Error Compensated Quantized SGD and its Applications to Large-scale Distributed Optimization

Jiaxiang Wu [1] Weidong Huang [1] Junzhou Huang [1] Tong Zhang [1]

Abstract

Large-scale distributed optimization is of great importance in various applications. For data-parallel based distributed learning, the inter-node gradient communication often becomes the performance bottleneck. In this paper, we propose the error compensated quantized stochastic gradient descent algorithm to improve the training efficiency. Local gradients are quantized to reduce the communication overhead, and accumulated quantization error is utilized to speed up the convergence. Furthermore, we present theoretical analysis on the convergence behaviour, and demonstrate its advantage over competitors. Extensive experiments indicate that our algorithm can compress gradients by a factor of up to two magnitudes without performance degradation.

1. Introduction

Due to the explosive growth of data in recent years, large-scale machine learning has attracted increasing attention in various domains, such as computer vision and speech recognition. Distributed optimization is one of the core building blocks in these applications, where the training data is often too massive to be efficiently handled by a single computation node.

A commonly used distributed learning framework is data parallelism, in which the whole data set is split and stored on multiple nodes within a cluster. Each node computes its local gradients and communicates gradients with other nodes to update model parameters. For such learning systems, the time consumption can be roughly categorized as either computation or communication. The communication often becomes the performance bottleneck, especially for large clusters and/or models with tons of parameters.

[1]Tencent AI Lab, Shenzhen, China. Correspondence to: Jiaxiang Wu <jonathanwu@tencent.com>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

There have been several works attempting to improve the efficiency of distributed learning by reducing the communication cost. Some methods focused on quantizing gradients into the fixed-point numbers (Zhou et al., 2016; Alistarh et al., 2017), so that much fewer bits are needed to be transimitted. More aggressive quantization, such as the binary or ternary representation, has also been investigated in (Seide et al., 2014; Strom, 2015; Wen et al., 2017). Other methods imposed sparsity onto gradients during communication, where only a small fraction of gradients get exchanged across nodes in each iteration (Wangni et al., 2017; Lin et al., 2018).

The underlying ideas of these methods are basically to compress gradients into some special form, in which each entry can be represented by much fewer bits than the original 32-bit floating-point number. Such compression introduces extra stochastic noises, *i.e.* quantization error, into the optimization process, and will slow down the convergence or even leads to divergence. 1Bit-SGD (Seide et al., 2014) adopted the error feedback scheme, which was to compensate the current local gradients with quantization error from the last iteration, before feeding it into the quantization function. Although authors stated that this improved the convergence behaviour, no theoretical analysis was given to evidence its effectiveness.

In this paper, we propose the error compensated quantized stochastic gradient descent method, namely ECQ-SGD. Our algorithm also utilizes the error feedback scheme, but here we accumulate all the previous quantization errors, rather than only from the last iteration as in 1Bit-SGD. Although empirical evaluation shows that this modification leads to faster and more stable convergence than many baseline methods, it is non-trivial to establish the theoretical guarantee for this phenomenon.

In (Alistarh et al., 2017), authors proved that for their proposed QSGD algorithm, the number of iterations required to reach a certain sub-optimality gap is proportional to the variance bound of stochastic quantized gradients. However, this cannot explain our method's convergence behaviour, since our quantized gradients are biased estimations, unlike in QSGD. Actually, the variance bound of quantized gradients is even larger than that in QSGD, due to the accumulated quantization error. In order to address this issue, we present the convergence analysis from another perspective, and prove that our algorithm has a tighter worst-case error bound than QSGD with properly chosen hyper-parameters. It can be shown that our proposed error feedback scheme can well suppress the quantization error's contribution to the error bound, leading to a smaller sub-optimality gap than QSGD, as we observed in experiments.

The remaining part of this paper will be organized as follows. We briefly review related works in Section 2, and present preliminaries in Section 3. In Section 4 and 5, we propose the ECQ-SGD algorithm and its theoretical analysis. Extensive experiments are carried out in Section 6, followed by the final conclusions.

2. Related Works

There have been several works proposed to speed up stochastic gradient descent in the context of distributed learning. Some adopt asynchronous update to decouple computation from communication, while others focus on reducing the communication overhead with gradient quantization or sparsification.

Asynchronous SGD. Hogwild! (Recht et al., 2011) is a lock-free implementation of paralleled SGD that can achieve a nearly optimal rate of convergence for certain problems. (Dean et al., 2012) proposes the *DistBelief* framework, which adopts asynchronous SGD to train deep networks under a distributed setting. The convergence behaviour of asynchronous SGD have been extensively analysed in many works (Chaturapruek et al., 2015; Zhao & Li, 2016; Zheng et al., 2017; De Sa et al., 2017).

Gradient Quantization. In (Seide et al., 2014), 1Bit-SGD is proposed to quantize each gradient component to either 1 or -1 with zero-thresholding. An error feedback scheme is introduced during quantization, to compensate the quantization error from the last iteration. Similar ideas are adopted in (Strom, 2015), which accumulates local gradients across iterations, and only transmits gradient components exceeding a pre-selected threshold. Wen et al. further extend this idea and compress gradients into ternary values with a stochastic quantization function to ensure the unbiasness (Wen et al., 2017). Quantized SGD (Alistarh et al., 2017) randomly quantizes gradients using uniformly distributed quantization points, and detailed analysis is presented to address its convergence. ZipML (Zhang et al., 2017) introduces an optimal quantization strategy via dynamically choosing quantization points based on the distribution. Zhou et al. propose the DoReFa-Net to train convolutional networks with inputs, weights, and gradients all quantized into fixed-point numbers (Zhou et al., 2016).

