Moniqua: Modulo Quantized Communication in Decentralized SGD

Yucheng Lu [1] **Christopher De Sa** [1]

Abstract

Running Stochastic Gradient Descent (SGD) in a decentralized fashion has shown promising results. In this paper we propose Moniqua, a technique that allows decentralized SGD to use quantized communication. We prove in theory that Moniqua communicates a provably bounded number of bits per iteration, while converging at the same asymptotic rate as the original algorithm does with full-precision communication. Moniqua improves upon prior works in that it (1) requires zero additional memory, (2) works with 1-bit quantization, and (3) is applicable to a variety of decentralized algorithms. We demonstrate empirically that Moniqua converges faster with respect to wall clock time than other quantized decentralized algorithms. We also show that Moniqua is robust to very low bit-budgets, allowing 1-bit-per-parameter communication without compromising validation accuracy when training ResNet20 and ResNet110 on CIFAR10.

1. Introduction

Stochastic gradient descent (SGD), as a widely adopted optimization algorithm for machine learning, has shown promising performance when running in parallel (Zhang, 2004; Bottou, 2010; Dean et al., 2012; Goyal et al., 2017). However, the communication bottleneck among workers[1] can substantially slow down the training (Alistarh, 2018). State-of-the-art frameworks such as TensorFlow (Abadi et al., 2016), CNTK (Seide & Agarwal, 2016) and MXNet (Chen et al., 2015) are built in a centralized fashion, where workers exchange gradients either via a centralized parameter server (Li et al., 2014a;b) or the MPI AllReduce operation (Gropp et al., 1999). Such a design, however, puts heavy pressure on the central server and strict requirements on the underlying network. In other words, when the underlying network is poorly constructed, i.e. high latency or low bandwidth, it can easily cause degradation of training performance due to communication congestion in the central server or stragglers (slow workers) in the system.

There are two general approaches to deal with these problems: (1) decentralized training (Lian et al., 2017a;b; Tang et al., 2018b; Hendrikx et al., 2018) and (2) quantized communication[2] (Zhang et al., 2017; Alistarh et al., 2017; Wen et al., 2017). In decentralized training, all the workers are connected to form a graph and each worker communicates only with neighbors by averaging model parameters between two adjacent optimization steps. This balances load and is robust to scenarios where workers can only be partially connected or the communication latency is high. On the other hand, quantized communication reduces the amount of data exchanged among workers, leading to faster convergence with respect to wall clock time (Alistarh et al., 2017; Seide et al., 2014; Doan et al., 2018; Zhang et al., 2017; Wang et al., 2018). This is especially useful when the communication bandwidth is restricted.

At this point, a natural question is: *Can we apply quantized communication to decentralized training, and thus benefit from both of them?* Unfortunately, directly combining them together negatively affects the convergence rate (Tang et al., 2018a). This happens because existing quantization techniques are mostly designed for centralized SGD, where workers communicate via exchanging gradients (Alistarh et al., 2017; Seide et al., 2014; Wangni et al., 2018). Gradients are robust to quantization since they get smaller in magnitude near local optima and in some sense carry less information, causing quantization error to approach zero (De Sa et al., 2018). In contrast, decentralized workers are communicating the model parameters, which do not necessarily get smaller around local optima and thus the quantization error does not approach zero without explicitly increasing precision (Tang et al., 2018c). Previous work

[1]Department of Computer Science, Cornell University, Ithaca, New York, United States. Correspondence to: Yucheng Lu <yl2967@cornell.edu>, Christopher De Sa <cdesa@cs.cornell.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

[1]A worker could refer to any computing unit that is capable of computing, communicating and has local memory such as CPU, GPU, or even a single thread, etc.

[2]For brevity, in this paper we generally refer to lossy compression methods including quantization, sparsification, etc, as "quantization."

solved this problem by adding an error tracker to compensate for quantization errors (Tang et al., 2019) or adding replicas of neighboring models and focusing on quantizing model-difference which does approach zero (Koloskova et al., 2019; Tang et al., 2018a). However, these methods have limitations in that: (1) the extra replicas or error tracking incurs substantial memory overhead that is proportional to size of models and the graph (more details in Section 2); and (2) these methods are either limited to constant step size or biased quantizers (Koloskova et al., 2019; Tang et al., 2018a; 2019).

To address these problems, in this paper we propose Moniqua, an additional-memory-free method for decentralized training to use quantized communication. Moniqua supports non-constant step size and biased quantizers. Our contribution can be summarized as follows:

- We show by example that naively quantizing communication in decentralized training can fail to converge asymptotically. (Section 3)
- We propose **Moniqua**, a general algorithm that uses **mo**dular arithmetic for commu**ni**cation **qua**ntization in decentralized training. We prove applying Moniqua achieves the same asymptotic convergence rate as the baseline full-precision algorithm (D-PSGD) while supporting extreme low bit-budgets. (Section 4)
- We apply Moniqua to decentralized algorithms with variance reduction and asynchronous communication (D^2 and AD-PSGD) and prove Moniqua enjoys the same asymptotic rate as with full-precision communication when applied to these cases. (Section 5)
- We empirically evaluate Moniqua and show it outperforms all the related algorithms given an identical quantizer. We also show Moniqua is scalable and works with 1-bit quantization. (Section 6)

Intuition behind Moniqua. In decentralized training, workers communicate to average their model parameters (Lian et al., 2017a). As the algorithm converges, all the workers will approach the same stationary point as they reach consensus (Tang et al., 2018a). As a result, the difference in the same coordinate of models on two workers is becoming small. Suppose x and y are the ith coordinates of models on workers w_x and w_y, respectively. If we somehow know in advance that $|x - y| < \theta$, then if w_y needs to obtain x, it suffices to fetch $x \bmod 2\theta$ rather than x from w_x. Note that $x \bmod 2\theta$ is generally a smaller number than x, which means to obtain the same absolute error, fewer bits are needed compared to fetching x directly. Formally, this intuition is captured in the following lemma.

Lemma 1. *Define the modulo operation* mod *as the follows. For any $z \in \mathbb{R}$ and $a \in \mathbb{R}^+$,*

$$\{z \bmod a\} = \{z + na | n \in \mathbb{N}\} \cap [-a/2, a/2) \quad (1)$$

then for any $x, y \in \mathbb{R}$, if $|x - y| < \theta$, then

$$x = (x \bmod 2\theta - y \bmod 2\theta) \bmod 2\theta + y.$$

2. Related Work

Decentralized Stochastic Gradient Descent (SGD). Decentralized algorithms (Mokhtari & Ribeiro, 2015; Sirb & Ye, 2016; Lan et al., 2017; Wu et al., 2018b) have been widely studied with consideration of communication efficiency, privacy and scalability. In the domain of large-scale machine learning, D-PSGD was the first Decentralized SGD algorithm that was proven to enjoy the same asymptotic convergence rate $O(1/\sqrt{Kn})$ (where K is the number of total iterations and n is the number of workers) as centralized algorithms (Lian et al., 2017a). After D-PSGD came D^2, which improves D-PSGD and is applicable to the case where workers are not sampling from identical data sources (Tang et al., 2018b). Another extension was AD-PSGD, which lets workers communicate *asynchronously* and has a convergence rate of $O(1/\sqrt{K})$ (Lian et al., 2017b). Other relevant work includes: He et al. (2018), which investigates decentralized learning on linear models; Nazari et al. (2019), which introduces decentralized algorithms with online learning; Zhang & You (2019), which analyzes the case when workers cannot mutually communicate; and Assran et al. (2018), which investigates Decentralized SGD specifically for deep learning.

