Journal of Machine Learning Research 22 (2021) 1-43 Submitted 3/20; Revised 1/21; Published 4/21

NUQSGD: Provably Communication-efficient Data-parallel SGD via Nonuniform Quantization

Ali Ramezani-Kebrya* ALI.RAMEZANI@EPFL.CH
École Polytechnique Fédérale de Lausanne
Route Cantonale, 1015 Lausanne, Switzerland

Fartash Faghri FAGHRI@CS.TORONTO.EDU
Dept. of Computer Science, Univ. of Toronto and Vector Institute
Toronto, ON M5T 3A1, Canada

Ilya Markov ILIA.MARKOV@IST.AC.AT
Institute of Science and Technology Austria
3400 Klosterneuburg, Austria

Vitalii Aksenov VITALII.AKSENOV@IST.AC.AT
Institute of Science and Technology Austria
3400 Klosterneuburg, Austria

Dan Alistarh DAN.ALISTARH@IST.AC.AT
Institute of Science and Technology Austria
3400 Klosterneuburg, Austria

Daniel M. Roy DANIEL.ROY@UTORONTO.CA
Dept. of Statistical Sciences, Univ. of Toronto and Vector Institute
Toronto, ON M5S 3G3, Canada

Editor: Sathiya Keerthi

Abstract

As the size and complexity of models and datasets grow, so does the need for communication-efficient variants of stochastic gradient descent that can be deployed to perform parallel model training. One popular communication-compression method for data-parallel SGD is QSGD (Alistarh et al., 2017), which quantizes and encodes gradients to reduce communication costs. The baseline variant of QSGD provides strong theoretical guarantees, however, for practical purposes, the authors proposed a heuristic variant which we call QSGDinf, which demonstrated impressive empirical gains for distributed training of large neural networks. In this paper, we build on this work to propose a new gradient quantization scheme, and show that it has both stronger theoretical guarantees than QSGD, and matches and exceeds the empirical performance of the QSGDinf heuristic and of other compression methods.

Keywords: Communication-efficient SGD, Quantization, Gradient Compression, Data-parallel SGD, Deep Learning

∗. Work performed at the Vector Institute and University of Toronto.

©2021 Ali Ramezani-Kebrya, Fartash Faghri, Ilya Markov, Vitalii Aksenov, Dan Alistarh, and Daniel M. Roy.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v22/20-255.html.

1. Introduction

Deep learning is booming thanks to enormous datasets and very large models, leading to the fact that the largest datasets and models can no longer be trained on a single machine. One common solution to this problem is to use distributed systems for training. The most common algorithms underlying deep learning are stochastic gradient descent (SGD) and its variants, which led to a significant amount of research on building and understanding distributed versions of SGD.

Implementations of SGD on distributed systems and data-parallel versions of SGD are scalable and take advantage of multi-GPU systems. Data-parallel SGD, in particular, has received significant attention due to its excellent scalability properties (Zinkevich et al., 2010; Bekkerman et al., 2011; Recht et al., 2011; Dean et al., 2012; Coates et al., 2013; Chilimbi et al., 2014; Li et al., 2014; Duchi et al., 2015; Xing et al., 2015; Zhang et al., 2015; Alistarh et al., 2017). In data-parallel SGD, a large dataset is partitioned among K processors. These processors work together to minimize an objective function. Each processor has access to the current parameter vector of the model. At each SGD iteration, each processor computes an updated stochastic gradient using its own local data. It then shares the gradient update with its peers. The processors collect and aggregate stochastic gradients to compute the updated parameter vector.

Increasing the number of processing machines reduces the computational costs significantly. However, the communication costs to share and synchronize huge gradient vectors and parameters increases dramatically as the size of the distributed systems grows. Communication costs may thwart the anticipated benefits of reducing computational costs. Indeed, in practical scenarios, the communication time required to share stochastic gradients and parameters is the main performance bottleneck (Recht et al., 2011; Li et al., 2014; Seide et al., 2014; Strom, 2015; Alistarh et al., 2017). Reducing communication costs in data-parallel SGD is an important problem.

One promising solution to the problem of reducing communication costs of data-parallel SGD is gradient compression, *e.g.,* through gradient quantization (Dean et al., 2012; Seide et al., 2014; Sa et al., 2015; Gupta et al., 2015; Abadi et al., 2016; Zhou et al., 2018; Alistarh et al., 2017; Wen et al., 2017; Bernstein et al., 2018). (This should not be confused with weight quantization/sparsification, as studied by Wen et al. (2016); Hubara et al. (2016); Park et al. (2017); Wen et al. (2017), which we do not discuss here.) Unlike full-precision data-parallel SGD, where each processor is required to broadcast its local gradient in full-precision, *i.e.,* transmit and receive huge full-precision vectors at each iteration, quantization requires each processor to transmit only a few communication bits per iteration for each component of the stochastic gradient.

One popular such proposal for communication-compression is quantized SGD (QSGD), due to Alistarh et al. (2017). In QSGD, stochastic gradient vectors are normalized to have unit L^2 norm, and then compressed by quantizing each element to a uniform grid of quantization levels using a randomized method. While most lossy compression schemes do not provide convergence guarantees, QSGD's quantization scheme is designed to be unbiased, which implies that the quantized stochastic gradient is itself a stochastic gradient, only with higher variance determined by the dimension and number of quantization levels. As a result, a number of theoretical guarantees are established for QSGD, including that it converges

under standard assumptions. By changing the number of quantization levels, QSGD allows the user to trade-off communication bandwidth and convergence time.

Despite their theoretical guarantees based on quantizing after L^2 normalization, Alistarh et al. opt to present empirical results using L^∞ normalization. We call this variation QSGDinf. While the empirical performance of QSGDinf is strong, their theoretical guarantees on the number of bits transmitted no longer apply. Indeed, in our own empirical evaluation of QSGD, we find the variance induced by quantization is substantial, and the performance is far from that of SGD and QSGDinf.

Given the popularity of this scheme, it is natural to ask one can obtain guarantees as strong as those of QSGD while matching the practical performance of the QSGDinf heuristic. In this work, we answer this question in the affirmative by providing a new quantization scheme which fits into QSGD in a way that allows us to establish stronger theoretical guarantees on the variance, bandwidth, and cost to achieve a prescribed gap. Instead of QSGD's uniform quantization scheme, we use an unbiased nonuniform logarithmic scheme, reminiscent of those introduced in telephony systems for audio compression (Cattermole, 1969). We call the resulting algorithm *nonuniformly quantized stochastic gradient descent* (NUQSGD). Like QSGD, NUQSGD is a quantized data-parallel SGD algorithm with strong theoretical guarantees that allows the user to trade off communication costs with convergence speed. Unlike QSGD, NUQSGD has strong empirical performance on deep models and large datasets, matching that of QSGDinf. Beyond just the stronger theoretical guarantees, NUQSGD also allows us to employ non-trivial coding to the quantized gradients, as its code-length guarantees also hold in practice. Specifically, we provide a new efficient implementation for these schemes using a modern computational framework (Pytorch), and benchmark it on classic large-scale image classification tasks. Results showcase the practical performance of NUQSGD, which can surpass that of QSGDinf and of SignSGD with Error Feedback (EF) (Karimireddy et al., 2019) when employing gradient coding, both in terms of communication-compression and end-to-end training time.

The intuition behind the nonuniform quantization scheme underlying NUQSGD is that, after L^2 normalization, many elements of the normalized stochastic gradient will be near-zero. By concentrating quantization levels near zero, we are able to establish stronger bounds on the excess variance. These bounds decrease rapidly as the number of quantization levels increases. In fact, we provide a lower bound showing that our variance bound is *tight* w.r.t. the model dimension. We establish convergence guarantees for NUQSGD under standard assumptions for convex and nonconvex problems. Given the importance of momentum-based methods, we establish convergence guarantees for communication-efficient variants of SGD with momentum. We have derived a tight worst-case variance upper bound for a fixed set of arbitrary levels, expressed as the solution to an integer program with quadratic constraints. We can relax the program to obtain a quadratic program. A coarser analysis yields an upper bound expressed as the solution to a linear program, which is more amenable to analysis. Finally, by combining the variance bound with a bound on the expected code-length, we obtain a bound on the total communication costs of achieving an expected suboptimality gap. The resulting bound is stronger than the one provided by QSGD.

To study how quantization affects convergence on state-of-the-art deep models, we compare NUQSGD, QSGD, QSGDinf, and EF-SignSGD focusing on training loss, variance, and test accuracy on standard deep models and large datasets. Using the same number of

bits per iteration, experimental results show that NUQSGD has smaller variance than QSGD, as predicted. The smaller variance also translates to improved optimization performance, in terms of both training loss and test accuracy. We also observe that NUQSGD matches the performance of QSGDinf in terms of variance and loss/accuracy. Further, our distributed implementation shows that the resulting algorithm considerably reduces communication cost of distributed training, without adversely impacting accuracy. Our empirical results show that NUQSGD can provide faster end-to-end parallel training relative to data-parallel SGD, QSGD, and EF-SignSGD on the ImageNet dataset (Deng et al., 2009), in particular when combined with non-trivial coding of the quantized gradients.

1.1 Summary of Contributions

- We propose a non-uniform gradient quantization method and establish strong theoretical guarantees for its excess variance and communication costs. These bounds are strictly stronger than those known for QSGD. In addition, we establish a lower bound on the variance that shows our bound is tight.

- We proceed to establish stronger convergence guarantees for NUQSGD for convex and nonconvex problems, under standard assumptions. We establish convergence guarantees for communication-efficient variants of SGD with momentum.

- For generally spaced levels, we derive tight worst-case variance upper bounds expressed as an integer quadratic program and present several relaxations of this bound.

- We demonstrate that NUQSGD has strong empirical performance on deep models and large datasets, both in terms of accuracy and scalability. Thus, NUQSGD closes the gap between the theoretical guarantees of QSGD and the empirical performance of QSGDinf.

1.2 Related Work

Seide et al. (2014) proposed SignSGD, an efficient heuristic scheme to reduce communication costs drastically by quantizing each gradient component to two values. (This scheme is sometimes also called 1bitSGD (Seide et al., 2014).) Bernstein et al. (2018) later provided convergence guarantees for a variant of SignSGD. Note that the quantization employed by SignSGD is not unbiased, and so a new analysis was required. As the number of levels is fixed, SignSGD does not provide any trade-off between communication costs and convergence speed. Karimireddy et al. (2019) proposed EF-SignSGD, which is an improved version of SignSGD. We compare NUQSGD and EF-SignSGD empirically.

Sa et al. (2015) introduced Buckwild!, a lossy compressed SGD with convergence guarantees. The authors provided bounds on the error probability of SGD, assuming convexity, gradient sparsity, and unbiased quantizers.

Wen et al. (2017) proposed TernGrad, a stochastic quantization scheme with three levels. TernGrad also significantly reduces communication costs and obtains reasonable accuracy with a small degradation to performance compared to full-precision SGD. TernGrad can be viewed as a special case of QSGDinf with three quantization levels.

NUQSGD uses a logarithmic quantization scheme.[1] Such schemes have long been used, e.g. in telephony systems for audio compression (Cattermole, 1969). Logarithmic quantization schemes have appeared in other contexts recently: Hou and Kwok (2018) studied weight distributions of long short-term memory networks and proposed to use logarithm quantization for network compression. Miyashita et al. (2016) and Lee et al. (2017) proposed logarithmic encodings to represent weights and activations. Li and Sa (2019) obtained dimension-free bounds for logarithmic quantization of weights. Zhang et al. (2017) proposed a gradient compression scheme and introduced an optimal quantization scheme, but for the setting where the points to be quantized are known in advance. As a result, their scheme is not applicable to the communication setting of quantized data-parallel SGD.

2. Preliminaries: Data-parallel SGD and Convergence

We consider a high-dimensional machine learning model, parametrized by a vector $\mathbf{w} \in \mathbb{R}^d$. Let $\Omega \subseteq \mathbb{R}^d$ denote a closed and convex set. Our objective is to minimize $f : \Omega \to \mathbb{R}$, which is an unknown, differentiable, and β-smooth function. The following summary is based on (Alistarh et al., 2017).