Gradient Sparsification. The gradient dropping method is proposed in (Aji & Heafield, 2017) to introduce sparsity into gradients to reduce the communication cost. In (Wangni et al., 2017), gradient sparsification is modelled as a linear programming problem, aiming to minimize the variance increase of quantized gradients. Lin et al. propose the deep gradient compression algorithm, utilizing momentum correction, gradient clipping, momentum factor masking, and warm-up training to achieve higher sparsity without losing the accuracy (Lin et al., 2018).

3. Preliminaries

We consider the following unconstrained optimization:

$$\min_{\mathbf{w}} \; f(\mathbf{w}) \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^d$ and $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a convex and differentiable function we wish to minimize. Often, the objective function f is defined on a set of training samples $\mathcal{D} = \{\mathbf{x}_i\}$, and the need for distributed optimization arises when the training set is too large to fit into a single node.

Assume we are solving this distributed optimization problem in a data-parallel manner. The full set $\mathcal{D}$ is evenly distributed across P nodes, and the data subset located at the p-th node is denoted as $\mathcal{D}_p$. Formally, we wish to optimize:

$$\min_{\mathbf{w}} \sum_{p=1}^{P} \sum_{\mathbf{x}_i \in \mathcal{D}_p} f(\mathbf{w}; \mathbf{x}_i) \tag{2}$$

Figure 1 depicts how model parameters $\mathbf{w}$ are updated via distributed SGD. Every node initializes its local model replica using the same random seed, to ensure the consistency of all model replicas. In the t-th iteration, each node randomly selects a mini-batch of training samples, computes local gradients, and then broadcasts to all the other nodes. When one node gathers all the local gradients sent by other nodes, global gradients can be computed and used to update model parameters.

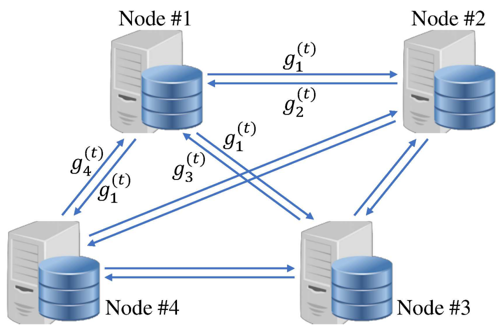

Figure 1. Distributed optimization under the data-parallel setting.

4. Error Compensated Quantized SGD

For distributed optimization under the data-parallel setting, it is required to exchange local gradients between every two nodes in each iteration. For large-scale distributed optimization with massive number of model parameters, *e.g.* training a convolutional neural network, gradient communication may become the performance bottleneck.

One possible solution is to quantize local gradients before transmission, so as to reduce the communication cost. In this section, we propose the ECQ-SGD (Error Compensated Quantized SGD) algorithm, which updates model parameters with quantized local gradients. In each iteration, current local gradients are compensated with accumulated quantization error from all the previous iterations, and then fed into a stochastic quantization function for compression.

Let $Q : \mathbb{R}^d \rightarrow \mathcal{C}^d$ be an unbiased stochastic quantization function, which maps each component in a d-dimensional vector into some element from the quantization codebook $\mathcal{C}$. The codebook usually only contains limited number of elements, so the quantized vector can be efficiently encoded. In each iteration, each node quantizes its local gradients before broadcasting:

$$\tilde{\mathbf{g}}_p^{(t)} = Q(\mathbf{g}_p^{(t)}) \tag{3}$$

where $\mathbf{g}_p^{(t)}$ is the local gradients of the p-th node at the t-th iteration, and $\tilde{\mathbf{g}}_p^{(t)}$ is its quantized counterpart.

When a node receives all the local gradients sent by other nodes, it computes the global gradients and updates its local model replica via:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \cdot \tilde{\mathbf{g}}^{(t)} = \mathbf{w}^{(t)} - \frac{\eta}{P} \sum_{p=1}^{P} \tilde{\mathbf{g}}_p^{(t)} \tag{4}$$

where $\eta > 0$ is the learning rate.

The core idea of ECQ-SGD is that when quantizing local gradients, both the current gradients and previously accumulated quantization error should be taken into consideration. Specifically, we use $\mathbf{h}_p^{(t)}$ to denote the accumulated quantization error of the p-th node at the t-th iteration:

$$\mathbf{h}_p^{(t)} = \sum_{t'=0}^{t-1} \beta^{t-1-t'} (\mathbf{g}_p^{(t')} - \tilde{\mathbf{g}}_p^{(t')}) \tag{5}$$

where β is the time decaying factor ($0 \leq \beta \leq 1$). Note that the accumulated quantization error can be incrementally updated:

$$\mathbf{h}_p^{(t)} = \beta \mathbf{h}_p^{(t-1)} + (\mathbf{g}_p^{(t-1)} - \tilde{\mathbf{g}}_p^{(t-1)}) \tag{6}$$

where $\mathbf{h}_p^{(0)} = \mathbf{0}$. The quantized local gradients are now computed by applying the quantization function to the compensated gradients:

$$\tilde{\mathbf{g}}_p^{(t)} = Q(\mathbf{g}_p^{(t)} + \alpha \mathbf{h}_p^{(t)}) \tag{7}$$

where α is the compensation coefficient ($\alpha \geq 0$).

