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 communicate via exchanging gradients (Alistarh et al., 2017; Seide et al., 2014; Doan et al., 2018; Zhang et al., 2017; Wang 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 2 for brevity, in this paper we generally refer to lossy compression methods including quantization, sparsification, etc, as “quantization.”


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

2For 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 modular arithmetic for communication quantization 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 $i$th 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 $x \mod 2\theta$ rather than $x$ from $w_x$. Note that $x \mod 2\theta$ 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 $a \in \mathbb{R}^+$,

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

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

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

**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...
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. 1 bit 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 \( Q_{\delta} \) that has bounded error

$$
\| Q_{\delta}(x) - x \|_\infty \leq \delta \quad \text{when} \quad x \in \left[ -\frac{1}{2}, \frac{1}{2} \right]^d
$$

where \( \delta \) is some constant. Note that in this assumption, we do not assume any bound for \( x \) 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 \( \delta \) denotes more fine-grained quantization requiring more bits. For example, a biased linear quantizer can achieve (2) by rounding any coordinate of \( x \) to the nearest number in the set \( \{ 2\delta n \mid n \in \mathbb{Z} \} \); this will require about \( \delta^{-1} \) 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 \( x \in \mathbb{R}^d \) and a local dataset \( D_i \) and collaborates to optimize

$$
\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^{n} E_{\xi \sim D_i} f_i(x; \xi).
$$

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

$$
x_{k+1,i} = x_{k,i} - \alpha_k \tilde{g}_{k,i}
$$

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

---

### 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).

<table>
<thead>
<tr>
<th></th>
<th>DCD-PSGD</th>
<th>ECD-PSGD</th>
<th>ChocoSGD</th>
<th>DeepSqueeze</th>
<th>Moniqua</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports biased quantizers</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports 1-bit quantization</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Works beyond D-PSGD</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Non-constant Step Size</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Additional Memory</td>
<td>( \Theta(md) )</td>
<td>( \Theta(md) )</td>
<td>( \Theta(md) )</td>
<td>( \Theta(nd) )</td>
<td>0</td>
</tr>
</tbody>
</table>
approach with quantizer $Q_δ$ can be represented by
\[ x_{k+1,i} = x_{k,i} W_{ii} + \sum_{j \neq i} Q_δ(x_{k,j}) W_{ji} - \alpha_k \hat{g}_{k,i} \tag{4} \]

Based on Equation 4, we obtain the following theorem.

**Theorem 1.** For some constant $δ$, suppose that we use an unbiased linear quantizer $Q_δ$ with representable points \( \{δn \mid n \in \mathbb{Z} \} \) to learn on the quadratic objective function \( f(x) = (x - δ/2)^T (x - δ/2)/2 \) with the direct quantization approach (4). Let $φ$ denote the smallest value of a non-zero entry in $W$. Regardless of what step size we adopt, it will always hold for all iterations $k$ and local model indices $i$ that $\mathbb{E} \|\nabla f(x_{k,i})\|^2 \geq \frac{φ^2 \delta^2}{8(1 + φ^2)}$. 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 $Q_δ$. Consider the two-scaler example from Section 1. Suppose we know $y$ and $|x - y| < \theta$ and need to fetch $x$ from a remote host via a quantizer $Q_δ$ to recover $x$. We’ve shown in Section 3 that fetching and using $Q_δ(x)$ leads to divergence. Instead, we define a parameter $B_θ = (2\theta)/(1 - 2\delta)$ and then use the modulo operation and fetch $Q_δ((x/B_θ) \mod 1)$ from the remote host, from which we can approximately recover $x$ as
\[ \hat{x} = (B_θ Q_δ((x/B_θ) \mod 1)) - y \mod B_θ + y \tag{5} \]

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 $\theta$, as shown in the following lemma.