Recall that a function f is **β-smooth** if, for all $\mathbf{u}, \mathbf{v} \in \Omega$, we have $\|\nabla f(\mathbf{u}) - \nabla f(\mathbf{v})\| \leq \beta\|\mathbf{u} - \mathbf{v}\|$, where $\|\cdot\|$ denotes the Euclidean norm. Let $(\mathcal{S}, \Sigma, \mu)$ be a probability space (and let $\mathbb{E}$ denote expectation). Assume we have access to stochastic gradients of f, *i.e.,* we have access to a function $g : \Omega \times \mathcal{S} \to \mathbb{R}^d$ such that, if $s \sim \mu$, then $\mathbb{E}[g(\mathbf{w}, s)] = \nabla f(\mathbf{w})$ for all $\mathbf{w} \in \Omega$. In the rest of the paper, we let $g(\mathbf{w})$ denote the stochastic gradient for notational simplicity. The update rule for conventional full-precision projected SGD is $\mathbf{w}_{t+1} = \mathbf{P}_\Omega(\mathbf{w}_t - \alpha g(\mathbf{w}_t))$, where $\mathbf{w}_t$ is the current parameter input, α is the learning rate, and $\mathbf{P}_\Omega$ is the Euclidean projection onto Ω.

We say the stochastic gradient has a **second-moment upper bound** B when $\mathbb{E}[\|g(\mathbf{w})\|^2] \leq B$ for all $\mathbf{w} \in \Omega$. Similarly, the stochastic gradient has a **variance upper bound** σ^2 when $\mathbb{E}[\|g(\mathbf{w}) - \nabla f(\mathbf{w})\|^2] \leq \sigma^2$ for all $\mathbf{w} \in \Omega$. Note that a second-moment upper bound implies a variance upper bound, because the stochastic gradient is unbiased.

We have classical convergence guarantees for conventional full-precision SGD given access to stochastic gradients at each iteration:

Theorem 1 (Bubeck 2015, Theorem 6.3) *Let $f : \Omega \to \mathbb{R}$ denote a convex and β-smooth function and let $R^2 \triangleq \sup_{\mathbf{w}\in\Omega} \|\mathbf{w} - \mathbf{w}_0\|^2$. Suppose that the projected SGD update is executed for T iterations with $\alpha = 1/(\beta + 1/\gamma)$ where $\gamma = R\sqrt{1/T}/\sigma$. Given repeated and independent access to stochastic gradients with a variance upper bound σ^2, projected SGD satisfies*

$$\mathbb{E}\Big[f\big(\frac{1}{T}\sum_{t=0}^{T}\mathbf{w}_t\big)\Big] - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq R\sqrt{\frac{\sigma^2}{T}} + \frac{\beta R^2}{2T}. \tag{1}$$

Following (Alistarh et al., 2017), we consider data-parallel SGD, a synchronous distributed framework consisting of K processors that partition a large dataset among themselves. This

1. After the completion of this work, we became aware of earlier, independent work by Horváth et al. (2019), which introduces gradient quantization to exponentially spaced levels (powers of 1/2). They devise variance bounds for L^p normalization. We obtain tighter variance bound for L^2 normalization, and extend our consideration to any arbitrary sequence of levels beyond powers of 1/2.

Input: local data, local copy of the parameter vector $\mathbf{w}_t$, learning rate α, and K

for $t = 1$ ***to*** T **do**
  for $i = 1$ ***to*** K **do** // each transmitter processor (in parallel)
    Compute $g_i(\mathbf{w}_t)$; // stochastic gradient
    Encode $c_{i,t} \leftarrow \text{ENCODE}\big(g_i(\mathbf{w}_t)\big)$;
    Broadcast $c_{i,t}$ to all processors;
  for $l = 1$ ***to*** K **do** // each receiver processor (in parallel)
    for $i = 1$ ***to*** K **do** // each transmitter processor
      Receive $c_{i,t}$ from processor i for each i;
      Decode $\hat{g}_i(\mathbf{w}_t) \leftarrow \text{DECODE}\big(c_{i,t}\big)$;
    Aggregate $\mathbf{w}_{t+1} \leftarrow \mathbf{P}_\Omega(\mathbf{w}_t - \frac{\alpha}{K}\sum_{i=1}^K \hat{g}_i(\mathbf{w}_t))$;

Algorithm 1: Data-parallel (synchronized) SGD.

framework models real-world systems with multiple GPU resources. Each processor keeps a local copy of the parameter vector and has access to independent and private stochastic gradients of f.

At each iteration, each processor computes its own stochastic gradient based on its local data and then broadcasts it to all peers. Each processor receives and aggregates the stochastic gradients from all peers to obtain the updated parameter vector. In detail, the update rule for full-precision data-parallel SGD is $\mathbf{w}_{t+1} = \mathbf{P}_\Omega(\mathbf{w}_t - \frac{\alpha}{K}\sum_{l=1}^K \overline{g}_l(\mathbf{w}_t))$ where $\overline{g}_l(\mathbf{w}_t)$ is the stochastic gradient computed and broadcasted by processor l. Provided that $\overline{g}_l(\mathbf{w}_t)$ is a stochastic gradient with a variance upper bound σ^2 for all l, then $\frac{1}{K}\sum_{l=1}^K \overline{g}_l(\mathbf{w}_t)$ is a stochastic gradient with a variance upper bound $\frac{\sigma^2}{K}$. Thus, aggregation improves convergence of SGD by reducing the first term of the upper bound in (1). Assume each processor computes a mini-batch gradient of size J. Then, this update rule is essentially a mini-batched update with size JK.

Data-parallel SGD is described in Algorithm 1. Full-precision data-parallel SGD is a special case of Algorithm 1 with identity encoding and decoding mappings. Otherwise, the decoded stochastic gradient $\hat{g}_i(\mathbf{w}_t)$ is likely to be different from the original local stochastic gradient $g_i(\mathbf{w}_t)$.

By Theorem 1, we have the following convergence guarantees for full-precision data-parallel SGD:

Corollary 2 (Alistarh et al. 2017, Corollary 2.2) *Let f, R, and γ be as defined in Theorem 1 and let $\epsilon > 0$. Suppose that the projected SGD update is executed for T iterations with $\alpha = 1/(\beta + \sqrt{K}/\gamma)$ on K processors, each with access to independent stochastic gradients of f with a second-moment bound B. The smallest T for the full-precision data-parallel SGD that guarantees $\mathbb{E}\big[f(\frac{1}{T}\sum_{t=0}^T \mathbf{w}_t)\big] - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \epsilon$ is $T_\epsilon = O\big(R^2 \max(\frac{B}{K\epsilon^2}, \frac{\beta}{2\epsilon})\big)$.*

3. Nonuniformly Quantized Stochastic Gradient Descent

Data-parallel SGD reduces computational costs significantly. However, the communication costs of broadcasting stochastic gradients is the main performance bottleneck in large-scale distributed systems. In order to reduce communication costs and accelerate training, Alistarh

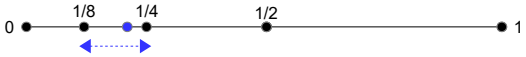

Figure 1: An example of nonuniform stochastic quantization with $s = 3$. The point between the arrows represents the value of the normalized coordinate. It will be quantized to either 1/8 or 1/4. In this case, the point is closer to 1/4, and so will be more likely to be quantized to 1/4. The probabilities are chosen so that the mean of the quantization is the unquantized coordinate's value.

et al. (2017) introduced a compression scheme that produces a compressed and unbiased stochastic gradient, suitable for use in SGD.

At each iteration of QSGD, each processor broadcasts an encoding of its own compressed stochastic gradient, decodes the stochastic gradients received from other processors, and sums all the quantized vectors to produce a stochastic gradient. In order to compress the gradients, every coordinate (with respect to the standard basis) of the stochastic gradient is normalized by the Euclidean norm of the gradient and then stochastically quantized to one of a small number quantization levels distributed uniformly in the unit interval. The stochasticity of the quantization is necessary to not introduce bias.

Alistarh et al. (2017) give a simple argument that provides a *lower* bound on the number of coordinates that are quantized to zero in expectation. Encoding these zeros efficiently provides communication savings at each iteration. However, the cost of their scheme is greatly increased variance in the gradient, and thus slower overall convergence. In order to optimize overall performance, we must balance communication savings with variance.

By simple counting arguments, the distribution of the (normalized) coordinates cannot be uniform. Indeed, this is the basis of the lower bound on the number of zeros. These arguments make no assumptions on the data distribution, and rely entirely on the fact that the quantities being quantized are the coordinates of a unit-norm vector. Uniform quantization does not capture the properties of such vectors, leading to substantial gradient variance.

3.1 Nonuniform Quantization

In this paper, we propose and study a new scheme to quantize normalized gradient vectors. Instead of uniformly distributed quantization levels, as proposed by Alistarh et al. (2017), we consider quantization levels that are nonuniformly distributed in the unit interval, as depicted in Figure 1. In order to obtain a quantized gradient that is suitable for SGD, we need the quantized gradient to remain unbiased. Alistarh et al. (2017) achieve this via a randomized quantization scheme, which can be easily generalized to the case of nonuniform quantization levels.

Using a carefully parametrized generalization of the unbiased quantization scheme introduced by Alistarh et al., we can control both the cost of communication and the variance of the gradient. Compared to a uniform quantization scheme, our scheme reduces quantization error and variance by better matching the properties of normalized vectors. In particular, by increasing the number of quantization levels near zero, we obtain a stronger

variance bound. Empirically, our scheme also better matches the distribution of normalized coordinates observed on real datasets and networks.

We now describe the nonuniform quantization scheme: Let $s \in \{1, 2, \cdots\}$ be the number of internal quantization levels, and let $\mathcal{L} = (l_0, l_1, \cdots, l_{s+1})$ denote the sequence of quantization levels, where $l_0 = 0 < l_1 < \cdots < l_{s+1} = 1$. For $r \in [0, 1]$, let $\tilde{s}(r)$ and $p(r)$ satisfy $l_{\tilde{s}(r)} \leq r \leq l_{\tilde{s}(r)+1}$ and $r = \big(1-p(r)\big)l_{\tilde{s}(r)} + p(r)l_{\tilde{s}(r)+1}$, respectively. Define $\tau(r) = l_{\tilde{s}(r)+1} - l_{\tilde{s}(r)}$. Note that $\tilde{s}(r) \in \{0, 1, \cdots, s\}$.

Definition 3 *The nonuniform quantization of a vector $\mathbf{v} \in \mathbb{R}^d$ is*

$$Q_s(\mathbf{v}) \triangleq [Q_s(v_1), \cdots, Q_s(v_d)]^T \quad \textit{where} \quad Q_s(v_i) = \|\mathbf{v}\| \cdot \text{sign}(v_i) \cdot h_i(\mathbf{v}, s) \tag{2}$$

where, letting $r_i = |v_i|/\|\mathbf{v}\|$, the $h_i(\mathbf{v}, s)$'s are independent random variables such that $h_i(\mathbf{v}, s) = l_{\tilde{s}(r_i)}$ with probability $1 - p(r_i)$ and $h_i(\mathbf{v}, s) = l_{\tilde{s}(r_i)+1}$ otherwise.

We note that the distribution of $h_i(\mathbf{v}, s)$ satisfies $\mathbb{E}[h_i(\mathbf{v}, s)] = r_i$ and achieves the minimum variance over all distributions that satisfy $\mathbb{E}[h_i(\mathbf{v}, s)] = r_i$ with support $\mathcal{L}$. We first focus on a special case of nonuniform quantization with $\hat{\mathcal{L}} = (0, 1/2^s, \cdots, 2^{s-1}/2^s, 1)$ as the quantization levels. In Section 4.1, we extend our consideration to any arbitrary sequence of levels.

The intuition behind this quantization scheme is that it is very unlikely to observe large values of r_i in the stochastic gradient vectors of machine learning models. Stochastic gradients are observed to be dense vectors (Bernstein et al., 2018). Hence, it is natural to use fine intervals for small r_i values to reduce quantization error and control the variance.

After quantizing the stochastic gradient with a small number of discrete levels, each processor must encode its local gradient into a binary string for broadcasting. We describe this encoding in Appendix A.

4. Theoretical Guarantees

In this section, we provide theoretical guarantees for NUQSGD, giving variance and code-length bounds, and using these in turn to compare NUQSGD and QSGD. Please note that the proofs of Theorems 4, 5, and 13 are provided in Appendices B, C, and D respectively.