Here, we adopt a stochastic quantization function with uniformly distributed quantization points, similar to QSGD (Alistarh et al., 2017), where the i-th component is quantized as:

$$\tilde{g}_i = \|\mathbf{g}\| \cdot \text{sgn}(g_i) \cdot \xi(|g_i|; \|\mathbf{g}\|) \tag{8}$$

where $\|\mathbf{g}\|$ acts as the scaling factor (possible choices include l_2-norm and l_∞-norm), and $\xi(\cdot)$ is a stochastic function which maps a scalar to some element in $\left\{0, \frac{1}{s}, \ldots, 1\right\}$:

$$\xi(|g_i|; \|\mathbf{g}\|) = \begin{cases} \frac{l}{s}, & \text{with probability } l + 1 - s \cdot \frac{|g_i|}{\|\mathbf{g}\|} \\ \frac{l+1}{s}, & \text{otherwise} \end{cases} \tag{9}$$

when $|g_i| / \|\mathbf{g}\|$ falls in the interval $\left[\frac{l}{s}, \frac{l+1}{s}\right)$. The hyper-parameter s defines the number of non-zero quantization levels: a larger s leads to more fine-grained quantization, at the cost of increased communication cost. From now on, we use $Q_s(\cdot)$ to denote the quantization function with s non-zero quantization levels.

After quantization, we only need $r = \lceil \log_2(2s+1) \rceil$ bits to encode each quantized $\tilde{g}_i$, and one floating-point number to represent the scaling factor $\|\mathbf{g}\|$. The overall communication cost is $32 + dr$ bits ($r \ll 32$), which is much smaller than $32d$ bits needed by the original 32-bit full-precision gradients. More efficient entropy encoding schemes, *e.g.* Huffman encoding (Huffman, 1952), can further reduce the communication cost. Let d_k denote the number of dimensions assigned to the k-th quantization level, then the overall encoding length is at most $\sum_{k=1}^{2s+1} d_k \log_2 \frac{d}{d_k}$ bits.

We summarize the overall workflow in Algorithm 1. Since all the local gradients are quantized before transmission, the communication overhead can be greatly reduced. This is crucial to the learning efficiency, especially when the inter-node communication is the performance bottleneck.

5. Theoretical Analysis

In this section, we analyse the convergence behaviour of the proposed ECQ-SGD algorithm. We start with the discussion on the variance bound of quantization error. After that, we build up the error bound for quadratic optimization problems, and demonstrate ECQ-SGD's advantage over QSGD.

5.1. Variance Bound of Quantization Error

We define the quantization error as the difference between the compensated local gradients and its quantization results:

$$\boldsymbol{\varepsilon}_p^{(t)} = \tilde{\mathbf{g}}_p^{(t)} - \left(\mathbf{g}_p^{(t)} + \alpha \mathbf{h}_p^{(t)}\right) \tag{10}$$

If the quantization function uses l_2-norm as the scaling factor, then it is identical to the one used in QSGD. In this case,

Algorithm 1 Error Compensated Quantized SGD

```
Input: distributed data $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_P$
initialize model parameters $\mathbf{w}^{(0)}$, and reset $\mathbf{h}_p^{(0)} \leftarrow \mathbf{0}$
for $t = 0, \dots, T-1$ do
  // 1. Gradient Computation and Communication
  for $p = 1, \dots, P$ do
    randomly select a mini-batch $\mathcal{D}_p^{(t)}$
    compute local gradients $\mathbf{g}_p^{(t)} = \nabla f(\mathbf{w}^{(t)}; \mathcal{D}_p^{(t)})$
    compute quantized local gradients $\tilde{\mathbf{g}}_p^{(t)}$ with (7)
    broadcast quantized local gradients $\tilde{\mathbf{g}}_p^{(t)}$
    update the accumulated quantization error $\mathbf{h}_p^{(t+1)}$
    with (6)
  end for

  // 2. Model Update
  for $p = 1, \dots, P$ do
    receive quantized local gradients $\{\tilde{\mathbf{g}}_p^{(t)}\}$
    compute global gradients $\tilde{\mathbf{g}}^{(t)} = \sum_p \tilde{\mathbf{g}}_p^{(t)} / P$
    update model parameters $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \cdot \tilde{\mathbf{g}}^{(t)}$
  end for
end for
```

we can directly borrow their conclusions on the following two properties of quantization error:

Lemma 1. *(Alistarh et al., 2017) For any vector $\mathbf{v} \in \mathbb{R}^d$, let $\boldsymbol{\varepsilon} = Q_s(\mathbf{v}) - \mathbf{v}$ denote the quantization error, then we have:*

- *Unbiasness:* $\mathbb{E}[\boldsymbol{\varepsilon}] = \mathbf{0}$
- *Bounded variance:* $\mathbb{E}\|\boldsymbol{\varepsilon}\|_2^2 \leq \min\left(\frac{d}{s^2}, \frac{\sqrt{d}}{s}\right) \cdot \|\mathbf{v}\|_2^2$

Here, we assume that the second moment of local gradients are bounded, *i.e.* $\|\mathbf{g}_p^{(t)}\|_2^2 \leq B, \forall (p,t)$. Under this assumption, the quantization error in QSGD satisfies:

$$\mathbb{E}\|\boldsymbol{\varepsilon}_p^{(t)}\|_2^2 \leq \gamma B \tag{11}$$

where $\gamma = \min(d/s^2, \sqrt{d}/s)$.

In ECQ-SGD, however, the vector to be quantized is a linear combination of current local gradients and accumulated quantization error. Still, we can derive a slightly relaxed bound for the quantization error's second moment:

Lemma 2. *For the p-th node, its quantization error at the t-th iteration satisfies:*

$$\mathbb{E}\|\boldsymbol{\varepsilon}_p^{(t)}\|_2^2 \leq \left(1 + \alpha^2\gamma \cdot \frac{1-\lambda^t}{1-\lambda}\right) \cdot \gamma B \tag{12}$$

where $\lambda = \alpha^2\gamma + (\beta - \alpha)^2$.

