether the error feedback term helps the convergence. As it is shown in the middle column of Figure 4, without error feedback, the model performs worse on both the training phase and the test phase. The right column shows the result when we test on different workers. In the training part, the more workers we use, the faster convergence we obtain. However, we will have a worse testing accuracy. This is an issue related to the generalization of the network when the batch size gets larger, and we will not discuss it here, due to the space limit.

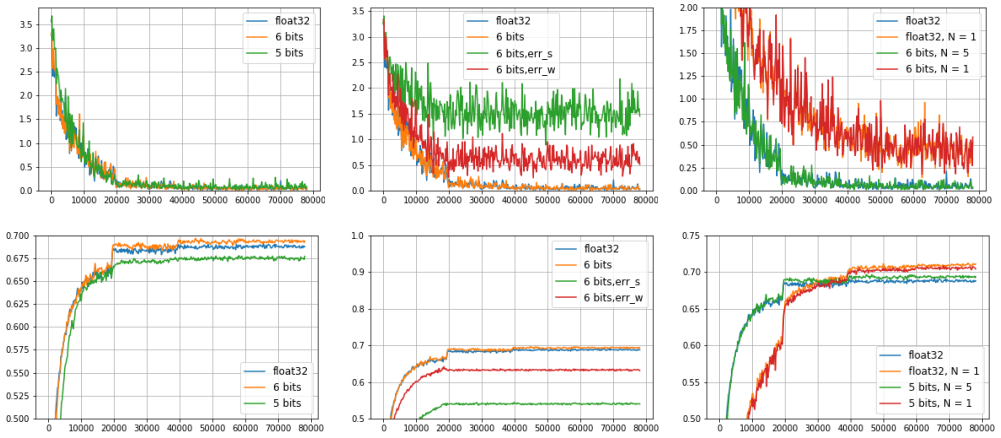

Figure 4: Results of image classification. The upper row shows the training loss and the bottom row shows the testing accuracy. "err_s", represents Efficient-Adam without error feedback on server and "err_w" represents Efficient-Adam without error feedback on workers.

6 Conclusion

We proposed a communication efficient adaptive stochastic gradient descent algorithm for optimizing the non-convex stochastic problem, dubbed *Efficient-Adam*. In the algorithm, we used a two-side quantization to reduce the communication overhead and two error feedback terms to compensate for the quantization error to encourage the convergence. With the more practical assumptions, we established a theoretical convergence result for the algorithm. Besides, we established the communication complexity and iteration complexity under some certain quantization functions. On the other hand, when the quantization operators are generalized to compressors, Efficient-Adam can achieve the same convergence rate as full-precision Adam. Lastly, we applied the algorithm to a toy task and real-world image task. The experimental results confirm the efficacy of the proposed Efficient-Adam.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

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

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. On the convergence of adam and adagrad. *arXiv preprint arXiv:2003.02395*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

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

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Lu Hou, Ruiliang Zhang, and James T. Kwok. Analysis of quantized models. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryM_IoAqYX`.

Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, pp. 2525–2536, 2018.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. *arXiv preprint arXiv:1901.09847*, 2019.

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

Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report TR-2009, University of Toronto, Toronto*, 2009.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pp. 583–598, 2014a.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670, 2014b.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 5330–5340, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Wei Liu, Li Chen, Yunfei Chen, and Wenyi Zhang. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems*, 31(8):1754–1766, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Alexander Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, 2010.

Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pp. 4447–4458, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Hanlin Tang, Xiangru Lian, Shuang Qiu, Lei Yuan, Ce Zhang, Tong Zhang, and Ji Liu. Deepsqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1907.07346*, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1299–1309, 2018.

Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. *arXiv preprint arXiv:1806.01811*, 2018.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pp. 1509–1519, 2017.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2019.

Shuai Zheng, Ziyue Huang, and James Kwok. Communication-efficient distributed blockwise momentum sgd with error-feedback. In *Advances in Neural Information Processing Systems*, pp. 11446–11456, 2019.

Fangyu Zou, Li Shen, Zequn Jie, Ju Sun, and Wei Liu. Weighted adagrad with unified momentum. *arXiv preprint arXiv:1808.03408*, 2018.

Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11127–11135, 2019.