Theorem 4 (Variance bound) *Let $\mathbf{v} \in \mathbb{R}^d$. The nonuniform quantization of $\mathbf{v}$ satisfies $\mathbb{E}[Q_s(\mathbf{v})] = \mathbf{v}$. Then we have*

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \epsilon_Q \|\mathbf{v}\|^2 \tag{3}$$

where $\epsilon_Q = (1/8 + 2^{-2s-2}d)\mathbb{1}\{d < 2^{2s+1}\} + (2^{-s}\sqrt{d} - 7/8)\mathbb{1}\{d \geq 2^{2s+1}\}$ and $\mathbb{1}$ denotes the indicator function. Provided that $s \leq \log(d)/2$, then it also holds that $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \hat{\epsilon}_Q \|\mathbf{v}\|^2$ where $\hat{\epsilon}_Q = \min\{2^{-2s}/4(d - 2^{2s}), 2^{-s}\sqrt{d - 2^{2s}}\} + O(s)$.

The result in Theorem 4 implies that if $g(\mathbf{w})$ is a stochastic gradient with a second-moment bound η, then $Q_s(g(\mathbf{w}))$ is a stochastic gradient with a variance upper bound $\epsilon_Q \eta$. Note that the variance upper bound decreases with the number of quantization levels. In the range of $s = o(\log(d))$, $\hat{\epsilon}_Q$ decreases with s, which is because the first term in the upper

bound decreases exponentially fast in s. To obtain $\hat{\epsilon}_Q$, we establish upper bounds on the number of coordinates of $\mathbf{v}$ falling into intervals defined by $\hat{\mathcal{L}}$. Our bound is tighter than the bound of Horváth et al. (2019).

Theorem 5 (Code-length bound) *Let $\mathbf{v} \in \mathbb{R}^d$. Provided d is large enough to ensure $2^{2s} + \sqrt{d}2^s \leq d/e$, the expectation $\mathbb{E}[|\text{ENCODE}(\mathbf{v})|]$ of the number of communication bits needed to transmit $Q_s(\mathbf{v})$ is bounded above by*

$$N_Q = C + 3n_{s,d} + (1+o(1))n_{s,d}\log\Big(\frac{d}{n_{s,d}}\Big) + (1+o(1))n_{s,d}\log\log\Big(\frac{8(2^{2s}+d)}{n_{s,d}}\Big) \qquad (4)$$

where $C = b - (1+o(1))$ and $n_{s,d} = 2^{2s} + 2^s\sqrt{d}$.[2]

Theorem 5 provides a bound on the expected number of communication bits to encode the quantized stochastic gradient. Note that $2^{2s} + \sqrt{d}2^s \leq d/e$ is a mild assumption in practice. As one would expect, the bound, (4), increases monotonically in d and s. In the sparse case, if we choose $s = o(\log d)$ levels, then the upper bound on the expected code-length is $O\big(2^s\sqrt{d}\log\big(\frac{\sqrt{d}}{2^s}\big)\big)$.

Combining the upper bounds above on the variance and code-length, Corollary 2 implies the following guarantees for NUQSGD:

Theorem 6 (NUQSGD for smooth convex optimization) *Let f and R be defined as in Theorem 1, let ϵ_Q be defined as in Theorem 4, let $\epsilon > 0$, $\hat{B} = (1+\epsilon_Q)B$, and let $\gamma > 0$ be given by $\gamma^2 = R^2/(\hat{B}T)$. With* ENCODE *and* DECODE *defined as in Appendix A, suppose that Algorithm 1 is executed for T iterations with a learning rate $\alpha = 1/(\beta + \sqrt{K}/\gamma)$ on K processors, each with access to independent stochastic gradients of f with a second-moment bound B. Then $T_\epsilon = O\big(\max\big(\frac{\hat{B}}{K\epsilon^2}, \frac{\beta}{2\epsilon}\big)R^2\big)$ iterations suffice to guarantee $\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=0}^{T}\mathbf{w}_t\right)\right] - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \epsilon$. In addition, NUQSGD requires at most N_Q communication bits per iteration in expectation.*

Proof Let $g(\mathbf{w})$ and $\hat{g}(\mathbf{w})$ denote the full-precision and decoded stochastic gradients, respectively. Then

$$\mathbb{E}[\|\hat{g}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2] \leq \mathbb{E}[\|g(\mathbf{w}) - \nabla f(\mathbf{w})\|^2] + \mathbb{E}[\|\hat{g}(\mathbf{w}) - g(\mathbf{w})\|^2]. \qquad (5)$$

By Theorem 4, $\mathbb{E}[\|\hat{g}(\mathbf{w}) - g(\mathbf{w})\|^2] \leq \epsilon_Q\mathbb{E}[\|g(\mathbf{w})\|^2]$. By assumption, $\mathbb{E}[\|g(\mathbf{w})\|^2] \leq B$. Noting $g(\mathbf{w})$ is unbiased, $\mathbb{E}[\|\hat{g}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2] \leq (1+\epsilon_Q)B$. The result follows by Corollary 2. ■

In the following theorem, we show that for any given set of levels, there exists a distribution of points with dimension d such that the variance is in $\Omega(\sqrt{d})$, and so our bound is tight in d.

Theorem 7 (Lower bound) *Let $d \in \mathbb{Z}^{>0}$ and let $(0, l_1, \cdots, l_s, 1)$ denote an arbitrary sequence of quantization levels. Provided $d \geq (2/l_1)^2$, there exists a vector $\mathbf{v} \in \mathbb{R}^d$ such that the variance of unbiased quantization of $\mathbf{v}$ is lower bounded by $\|\mathbf{v}\|^2 l_1\sqrt{d}/2$, i.e., the variance is in $\Omega(\sqrt{d})$.*

2. In practice, we use standard 32-bit floating point encoding, *i.e.,* $b = 32$.

Proof The variance of $Q_s(\mathbf{v})$ for general sequence of quantization levels is given by

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] = \|\mathbf{v}\|^2 \sum_{i=1}^{d} \sigma^2(r_i).$$

If $r \in [l_{\tilde{s}(r)}, l_{\tilde{s}(r)+1}]$, the variance $\sigma^2(r)$ can be expressed as

$$\sigma^2(r) = \tau(r)^2 p(r)\big(1 - p(r)\big) = (l_{\tilde{s}(r)+1} - r)(r - l_{\tilde{s}(r)}). \tag{6}$$

We consider $\mathbf{v}_0 = [r, r, \cdots, r]^T$ for $r \neq 0$. The normalized coordinates is $\hat{\mathbf{v}}_0 = [1/\sqrt{d}, \cdots, 1/\sqrt{d}]^T$.

Using (6) and noting $1/\sqrt{d} < l_1$, we have

$$\sigma^2(r_0) = 1/\sqrt{d}\big(l_1 - 1/\sqrt{d}\big) \geq l_1/(2\sqrt{d}). \tag{7}$$

Summing variance of all coordinates and applying (7), the variance of $Q_s(\mathbf{v}_0)$ is lower bounded by

$$\mathbb{E}[\|Q_s(\mathbf{v}_0) - \mathbf{v}_0\|^2] = \|\mathbf{v}_0\|^2 d\sigma^2(r) \geq \|\mathbf{v}_0\|^2 l_1 \sqrt{d}/2. \tag{8}$$

■

We can obtain convergence guarantees to various learning problems where we have convergence guarantees for SGD under standard assumptions. On nonconvex problems, (weaker) convergence guarantees can be established along the lines of, e.g., (Ghadimi and Lan, 2013, Theorem 2.1). In particular, NUQSGD is guaranteed to converge to a local minima for smooth general loss functions.

Theorem 8 (NUQSGD for smooth nonconvex optimization) *Let $f : \Omega \to \mathbb{R}$ denote a possibly nonconvex and β-smooth function. Let $\mathbf{w}_0 \in \Omega$ denote an initial point, ϵ_Q be defined as in Theorem 4, $T \in \mathbb{Z}^{>0}$, and $f^* = \inf_{\mathbf{w}\in\Omega} f(\mathbf{w})$. Suppose that Algorithm 1 is executed for T iterations with a learning rate $\alpha < 2/\beta$ on K processors, each with access to independent stochastic gradients of f with a second-moment bound B. Then there exists a random stopping time $R \in \{0, \cdots, T\}$ such that NUQSGD guarantees $\mathbb{E}[\|\nabla f(\mathbf{w}_R)\|^2] \leq \beta(f(\mathbf{w}_0) - f^*)/T + 2(1 + \epsilon_Q)B/K$.*

4.1 Worst-case Variance Analysis

In this section, we first derive a tight worst-case variance upper bound by optimizing over the distribution of normalized coordinates for an arbitrary sequence of levels, expressed as a solution to an integer program with quadratic constraints. We then relax the program to obtain a quadratically constrained quadratic program (QCQP). A coarser analysis yields an upper bound expressed as a solution to a linear program (LP), which is more amenable to analysis. We solve this LP analytically for the special case of $s = 1$ and show the optimal level is at $1/2$.

Then, for an exponentially spaced collection of levels of the form $(0, p^s, \cdots, p^2, p, 1)$ for $p \in (0, 1)$ and an integer number of levels, s, we write the expression of QCQP and solve it

efficiently using standard solvers. We have a numerical method for finding the optimal value of p that minimizes the worst-case variance, for any given s and d. Through the worst-case analysis, we gain insight into the behaviour of the variance upper bound. We show that our current scheme is nearly optimal (in the worst-case sense) in some cases. Using these techniques we can obtain slightly tighter bounds numerically.

4.1.1 Generally Spaced Levels

Let $\mathcal{L} = (l_0, l_1, \cdots, l_s, l_{s+1})$ denote an arbitrary sequence of quantization levels where $l_0 = 0 < l_1 < \cdots < l_{s+1} = 1$. Recall that, for $r \in [0,1]$, we define $\tilde{s}(r)$ and $p(r)$ such that they satisfy $l_{\tilde{s}(r)} \leq r \leq l_{\tilde{s}(r)+1}$ and $r = \big(1-p(r)\big)l_{\tilde{s}(r)} + p(r)l_{\tilde{s}(r)+1}$, respectively. Define $\tau(r) = l_{\tilde{s}(r)+1} - l_{\tilde{s}(r)}$. Note that $\tilde{s}(r) \in \{0, 1, \cdots, s\}$. Then, $h_i(\mathbf{v}, s)$'s are defined in two cases based on which quantization interval r_i falls into:

1) If $r_i \in [0, l_1]$, then

$$h_i(\mathbf{v}, s) = \begin{cases} 0 & \text{with probability } 1 - p_1(r_i, \mathcal{L}); \\ l_1 & \text{otherwise} \end{cases} \tag{9}$$

where $p_1\big(r, \mathcal{L}\big) = r/l_1$.

2) If $r_i \in [l_{j-1}, l_j]$ for $j = 1, \cdots, s+1$, then

$$h_i(\mathbf{v}, s) = \begin{cases} l_{j-1} & \text{with probability } 1 - p_2(r_i, \mathcal{L}); \\ l_j & \text{otherwise} \end{cases} \tag{10}$$

where $p_2\big(r, \mathcal{L}\big) = (r - l_{j-1})/\tau_{j-1}$.

Let $\mathcal{S}_j$ denote the coordinates of vector $\mathbf{v}$ whose elements fall into the $(j+1)$-th bin, *i.e.,* $\mathcal{S}_j \triangleq \{i : r_i \in [l_j, l_{j+1}]\}$ for $j = 0, \cdots, s$. Let $d_j \triangleq |\mathcal{S}_j|$.

Following Lemma 14 and steps in Theorem 4, we can show that

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \|\mathbf{v}\|^2 (\min\{\tau_0^2 d_0/4, \tau_0\sqrt{d_0}\} + \sum_{j=1}^{s} \min\{\tau_j^2 d_j/4, \tau_j(\sqrt{d_j} - l_j d_j)\}). \tag{11}$$

Theorem 9 (QCQP bound) *Let $\mathbf{v} \in \mathbb{R}^d$. An upper bound on the nonuniform quantization of $\mathbf{v}$ is given by $\epsilon_{QP}\|\mathbf{v}\|^2$ where ϵ_{QP} is the optimal value of the following QCQP:*

$$\begin{aligned} \mathcal{Q}_1 : \quad & \max_{(d_0,\cdots,d_s,z_0,\cdots,z_s)} \sum_{j=0}^{s} z_j \\ & \text{subject to } d - d_0 - \cdots - d_j \leq (1/l_{j+1})^2,\ j = 0, \cdots, s-1, \\ & \sum_{j=0}^{s} d_j \leq d,\ z_0 \leq \tau_0^2 d_0/4,\ z_0^2 \leq \tau_0^2 d_0, \\ & z_j \leq \tau_j^2 d_j/4,\ z_j^2 + \tau_j^2 l_j^2 d_j^2 + 2\tau_j l_j d_j z_j \leq \tau_j^2 d_j,\ j = 1, \cdots, s, \\ & d_j \geq 0,\ j = 0, \cdots, s. \end{aligned}$$

Proof Following Lemma 17, we have

$$d - d_0 - d_1 - \cdots - d_j \leq (1/l_{j+1})^2 \tag{12}$$

for $j = 0, \cdots, s-1$.