Proof. Please refer to the supplementary material. □

In ECQ-SGD, we usually select hyper-parameters (α, β) to satisfy $\lambda < 1$ (*i.e.* let $0 < \alpha < \frac{2}{\gamma+1}$ and $\beta = 1$), ensuring that the following variance bound of quantization error holds for any iteration t:

$$\mathbb{E}\|\boldsymbol{\varepsilon}_p^{(t)}\|_2^2 < \left(1 + \frac{\alpha^2\gamma}{1-\lambda}\right) \cdot \gamma B \tag{13}$$

Therefore, as long as $\frac{\alpha^2\gamma}{1-\lambda}$ is small, the variance bound of quantization error in the ECQ-SGD algorithm is still very close to that of QSGD. Also, it is worth noting that this upper bound does not depend on t, indicating that the variance bound will not diverge during the optimization.

Based on the above results, one can derive that the quantized local gradients' variance bound in ECQ-SGD is also larger than that in QSGD. According to the convergence analysis in QSGD, the number of iterations required to reach certain sub-optimality is proportional to the stochastic gradients' variance bound. This implies that following the convergence analysis in QSGD, one would conclude that ECQ-SGD should converge slower than QSGD, which actually conflicts with experimental results (as shown later). Therefore, we need to analyse ECQ-SGD's convergence behaviour from another perspective to explain its advantage over QSGD as observed in practice.

5.2. Convergence for Quadratic Optimization

Now we analyse how model parameters converge to the optimal solution in ECQ-SGD. Consider the following convex quadratic optimization:

$$\min_{\mathbf{w}} \sum_{p=1}^{P} \sum_{(\mathbf{A}_i, \mathbf{b}_i) \in \mathcal{D}_p} \frac{1}{2}\mathbf{w}^T\mathbf{A}_i\mathbf{w} + \mathbf{b}_i^T\mathbf{w} \tag{14}$$

where the whole data set $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_P$ is evenly split and stored at P nodes. By summing up all $\mathbf{A}_i$s and $\mathbf{b}_i$s, the above optimization is equivalent to:

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} + \mathbf{b}^T\mathbf{w} \tag{15}$$

where $\mathbf{A} = \sum_{(\mathbf{A}_i, \mathbf{b}_i) \in \mathcal{D}} \mathbf{A}_i$ and $\mathbf{b} = \sum_{(\mathbf{A}_i, \mathbf{b}_i) \in \mathcal{D}} \mathbf{b}_i$. The smallest and largest singular values of $\mathbf{A}$ are denoted as a_1 and a_2, respectively, *i.e.* $a_1\mathbf{I} \preceq \mathbf{A} \preceq a_2\mathbf{I}$. Here, we assume $a_1 > 0$ to ensure the strong convexity, which leads to the closed-form optimal solution $\mathbf{w}^* = -\mathbf{A}^{-1}\mathbf{b}$.

In each iteration, every node constructs a mini-batch of training samples $\{(\mathbf{A}_i, \mathbf{b}_i)\}$, uniformly sampled from its local data subset. The resulting local gradients can be represented as the underlying true gradients plus a stochastic noise term, which is:

$$\mathbf{g}_p^{(t)} = \mathbf{A}\mathbf{w}^{(t)} + \mathbf{b} + \boldsymbol{\xi}_p^{(t)} \tag{16}$$

where $\{\boldsymbol{\xi}_p^{(t)}\}$ are *i.i.d.* random noises with zero mean. The quantized local gradients are given by:

$$\tilde{\mathbf{g}}_p^{(t)} = \mathbf{A}\mathbf{w}^{(t)} + \mathbf{b} + \boldsymbol{\xi}_p^{(t)} + \alpha\mathbf{h}_p^{(t)} + \boldsymbol{\varepsilon}_p^{(t)} \tag{17}$$

Recall the update rule of ECQ-SGD, we have:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta(\mathbf{A}\mathbf{w}^{(t)} + \mathbf{b} + \boldsymbol{\xi}^{(t)} + \alpha\mathbf{h}^{(t)} + \boldsymbol{\varepsilon}^{(t)}) \tag{18}$$

where auxiliary variables are defined as follows:

$$\boldsymbol{\xi}^{(t)} = \frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\xi}_p^{(t)},\ \mathbf{h}^{(t)} = \frac{1}{P}\sum_{p=1}^{P}\mathbf{h}_p^{(t)},\ \boldsymbol{\varepsilon}^{(t)} = \frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\varepsilon}_p^{(t)} \tag{19}$$

From Lemma 2, we can derive the variance bound for the pseudo quantization error:

$$\mathbb{E}\|\boldsymbol{\varepsilon}^{(t)}\|_2^2 \leq \left[1 + \alpha^2\gamma \cdot \frac{1-\lambda^t}{1-\lambda}\right] \cdot \frac{\gamma B}{P} \tag{20}$$

Below, we present the worst-case upper bound on the expected distance between $\mathbf{w}^{(t+1)}$, solution obtained in the $(t+1)$-th iteration, and $\mathbf{w}^*$, the optimal solution.