Quantized Communication in Centralized SGD. Prior research on quantized communication is often focused on centralized algorithms, such as randomized quantization (Doan et al., 2018; Suresh et al., 2017; Zhang et al., 2017) and randomized sparsification (Wangni et al., 2018; Stich et al., 2018; Wang et al., 2018; Alistarh et al., 2018). Many examples of prior work focus on studying quantization in the communication of deep learning tasks specifically (Han et al., 2015; Wen et al., 2017; Grubic et al., 2018). Alistarh et al. (2017) proposes QSGD, which uses an encoding-efficient scheme, and discusses its communication complexity. Another method, 1bitSGD, quantizes exchanged gradients with one bit per parameter and shows great empirical success on speech recognition (Seide et al., 2014). Other work discusses the convergence rate under sparsified or quantized communication (Jiang & Agrawal, 2018; Stich et al., 2018). Acharya et al. (2019) theoretically analyzes sublinear communication for distributed training.

Quantized Communication in Decentralized SGD. Quantized communication for decentralized algorithms is a rising topic in the optimization community. Previous work has proposed decentralized algorithms with quantized communication for strongly convex objectives (Reisizadeh et al., 2018). Following that, Tang et al. (2018a) proposes DCD/ECD-PSGD, which quantizes communication via estimating model difference. Furthermore, Tang

Table 1. Comparison among Moniqua and baseline algorithms, where workers form a graph with n vertices and m edges. d refers to the model dimension. Detailed discussion can be found in Section 2. The additional memory refers to the space complexity required additional to the baseline full-precision communication decentralized training algorithm (D-PSGD).

	DCD-PSGD	ECD-PSGD	ChocoSGD	DeepSqueeze	**Moniqua**
Supports biased quantizers	No	No	Yes	Yes	**Yes**
Supports 1-bit quantization	No	No	Yes	No	**Yes**
Works beyond D-PSGD	No	No	No	No	**Yes**
Non-constant Step Size	No	No	No	No	**Yes**
Additional Memory	$\Theta(md)$	$\Theta(md)$	$\Theta(md)$	$\Theta(nd)$	**0**

et al. (2019) proposes DeepSqueeze, which applies an error-compensation method (Wu et al., 2018a) to decentralized setting. Koloskova et al. (2019) proposed ChocoSGD, a method that lets workers estimate remote models with a local estimator, which supports arbitrary quantization by tuning the communication matrix.

How Moniqua improves on prior works. We summarize the comparison among Moniqua and other baseline algorithms in Table 2. Specifically, Moniqua works with a wider range of quantizers (those with biased estimation or extremely restricted precision, e.g. 1bit per parameter) with theoretical guarantees. It enjoys several statistical benefits such as supporting non-constant step sizes and can be extended to different scenarios that are beyond synchronous setting (D-PSGD). Most importantly, it prevents the algorithms from trading memory with bandwidth, requiring zero additional memory in the implementation.

3. Setting and Notation

In this section, we introduce our notation and the general assumptions we will make about the quantizers for our results to hold. Then we describe D-PSGD (Lian et al., 2017a), the basic algorithm for Decentralized SGD, and we show how naive quantization can fail in decentralized training.

Quantizers. Throughout this paper, we assume that we use a quantizer $\mathcal{Q}_\delta$ that has bounded error

$$\left\|\mathcal{Q}_\delta(\boldsymbol{x}) - \boldsymbol{x}\right\|_\infty \leq \delta \quad \text{when} \quad \boldsymbol{x} \in \left[-\tfrac{1}{2}, \tfrac{1}{2}\right)^d \tag{2}$$

where δ is some constant. Note that in this assumption, we do not assume any bound for $\boldsymbol{x}$ outside $\left[-\frac{1}{2}, \frac{1}{2}\right)^d$: as will be shown later, a bound in this region is sufficient for our theory to hold. This assumption holds for both linear (Gupta et al., 2015; De Sa et al., 2017) and non-linear (Stich, 2018; Alistarh et al., 2017) quantizers. In general, a smaller δ denotes more fine-grained quantization requiring more bits. For example, a biased linear quantizer can achieve (2) by rounding any coordinate of $\boldsymbol{x}$ to the nearest number in the set $\{2\delta n \mid n \in \mathbb{Z}\}$; this will require about δ^{-1} quantization points to cover the interval $[-1/2, 1/2)$, so such a linear quantizer can satisfy (2) using only $\lceil \log_2\left(\frac{1}{2\delta}+1\right)\rceil$ bits (Li et al., 2017; Gupta et al., 2015).

Decentralized parallel stochastic gradient descent (D-PSGD). D-PSGD (Lian et al., 2017a) is the first and most basic Decentralized SGD algorithm. In D-PSGD, n workers are connected to form a graph. Each worker i stores a copy of model $\boldsymbol{x} \in \mathbb{R}^d$ and a local dataset $\mathcal{D}_i$ and collaborates to optimize

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i} f_i(\boldsymbol{x};\xi)}_{f_i(\boldsymbol{x})}. \tag{3}$$

where ξ is a data sample from $\mathcal{D}_i$. In each iteration of D-PSGD, worker i computes a local gradient sample using $\mathcal{D}_i$. Then it *averages* its model parameters with its neighbors according to a symmetric and doubly stochastic matrix $\boldsymbol{W}$, where $\boldsymbol{W}_{ij}$ denotes the ratio worker j averages from worker i. Formally: Let $\boldsymbol{x}_{k,i}$ and $\tilde{\boldsymbol{g}}_{k,i}$ denote local model and sampled gradient on worker i at k-th iteration, respectively. Let α_k denote the step size. The update rule of D-PSGD can be expressed as:

$$\begin{aligned}\boldsymbol{x}_{k+1,i} &= \sum\nolimits_{j=1}^{n} \boldsymbol{x}_{k,j}\boldsymbol{W}_{ji} - \alpha_k\tilde{\boldsymbol{g}}_{k,i} \\ &= \underbrace{\boldsymbol{x}_{k,i} - \sum\nolimits_{j=1}^{n}(\boldsymbol{x}_{k,i} - \boldsymbol{x}_{k,j})\boldsymbol{W}_{ji}}_{\text{communicate to reduce difference}} \underbrace{-\alpha_k\tilde{\boldsymbol{g}}_{k,i}}_{\text{gradient step}}\end{aligned}$$

From (3) we can see the update of a single local model contains two parts: communication to reduce model difference and a gradient step. Lian et al. (2017a) shows that all local models in D-PSGD reach the same stationary point.