The problem of optimizing $(d_0, \cdots, d_s)$ to maximize the variance upper bound (11) subject to (12) is given by

$$\begin{aligned}
\mathcal{R}_1: \quad & \max_{(d_0,\cdots,d_s)} \sum_{j=0}^{s} \min\{\tau_j^2 d_j/4, \tau_j(\sqrt{d_j} - l_j d_j)\} \\
& \text{subject to (12), } j = 0, \cdots, s-1, \\
& \sum_{j=0}^{s} d_j \leq d, \qquad (13) \\
& d_j \in \mathbb{Z}^{\geq 0}, \ j = 0, \cdots, s. \qquad (14)
\end{aligned}$$

Let $z_j \triangleq \min\{\tau_j^2 d_j, \tau_j\sqrt{d_j}\}$ denote an auxiliary variable for $j = 0, \cdots, s$. Problem $\mathcal{R}_1$ can be rewritten as

$$\begin{aligned}
\mathcal{R}_2: \quad & \max_{(d_0,\cdots,d_s,z_0,\cdots,z_s)} \sum_{j=0}^{s} z_j \\
& \text{subject to (12), (13), and (14),} \\
& z_0 \leq \tau_0^2 d_0/4, \ z_0^2 \leq \tau_0^2 d_0, \\
& z_j \leq \tau_j^2 d_j/4, \ z_j^2 + \tau_j^2 l_j^2 d_j^2 + 2\tau_j l_j d_j z_j \leq \tau_j^2 d_j, \ j = 1, \cdots, s.
\end{aligned} \tag{15}$$

The variance optimization problem $\mathcal{R}_2$ is an integer nonconvex problem. We can obtain an upper bound on the optimal objective of problem $\mathcal{R}_2$ by relaxing the integer constraint as follows. The resulting QSQP is shown as follows:

$$\begin{aligned}
\mathcal{Q}_1: \quad & \max_{(d_0,\cdots,d_s,z_0,\cdots,z_s)} \sum_{j=0}^{s} z_j \\
& \text{subject to } d_j \geq 0, \ j = 0, \cdots, s, \\
& \qquad (12), \ (13), \text{ and } (15).
\end{aligned} \tag{16}$$

Note that problem $\mathcal{Q}_1$ can be solved efficiently using standard standard interior point-based solvers, *e.g.,* CVX (Boyd and Vandenberghe, 2004). ■

In the following, we develop a coarser analysis that yields an upper bound expressed as the optimal value to an LP.

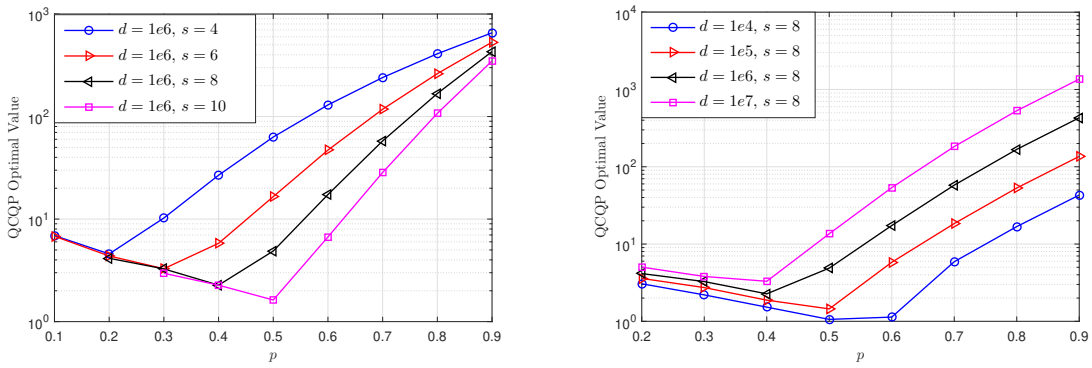

Figure 2: Optimal value of problem $\mathcal{Q}_1$ versus $p \in [0, 1]$ for exponentially spaced collection of levels of the form $(0, p^s, \cdots, p^2, p, 1)$.

Theorem 10 (LP bound) *Let $\mathbf{v} \in \mathbb{R}^d$. An upper bound on the nonuniform quantization of $\mathbf{v}$ is given by $\epsilon_{LP}\|\mathbf{v}\|^2$ where ϵ_{LP} is the optimal value of the following LP:*

$$\begin{aligned} \mathcal{P}_1 : \quad & \max_{(d_0,\cdots,d_s)} \sum_{j=0}^{s} \tau_j^2 d_j/4 \\ & \text{subject to } d - d_0 - \cdots - d_j \le (1/l_{j+1})^2,\ j = 0, \cdots, s-1, \\ & \sum_{j=0}^{s} d_j \le d, \\ & d_j \ge 0,\ j = 0, \cdots, s. \end{aligned}$$

Proof The proof follows the steps in the proof of Theorem 9 for the problem of optimizing $(d_0, \cdots, d_s)$ to maximize the following upper bound

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \le \|\mathbf{v}\|^2 \sum_{j=0}^{s} \tau_j^2 d_j/4. \tag{17}$$

■

The LP bound can be solved exactly in some simple cases. In Appendix E, we present the optimal solution for the special case with $s = 1$.

4.1.2 Exponentially Spaced Levels

In this section, we focus on the special case of exponentially spaced collection of levels of the form $\mathcal{L}_p = (0, p^s, \cdots, p^2, p, 1)$ for $p \in (0, 1)$ and an integer number of levels, s. In this case, we have $\tau_0 = p^s$ and $\tau_j = (1-p)p^{s-j}$ for $j = 1, \cdots, s$.

For any given s and d, we can solve the corresponding quadratic and linear programs efficiently to find the worst-case variance bound. As a bonus, we can find the optimal value of p that minimizes the worst-case variance bound. In Figure 2, we show the numerical

results obtained by solving QCQP $\mathcal{Q}_1$ with $\mathcal{L}_p$ versus p using CVX (Boyd and Vandenberghe, 2004). In Figure 2 (left), we fix d and vary s, while in Figure 2 (right), we fix s and vary d. As expected, we note that the variance upper bound increases as d increases and the variance upper bound decreases as s increases. We observe that our current scheme is nearly optimal (in the worst-case sense) in some cases. Further, the optimal value of p shifts to the right as d increases and shifts to the left as s increases.

4.2 NUQSGD with Momentum

We have convergence guarantees for NUQSGD with momentum along the lines of, e.g., (Yan et al., 2018) where convergence guarantees for momentum-based methods are established under standard assumptions.

The update rule for full-precision SGD with momentum is

$$\begin{aligned} \mathbf{y}_{t+1} &= \mathbf{w}_t - \alpha g(\mathbf{w}_t) \\ \mathbf{y}^l_{t+1} &= \mathbf{w}_t - l\alpha g(\mathbf{w}_t) \\ \mathbf{w}_{t+1} &= \mathbf{y}_{t+1} + \mu\big(\mathbf{y}^l_{t+1} - \mathbf{y}^l_t\big) \end{aligned} \tag{18}$$

where $\mathbf{w}_t$ is the current parameter input and $\mu \in [0, 1)$ is the momentum parameter (Yan et al., 2018). Note that the heavy-ball method (Polyak, 1964) and Nesterov's accelerated gradient method (Nesterov, 1983) are obtained by substituting $l = 0$ and $l = 1$, respectively.

One obtains data-parallel SGD with momentum by taking Algorithm 1 and replacing step 10 with (18). We first establish the convergence guarantees for convex optimization in the following theorem.

Theorem 11 (NUQSGD with momentum for convex optimization) *Let $f : \mathbb{R}^d \to \mathbb{R}$ denote a convex function with $\|\nabla f(\mathbf{w})\| \leq V$ for all $\mathbf{w}$. Let $\mathbf{w}_0$ denote an initial point, $\mathbf{w}^* = \arg\min f(\mathbf{w})$, $\hat{\mathbf{w}}_T = 1/T \sum_{t=0}^{T} \mathbf{w}_t$, and ϵ_Q be defined as in Theorem 4.*

Suppose that NUQSGD with momentum is executed for T iterations with a learning rate $\alpha > 0$ on K processors, each with access to independent stochastic gradients of f with a second-moment bound B. Then NUQSGD with momentum satisfies

$$\begin{aligned} \mathbb{E}[f(\hat{\mathbf{w}}_T)] - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \frac{\mu(f(\mathbf{w}_0) - f(\mathbf{w}^*))}{(1-\mu)(T+1)} + \frac{(1-\mu)\|\mathbf{w}_0 - \mathbf{w}^*\|^2}{2\alpha(T+1)} \\ + \frac{\alpha(1+2l\mu)(V^2 + (1+\epsilon_Q)B/K)}{2(1-\mu)}. \end{aligned} \tag{19}$$

On nonconvex problems, (weaker) convergence guarantees can be established for NUQSGD with momentum. In particular, NUQSGD with momentum is guaranteed to converge to a local minima for smooth general loss functions.

Theorem 12 (NUQSGD with momentum for smooth nonconvex optimization) *Let $f : \mathbb{R}^d \to \mathbb{R}$ denote a possibly nonconvex and β-smooth function with $\|\nabla f(\mathbf{w})\| \leq V$ for all $\mathbf{w}$. Let $\mathbf{w}_0$ denote an initial point, $\mathbf{w}^* = \arg\min f(\mathbf{w})$, and ϵ_Q be defined as in Theorem 4.*

Suppose that NUQSGD with momentum is executed for T iterations with $\alpha = \min\{(1-\mu)/(2\beta), C/\sqrt{T+1}\}$ for some $C > 0$ on K processors, each with access to independent

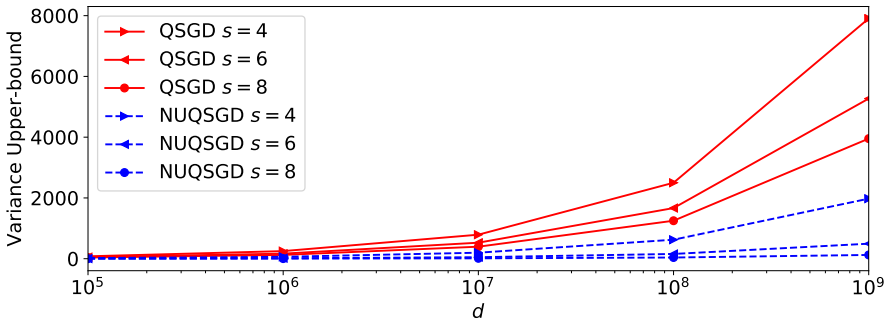

Figure 3: Variance upper bounds.

stochastic gradients of f with a second-moment bound B. Then NUQSGD with momentum satisfies

$$\min_{t=0,\cdots,T} \mathbb{E}[\|\nabla f(\mathbf{w}_t)\|^2] \leq \frac{2(f(\mathbf{w}_0) - f(\mathbf{w}^*))(1-\mu)}{\alpha(T+1)} + \frac{C}{(1-\mu)^3\sqrt{T+1}}\tilde{V} \tag{20}$$

where $\tilde{V} = \beta\big(\mu^2((1-\mu)l-1)^2 + (1-\mu)^2\big)(V^2 + (1+\epsilon_Q)B/K)$.

In Appendices G and H, we extend our analysis to asynchronous and decentralized settings.

4.3 NUQSGD vs QSGD and QSGDinf

How do QSGD and NUQSGD compare in terms of bounds on the expected number of communication bits required to achieve a given suboptimality gap ϵ? The quantity that controls our guarantee on the convergence speed in both algorithms is the variance upper bound, which in turn is controlled by the quantization schemes. Note that the number of quantization levels, s, is usually a small number in practice. On the other hand, the dimension, d, can be very large, especially in overparameterized networks. In Figure 3, we show that the quantization scheme underlying NUQSGD results in substantially smaller variance upper bounds for plausible ranges of s and d. Note that these bounds do not make any assumptions on the dataset or the structure of the network.

For any (nonrandom) number of iterations T, an upper bound, $\overline{N}_A$, holding uniformly over iterations $k \leq T$ on the expected number of bits used by an algorithm A to communicate the gradient on iteration k, yields an upper bound $T\overline{N}_A$, on the expected number of bits communicated over T iterations by algorithm A. Taking $T = T_{A,\epsilon}$ to be the (minimum) number of iterations needed to guarantee an expected suboptimality gap of ϵ based on the properties of A, we obtain an upper bound, $\zeta_{A,\epsilon} = T_{A,\epsilon}\overline{N}_A$, on the expected number of bits of communicated on a run expected to achieve a suboptimality gap of at most ϵ.