Theorem 1. *Let $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} + \mathbf{b}^T\mathbf{w}$ be the objective function to be minimized, whose optimal solution is denoted as $\mathbf{w}^*$, and $R^2 = \sup_{\mathbf{w}\in\mathbb{R}^d}\|\mathbf{w}-\mathbf{w}^*\|_2^2$. Assume the stochastic noise in the mini-batch's gradients satisfies $\mathbb{E}\|\boldsymbol{\xi}^{(t)}\|_2^2 \leq \sigma^2$, then the error bound in the ECQ-SGD algorithm satisfies:*

$$\begin{aligned}\mathbb{E}\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|_2^2 \leq R^2\|\mathbf{H}^{t+1}\|_2^2 + \eta^2\sigma^2\sum_{t'=0}^{t}\|\mathbf{H}^{t'}\|_2^2 \\ + \eta^2\mathbb{E}\|\boldsymbol{\varepsilon}^{(t)}\|_2^2 + \eta^2\sum_{t'=0}^{t-1}\|\boldsymbol{\Theta}^{(t')}\|_2^2 \cdot \mathbb{E}\|\boldsymbol{\varepsilon}^{(t')}\|_2^2\end{aligned} \tag{21}$$

where $\mathbf{H} = \mathbf{I} - \eta\mathbf{A}$ and:

$$\boldsymbol{\Theta}^{(t')} = \mathbf{H}^{t-t'} - \sum_{t''=t'+1}^{t}\alpha(\beta-\alpha)^{t''-t'-1}\mathbf{H}^{t-t''} \tag{22}$$

Proof. Please refer to the supplementary material. □

It is worth mentioning that our algorithm only differs from QSGD in the last two terms of error bound given above; the other two terms are identical in both algorithms. Now we analyse $\boldsymbol{\Theta}^{(t')}$'s value to reveal the difference between ECQ-SGD and QSGD. Firstly, we can prove that:

Lemma 3. *If the learning rate satisfies $\eta a_1 < 1$, then:*

$$\boldsymbol{\Theta}^{(t')} \preceq \left(1 - \frac{\alpha}{1-\eta a_1}\frac{1-\nu^{t-t'}}{1-\nu}\right) \cdot \mathbf{H}^{t-t'} \tag{23}$$

where $\nu = (\beta-\alpha)/(1-\eta a_1)$.

By letting $0 < \alpha < 1$ and $\beta = 1 - \eta a_1$, the inequality in Lemma 3 is simplified to $\boldsymbol{\Theta}^{(t')} \preceq \nu^{t-t'} \cdot \mathbf{H}^{t-t'}$. Also, we have $0 < \nu < 1$ from its definition, which implies that in ECQ-SGD, the multiplier of each previous quantization error consistently precedes its counterpart in QSGD (where $\nu = 1$ due to $\alpha = 0$).

To simplify further discussions, we introduce the auxiliary variable $\tau^{(t')}$, defined as:

$$\tau^{(t')} = \sup\left(\|\boldsymbol{\Theta}^{(t')}\|_2^2 \cdot \mathbb{E}\|\boldsymbol{\varepsilon}^{(t')}\|_2^2\right) \tag{24}$$

which stands for the upper bound of quantization error $\boldsymbol{\varepsilon}^{(t')}$'s contribution to the final error bound (21). Similarly, the upper bound of quantization error's contribution in QSGD is given by:

$$\tau_{QSGD}^{(t')} = \|\mathbf{H}^{t-t'}\|_2^2 \cdot \frac{\gamma B}{P} \tag{25}$$

which is actually a special case of ECQ-SGD when $\alpha = 0$.

Recall that in (13), by ensuring $\lambda < 1$, the variance bound of quantization error in ECQ-SGD is relatively close to that of QSGD. Therefore, with properly chosen hyper-parameters, ECQ-SGD's reduction ratio in $\tau^{(t')}$ (comparing against QSGD) approaches to zero as the time gap $(t-t')$ grows to infinity:

Lemma 4. *If $\beta = 1 - \eta a_1$ and $\alpha > 0$ satisfies that $\lambda = \alpha^2\gamma + (\beta-\alpha)^2 < 1$, then we have:*

$$\lim_{(t-t')\to\infty}\frac{\tau^{(t')}}{\tau_{QSGD}^{(t')}} = 0 \tag{26}$$

The detailed proof for Lemma 3 and 4 can be found in the supplementary material. In practice, since the learning rate $\eta \ll \frac{1}{a_1}$, we can simply set $\beta = 1$ and α to some small positive number to approximately satisfy the requirements in Lemma 4.

Lemma 4 implies that as the iteration goes on, ECQ-SGD can better suppress all the previous quantization errors' contribution to the error bound (21) than QSGD. An intrinsic understanding is that with the error feedback scheme, quantization errors from different iterations are cancelled out when evaluating the final error bound, leading to a tighter worst-case upper bound than that of QSGD.

6. Experiments

In this section, we demonstrate the effectiveness of our proposed ECQ-SGD algorithm with extensive experiments. We start with linear models for convex optimization, and then extend to non-convex optimization with deep convolutional neural networks. We further analyse the scalability for large-scale scenarios, and finish the discussion with a detailed study on choices of hyper-parameters.