Failure with naive quantization. Here, we illustrate why naively quantizing communication in decentralized training —directly quantizing the exchanged data—can fail to converge asymptotically even on a simple problem. This naive approach with quantizer $\mathcal{Q}_\delta$ can be represented by

$$\boldsymbol{x}_{k+1,i} = \boldsymbol{x}_{k,i}\boldsymbol{W}_{ii} + \sum\nolimits_{j\neq i} \mathcal{Q}_\delta(\boldsymbol{x}_{k,j})\boldsymbol{W}_{ji} - \alpha_k \tilde{\boldsymbol{g}}_{k,i} \quad (4)$$

Based on Equation 4, we obtain the following theorem.

Theorem 1. *For some constant δ, suppose that we use an unbiased linear quantizer $\mathcal{Q}_\delta$ with representable points $\{\delta n \mid n \in \mathbb{Z}\}$ to learn on the quadratic objective function $f(\boldsymbol{x}) = (\boldsymbol{x} - \delta\mathbf{1}/2)^\top(\boldsymbol{x} - \delta\mathbf{1}/2)/2$ with the direct quantization approach (4). Let ϕ denote the smallest value of a non-zero entry in $\boldsymbol{W}$. Regardless of what step size we adopt, it will always hold for all iterations k and local model indices i that $\mathbb{E}\left\|\nabla f(\boldsymbol{x}_{k,i})\right\|^2 \geq \frac{\phi^2\delta^2}{8(1+\phi^2)}$. That is, the local iterates will fail to asymptotically converge to a region of small gradient magnitude in expectation.*

Theorem 1 shows that naively quantizing communication in decentralized SGD, even with an unbiased quantizer, any local model can fail to converge on a simple quadratic objective. This is not satisfying, since, it implies we would need more advanced quantizers which are likely to require more system resources such as memory. In the following section, we propose a technique, Moniqua, that solves this problem.

4. Moniqua

In Section 1, we described the basic idea behind Moniqua: to use modular arithmetic to decrease the magnitude of the numbers we are quantizing. We now describe how Moniqua implements this intuition with a given quantizer $\mathcal{Q}_\delta$. Consider the two-scalar example from Section 1. Suppose we know y and $|x-y| < \theta$ and need to fetch x from a remote host via a quantizer $\mathcal{Q}_\delta$ to recover x. We've shown in Section 3 that fetching and using $\mathcal{Q}_\delta(x)$ leads to divergence. Instead, we define a parameter $B_\theta = (2\theta)/(1-2\delta)$ and then use the modulo operation and fetch $\mathcal{Q}_\delta\left((x/B_\theta) \bmod 1\right)$ from the remote host, from which we can approximately recover x as

$$\hat{x} = \left(B_\theta \mathcal{Q}_\delta\left((x/B_\theta) \bmod 1\right) - y\right) \bmod B_\theta + y. \quad (5)$$

Note that inside the quantizer we rescale x to x/B_θ, which is required for (2) to apply. This approach has quantization error bounded proportional to the original bound θ, as shown in the following lemma.

Lemma 2. *For any scalars $x, y \in \mathbb{R}$, if $|x-y| < \theta$ and if $\delta < \frac{1}{2}$, then if we set $B_\theta = (2\theta)/(1-2\delta)$ and $\hat{x}$ as in (5),*

$$|\hat{x} - x| \leq \delta B_\theta = \theta \cdot (2\delta)/(1-2\delta).$$

Importantly, since the quantization error is decreasing with θ, if we are able to prove a decentralized algorithm approaches consensus and use this proof to give a bound of the form $|x-y| < \theta$, this bound will give us a compression procedure (5) with smaller error as our consensus bound improves. We formalize this approach as Moniqua (Algorithm 1). (Note that all the division and mod operations in Algorithm 1 act element-wise.)

Algorithm 1 Pseudo-code of Moniqua on worker i

```
Require: initial point $x_{0,i} = x_0$, step size $\{\alpha_k\}_{k\geq 0}$, the a priori bound $\{\theta_k\}_{k\geq 0}$, communication matrix $W$, number of iterations $K$, quantizer $\mathcal{Q}_\delta$, neighbor list $\mathcal{N}_i$
1: for $k = 0, 1, 2, \cdots, K-1$ do
2:     Compute a local stochastic gradient $\tilde{g}_{k,i}$ with data sample $\xi_{k,i}$ and current weight $x_{k,i}$
3:     Send modulo-ed model to neighbors:
           $q_{k,i} = \mathcal{Q}_\delta\left((x_{k,i}/B_{\theta_k}) \bmod 1\right)$
4:     Compute local biased term $\hat{x}_{k,i}$ as:
           $\hat{x}_{k,i} = q_{k,i}B_{\theta_k} - x_{k,i} \bmod B_{\theta_k} + x_{k,i}$
5:     Recover model received from worker $j$ as:
           $\hat{x}_{k,j} = \left(q_{k,j}B_{\theta_k} - x_{k,i}\right) \bmod B_{\theta_k} + x_{k,i}$
6:     Average with neighboring workers:
           $x_{k+\frac{1}{2},i} \leftarrow x_{k,i} + \sum_{j\in\mathcal{N}_i} (\hat{x}_{k,j} - \hat{x}_{k,i})W_{ji}$
7:     Update the local weight with local gradient:
           $x_{k+1,i} \leftarrow x_{k+\frac{1}{2},i} - \alpha_k \tilde{g}_{k,i}$
8: end for
9: return averaged model $\overline{X}_K = \frac{1}{n}\sum_{i=1}^n x_{K,i}$
```

Note that in line 4 and 6, we compute and cancel out a local biased term, this is to cancel out the extra noise which may be brought to the averaged model. As we will show in the supplementary material, cancelling out this local biased term reduces extra noise to the algorithm. And in Algorithm 1, we consider the general case where θ can be a iteration dependent bound. As will be shown later, a constant θ also guarantees convergence.

We now proceed to analyze the convergence rate of Algorithm 1. We use the following common assumptions for analyzing decentralized optimization algorithms (Lian et al., 2017a; Tang et al., 2018a; Koloskova et al., 2019).

(A1) Lipschitzian gradient. All the functions f_i have L-Lipschitzian gradients.

$$\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$$

(A2) Spectral gap. The communication matrix $\boldsymbol{W}$ is a symmetric doubly stochastic matrix and

$$\max\{|\lambda_2(\boldsymbol{W})|, |\lambda_n(\boldsymbol{W})|\} = \rho < 1,$$

where $\lambda_i(\boldsymbol{W})$ denotes the the ith largest eigenvalue of $\boldsymbol{W}$.

(A3) Bounded variance. There exist non-negative constants σ and $\varsigma \in \mathbb{R}$ such that

$$\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left\|\nabla\tilde{f}_i(\boldsymbol{x};\xi_i)-\nabla f_i(\boldsymbol{x})\right\|^2 \leq\sigma^2$$
$$\mathbb{E}_{i\sim\{1,\cdots,n\}}\left\|\nabla f_i(\boldsymbol{x})-\nabla f(\boldsymbol{x})\right\|^2 \leq\varsigma^2$$

where $\nabla\tilde{f}_i(\boldsymbol{x};\xi_i)$ denotes the gradient sample on worker i computed via data sample ξ_i.