Theorem 13 (Expected number of communication bits) *Provided that $s = o(\log(d))$ and $\frac{2\hat{B}}{K\epsilon^2} > \frac{\beta}{\epsilon}$, $\zeta_{\text{NUQSGD},\epsilon} = O\big(\frac{1}{\epsilon^2}\sqrt{d(d-2^{2s})}\log\big(\frac{\sqrt{d}}{2^s}\big)\big)$ and $\zeta_{\text{QSGD},\epsilon} = O(\frac{1}{\epsilon^2}d\log\sqrt{d})$.*

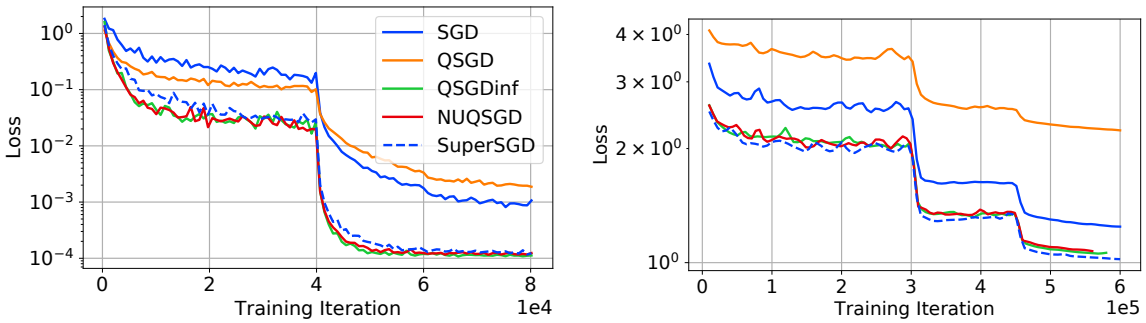

Figure 4: Training loss on CIFAR10 (left) and ImageNet (right) for ResNet models. QSGD, QSGDinf, and NUQSGD are trained by simulating the quantization and dequantizing of the gradients from 8-GPUs. On CIFAR10, SGD refers to the single-GPU training versus on ImageNet it refers to 2-GPU setup in the original ResNet paper. SGD is shown to highlight the significance of the gap between QSGD and QSGDinf. SuperSGD refers to simulating full-precision distributed training without quantization. SuperSGD is impractical in scenarios with limited bandwidth.

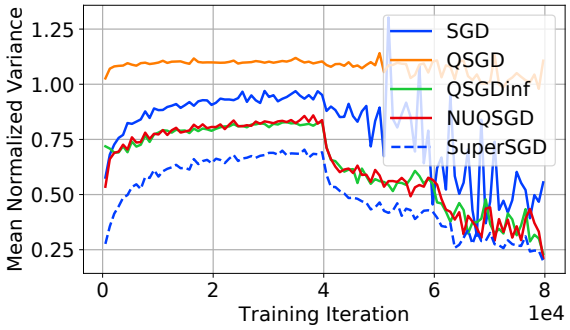

Figure 5: Estimated normalized variance on CIFAR10 on the trajectory of single-GPU SGD. Variance is measured for fixed model snapshots during training. Notice that the variance for NUQSGD and QSGDinf is lower than SGD for almost all the training and it decreases after the learning rate drops.

Focusing on the dominant terms in the expressions of overall number of communication bits required to guarantee a suboptimality gap of ϵ, we observe that NUQSGD provides slightly stronger guarantees. Note that our stronger guarantees come without any assumption about the data.

In Appendix I, we show that there exist vectors for which the variance of quantization under NUQSGD is guaranteed to be smaller than that under QSGDinf.

5. Experimental Evaluation

In this section, we examine the practical performance of NUQSGD in terms of both convergence (accuracy) and speedup. The goal is to empirically examine whether NUQSGD can provide the similar performance and accuracy compared to the QSGDinf heuristic, which

has no theoretical compression guarantees.[3] For this, we implement and test these three methods (NUQSGD, QSGD, and QSGDinf), together with the distributed full-precision SGD baseline, which we call SuperSGD. Additionally, we will compare practical performance against a variant of SignSGD with EF (Karimireddy et al., 2019). We split our study across two axes: first, we validate our theoretical analysis by examining the variance induced by the methods, as well as their convergence in terms of loss/accuracy versus number of samples processed. Second, we provide an efficient implementation of all four methods in Pytorch using the Horovod communication back-end (Sergeev and Del Balso, 2018), a communication back-end supporting Pytorch, Tensorflow and MXNet. We adapted Horovod to efficiently support quantization and gradient coding, and examine speedup relative to the full-precision baseline. Further, we examine the effect of quantization on training performance by measuring loss, variance, accuracy, and speedup for ResNet models (He et al., 2016) applied to ImageNet and CIFAR10 (Krizhevsky, 2009).

Convergence and Variance. Our first round of experiments examine the impact of quantization on solution quality. We evaluate these methods on two image classification datasets: ImageNet and CIFAR10. We train ResNet110 on CIFAR10 and ResNet18 on ImageNet with mini-batch size 128 and base learning rate 0.1. In all experiments, momentum and weight decay are set to 0.9 and 10^{-4}, respectively. The bucket size (quantization granularity) and the number of quantization bits are set to 8192 and 4, respectively. We observed similar trends in experiments with various bucket sizes and number of bits per entry. We simulate a scenario with k GPUs for all three quantization methods by estimating the gradient from k independent mini-batches and aggregating them after quantization and dequantization.

In Figure 4 (left and right), we show the training loss with 8 GPUs. We observe that NUQSGD and QSGDinf have lower training loss compared to QSGD on ImageNet. We observe a significant gap in training loss on CIFAR10, where the gap grows as training proceeds. We also observed similar performance gaps in test accuracy (provided in Appendix J). In particular, unlike NUQSGD, QSGD does not achieve the test accuracy of full-precision SGD. Figure 5 shows the mean normalized variance of the gradient (defined formally in Appendix J) versus the training iterations, on the trajectory of single-GPU SGD on CIFAR10. These observations validate our theoretical results that NUQSGD has smaller variance for large models with small number of quantization bits, and support the intuition that this can impact solution quality.

In Figure 6, we show the test accuracy for training ResNet110 on CIFAR10 and validation accuracy for training ResNet34 on ImageNet from random initialization until convergence. Similar to the training loss performance, we observe that NUQSGD and QSGDinf outperform QSGD in terms of test accuracy in both experiments. In both experiments, unlike NUQSGD, QSGD does not recover the test accuracy of SGD. The gap between NUQSGD and QSGD on ImageNet is significant. We argue that this is achieved because NUQSGD and QSGDinf have lower variance relative to QSGD. It turns out both training loss and generalization error can benefit from the reduced variance.

3. We also provide a generic implementation in Horovod (Sergeev and Del Balso, 2018), a communication back-end which can support a range of modern frameworks such as Tensorflow, Keras, Pytorch, and MXNet.

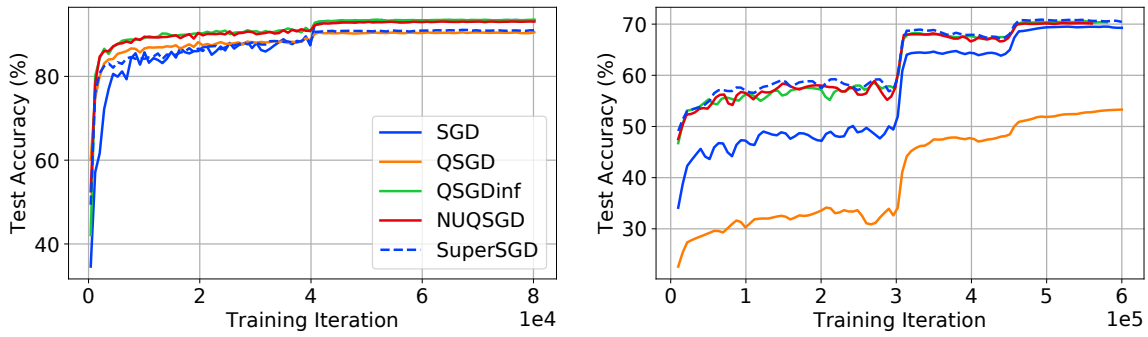

Figure 6: Accuracy on the hold-out set on CIFAR10 (left) and on ImageNet (right) for training ResNet models from random initialization until convergence. For CIFAR10, the hold-out set is the test set and for ImageNet, the hold-out set is the validation set.

Efficient Implementation and Speedup. To examine speedup behavior, we implemented all quantization methods in Horovod (Sergeev and Del Balso, 2018). Doing so efficiently requires non-trivial refactoring of this framework, since it does not support communication compression—our framework will be open-sourced upon publication. For performance reasons, our implementation diverges slightly from the theoretical analysis. First, the Horovod framework applies "tensor fusion" to multiple layers, by merging the resulting gradient tensors for more efficient transmission. This causes the gradients for different layers to be quantized together, which can lead to loss of accuracy (due to e.g. different normalization factors across the layers). We addressed this by tuning the way in which tensor fusion is applied to the layers such that it minimizes the accuracy loss. Second, we noticed that quantizing the gradients corresponding to the biases has an adverse effect on accuracy; since the communication impact of biases is negligible, we transmit them at full precision. We apply these steps to all methods. We implemented compression and de-compression via efficient CUDA kernels.

Efficient Encoding. One of the advantages of NUQSGD is that it provides both good practical accuracy, as well as bounds on the code-length of the transmitted gradients in actual executions. This should be contrasted with QSGD (which provides such bounds, but, as seen above, loses accuracy), and QSGDinf, whose analysis no longer implies any bounds on the expected code-length. (The QSGD analysis is only performed for L2 normalization.) We leverage this fact by employing Huffman coding on an initial sample of gradients, in order to determine a non-trivial encoding of the integer levels sent. We then employ this coding for the rest of the execution. This variant is called encoded NUQSGD.

Our baselines are full-precision SGD (SuperSGD), EF-SignSGD (Karimireddy et al., 2019), and the QSGDinf heuristic, which we compare against the 4-bit and 8-bit NUQSGD variants executing the same pattern. The implementation of the QSGDinf heuristic provides virtually identical convergence numbers, and is sometimes omitted for visibility. QSGD yields inferior convergence on this dataset and is therefore omitted. All variants are implemented using a standard all-to-all reduction pattern. Figure 7 (left and right) shows the execution

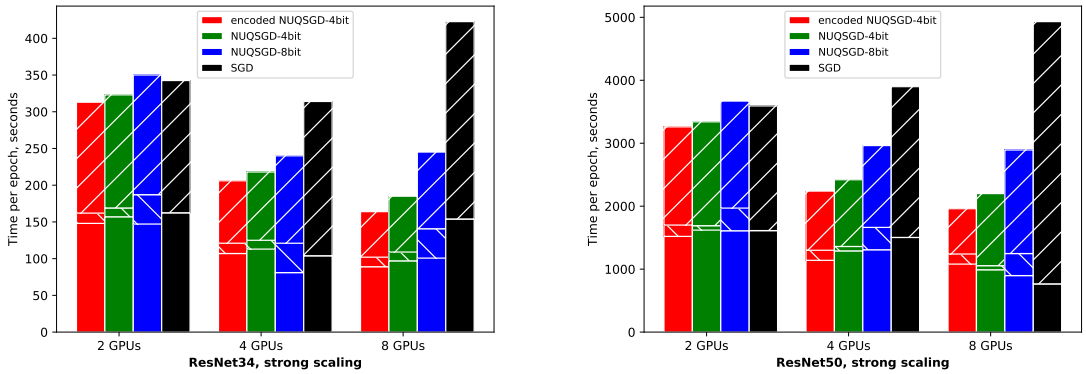

Figure 7: Scalability behavior for NUQSGD versus the full-precision baseline when training ResNet34 (left) and ResNet50 (right) on ImageNet. Both experiments examine *strong scaling*, splitting a global batch of size 256 onto the available GPUs, for 2, 4, and 8 nodes. Each time bar is split into computation (bottom), encoding cost (middle), and transmission cost (top). Notice the negative scalability of the SGD baseline in both scenarios. By contrast, the 4-bit communication-compressed implementation achieves positive scaling, while the 8-bit variant stops scaling between 4 and 8 nodes due to the higher communication and encoding costs, while the encoding variant offers further compression.