**Lemma 2.** For any scalars $x, y \in \mathbb{R}$, if $|x - y| < \theta$ and if $δ < \frac{1}{2}$, then if we set $B_θ = (2\theta)/(1 - 2δ)$ and $\hat{x}$ as in (5),
\[ |\hat{x} - x| \leq δ B_θ = θ \cdot (2δ)/(1 - 2δ). \]

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 \( \{|g_{k,i}| \}_{k \geq 0} \), communication matrix $W$, number of iterations $K$, quantizer $Q_δ$, neighbor list $N_i$

1: for $k = 0, 1, 2, \cdots, K - 1$
2: Compute a local stochastic gradient $\hat{g}_{k,i}$ with data sample $\xi_{k,i}$ and current weight $x_{k,i}$
3: Send modulo-ed model to neighbors:
   \[ q_{k,i} = Q_δ((x_{k,i}/B_θ) \mod 1) \]
4: Compute local biased term $\hat{x}_{k,i}$ as:
   \[ \hat{x}_{k,i} = q_{k,i} B_θ - x_{k,i} \mod B_θ + x_{k,i} \]
5: Recover model received from worker $j$ as:
   \[ \hat{x}_{k,j} = (q_{k,j} B_θ - x_{k,j}) \mod B_θ + x_{k,j} \]
6: Average with neighboring workers:
   \[ x_{k+\frac{1}{2},i} = x_{k,i} + \sum_{j \in N_i} (\hat{x}_{k,j} - \hat{x}_{k,i}) W_{ji} \]
7: Update the local weight with local gradient:
   \[ x_{k+1,i} = x_{k+\frac{1}{2},i} - \alpha_k \hat{g}_{k,i} \]
8: end for
9: return averaged model $\hat{X}_K = \frac{1}{n} \sum_{i=1}^{n} x_{K,i}$

We now proceed to analyze the convergence rate of Algorithm 1. (Note that all the division and mod operations in Algorithm 1 act element-wise.)

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 an 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.
\[ \|\nabla f_i(x) - \nabla f_i(y)\| \leq L \|x - y\|, \forall x, y \in \mathbb{R}^d \]

(A2) **Spectral gap.** The communication matrix $W$ is a symmetric doubly stochastic matrix and
\[ \max \{λ_2(W), |λ_n(W)|\} = \rho < 1, \]
where $λ_i(W)$ denotes the the $i$th largest eigenvalue of $W$.

(A3) **Bounded variance.** There exist non-negative con-
Theorem 2. The convergence rate is not negatively impacted by the quantization.

Corollary 1. If we adopt a step size scheme where $\alpha_k = \frac{2\alpha G_\infty C_\alpha \log(16n)}{1-\rho}$ and $\delta \leq \frac{8G_\infty^2 \eta \log(16n) + 2(1-\eta \rho)}{4\alpha \rho}$, then Algorithm 1 converges at the following rate:

$$\sum_{k=0}^{K-1} \left( \frac{E}{n} \| \nabla f(x_k) \|^2 \right)^2 \leq 4E f(0) - E f^* + 2\alpha^2 \sum_{k=0}^{K-1} \alpha_k^2 + \frac{8(\sigma^2 + 3\gamma^2)L^2}{(1-\rho)^2} \sum_{k=0}^{K-1} \alpha_k^2 + \frac{8G_\infty^2 d L^2}{(1-\rho)^2 C_\gamma^2} \sum_{k=0}^{K-1} \alpha_k^2$$

where $f^* = \inf_x f(x)$.

Theorem 2 shows that the priori bound $\theta_k$ 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 $\alpha_k = \frac{2\alpha G_\infty C_\alpha \log(16n)}{1-\rho}$ in Theorem 2, then the output of Algorithm 1 converges at the asymptotic rate

$$\frac{1}{K} \sum_{k=0}^{K-1} E \| \nabla f(x_k) \|^2 \leq \frac{\sigma^2}{K} + \frac{\gamma^2}{K^4} + \frac{(\sigma^2 + G_\infty^2 d)n}{\sigma^2 K + n}.$$

Consistent with D-PSGD. Note that D-PSGD converges at the asymptotic rate of $O(\sigma/\sqrt{nK} + \gamma^2/K^4 + n/K)$, 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 $l_\infty$-norm to bound sample variance and gradient magnitude, respectively. Note that, when $d$ gets larger, the variance $\sigma^2$ 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 ($Q_\delta$). If we use nearest neighbor rounding (Gupta et al., 2015) with a linear quantizer as $Q_\delta$ in Theorem 2, it suffices to use at each step a number of bits $B$ for each parameter sent, where

$$B \leq \left\lfloor \log_2 \left( \frac{1}{2 \rho} + 1 \right) \right\rfloor = \left\lfloor \log_2 \left( \frac{4 \log_2(16n) + 3}{1-\rho} \right) \right\rfloor.$$

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 $W$ 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 $\bar{W} = \gamma W + (1 - \gamma) I$ and tune $\gamma$ as a hyperparameter. We formalize this result in the following Theorem.