(A4) **Initialization.** All the local models are initialized with the same weight: $\boldsymbol{x}_{0,i}=\boldsymbol{x}_0$ for all i, and without loss of generality $\boldsymbol{x}_0=\mathbf{0}^d$.

(A5) **Bounded gradient magnitude.** For some constant G_∞, the norm of a sampled gradient is bounded by $\left\|\tilde{\boldsymbol{g}}_{k,i}\right\|_\infty\leq G_\infty$, for all i and k.

Lemma 2 states that the error bound from quantization is proportional to θ. In other words, a tight estimation or choice on the θ will lead to smaller quantization error in the algorithm. We present these parameter choices in Theorem 2, along with the resulting convergence rate for Moniqua.

Theorem 2. *Consider adopting a non-increasing step size scheme $\{\alpha_t\}_{t\geq 0}$ such that there exists constant $C_\alpha>0$ and η $(0<\eta\leq 1)$ that for any $k,t\geq 0$, $\frac{\alpha_k}{\alpha_{k+t}}\leq C_\alpha\eta^t$, set $\theta_k=\frac{2\alpha_k G_\infty C_\alpha\log(16n)}{1-\eta\rho}$ and $\delta=\frac{1-\eta\rho}{8C_\alpha^2\eta\log(16n)+2(1-\eta\rho)}$, then Algorithm 1 converges at the following rate:*

$$\sum_{k=0}^{K-1}\alpha_k\mathbb{E}\left\|\nabla f(\overline{\boldsymbol{X}}_k)\right\|^2\leq 4(\mathbb{E}f(\mathbf{0})-\mathbb{E}f^*)+\frac{2\sigma^2L}{n}\sum_{k=0}^{K-1}\alpha_k^2+\frac{8(\sigma^2+3\varsigma^2)L^2}{(1-\rho)^2}\sum_{k=0}^{K-1}\alpha_k^3+\frac{8G_\infty^2dL^2}{(1-\rho)^2C_\alpha^2}\sum_{k=0}^{K-1}\alpha_k^3$$

where $f^*=\inf_{\boldsymbol{x}}f(\boldsymbol{x})$.

Theorem 2 shows that the priori bound θ_k is proportional to the step size and increases at the logarithmic speed when system size n increases. The two-constant assumption on the step size prevents it from decreasing too fast. As a rapidly decreasing step size would prevent us from obtaining such a priori bound in theory. This assumption generally holds for most of the step size schemes. Just as baseline algorithms, by setting step size to a constant, we can obtain a concrete convergence bound as shown in the following corollary.

Corollary 1. *If we adopt a step size scheme where $\alpha_k=\frac{1}{\varsigma^{2/3}K^{1/3}+\sigma\sqrt{K/n}+2L}$ in Theorem 2, then the output of Algorithm 1 converges at the asymptotic rate*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f(\overline{\boldsymbol{X}}_k)\right\|^2\lesssim\frac{1}{K}+\frac{\sigma}{\sqrt{nK}}+\frac{\varsigma^{\frac{2}{3}}}{K^{\frac{2}{3}}}+\frac{(\sigma^2+G_\infty^2d)n}{\sigma^2K+n}.$$

Consistent with D-PSGD. Note that D-PSGD converges at the asymptotic rate of $O(\sigma/\sqrt{nK}+\varsigma^{\frac{2}{3}}/K^{\frac{2}{3}}+n/K)$, and thus Moniqua has the same asymptotic rate as D-PSGD (Lian et al., 2017a). That is, the asymptotic convergence rate is not negatively impacted by the quantization.

Robust to large d. In Assumptions (A3) and (A5), we use l_2-norm and l_∞-norm to bound sample variance and gradient magnitude, respectively. Note that, when d gets larger, the variance σ^2 will also tend to grow proportionally. So, the last term will tend to remain n/K asymptotically with large d.

Bound on the Bits. The specific number of bits required by Moniqua depends on the underlying quantizer ($\mathcal{Q}_\delta$). If we use nearest neighbor rounding (Gupta et al., 2015) with a linear quantizer as $\mathcal{Q}_\delta$ in Theorem 2, it suffices to use at each step a number of bits $\mathcal{B}$ for each parameter sent, where

$$\mathcal{B}\leq\left\lceil\log_2\left(\frac{1}{2\delta}+1\right)\right\rceil=\left\lceil\log_2\left(\frac{4\log_2(16n)}{1-\rho}+3\right)\right\rceil$$

Note that this bound is independent of model dimension d. When the system scales up, the number of required bits grows at a rate of $O(\log\log n)$. Note that, this is a general bound on the number of bits required by Moniqua using the same communication matrix $\boldsymbol{W}$ as the baseline. To enforce a even more restricted bit-budget (e.g. 1 bit), Moniqua can still converge at the same rate by adjusting the communication matrix.

1-bit Quantization. We can also add a consensus step (Tang et al., 2019; Koloskova et al., 2019) to allow Moniqua to use 1 bit per number. Specifically, we adopt a slack communication matrix $\overline{\boldsymbol{W}}=\gamma\boldsymbol{W}+(1-\gamma)\boldsymbol{I}$ and tune γ as a hyperparameter. We formalize this result in the following Theorem.

Theorem 3. *Consider using a communication matrix in the form of $\overline{\boldsymbol{W}}=\gamma\boldsymbol{W}+(1-\gamma)\boldsymbol{I}$. If we set $\theta=\frac{2\alpha G_\infty\log(16n)}{\gamma(1-\rho)}$, $\gamma=\frac{2}{1-\rho+\frac{16\delta^2}{(1-2\delta)^2}\cdot\frac{64\log(4n)\log(K)}{1-\rho}}$, and $\alpha=\frac{1}{\varsigma^{\frac{2}{3}}K^{\frac{1}{3}}+\sigma\sqrt{\frac{K}{n}}+2L}$, then the output of Algorithm 1 converges at the asymptotic rate*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f(\overline{\boldsymbol{X}}_k)\right\|^2\lesssim\frac{\sigma}{\sqrt{nK}}+\frac{1}{K}+\frac{\varsigma^{\frac{2}{3}}\delta^4\log^2(n)\log^2(K)}{K^{\frac{2}{3}}(1-2\delta)^4}+\frac{\sigma^2n\delta^4\log^2(n)\log^2(K)}{(\sigma^2K+n)(1-2\delta)^4}+\frac{n\delta^6\log^4(n)\log^2(K)}{(\sigma^2K+n)(1-2\delta)^6}$$

Note that the dominant term in Lemma 3 is still $O(\sigma/\sqrt{nK})$, which means Moniqua converges at the asymptotic rate the same as full precision D-PSGD (Lian et al., 2017a) even with more restricted bits-budget. Note that in Theorem 3, the only requirement on the quantizer is $\delta<\frac{1}{2}$. Considering the properties of our quantizer (2), this version of Moniqua allowes us to use 1 bit in general per parameter.