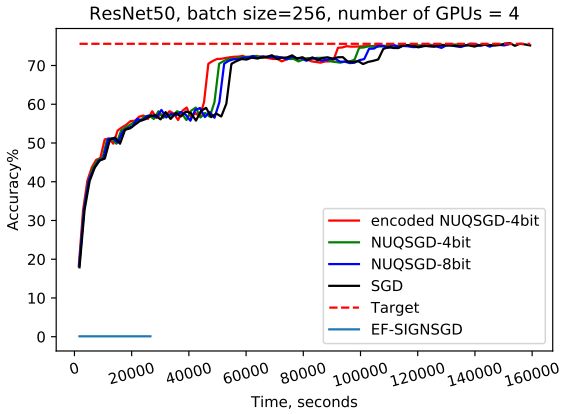

Figure 8: End-to-end training time for ResNet50/ImageNet for NUQSGD and EF-SignSGD (which diverges) versus the SuperSGD baseline.

time per epoch for ResNet34 and ResNet50 models on ImageNet, on a cluster machine with 8 NVIDIA 2080 Ti GPUs, for the hyper-parameter values quoted above.[4]

The results confirm the efficiency and scalability of the compressed variant, mainly due to the reduced communication volume. We note that the overhead of compression and decompression is less than 1% of the batch computation time for vanilla NUQSGD,

4. We use the following hardware: CPU information: Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz, 24 cores. GPU2GPU Bandwidth: unidirectional 10GB/s, Bidirectional 15GB/s.

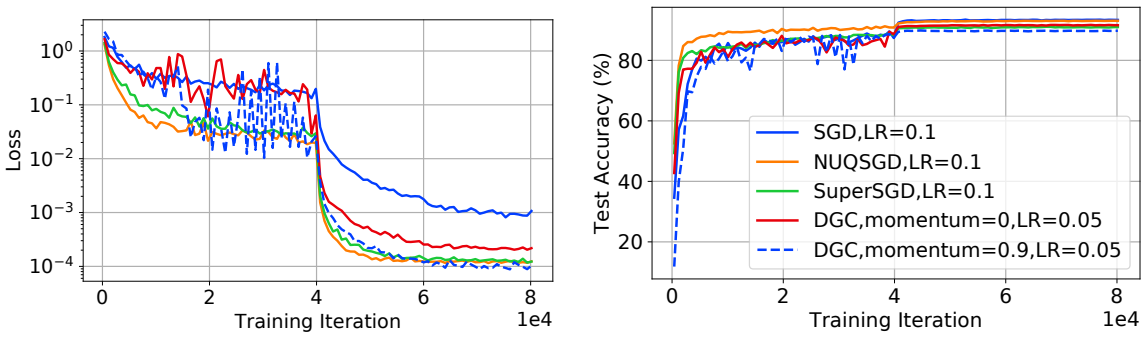

Figure 9: Training loss (left) and Test accuracy (right) on CIFAR10 for ResNet110. We set the ratio of compression for DGC to be roughly the same as NUQSGD. In particular, we compare compression methods at compression ratio = 4/32, *i.e.,* NUQSGD with 4 bits and DGC at 12.5% compression. At this rate, both methods will have approximately the same communication cost, *i.e.,* comparison in simulation is representative of real-time performance. We tune the learning rate and momentum for DGC and show its best performance.

and of $< 4\%$ for the gradient coding variant. We also note that the smallest reduction times are obtained by encoded NUQSGD, which send on average approximately 2.7 bits per component, a reduction of 67% over standard NUQSGD.

Figure 8 presents end-to-end speedup numbers (time versus accuracy) for ResNet50 on ImageNet, executed on 4 GPUs, under the same hyperparameter settings as the full-precision baseline, with bucket size 512. First, notice that all NUQSGD variants match the target accuracy of the 32-bit model, with non-trivial speedup over the standard data-parallel variant, directly proportional to the per-epoch speedup. The QSGDinf heuristic yields similar accuracy and performance, and is therefore omitted. We found that unfortunately EF-SignSGD does not converge under these standard hyperparameter settings. To address this issue, we performed a non-trivial amount of hyperparameter tuning for this algorithm: in particular, we found that the scaling factors and the bucket size must be carefully adjusted for convergence on ImageNet. We were able to recover full accuracy with EF-SignSGD on ResNet50, but at the cost of quantizing into buckets of size 64. In this setting, the algorithm sends 32 bits of scaling data for every 64 entries, and the GPU implementation becomes less efficient due to error computation and reduced parallelism. The end-to-end speedup of this tuned variant is inferior to baseline 4-bit NUQSGD, and only slightly superior to that of 8-bit NUQSGD. Please see Figure 18 in the Appendix J and the accompanying text for details.

We therefore conclude that NUQSGD offers competitive or superior performance w.r.t. QSGDinf and EF-SignSGD, while providing strong convergence guarantees, and that it allows additional savings due to the fact that gradients can be encoded.

Comparison with DGC. In this section, we make a comparison with DGC (Lin et al., 2018). DGC is essentially a sparsification method, which leverages momentum correction and local gradient clipping to recover accuracy.

In Figure 9, we show simulation results to compare convergence and generalization of DGC with those of NUQSGD and full-precision SGD. We set the ratio of compression for

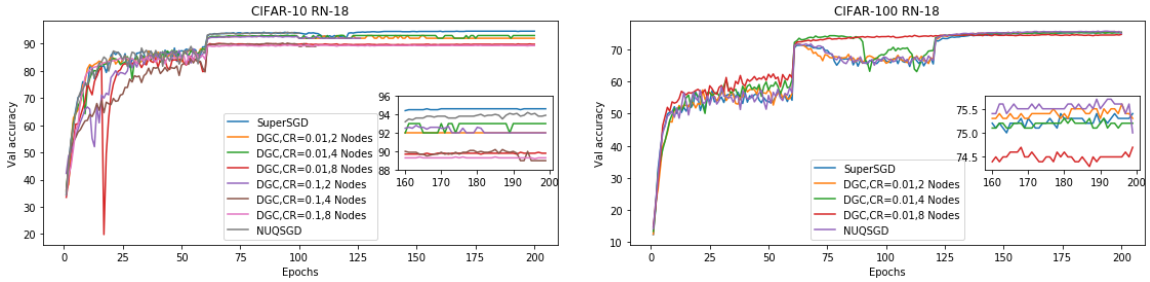

Figure 10: Test Accuracy on CIFAR10 (left) and CIFAR100 (right) for ResNet18. For NUQSGD and SuperSGD, we use 8 GPUs. For NUQSGD, we use the same hyperparameters that are tuned for the full-precision baseline. However, we tune hyperparameters for DGC at 1% and 10% compression ratios.

DGC to be roughly the same as NUQSGD. In particular, we compare compression methods at compression ratio = 4/32, *i.e.,* NUQSGD with 4 bits and DGC at 12.5% compression. At this rate, both methods will have approximately the same communication cost, *i.e.,* comparison in simulation is representative of real-time performance. We show the results for NUQSGD, DGC, and full-precision baselines on CIFAR10. We tune the learning rate and momentum for DGC and show its best performance.

Our results show that by tuning DGC carefully, it can achieve full-precision performance albeit some noisy curves at the beginning. On the other hand, NUQSGD provides communication efficiency on the fly where practitioners can reuse the same hyperparameters that are tuned for full-precision schemes. NUQSGD is also easier to implement efficiently in practice. Furthermore, NUQSGD enjoys strong theoretical guarantees. Finally, we note that NUQSGD can be used as the encoding function on top of DGC to further reduce its communication costs. Hence, DGC and NUQSGD can be viewed as complementary methods for practitioners. Additional simulation results are presented in Appendix J.

To further evaluate the performance of DGC over different numbers of GPUs, compression ratios, and datasets, in Figure 10, we show validation accuracy when we train ResNet18 on CIFAR10 and CIFAR100. For NUQ and the full-precision baseline (SuperSGD), we use 8 GPUs. For NUQSGD, we use the same hyperparameters that are tuned for the full-precision baseline. However, we tune hyperparameters for DGC at 1% and 10% compression ratios. We note that unlike NUQSGD, DGC's performance degrades as we increase the number of GPUs. We observe DGC is less stable for simpler datasets such as CIFAR10. In sum, while NUQSGD offers lower *peak* compression relative to DGC, NUQSGD is more practical. Additionally, NUQSGD offers strong theoretical guarantees relative to DGC.

Comparison with TernGrad and ATOMO. We ported the original ATOMO (Wang et al., 2018) and TernGrad (Wen et al., 2017) code to our framework. In Table 1, we present results of training ResNet20 on CIFAR10 on Google Cloud (4 nodes in a bandwidth-constrained setting) under the standard training parameters. Our experiments show that NUQSGD achieves slightly higher accuracy than ATOMO at a slightly higher compression ratio. (We did not use Huffman coding here.) ATOMO's time/step is high due to its computation cost (we have used its original implementation, but further optimizations might be possible). Unfortunately, TernGrad did not converge for the standard (sequential)

Method	Accuracy	Time/step (s)	Compression%
SuperSGD	91.5	205	100
NUQSGD (3bit)	90.94	85	9
ATOMO (rank=3)	90.5	297	17
TernGrad	did not converge for standard hyperparameters		

Table 1: Training ResNet20 on CIFAR-10 using on 4 nodes under standard training parameters.

hyperparameters. Lin et al. (2018) showed that TernGrad loses accuracy at ImageNet scale. Its average compression ratio (2 bits/entry) is similar to that of NUQSGD with Huffman coding (approximately 2.4 bits/entry).

6. Conclusions

We study data-parallel and communication-efficient version of stochastic gradient descent. Building on QSGD (Alistarh et al., 2017), we study a nonuniform quantization scheme. We establish upper bounds on the variance of nonuniform quantization and the expected code-length. The former decreases as the number of quantization levels increases, while the latter increases with the number of quantization levels. Thus, this scheme provides a trade-off between the communication efficiency and the convergence speed. We compare NUQSGD and QSGD in terms of their variance bounds and the expected number of communication bits required to meet a certain convergence error, and show that NUQSGD provides stronger guarantees. Experimental results are consistent with our theoretical results and confirm that NUQSGD matches the performance of QSGDinf when applied to practical deep models and datasets including ImageNet. Thus, NUQSGD closes the gap between the theoretical guarantees of QSGD and empirical performance of QSGDinf. One limitation of our study which we aim to address in future work is that we focus on all-to-all reduction patterns, which interact easily with communication compression. In particular, we aim to examine the interaction between more complex reduction patterns, such as ring-based reductions (Hannun et al., 2014), which may yield superior performance in bandwidth-bottlenecked settings, but which interact with communication-compression in non-trivial ways, since they may lead a gradient to be quantized at each reduction step.

Acknowledgement

Ramezani-Kebrya was supported by an NSERC Postdoctoral Fellowship. Faghri was supported by an OGS Scholarship. Alistarh, Markov, and Aksenov were supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 805223 ScaleML). Roy was supported by an NSERC Discovery Grant. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.[5]

5. www.vectorinstitute.ai/#partners

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467, 2016.

Dan Alistarh, Demjan Grubic, Jerry Z. Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2017.

Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *Proc. International Conference on Machine Learning (ICML)*, 2018.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends ® in Machine Learning*, 8(3-4):231–358, 2015.

Kenneth W. Cattermole. *Principles of Pulse Code Modulation*. Iliffe, 1969.

Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project Adam: Building an efficient and scalable deep learning training system. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014.

Adam Coates, Brody Huval, Tao Wang, David Wu, Bryan Catanzaro, and Andrew Y. Ng. Deep learning with COTS HPC systems. In *Proc. International Conference on Machine Learning (ICML)*, 2013.

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Z. Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Y. Ng. Large scale distributed deep networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2012.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

John C. Duchi, Sorathan Chaturapruek, and Christopher Ré. Asynchronous stochastic convex optimization. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2015.

Peter Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *Proc. International Conference on Machine Learning (ICML)*, 2015.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. arXiv:1412.5567, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Samuel Horváth, Chen-Yu Ho, Ludovít Horváth, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. Natural compression for distributed deep learning. arXiv:1905.10988v1, 2019.

Lu Hou and James T. Kwok. Loss-aware weight quantization of deep networks. In *Proc. International Conference on Learning Representations (ICLR)*, 2018.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U. Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *Proc. International Conference on Machine Learning (ICML)*, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. Technical report, University of Toronto.