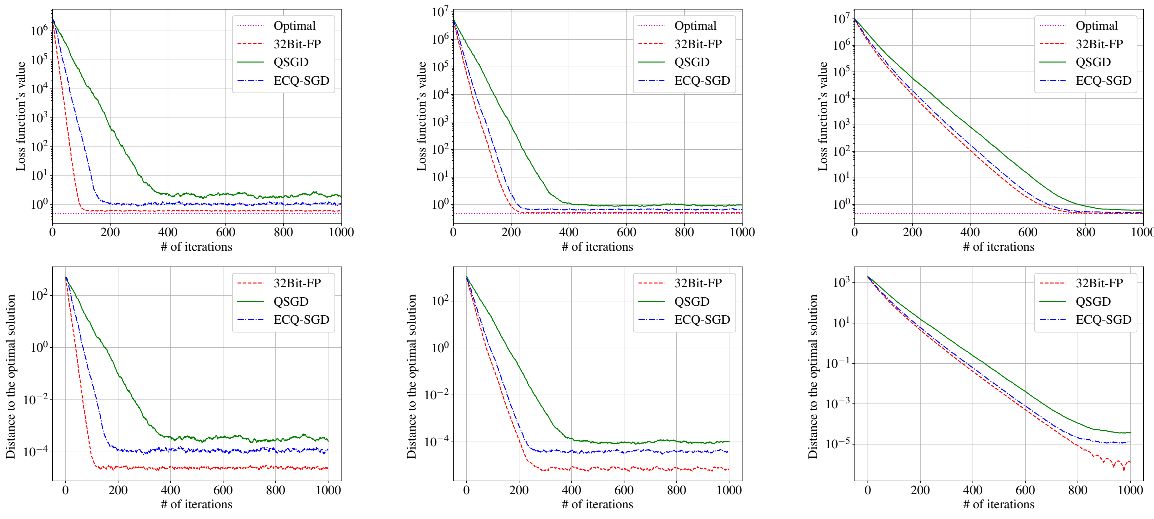

Figure 2. Comparison on the loss function's value and distance to the optimal solution (left: *Syn-256*; middle: *Syn-512*; right: *Syn-1024*).

6.1. Linear Models

In the previous convergence analysis, we claim that for convex quadratic optimization, ECQ-SGD can reduce the quantization error's contribution to the error bound, which leads to better solution after convergence. Now we verify this by comparing the convergence behaviour of ECQ-SGD and QSGD for training linear regression models.

Here we start with three synthetic datasets: Syn-256, Syn-512, and Syn-1024. Each dataset consists of 10k training samples, and the suffix denotes the feature dimension d. The training sample is generated as $y_i = \mathbf{w}^{*T}\mathbf{x}_i + \epsilon_i$, where $\mathbf{w}^* \in \mathbb{R}^d$ is the underlying model parameters we wish to obtain, and $\{\epsilon_i\}$ are *i.i.d.* random noises. The learning rate is set to 0.02, and both QSGD and ECQ-SGD use l_2-norm as the scaling factor and 4 non-zero quantization levels, *i.e.* $s = 4$. For ECQ-SGD, we let $\alpha = 0.2$ and $\beta = 0.9$.

In Figure 2, we compare the loss function's value (top) and distance to the optimal solution (bottom) in each iteration. For all three datasets, the convergence of loss function's value of ECQ-SGD is more close to the 32-bit full-precision SGD, and significantly faster than QSGD. On the other hand, the gap between QSGD (or ECQ-SGD) and 32Bit-FP in the distance to the optimal solution measures the quantization error's contribution to the error bound as defined in (21). The distance gap of ECQ-SGD is clearly smaller than QSGD, indicating that the quantization error's contribution to the error bound is well suppressed.

Now we compare the run-time speed between QSGD and ECQ-SGD on a larger dataset, Syn-20k, which consists of 50k training samples and the feature dimension is 20k. In Figure 3, we report the decomposed time consumption and test loss (in brackets) of various methods after 1k iterations. We discover that ECQ-SGD achieves similar test loss with 32Bit-FP in a shorter time than both 32Bit-FP and QSGD. Although ECQ-SGD requires extra encoding and decoding time, the overall training speed is still improved due to the reduction in the gradient communication time.

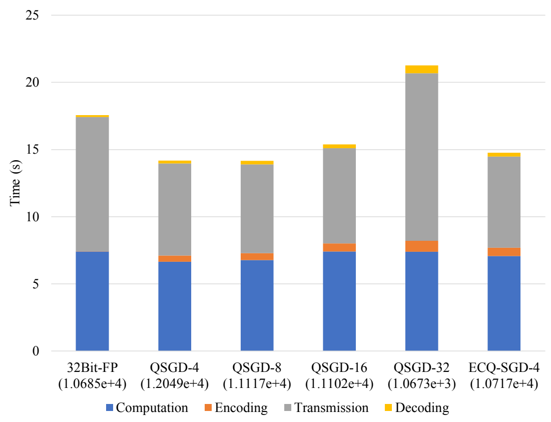

Figure 3. Comparison on the decomposed time consumption and test loss (in brackets) on the Syn-20K dataset. The suffix in QSGD and ECQ-SGD represents s, the number of non-zero quantization levels. The time consumption is for 1k iterations in total.

Furthermore, we extend the evaluation to two publicly available datasets, *YearPredictionMSD* for regression and *gisette* for classification (Chang & Lin, 2011). Linear regression and logistic regression models are trained with different gradient quantization methods on these two datasets respectively. The comparison on the loss function's value and communication cost, averaged from five random runs, is as depicted in Figure 4 and Table 1. Here we use squared l_2-loss for regression and log-loss for classification. The communication cost is measured by the total number of bits to encode gradients after 1k iterations. We use entropy encoding to fully exploit the sparsity after gradient quantization for all methods.

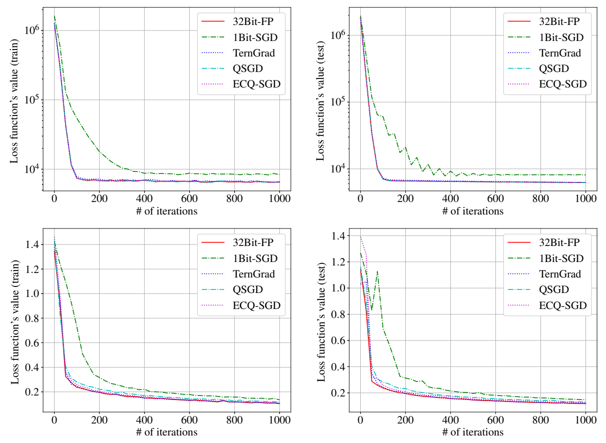

Figure 4. Comparison on the loss function's value for training linear models with various methods (top: *YearPredictionMSD*; bottom: *gisette*; left: training loss; right: test loss).