5. Scalable Moniqua

So far, we have discussed how Moniqua, along with baseline algorithms, modifies D-PSGD to use communication quantization. Note that the basic idea of using modular arithmetic in quantized communication is invariant to the algorithm being used. In light of this, in this section we show Moniqua is general enough to be applied on other decentralized algorithms that are beyond D-PSGD. Previous work has extended D-PSGD to D^2 (Tang et al., 2018b) (to make Decentralized SGD applicable to workers sampling from different data sources) and AD-PSGD (Lian et al., 2017b) (an asynchronous version of D-PSGD). In this section, we prove Moniqua is applicable to both of these algorithms.

Moniqua with Decentralized Data Decentralized data refers to the case where all the local datasets $\mathcal{D}_i$ are not identically distributed (Tang et al., 2018b). More explicitly, the outer variance $\mathbb{E}_{i\sim\{1,\cdots,n\}}\left\|\nabla f_i(\boldsymbol{x})-\nabla f(\boldsymbol{x})\right\|^2$ is no longer bounded by ς^2 as assumed in D-PSGD (Assumption (A3)). We apply Moniqua to D^2 (Tang et al., 2018b), a decentralized algorithm designed to tackle this problem by reduing the variance over time. Applying Moniqua on D^2 can be explicitly expressed[3] as:

$$\begin{aligned}\boldsymbol{X}_{k+\frac{1}{2}} &= 2\boldsymbol{X}_k - \boldsymbol{X}_{k-1} - \alpha_k\tilde{\boldsymbol{G}}_k + \alpha_{k-1}\tilde{\boldsymbol{G}}_{k-1}\\ \boldsymbol{X}_{k+1} &= \boldsymbol{X}_{k+\frac{1}{2}}\boldsymbol{W} + (\hat{\boldsymbol{X}}_{k+\frac{1}{2}} - \boldsymbol{X}_{k+\frac{1}{2}})(\boldsymbol{W}-\boldsymbol{I})\end{aligned}$$

where $\boldsymbol{X}_k$, $\tilde{\boldsymbol{G}}_k$ and $\hat{\boldsymbol{X}}_{k+\frac{1}{2}}$ are matrix in the shape of $\mathbb{R}^{d\times n}$, where their i-th column are $\boldsymbol{x}_{k,i}$, $\tilde{\boldsymbol{g}}_{k,i}$ and $\hat{\boldsymbol{x}}_{k+\frac{1}{2},i}$ respectively. And $\boldsymbol{X}_{-1}$ and $\tilde{\boldsymbol{G}}_{-1}$ are $\boldsymbol{0}^{d\times n}$ by convention. Based on this, we obtain the following convergence theorem.

Theorem 4. *If we apply Moniqua on D^2 in a setting where $\theta = (6D_1 n + 8)\alpha G_\infty$, $\delta = \frac{1}{12nD_2+2}$ and $\alpha_k = \alpha = \frac{1}{\sigma\sqrt{K/n}+2L}$ where D_1 and D_2 are two constants[4], applying Moniqua on D^2 has the following asymptotic convergence rate:*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f(\overline{\boldsymbol{X}}_k)\right\|^2 \lesssim \frac{1}{K} + \frac{\sigma}{\sqrt{nK}} + \frac{(\sigma^2+G_\infty^2 d)n}{\sigma^2 K + n}.$$

Note that D^2 (Tang et al., 2018b) with full-precision communication has the asymptotic convergence rate of $O\left(\frac{1}{K}+\frac{\sigma}{\sqrt{nK}}+\frac{n}{K}\right)$, Moniqua on D^2 has the same asymptotic rate.

Moniqua with Asychronous Communication Both D-PSGD and D^2 are synchronous algorithms as they require global synchronization at the end of each iteration, which can become a bottleneck when such synchronization is not cheap. Another algorithm, AD-PSGD, avoids this overhead by letting workers communicate asynchronously (Lian et al., 2017b). In the analysis of AD-PSGD, an iteration represents a *single* gradient update on *one* randomly-chosen worker, rather than a synchronous bulk update of all the workers. This single-worker-update analysis models the asynchronous nature of the algorithm. Applying Moniqua on AD-PSGD can be explicitly expressed[5] as:

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k\boldsymbol{W}_k + (\hat{\boldsymbol{X}}_k - \boldsymbol{X}_k)(\boldsymbol{W}_k - \boldsymbol{I}) - \alpha_k\tilde{\boldsymbol{G}}_{k-\tau_k}$$

where $\boldsymbol{W}_k$ describes the communication behaviour between the kth and $(k+1)$th gradient update, and τ_k denotes the delay (measured as a number of iterations) between when the gradient is computed and updated to the model. Note that unlike D-PSGD, here $\boldsymbol{W}_k$ can be different at each update step and usually each individually has $\rho = 1$, so we can't expect to get a bound in terms of a bound on the spectral gap, as we did in Theorems 2 and 4. Instead, we require the following condition, which is inspired by the literature on Markov chain Monte Carlo methods: for some constant t_{mix} and for any k, $\forall\boldsymbol{\mu}\in\mathbb{R}^n$, if $\boldsymbol{e}_i^\top\boldsymbol{\mu}_i \geq 0$ and $\boldsymbol{1}^\top\boldsymbol{\mu} = 1$, it must hold that $\left\|\left(\prod_{i=1}^{t_{\text{mix}}}\boldsymbol{W}_{k+i}\right)\boldsymbol{\mu} - \frac{\boldsymbol{1}}{n}\right\|_1 \leq \frac{1}{2}$. We call this constant t_{mix} because it is effectively the *mixing time* of the time-inhomogeneous Markov chain with transition probability matrix $\boldsymbol{W}_k$ at time k (Levin & Peres, 2017). Note that this condition is more general than those used in previous work on AD-PSGD because it does not require that the $\boldsymbol{W}_k$ are sampled independently or in an unbiased manner. Using this, we obtain the following convergence theorem.

Theorem 5. *If we apply Moniqua on AD-PSGD in a setting where $\theta = 16t_{\text{mix}}\alpha G_\infty$, $\delta = \frac{1}{64t_{\text{mix}}+2}$ and $\alpha_k = \alpha = \frac{n}{2L+\sqrt{K(\sigma^2+6\varsigma^2)}}$, applying Moniqua on AD-PSGD has the following asymptotic convergence rate:*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left\|\nabla f(\overline{\boldsymbol{X}}_k)\right\|^2 \lesssim \frac{1}{K} + \frac{\sqrt{\sigma^2+6\varsigma^2}}{\sqrt{K}} + \frac{(\sigma^2+6\varsigma^2)t_{\text{mix}}^2 n^2}{(\sigma^2+6\varsigma^2)K+1} + \frac{n^2 t_{\text{mix}}^2 G_\infty^2 d}{(\sigma^2+6\varsigma^2)K+1}$$

Note that AD-PSGD (Lian et al., 2017b) with full-precision communication has the asymptotic convergence rate of $O\left(\frac{1}{K}+\frac{\sqrt{\sigma^2+6\varsigma^2}}{\sqrt{K}}+\frac{n^2}{K}\right)$, Moniqua obtains the same asymptotic rate.