Theorem 3. Consider using a communication matrix in the form of $\bar{W} = \gamma W + (1 - \gamma) I$. If we set $\theta = \frac{2\alpha G_\infty \log(16n)}{\gamma(1-\rho)}$, $\gamma = \frac{1}{\gamma(1-\rho) + \frac{1}{2} \log \log(n) \log(K)}$, and $\alpha = \frac{\gamma}{\gamma(1-\rho) + \frac{1}{2} \log \log(n) \log(K)}$, then the output of Algorithm 1 converges at the asymptotic rate

$$\frac{1}{K} \sum_{k=0}^{K-1} E \| \nabla f(x_k) \|^2 \leq \frac{\sigma^2}{K} + \frac{\gamma^2}{K^4} + \frac{\sigma^2 n \delta^2 \log^2(n) \log^2(K)}{(\sigma^2 K + n)(1 - 2\delta)^4} + \frac{n \delta^6 \log^4(n) \log^2(K)}{(\sigma^2 K + n)(1 - 2\delta)^6}.$$

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 allows 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...
Moniqua: Modulo Quantized Communication in Decentralized SGD

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 $D_i$ are not identically distributed (Tang et al., 2018b). More explicitly, the outer variance $E_{x \sim \{1, \ldots, n\}} \| \nabla f_i(x) - \nabla f(x) \|^2$ is no longer bounded by $\varsigma^2$ 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 reducing the variance over time. Applying Moniqua on $D^2$ can be explicitly expressed as:

$$X_{k+1} = X_k + \alpha_k \tilde{G} + \tilde{X}_{k+1} W + (\tilde{X}_{k+1} - \tilde{X}_{k+1})(W - I)$$

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

**Theorem 4.** If we apply Moniqua on $D^2$ in a setting where $\theta = 6D_1n + 8)\alpha G / (\delta K n) - \alpha G / (\delta K n)$ and $\alpha_k = \alpha = \frac{1}{\sqrt{n}}$ where $D_1$ and $D_2$ are two constants, applying Moniqua on $D^2$ has the following asymptotic convergence rate:

$$\frac{1}{K} \sum_{k=0}^{K-1} E \| \nabla f(X_k) \|^2 \leq 1 + \frac{\sigma^2 + G^2 d}{\alpha K n}.$$ 

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

Moniqua with Asynchronous 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 as:

$$X_{k+1} = X_k W_k + (\tilde{X}_k - X_k)(W_k - I) - \alpha_k \tilde{G}_k - \tau_k$$

where $W_k$ describes the communication behaviour between the $k$th and $(k + 1)$th gradient update, and $\tau_k$ 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 $W_k$ 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 $k$, $\forall \mu \in \mathbb{R}^n$, if $e_k^\top x_{k,i} \geq 0$ and $1^\top \mu = 1$, it must hold that $\left\| \left( \prod_{k=1}^{t_{mix}} W_k + 1 \right) \mu - \frac{1}{n} \right\| \leq \frac{1}{L}$. We call this constant $t_{mix}$ because it is effectively the mixing time of the time-inhomogeneous Markov chain with transition probability matrix $W_k$ 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 $W_k$ 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 $\theta = 16t_{mix} \alpha G / (\delta K n) - \alpha G / (\delta K n)$ and $\alpha_k = \alpha = \frac{1}{2L + \sqrt{K(\sigma^2 + 6\varsigma^4)}}$, applying Moniqua on AD-PSGD has the following asymptotic convergence rate:

$$\frac{1}{K} \sum_{k=0}^{K-1} E \| \nabla f(X_k) \|^2 \leq \frac{1}{K} + \frac{\sqrt{\sigma^2 + 6\varsigma^2}}{\sqrt{K}} + \frac{(\sigma^2 + 6\varsigma^2) t_{max} n^2}{K + 1}.$$

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-
Moniqua: Modulo Quantized Communication in Decentralized SGD

---

**Figure 1.** Performance of different algorithms under different network configurations

We investigate the 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. 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: \{Momentum = 0.9, Weight Decay = \( 5e^{-4} \), Batch Size = 128, Initial Step Size = 0.1, \( \theta_k = 2.0 \)\}. In the extreme-bit-budget experiment, we further use adopt the average ratio \( \gamma = 5e^{-3} \).