Table 1. Comparison on the loss function's value and overall communication cost for training linear models with various methods.

YearPredictionMSD	Loss	# of Bits	Ratio
32Bit-FP	$\mathbf{4.99e3}$	$2.88e6$	-
1Bit-SGD	$6.57e3$	$1.51e5$	$19.07\times$
TernGrad	$5.04e3$	$1.41e5$	$20.43\times$
QSGD	$5.11e3$	$\mathbf{1.17e5}$	$\mathbf{24.63}\times$
ECQ-SGD	$5.00e3$	$1.22e5$	$23.62\times$
Gisette	**Loss**	**# of Bits**	**Ratio**
32Bit-FP	$\mathbf{1.16e{-}1}$	$1.60e8$	-
1Bit-SGD	$1.47e{-}1$	$2.59e6$	$61.75\times$
TernGrad	$\mathbf{1.16e{-}1}$	$4.62e6$	$34.67\times$
QSGD	$1.48e{-}1$	$5.88e5$	$272.18\times$
ECQ-SGD	$\mathbf{1.16e{-}1}$	$\mathbf{5.68e5}$	$\mathbf{281.88}\times$

From Figure 4, we observe that all methods except 1Bit-SGD converge at similar speed, and the final performance is also almost identical to each other. This may due to the relatively small number of feature dimensions (90) of the *YearPredictionMSD* dataset, so that different gradient quantization methods do not have significant performance gap.

For the *gisette* dataset, whose feature dimension is 5000, both TernGrad and ECQ-SGD can still match the performance of the 32-bit full-precision SGD, while 1Bit-SGD and QSGD suffer more severe performance degradation. On the other hand, ECQ-SGD achieves much higher compression ratio ($281.88\times$) than TernGrad ($34.67\times$).

6.2. Convolutional Neural Networks

Now we evaluate the ECQ-SGD algorithm for training convolutional neural networks, which is a highly non-convex optimization problem.

The experiments are carried out on the CIFAR-10 dataset (Krizhevsky, 2009), which consists of 60k images from 10 categories. We follow the common protocol, using 50k images for training and the remaining 10k images for evaluation. We train the ResNet-20 model (He et al., 2016) with different gradient quantization methods, and the results are as reported in Figure 5. For all methods, the batch size is set to 128, and the learning rate starts from 0.1, divided by 10 at 40k and 60k iterations. The training process is terminated at the end of the 200-th epoch ($\sim$78k iterations).

In the first column of Figure 5, we compare the loss function's value and overall communication overhead of various methods. The hyper-parameters of each method[1] are selected to achieve negligible accuracy loss, comparing with the 32-bit full-precision baseline. We discover that all methods converge at a similar speed, but ECQ-SGD offers over $80\times$ reduction in the communication cost and significantly outperforms other gradient quantization methods.

In the second and third column of Figure 5, we present more detailed comparison against QSGD, since it is the most relevant one with our method. Different scaling factors are used: l_2-norm for the second column, and l_∞-norm for the third column. Both methods use a bucket size of 4096 for fair comparison. We observe that ECQ-SGD is consistently superior to QSGD in both convergence speed and classification accuracy, while the reduction in the communication cost of these two methods are similar under the same hyper-parameter settings.

6.3. Performance Model

We adopt the performance model proposed in (Yan et al., 2015) to evaluate the scalability of our ECQ-SGD algorithm. Lightweight profiling on the computation and communication time is carried out to estimate the learning efficiency for larger clusters. Major hardware specifications are as follows: Intel Xeon E5-2680 CPU, Nvidia Tesla P40 GPU (8 units per node), and Mellanox ConnectX-3 Pro network card (40Gb/s connectivity).

In Figure 6, we report the throughput for training a ResNet-50 model on the ILSVRC-12 dataset (Russakovsky et al., 2015). For training with 512 GPUs, ECQ-SGD achieves 143.5% speed-up over the vanilla SGD (66.42k vs. 27.28k

[1]For 1Bit-SGD, the bucket size is set to 3. For TernGrad, the bucket size is set to 16. For QSGD, the bucket size is set to 512, using l_∞-norm as the scaling factor and $s = 4$. For ECQ-SGD, the bucket size is set to 4096, using l_2-norm as the scaling factor, and $s = 4$, $\alpha = 0.01$, and $\beta = 1.0$.