Since adopting a slack matrix to enable 1-bit quantization in these two algorithms will be similar to the case in Theorem 3, we omit the discussion here for brevity.

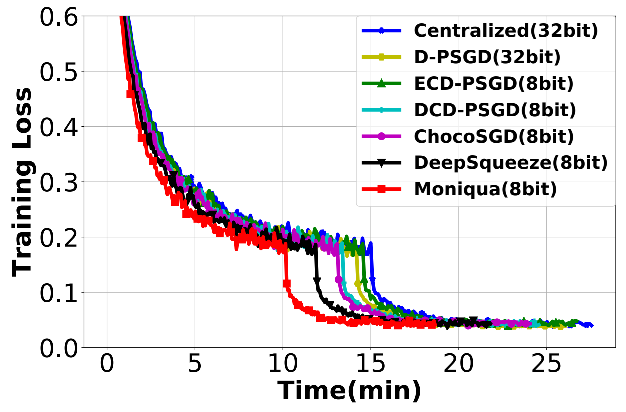
6. Experiments

In this section, we evaluate Moniqua empirically. First, we compare Moniqua and other quantized decentralized train-

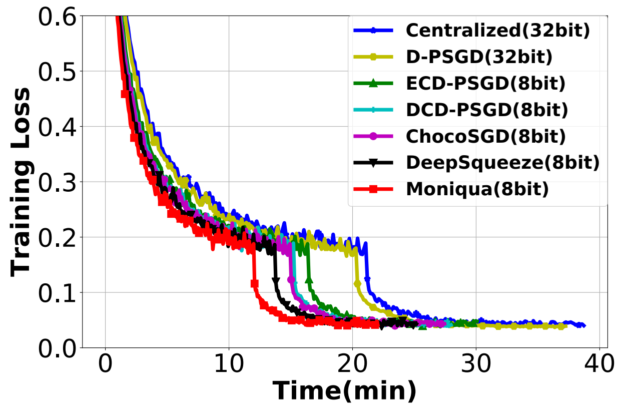
[3]For brevity, the detailed pseudo code can be found in the supplemenraty material, Section G.

[4]they only depend on the eigenvalues of W (definition can be found in supplementary material, section G)

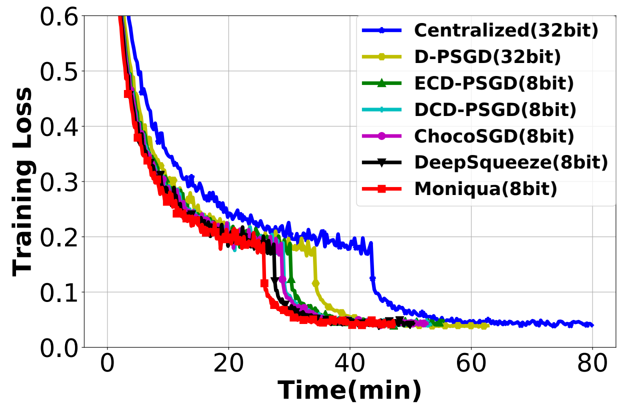
[5]For brevity, the detailed pseudo code can be found in the supplemenraty material, section H.



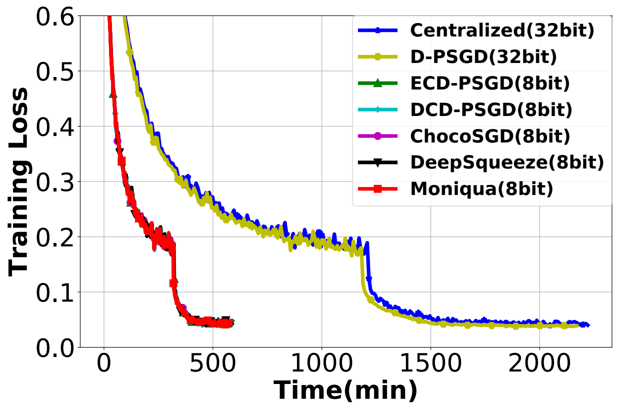
(a) Train Loss vs Time(s), Bandwidth=200Mbps, Latency=0.15ms



(b) Train Loss vs Time(s), Bandwidth=100Mbps, Latency=0.15ms



(c) Train Loss vs Time(s), Bandwidth=100Mbps, Latency=1.0ms



(d) Train Loss vs Time(s), Bandwidth=1.0Mbps, Latency=1.0ms

Figure 1. Performance of different algorithms under different network configurations

ing algorithms' convergence under different network configurations. Second, we compare the validation performance of them under extreme bit-budget. Then we investigate Moniqua's scalability on D^2 and AD-PSGD. Finally, we introduce several useful techniques for running Moniqua efficiently.

Setting and baselines. All the models and training scripts in this section are implemented in PyTorch and run on Google Cloud Platform. We launch one instance as one worker in previous formulation, each configured with a 2-core CPU with 4 GB memory and an NVIDIA Tesla P100 GPU. We use MPICH as the communication backend. All the instances are running Ubuntu 16.04, and latency and bandwidth on the underlying network are configured using the `tc` command in Linux. Throughout our experiments, we adopt the commonly used (Gupta et al., 2015; Li et al., 2017) stochastic rounding[6]. We compare Moniqua with the following baselines: Centralized (implemented as MPI AllReduce operation), D-PSGD (Lian et al., 2017a) with full-precision communication, DCD/ECD-PSGD (Tang et al., 2018a), ChocoSGD (Koloskova et al., 2019) and DeepSqueeze (Tang et al., 2019). In the experiment, we adopt the following hyperparameters for Moniqua: $\{\text{Momentum} = 0.9, \text{Weight Decay} = 5e-4, \text{Batch Size} = 128, \text{Initial Step Size} = 0.1, \theta_k = 2.0\}$. In the extreme-bit-budget experiment, we further use adopt the average ratio $\{\gamma = 5e-3\}$.

[6]Since several baselines are not applicable to biased quantizer, for fair comparison we consistently use stochastic rounding (unbiased).

Wall-clock time evaluation. We start by evaluating the performance of Moniqua and other baseline algorithms under different network configurations. We launch 8 workers connected in a ring topology and train a ResNet20 (He et al., 2016) model on CIFAR10 (Krizhevsky et al., 2014). For all the algorithms, we quantize each parameter into 8-bit representation.

We plot our results in Figure 1. We can see from Figures 1(a) to 1(b) that when the network bandwidth decreases, the curves begin to separate. AllReduce and full-precision D-

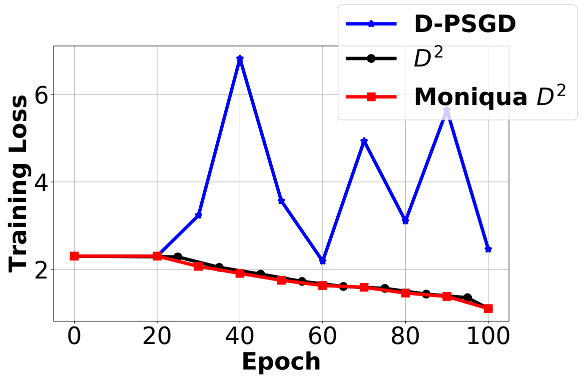
Table 2. Final test accuracy of ResNet20 and ResNet110 on CIFAR10 trained by different algorithms. ("diverge" means the algorithm cannot converge. "extra memory" means the extra memory required by different algorithms compared to full precision D-PSGD.)