**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.

---

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

<table>
<thead>
<tr>
<th></th>
<th>DCD-PSGD</th>
<th>ECD-PSGD</th>
<th>ChocoSGD</th>
<th>DeepSqueeze</th>
<th>Moniqua</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet20</td>
<td>budget: 1bit diverge</td>
<td>diverge</td>
<td>90.88 ± 0.13%</td>
<td>90.02 ± 0.22%</td>
<td>91.08 ± 0.19%</td>
</tr>
<tr>
<td></td>
<td>budget: 2bit diverge</td>
<td>36.32 ± 2.46%</td>
<td>91.09 ± 0.09%</td>
<td>91.12 ± 0.11%</td>
<td>91.13 ± 0.12%</td>
</tr>
<tr>
<td></td>
<td>extra memory (MB) 16.48</td>
<td>16.48</td>
<td>16.48</td>
<td>8.24</td>
<td>0</td>
</tr>
<tr>
<td>ResNet110</td>
<td>budget: 1bit diverge</td>
<td>diverge</td>
<td>91.24 ± 0.21%</td>
<td>91.80 ± 0.27%</td>
<td>92.97 ± 0.23%</td>
</tr>
<tr>
<td></td>
<td>budget: 2bit diverge</td>
<td>diverge</td>
<td>93.43 ± 0.12%</td>
<td>92.96 ± 0.17%</td>
<td>93.47 ± 0.18%</td>
</tr>
<tr>
<td></td>
<td>extra memory (MB) 103.68</td>
<td>103.68</td>
<td>103.68</td>
<td>51.84</td>
<td>0</td>
</tr>
</tbody>
</table>

Moniqua suffers 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 93.50%. 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, 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 $\theta$ empirically. We can see that the $\theta$ chosen will largely affect the running of Moniqua. In practice, there are several methods to effectively tune $\theta$. The first is to directly compute $\theta$ 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 $\theta$. Note that gradient is usually decreasing in magnitude as algorithm proceeds. In general the computed $\theta$ can be used throughout the training. The second method is to treat $\theta$ as a hyperparameter and use standard methods such as random search or grid search (Bergstra & Bengio, 2012) to tune $\theta$ until we find the correct $\theta$. The third method is to add verification. For instance, consider using stochastic rounding with quantization step being $\delta$. 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 $Q_\delta(x/\delta)$ mod $\theta/\delta$ to $M$, it will recover $Q_\delta(x/\delta)$ based on $y$. In addition, we can also send $H(Q_\delta(x/\delta))$, where $H$ is a hash function that takes the un-modded vector. When $M$ recovers $Q_\delta(x/\delta)$, it can detect whether the thing it recovered has the correct hash. If the $\theta$ 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 $\theta$. 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 $\theta$ with little cost. Besides, we found constant $\theta(s)$ suffice to perform well in the experiments, and thus in practice we usually do not need to modify $\theta$ in each iteration.

More efficient Moniqua. There are two techniques we have observed to improve the performance of Moniqua when using stochastic rounding: $Q_\delta(x) = \delta \lfloor \frac{x}{\delta} + u \rfloor$ (where $u$ is uniformly sampled from $[0, 1]$), $\forall x \in \mathbb{R}^d$. 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 $x$ and $y$ respectively, then the floored tensors $\lfloor \frac{x}{\delta} + u \rfloor$ and $\lfloor \frac{y}{\delta} + u \rfloor$ they send use the same randomly sampled value $u$. 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


Alistarh, D., Hoeffler, T., Johansson, M., Konstantinov, N., Khirirat, S., and Renggli, C. The convergence of sparsi-