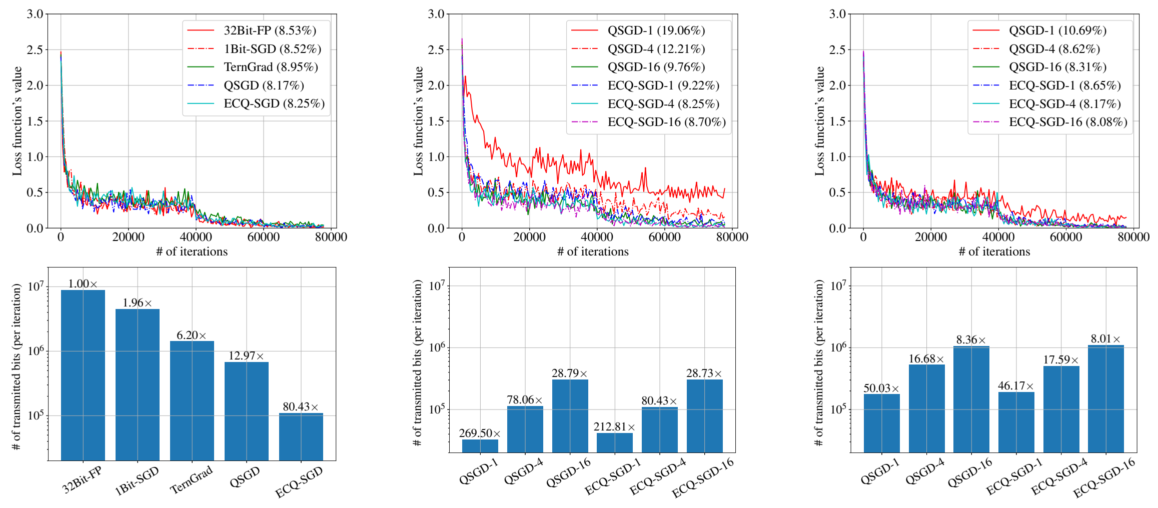

Figure 5. Comparison on the loss function's value and communication cost (measured by the number of transmitted bits) for training a LeNet model on the CIFAR-10 dataset. The classification accuracy is noted in the bracket. For the first two columns, both QSGD and ECQ-SGD use l_∞-norm as the scaling factor, and for the last column, l_2-norm is used.

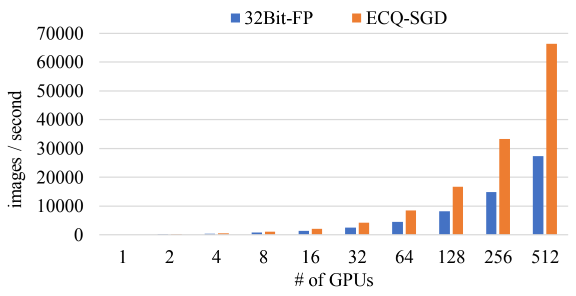

Figure 6. Comparison on the throughput for training a ResNet-50 model on the ILSVRC-12 dataset with different number of GPUs.

images per second). In our experiments, the connection bandwidth is relatively large and thus for clusters with a smaller bandwidth, it is expected that our method can achieve even higher speed-up.

6.4. Parameter Study

In Lemma 4, we present a guideline for choosing hyper-parameters α and β in ECQ-SGD. A simple practice is to set $\beta = 1$ and α to some small positive number satisfying $\alpha^2\gamma + (\beta - \alpha)^2 < 1$. Now we verify whether this is indeed optimal for training models to higher accuracy.

For quantitative evaluation, we train ResNet-20 models on the CIFAR-10 dataset with various hyper-parameters combinations, and the results are as reported in Table 2. When fixing α to 0.01, we discover that the lowest error rate is achieved when β is set to 1. On the other hand, when $\beta = 1$ is fixed, similar error rate is achieved by setting α to any value between 0.01 and 0.1. If α is too small, then the error feedback effect is greatly weaken, leading to similar performance with QSGD. If α is too large, then the constraint $\lambda = \alpha^2\gamma + (\beta - \alpha)^2 < 1$ might be violated, and the training process becomes less stable (*e.g.* when $\alpha = 0.15$, the optimization does not converge at all). The above observations are consistent with our previous analysis in Lemma 4.

Table 2. Comparison on the classification error rate of ResNet-20 on CIFAR-10, under various hyper-parameter combinations.

α	β	Err. Rate	α	β	Err. Rate
0.01	0.90	12.05%	0.001	1.00	9.18%
0.01	0.95	11.47%	0.003	1.00	9.06%
0.01	0.98	10.69%	0.010	1.00	8.25%
0.01	0.99	9.64%	0.030	1.00	8.18%
0.01	1.00	8.25%	0.100	1.00	8.22%
-	-	-	0.150	1.00	N/A

7. Conclusion

In this paper, we present the error compensated quantized SGD algorithm to improve the learning efficiency for large-scale distributed optimization. By introducing the error feedback scheme, the ECQ-SGD algorithm can effectively suppress the quantization error's contribution to the error bound. We analyse its convergence behaviour from the theoretical perspective, add demonstrate its advantage over the state-of-the-art QSGD algorithm. Experiments on convex linear models and non-convex convolutional neural networks demonstrate the efficacy of the proposed algorithm.

References

Aji, A. F. and Heafield, K. Sparse communication for distributed gradient descent. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1707–1718. 2017.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(27):1–27, 2011.

Chaturapruek, S., Duchi, J. C., and Ré, C. Asynchronous stochastic convex optimization: the noise is in the noise and sgd don't care. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1531–1539. 2015.

De Sa, C., Feldman, M., Ré, C., and Olukotun, K. Understanding and optimizing asynchronous low-precision stochastic gradient descent. In *Annual International Symposium on Computer Architecture (ISCA)*, pp. 561–574, 2017.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., aurelio Ranzato, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., and Ng, A. Y. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1223–1231. 2012.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, Sept 1952.

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

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations (ICLR)*, Apr 2018.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 693–701. 2011.

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. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Interspeech*, 2014.

Strom, N. Scalable distributed DNN training using commodity GPU cloud computing. In *Interspeech*, 2015.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. *CoRR*, abs/1710.09854, 2017.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. TernGrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1508–1518. 2017.

Yan, F., Ruwase, O., He, Y., and Chilimbi, T. Performance modeling and scalability optimization of distributed deep learning systems. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1355–1364, 2015.

Zhang, H., Li, J., Kara, K., Alistarh, D., Liu, J., and Zhang, C. ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning. In *International Conference on Machine Learning (ICML)*, pp. 4035–4043, Aug 2017.

Zhao, S.-Y. and Li, W.-J. Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2379–2385, 2016.

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

Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., and Zou, Y. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.