		DCD-PSGD	ECD-PSGD	ChocoSGD	DeepSqueeze	Moniqua
ResNet20	budget: 1bit	diverge	diverge	$90.88 \pm 0.13\%$	$90.02 \pm 0.22\%$	$91.08 \pm 0.19\%$
	budget: 2bit	diverge	$36.32 \pm 2.46\%$	$91.09 \pm 0.09\%$	$91.12 \pm 0.11\%$	$91.13 \pm 0.12\%$
	extra memory (MB)	16.48	16.48	16.48	8.24	0
ResNet110	budget: 1bit	diverge	diverge	$91.24 \pm 0.21\%$	$91.80 \pm 0.27\%$	$92.97 \pm 0.23\%$
	budget: 2bit	diverge	diverge	$93.43 \pm 0.12\%$	$92.96 \pm 0.17\%$	$93.47 \pm 0.18\%$
	extra memory (MB)	103.68	103.68	103.68	51.84	0

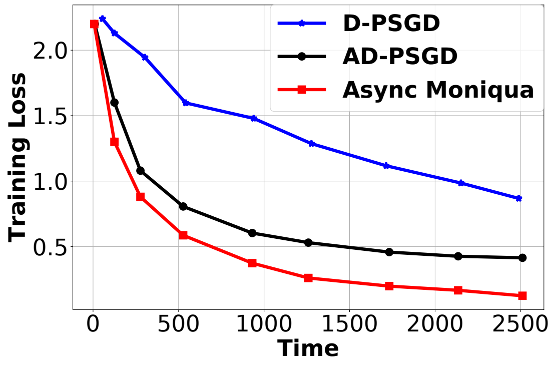
PSGD suffer the most, since they require a large volume of high-precision exchanged data. And from Figure 1(b) to Figure 1(c), when the network latency increases, AllReduce is severely delayed since it needs to transfer large volume of messages (such as handshakes between hosts to send data). On the other hand, from Figure 1(a) to Figure 1(b) and Figure 1(c), curves of all the quantized baselines (DCD/ECD-PSGD, ChocoSGD and DeepSqueeze) are getting closer to Moniqua. This is because, as shown in Figure 1(a), the extra updating of the replicas in DCD/ECD-PSGD and ChocoSGD as well as the error tracking in DeepSqueeze counteract the benefits from accelerated communication. However, when network bandwidth decreases or latency increases, communication becomes the bottleneck and makes these algorithms diverge from centralized SGD and D-PSGD. Delay between Moniqua and quantized baselines does not vary with the network since that only depends on the their extra local computation (error tracking and replica update). Figure 1(d) shows an extremely poor network, and we can see that all the quantized baselines are having similar convergence speed since now network is a serious overhead.

Extremely low bit-budget. We proceed to evaluate whether Moniqua and other baselines are able to achieve state-of-the-art accuracy under extremely low bit budgets. We train two different models: ResNet20 and ResNet110 on CIFAR10. State-of-the-art results (He et al., 2016) show that ResNet20 can achieve test accuracy of 91.25% while ResNet110 can achieve 93.57%. We enforce two strict bit-budget: 1bit and 2bit (per parameter). We plot the final test accuracy under different algorithms in Table 6. We can see that DCD-PSGD and ECD-PSGD are generally not able to converge. Among all the other algorithms, Moniqua achieves slightly better test accuracy while requiring no additional memory. By comparison, ChocoSGD and DeepSqueeze are able to get close to state-of-the art accuracy, but at the cost of incurring substantial memory overhead.

Scalability. We evaluate the performance of Moniqua when applied to D^2 (Tang et al., 2018b) and AD-PSGD (Lian et al., 2017b). First, we demonstrate how applying Moniqua to D^2 can handle decentralized data. We launch 10 workers,



(a) Training Loss vs Epoch (Decentralized Data)



(b) Training Loss vs Time(s) (Asynchronous Communication)

Figure 2. Performance of applying Moniqua on D^2 and AD-PSGD

collaborating to train a VGG16 (Simonyan & Zisserman, 2014) model on CIFAR10. Similar to the setting of D^2 (Tang et al., 2018b), we let each worker have exclusive access to 1 label (of the 10 labels total in CIFAR10). In this way, the data variance among workers is maximized. We plot the results in Figure 2(a). We observe that applying Moniqua on D^2 does not affect the convergence rate while D-PSGD can no longer converge because of the outer variance. Here we omit the wall clock time comparison since the communication volume is the same in comparison of

Moniqua and Centralized algorithm in Figure 1.

Next, we evaluate Moniqua on AD-PSGD. We launch 6 workers organized in a ring topology, collaborating to train a ResNet110 model on CIFAR10. We set the network bandwidth to be 20Mbps and latency to be 0.15ms. We plot the results in Figure 2(b). We can see that both AD-PSGD and asynchronous Moniqua outperform D-PSGD. Besides, Moniqua outperforms AD-PSGD in that communication is reduced, which is aligned with the intuition and theory.

Choosing θ empirically. We can see that the θ chosen will largely affect the running of Moniqua. In practice, there are several methods to effectively tune θ. The **first** is to directly compute θ via its expression. Specifically, we could first run a few epochs and keep track of the infinity norm of the gradient and then use expression in Theorem 2 to obtain θ. Note that gradient is usually decreasing in magnitude as algorithm proceeds. In general the computed θ can be used throughout the training. The **second** method is to treat θ as a hyperparameter and use standard methods such as random search or grid search (Bergstra & Bengio, 2012) to tune θ until we find the correct θ. The **third** method is to add verification. For instance, consider using stochastic rounding with quantization step being δ. Suppose we have $x \in \mathbb{R}$ and need to send it to machine M with y. If $|x - y| < \theta$, then if we send $\mathcal{Q}_\delta(x/\delta) \bmod \theta/\delta$ to M, it will recover $\mathcal{Q}_\delta(x/\delta)$ based on y. In addition, we can also send $H(\mathcal{Q}_\delta(x/\delta))$, where H is a hash function that takes the un-modded vector. When M recovers $\mathcal{Q}_\delta(x/\delta)$, it can detect whether the thing it recovered has the correct hash. If the θ is mistakenly chosen, M will detect any errors with high probability (Al-Riyami & Paterson, 2003). Note that compared to the model parameters, the output of hash function will not cause any overhead in general.

In the experiments of previous subsections, we mainly use the first method, which is sufficient for a good θ. The second method is a standard tuning protocol, but we do not usually use it in practice. The third method is optional to further guarantee the correctness of θ with little cost. Besides, we found constant θ(s) suffice to perform well in the experiments, and thus in practice we usually do not need to modify θ in each iteration.