Edward H. Lee, Daisuke Miyashita, Elaina Chai, Boris Murmann, and S. Simon Wong. LogNet: Energy-efficient neural networks using logarithmic computation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

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

Zheng Li and Christopher M. De Sa. Dimension-free bounds for low-precision training. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2015.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *Proc. International Conference on Learning Representations (ICLR)*, 2018.

Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. arXiv:1603.01025v2, 2016.

Yurii Nesterov. A method of solving a convex programming problem with convergence $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Jongsoo Park, Sheng Li, Wei Wen, Ping Tak Peter Tang, Hai Li, Yiran Chen, and Pradeep Dubey. Faster CNNs with direct sparse convolutions and guided pruning. In *Proc. International Conference on Learning Representations (ICLR)*, 2017.

Boris Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Benjamin Recht, Christopher Ré, Stephen J. Wright, and Feng Niu. HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2011.

Christopher M. De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of HOGWILD!-style algorithms. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2015.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Proc. INTERSPEECH*, 2014.

Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. arXiv:1802.05799, 2018.

Nikko Strom. Scalable distributed DNN training using commodity GPU cloud computing. In *Proc. INTERSPEECH*, 2015.

Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. Communication compression for decentralized training. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2018.

Hongyi Wang, Scott Sievert, Zachary Charles, Shengchao Liu, Stephen Wright, and Dimitris Papailiopoulos. ATOMO: Communication-efficient learning via atomic sparsification. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2018.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. TernGrad: Ternary gradients to reduce communication in distributed deep learning. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2017.

Eric P. Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE transactions on Big Data*, 1(2):49–67, 2015.

Yan Yan, Tianbao Yang, Zhe Li, Qihang Lin, and Yi Yang. A unified analysis of stochastic momentum methods for deep learning. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Hantian Zhang, Jerry Li, Kaan Kara, Dan Alistarh, Ji Liu, and Ce Zhang. ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning. In *Proc. International Conference on Machine Learning (ICML)*, 2017.

Sixin Zhang, Anna Choromanska, and Yann LeCun. Deep learning with elastic averaging SGD. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2015.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv:1606.06160, 2018.

Martin A. Zinkevich, Markus Weimer, Lihong Li, and Alex Smola. Parallelized stochastic gradient descent. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2010.

Encoding:
Place a 0 at the end of the string;
if $N == 1$ **then**
  Stop;
else
  Prepend binary(N) to the beginning ; `// Let` N' `denote # bits prepended minus 1`
  Encode N' recursively;
Decoding:
Start with $N = 1$;
if *the next bit* $== 0$ **then**
  Stop and return N;
else
  Read that bit plus N following bits and update N;

Algorithm 2: Elias recursive coding produces a bit string encoding of positive integers.

A. Encoding

By inspection, the quantized gradient $Q_s(\mathbf{v})$ is determined by the tuple $(\|\mathbf{v}\|, \boldsymbol{\rho}, \mathbf{h})$, where $\|\mathbf{v}\|$ is the norm of the gradient, $\boldsymbol{\rho} \triangleq [\text{sign}(v_1), \cdots, \text{sign}(v_d)]^T$ is the vector of signs of the coordinates v_i's, and $\mathbf{h} \triangleq [h_1(\mathbf{v}, s), \cdots, h_d(\mathbf{v}, s)]^T$ are the quantizations of the normalized coordinates. We can describe the ENCODE function (for Algorithm 1) in terms of the tuple $(\|\mathbf{v}\|, \boldsymbol{\rho}, \mathbf{h})$ and an encoding/decoding scheme ERC : $\{1, 2, \cdots\} \to \{0, 1\}^*$ and ERC^{-1} : $\{0, 1\}^* \to \{1, 2, \cdots\}$ for encoding/decoding positive integers.

The encoding, ENCODE($\mathbf{v}$), of a stochastic gradient is as follows: We first encode the norm $\|\mathbf{v}\|$ using b bits where, in practice, we use standard 32-bit floating point encoding. We then proceed in rounds, $r = 0, 1, \cdots$. On round r, having transmitted all nonzero coordinates up to and including t_r, we transmit $\text{ERC}(i_r)$ where $t_{r+1} = t_r + i_r$ is either (i) the index of the first nonzero coordinate of $\mathbf{h}$ after t_r (with $t_0 = 0$) or (ii) the index of the last nonzero coordinate. In the former case, we then transmit one bit encoding the sign $\rho_{t_{r+1}}$, transmit $\text{ERC}(\log(2^{s+1}h_{t_{r+1}}))$, and proceed to the next round. In the latter case, the encoding is complete after transmitting $\rho_{t_{r+1}}$ and $\text{ERC}(\log(2^{s+1}h_{t_{r+1}}))$.

The DECODE function (for Algorithm 1) simply reads b bits to reconstruct $\|\mathbf{v}\|$. Using ERC^{-1}, it decodes the index of the first nonzero coordinate, reads the bit indicating the sign, and then uses ERC^{-1} again to determines the quantization level of this first nonzero coordinate. The process proceeds in rounds, mimicking the encoding process, finishing when all coordinates have been decoded.

Like Alistarh et al. (2017), we use Elias recursive coding (Elias, 1975, ERC) to encode positive integers. ERC is simple and has several desirable properties, including the property that the coding scheme assigns shorter codes to smaller values, which makes sense in our scheme as they are more likely to occur. Elias coding is a universal lossless integer coding scheme with a recursive encoding and decoding structure.

The Elias recursive coding scheme is summarized in Algorithm 2. For any positive integer N, the following results are known for ERC (Alistarh et al., 2017):

1. $|\text{ERC}(N)| \leq \big(1 + o(1)\big) \log N + 1$;
2. $\text{ERC}(N)$ can be encoded and decoded in time $O(|\text{ERC}(N)|)$;

3. Decoding can be done without knowledge of an upper bound on N.

B. Proof of Theorem 4 (Variance Bound)

We first find a simple expression of the variance of $Q_s(\mathbf{v})$ for every arbitrary quantization scheme in the following lemma:

Lemma 14 *Let $\mathbf{v} \in \mathbb{R}^d$, $\mathcal{L} = (l_0, l_1, \cdots, l_{s+1})$, and fix $s \geq 1$. The variance of $Q_s(\mathbf{v})$ for general sequence of quantization levels is given by*

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] = \|\mathbf{v}\|^2 \sum_{i=1}^{d} \tau^2(r_i) p(r_i)\big(1 - p(r_i)\big) \tag{21}$$

where $r_i = |v_i|/\|\mathbf{v}\|$ and $p(r), \tilde{s}(r), \tau(r)$ are defined in Section 3.1.

Proof Noting the random quantization is i.i.d over elements of a stochastic gradient, we can decompose $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2]$ as:

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] = \sum_{i=1}^{d} \|\mathbf{v}\|^2 \sigma^2(r_i) \tag{22}$$

where $\sigma^2(r_i) = \mathbb{E}[(h_i(\mathbf{v}, s) - r_i)^2]$. Computing the variance of $h_i(\mathbf{v}, s)$, we can show that $\sigma^2(r_i) = \tau^2(r_i) p(r_i)\big(1 - p(r_i)\big)$. ■

In the following, we consider NUQSGD algorithm with $\hat{\mathcal{L}} = (0, 1/2^s, \cdots, 2^{s-1}/2^s, 1)$ as the quantization levels. Then, $h_i(\mathbf{v}, s)$'s are defined in two cases based on which quantization interval r_i falls into:

1) If $r_i \in [0, 2^{-s}]$, then

$$h_i(\mathbf{v}, s) = \begin{cases} 0 & \text{with probability } 1 - p_1(r_i, s); \\ 2^{-s} & \text{otherwise} \end{cases} \tag{23}$$

where $p_1\big(r, s\big) = 2^s r$.

2) If $r_i \in [2^{j-s}, 2^{j+1-s}]$ for $j = 0, \cdots, s-1$, then

$$h_i(\mathbf{v}, s) = \begin{cases} 2^{j-s} & \text{with probability } 1 - p_2(r_i, s); \\ 2^{j+1-s} & \text{otherwise} \end{cases} \tag{24}$$

where $p_2\big(r, s\big) = 2^{s-j} r - 1$. Note that $Q_s(\mathbf{0}) = \mathbf{0}$.

Let $\mathcal{S}_j$ denote the coordinates of vector $\mathbf{v}$ whose elements fall into the $(j+1)$-th bin, *i.e.*, $\mathcal{S}_0 \triangleq \{i : r_i \in [0, 2^{-s}]\}$ and $\mathcal{S}_{j+1} \triangleq \{i : r_i \in [2^{j-s}, 2^{j+1-s}]\}$ for $j = 0, \cdots, s-1$. Let $d_j \triangleq |\mathcal{S}_j|$. Applying the result of Lemma 14, we have

$$\begin{aligned} \mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] = \|\mathbf{v}\|^2 \tau_0^2 \sum_{i \in \mathcal{S}_0} p_1(r_i, s)(1 - p_1(r_i, s)) \\ + \|\mathbf{v}\|^2 \sum_{j=0}^{s-1} \tau_{j+1}^2 \sum_{i \in \mathcal{S}_{j+1}} p_2(r_i, s)\big(1 - p_2(r_i, s)\big) \end{aligned} \tag{25}$$

where $\tau_j \triangleq l_{j+1} - l_j$ for $j \in \{0, \cdots, s\}$.

The variance of $Q_s(\mathbf{v})$ can be also expressed as

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] = \|\mathbf{v}\|^2\Big(\sum_{r_i \in \mathcal{I}_0}(2^{-s} - r_i)r_i + \sum_{j=0}^{s-1}\sum_{r_i \in \mathcal{I}_{j+1}}(2^{j+1-s} - r_i)(r_i - 2^{j-s})\Big). \quad (26)$$

where $\mathcal{I}_0 \triangleq [0, 2^{-s}]$ and $\mathcal{I}_{j+1} \triangleq [2^{j-s}, 2^{j+1-s}]$ for $j = 0, \cdots, s-1$.

Inspired by the proof in (Horváth et al., 2019), we can find k that satisfies $(2^{j+1-s} - r)(r - 2^{j-s}) \leq kr^2$ for $r \in \mathcal{I}_{j+1}$. Expressing $r = 2^{j-s}\theta$, we can find k through solving

$$k = \max_{1 \leq \theta \leq 2}(2 - \theta)(\theta - 1)/\theta^2 = 1/8. \quad (27)$$

Substituting (27) into (26), an upper bound on $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2]$ is given by

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \|\mathbf{v}\|^2(1/8 + \sum_{r_i \in \mathcal{I}_0}(2^{-s} - r_i)r_i).$$

In the following, we derive three different bounds on $\sum_{r_i \in \mathcal{I}_0}(2^{-s} - r_i)r_i$, each gives us an upper bound on $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2]$.

Lemma 15 *Let $p \in (0, 1)$ and $r \in \mathcal{I}_0$. Then we have $r(2^{-s} - r) \leq K_p 2^{(-2+p)s} r^p$ where*

$$K_p = \Big(\frac{1/p}{2/p - 1}\Big)\Big(\frac{1/p - 1}{2/p - 1}\Big)^{(1-p)}. \quad (28)$$

Proof We can find K_p through solving $K_p = 2^{(2-p)s}\max_{r \in \mathcal{I}_0} r(2^{-s} - r)/r^p$. Expressing the optimization variable as $r = 2^{-s}\theta^{1/p}$, K_p can be obtained by solving this problem:

$$K_p = \max_{0 \leq \theta \leq 1} \theta^{1/p-1} - \theta^{2/p-1}. \quad (29)$$

We can solve (29) and obtain the optimal solution $\theta^* = \big(\frac{1/p-1}{2/p-1}\big)^p$. Substituting θ^* into (29), we obtain (28). ■

Note that using Hölder's inequality, we have

$$\sum_{r_i \in \mathcal{I}_0} r_i^p = \sum_{i \in \mathcal{S}_0} \frac{|v_i|^p}{\|\mathbf{v}\|^p} \leq \Big(\frac{\|\mathbf{v}\|_p}{\|\mathbf{v}\|}\Big)^p \leq d^{1-p/2}.$$

This gives us an upper bound on $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2]$:

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \|\mathbf{v}\|^2(1/8 + K_p 2^{(-2+p)s} d^{1-p/2}).$$

Furthermore, note that $r(2^{-s} - r) \leq 2^{-s}r$ and $r(2^{-s} - r) \leq 2^{-s}(2^{-s} - r)$ for $r \in \mathcal{I}_0$. This leads to the following upper bound on $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2]$:

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \|\mathbf{v}\|^2(1/8 + \min(2^{-s}\sqrt{d}, 2^{-2s}d)).$$

The final upper bound is obtained using the following lemma.