More efficient Moniqua. There are two techniques we have observed to improve the performance of Moniqua when using stochastic rounding: $Q_\delta(\boldsymbol{x}) = \delta\lfloor \frac{\boldsymbol{x}}{\delta} + \boldsymbol{u}\rfloor$ (where $\boldsymbol{u}$ is uniformly sampled from $[0, 1]$), $\forall \boldsymbol{x} \in \mathbb{R}^d$. The **first** is to use *shared randomness*, in which the same random seed is used for stochastic rounding on all the workers. That is, if two workers are exchanging tensors $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively, then the floored tensors $\lfloor \frac{\boldsymbol{x}}{\delta} + \boldsymbol{u}\rfloor$ and $\lfloor \frac{\boldsymbol{y}}{\delta} + \boldsymbol{u}\rfloor$ they send use the *same* randomly sampled value $\boldsymbol{u}$. This provably reduces the error due to quantization (more details are in the supplementary material). The **second** technique is to use a standard entropy compressor like `bzip` to further compress the communicated tensors. This can help further reduce the number of bits because the modulo operation in Moniqua can introduce some redundancy in the higher-order bits, which a traditional compression algorithm can easily remove.

7. Conclusions

In this paper we propose Moniqua, a simple unified method of quantizing the communication in decentralized training algorithms. Theoretically, Moniqua supports biased quantizer and non-convex problems, while enjoying the same asymptotic convergence rate as full-precision-communication algorithms without incurring storage or computation overhead. Empirically, we observe Moniqua converges faster than other related algorithms with respect to wall clock time. Additionally, Moniqua is robust to very low bits-budget.

Acknowledgement

Yucheng Lu was supported by Cornell PhD Fellowship. We thank A. Feder Cooper, Jerry Chee and anonymous reviewers from ICML 2020 for providing valuable feedback and suggestions on this paper. We thank Google Cloud Platform Research Credits program for providing computational resources.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Acharya, J., De Sa, C., Foster, D. J., and Sridharan, K. Distributed learning with sublinear communication. *arXiv preprint arXiv:1902.11259*, 2019.

Al-Riyami, S. S. and Paterson, K. G. Certificateless public key cryptography. In *International conference on the theory and application of cryptology and information security*, pp. 452–473. Springer, 2003.

Alistarh, D. A brief tutorial on distributed and concurrent machine learning. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pp. 487–488. ACM, 2018.

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*, pp. 1709–1720, 2017.

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

Assran, M., Loizou, N., Ballas, N., and Rabbat, M. Stochastic gradient push for distributed deep learning. *arXiv preprint arXiv:1811.10792*, 2018.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

De Sa, C., Feldman, M., Ré, C., and Olukotun, K. Understanding and optimizing asynchronous low-precision stochastic gradient descent. In *ACM SIGARCH Computer Architecture News*, volume 45, pp. 561–574. ACM, 2017.

De Sa, C., Leszczynski, M., Zhang, J., Marzoev, A., Aberger, C. R., Olukotun, K., and Ré, C. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012.

Doan, T. T., Maguluri, S. T., and Romberg, J. On the convergence of distributed subgradient methods under quantization. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 567–574. IEEE, 2018.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Gropp, W., Thakur, R., and Lusk, E. *Using MPI-2: Advanced features of the message passing interface*. MIT press, 1999.

Grubic, D., Tam, L., Alistarh, D., and Zhang, C. Synchronous multi-gpu deep learning with low-precision communication: An experimental study. *Proceedings of the EDBT 2018*, 2018.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pp. 1737–1746, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

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

He, L., Bian, A., and Jaggi, M. Cola: Decentralized linear learning. In *Advances in Neural Information Processing Systems*, pp. 4541–4551, 2018.

Hendrikx, H., Massoulié, L., and Bach, F. Accelerated decentralized optimization with local updates for smooth and strongly convex objectives. *arXiv preprint arXiv:1810.02660*, 2018.

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

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

Krizhevsky, A., Nair, V., and Hinton, G. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

Lan, G., Lee, S., and Zhou, Y. Communication-efficient algorithms for decentralized and stochastic optimization. *arXiv preprint arXiv:1701.03961*, 2017.

Levin, D. A. and Peres, Y. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Li, H., De, S., Xu, Z., Studer, C., Samet, H., and Goldstein, T. Training quantized nets: A deeper understanding. In *Advances in Neural Information Processing Systems*, pp. 5811–5821, 2017.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pp. 583–598, 2014a.

Li, M., Andersen, D. G., Smola, A. J., and Yu, K. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2014b.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. 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, 2017a.

Lian, X., Zhang, W., Zhang, C., and Liu, J. Asynchronous decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1710.06952*, 2017b.

Mokhtari, A. and Ribeiro, A. Decentralized double stochastic averaging gradient. In *Signals, Systems and Computers, 2015 49th Asilomar Conference on*, pp. 406–410. IEEE, 2015.

Nazari, P., Tarzanagh, D. A., and Michailidis, G. Dadam: A consensus-based distributed adaptive gradient method for online optimization. *arXiv preprint arXiv:1901.09109*, 2019.

Reisizadeh, A., Mokhtari, A., Hassani, S. H., and Pedarsani, R. Quantized decentralized consensus optimization. *CoRR*, abs/1806.11536, 2018. URL http://arxiv.org/abs/1806.11536.

Seide, F. and Agarwal, A. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2135–2135. ACM, 2016.

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 *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

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

Sirb, B. and Ye, X. Consensus optimization with delayed and stochastic gradients on decentralized networks. In *Big Data (Big Data), 2016 IEEE International Conference on*, pp. 76–85. IEEE, 2016.

Stich, S. U. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pp. 4452–4463, 2018.

Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, H. B. Distributed mean estimation with limited communication. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3329–3337. JMLR.org, 2017.

Tang, H., Gan, S., Zhang, C., Zhang, T., and Liu, J. Communication compression for decentralized training. In *Advances in Neural Information Processing Systems*, pp. 7663–7673, 2018a.

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. D2: Decentralized training over decentralized data. *arXiv preprint arXiv:1803.07068*, 2018b.

Tang, H., Yu, C., Renggli, C., Kassing, S., Singla, A., Alistarh, D., Liu, J., and Zhang, C. Distributed learning over unreliable networks. *arXiv preprint arXiv:1810.07766*, 2018c.

Tang, H., Lian, X., Qiu, S., Yuan, L., Zhang, C., Zhang, T., and Liu, J. Deepsqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1907.07346*, 2019.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems*, pp. 9850–9861, 2018.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1306–1316, 2018.

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*, pp. 1509–1519, 2017.

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

Wu, T., Yuan, K., Ling, Q., Yin, W., and Sayed, A. H. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 4(2):293–307, 2018b.

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*, pp. 4035–4043, 2017.

Zhang, J. and You, K. Asynchronous decentralized optimization in directed networks. *arXiv preprint arXiv:1901.08215*, 2019.

Zhang, T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 116. ACM, 2004.