Lemma 16 *Let $r \in \mathcal{I}_0$. Then we have $r(2^{-s} - r) \leq \hat{K} \sin\left(2^s \pi r\right)$ where $\hat{K} = 2^{-2s}/\pi$.*

Proof We can find $\hat{K}$ through solving $\hat{K} = \sup_{r \in (0, 2^{-s})} r(2^{-s} - r)/\sin(2^s \pi r)$. Expressing the optimization variable as $r = 2^{-s}\theta$, $\hat{K}$ can be obtained by solving this problem:

$$\hat{K} = 2^{-2s} \sup_{0<\theta<1} \theta(1-\theta)/\sin(\pi\theta) = 2^{-2s}/\pi. \tag{30}$$

■

Finally an upper bound on $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2]$ is given by:

$$\begin{aligned}\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] &\leq \|\mathbf{v}\|^2\Big(1/8 + 2^{-2s}|\mathcal{S}_0|/\pi \sin\left(2^s\pi/|\mathcal{S}_0| \sum_{i\in\mathcal{S}_0} \|v_i\|/\|\mathbf{v}\|\right)\Big) \\ &\leq \|\mathbf{v}\|^2\big(1/8 + 2^{-2s} d/\pi\big) \end{aligned} \tag{31}$$

where the first inequality follows from Jensen's inequality.

Combining these bounds, we have

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \epsilon_Q \|\mathbf{v}\|^2 \tag{32}$$

where $\epsilon_Q = 1/8 + \inf_{0<p<1} K_p 2^{(-2+p)s} d^{1-p/2}$ with $K_p = \big(\frac{1/p}{2/p-1}\big)\big(\frac{1/p-1}{2/p-1}\big)^{(1-p)}$.

Note that the optimal p to minimize ϵ_Q is obtained by minimizing:

$$\Xi(p) = \big(\frac{1/p}{2/p-1}\big)\big(\frac{1/p}{2/p-1}\big)^{1-p} \delta^{1-p}$$

where $\delta = \sqrt{d}/2^s$.

Differentiating $\Xi(p)$, the optimal p^* is given by

$$p^* = \begin{cases} \frac{\delta-2}{\delta-1}, & \delta \geq 2 \\ 0, & \delta < 2. \end{cases} \tag{33}$$

Substituting (33) into (32) gives (3).

For the second part of theorem, substituting $\tau_0 = 2^{-s}$ and $\tau_j = 2^{j-1-s}$ for $j \in \{1, \cdots, s\}$ into (25), we have

$$\begin{aligned}\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] &= \|\mathbf{v}\|^2 2^{-2s} \sum_{i\in\mathcal{S}_0} p_1(r_i, s)(1 - p_1(r_i, s)) \\ &\quad + \|\mathbf{v}\|^2 \sum_{j=0}^{s-1} 2^{2(j-s)} \sum_{i\in\mathcal{S}_{j+1}} p_2(r_i, s)\big(1 - p_2(r_i, s)\big) \\ &\leq \|\mathbf{v}\|^2 2^{-2s} \sum_{i\in\mathcal{S}_0} p_1(r_i, s) \\ &\quad + \|\mathbf{v}\|^2 \sum_{j=0}^{s-1} 2^{2(j-s)} \sum_{i\in\mathcal{S}_{j+1}} p_2(r_i, s) \end{aligned} \tag{34}$$

We first note that $\sum_{i\in\mathcal{S}_0} p_1(r_i,s) \le d$ and $\sum_{i\in\mathcal{S}_{j+1}} p_2(r_i,s) \le d$ for all j, *i.e.,* an upper bound on the variance of $Q_s(\mathbf{v})$ is given by $\mathbb{E}[\|Q_s(\mathbf{v})-\mathbf{v}\|^2] \le \|\mathbf{v}\|^2 d/3(2^{-2s+1}+1)$. Furthermore, we have

$$\sum_{i\in\mathcal{S}_0} p_1\big(r_i,s\big) \le \min\{d_0, 2^s\sqrt{d_0}\} \tag{35}$$

since $\frac{\sum_{i\in\mathcal{S}_0}|v_i|}{\|\mathbf{v}\|} \le \sqrt{d_0}$. Similarly, we have

$$\sum_{i\in\mathcal{S}_{j+1}} p_2\big(r_i,s\big) \le \min\{d_{j+1}, 2^{(s-j)}\sqrt{d_{j+1}}\}. \tag{36}$$

Considering the variance expression (26), note that $(2^{-s}-r)r_i \le 2^{-2s}/4$ for $r\in\mathcal{I}_0$ and $(2^{j+1-s}-r)(r-2^{j-s}) \le 2^{2j-2s}/4$ for $r\in\mathcal{I}_{j+1}$ for all j. This gives us an upper bound:

$$\mathbb{E}[\|Q_s(\mathbf{v})-\mathbf{v}\|^2] \le \|\mathbf{v}\|^2/4\big(2^{-2s}d_0 + \sum_{j=0}^{s-1} 2^{2j-2s}d_{j+1}\big). \tag{37}$$

Substituting the upper bounds in (35), (36), and (37) into (34), an upper bound on the variance of $Q_s(\mathbf{v})$ is given by

$$\begin{aligned}\mathbb{E}[\|Q_s(\mathbf{v})-\mathbf{v}\|^2] \le{}& \min\{2^{-2s}d_0/4, 2^{-s}\sqrt{d_0}\}\|\mathbf{v}\|^2 \\ &+ \sum_{j=0}^{s-1}\min\{2^{2(j-s)}d_{j+1}/4, 2^{j-s}\sqrt{d_{j+1}}\}\|\mathbf{v}\|^2.\end{aligned} \tag{38}$$

The upper bound in (38) cannot be used directly as it depends on $\{d_0,\cdots,d_s\}$. Note that d_j's depend on quantization intervals. In the following, we obtain an upper bound on $\mathbb{E}[\|Q_s(\mathbf{v})-\mathbf{v}\|^2]$, which depends only on d and s. To do so, we need to use this lemma inspired by (Alistarh et al., 2017, Lemma A.5): Let $\|\cdot\|_0$ count the number of nonzero components.

Lemma 17 *Let $\mathbf{v}\in\mathbb{R}^d$. The expected number of nonzeros in $Q_s(\mathbf{v})$ is bounded above by*

$$\mathbb{E}[\|Q_s(\mathbf{v})\|_0] \le 2^{2s} + \sqrt{d_0}2^s.$$

Proof Note that $d-d_0 \le 2^{2s}$ since

$$(d-d_0)2^{-2s} \le \sum_{i\notin\mathcal{S}_0} r_i^2 \le 1. \tag{39}$$

For each $i\in\mathcal{S}_0$, $Q_s(v_i)$ becomes zero with probability $1-2^s r_i$, which results in

$$\begin{aligned}\mathbb{E}[\|Q_s(\mathbf{v})\|_0] &\le d - d_0 + \sum_{i\in\mathcal{S}_0} r_i 2^s \\ &\le 2^{2s} + \sqrt{d_0}2^s.\end{aligned} \tag{40}$$

■

Using a similar argument as in the proof of Lemma 17, we have

$$d - d_0 - d_1 - \cdots - d_j \leq 2^{2(s-j)} \tag{41}$$

for $j = 0, 1, \cdots, s-1$. Define $b_j \triangleq d - 2^{2(s-j)}$ for $j = 0, \cdots, s-1$. Then

$$\begin{aligned} b_0 &\leq d_0 \\ b_1 &\leq d_1 + d_0 \\ &\vdots \quad \vdots \\ b_{s-1} &\leq d_0 + \cdots + d_{s-1}. \end{aligned} \tag{42}$$

Note that $d_s = d - d_0 - \cdots - d_{s-1}$.

We define

$$\begin{aligned} \tilde{d}_0 &\triangleq b_0 = d - 2^{2s} \\ \tilde{d}_1 &\triangleq b_1 - b_0 = 3 \cdot 2^{2(s-1)} \\ &\vdots \quad \vdots \\ \tilde{d}_{s-1} &\triangleq b_{s-1} - b_{s-2} = 12 \\ \tilde{d}_s &\triangleq d - \tilde{d}_0 - \tilde{d}_1 - \cdots - \tilde{d}_{s-1} = 4. \end{aligned} \tag{43}$$

Note that $\tilde{d}_0 \leq d_0$, $\tilde{d}_1 + \tilde{d}_0 \leq d_1 + d_0$, $\cdots$, $\tilde{d}_{s-1} + \cdots + \tilde{d}_0 \leq d_{s-1} + \cdots + d_0$, and $\tilde{d}_s + \cdots + \tilde{d}_0 = d_s + \cdots + d_0$.

Noting that the coefficients of the additive terms in the upper bound in (38) are monotonically increasing with j, we can find an upper bound on $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2]$ by replacing $(d_0, \cdots, d_s)$ with $(\tilde{d}_0, \cdots, \tilde{d}_s)$ in (38), which completes the proof.

C. Proof of Theorem 5 (Code-length Bound)

Let $|\cdot|$ denote the length of a binary string. In this section, we find an upper bound on $\mathbb{E}[|\text{ENCODE}(\mathbf{v})|]$, *i.e.,* the expected number of communication bits per iteration. Recall from Appendix A that the quantized gradient $Q_s(\mathbf{v})$ is determined by the tuple $(\|\mathbf{v}\|, \boldsymbol{\rho}, \mathbf{h})$. Write $i_1 < i_2 < \cdots < i_{\|\mathbf{h}\|_0}$ for the indices of the $\|\mathbf{h}\|_0$ nonzero entries of $\mathbf{h}$. Let $i_0 = 0$.

The encoding produced by ENCODE($\mathbf{v}$) can be partitioned into two parts, R and E, such that, for $j = 1, \ldots, \|\mathbf{h}\|_0$,

- R contains the codewords $\text{ERC}(i_j - i_{j-1})$ encoding the runs of zeros; and
- E contains the sign bits and codewords $\text{ERC}(\log\{2^{s+1} h_{i_j}\})$ encoding the normalized quantized coordinates.

Note that $\|[i_1, i_2 - i_1, \cdots, i_{\|\mathbf{h}\|_0} - i_{\|\mathbf{h}\|_0 - 1}]\|_1 \leq d$. Thus, by (Alistarh et al., 2017, Lemma A.3), the properties of Elias encoding imply that

$$|R| \leq \|\mathbf{h}\|_0 + (1 + o(1))\|\mathbf{h}\|_0 \log\Big(\frac{d}{\|\mathbf{h}\|_0}\Big). \tag{44}$$

We now turn to bounding $|E|$. The following result in inspired by (Alistarh et al., 2017, Lemma A.3).

Lemma 18 *Fix a vector* $\mathbf{q}$ *such that* $\|\mathbf{q}\|_p^p \le P$, *let* $i_1 < i_2 < \dots i_{\|\mathbf{q}\|_0}$ *be the indices of its* $\|\mathbf{q}\|_0$ *nonzero entries, and assume each nonzero entry is of form of* 2^k, *for some positive integer* k. *Then*

$$\sum_{j=1}^{\|\mathbf{q}\|_0} |\mathrm{ERC}(\log(q_{i_j}))| \le (1+o(1))\log\big(\frac{1}{p}\big) + \|\mathbf{q}\|_0 + (1+o(1))\|\mathbf{q}\|_0 \log\log\Big(\frac{P}{\|\mathbf{q}\|_0}\Big).$$

Proof Applying property (1) for ERC (end of Appendix A), we have

$$\begin{aligned}
\sum_{j=1}^{\|\mathbf{q}\|_0} |\mathrm{ERC}(\log(q_{i_j}))| &\le (1+o(1)) \sum_{j=1}^{\|\mathbf{q}\|_0} \log\log q_{i_j} + \|\mathbf{q}\|_0 \\
&\le (1+o(1))\log\big(\frac{1}{p}\big) + \|\mathbf{q}\|_0 \\
&\quad + (1+o(1)) \sum_{j=1}^{\|\mathbf{q}\|_0} \log\log q_{i_j}^p \\
&\le (1+o(1))\log\big(\frac{1}{p}\big) + \|\mathbf{q}\|_0 \\
&\quad + (1+o(1))\|\mathbf{q}\|_0 \log\log\big(\frac{P}{\|\mathbf{q}\|_0}\big)
\end{aligned}$$

where the last bound is obtained by Jensen's inequality. ■

Taking $\mathbf{q} = 2^{s+1}\mathbf{h}$, we note that $\|\mathbf{q}\|^2 = 2^{2s+2}\|\mathbf{h}\|^2$ and

$$\begin{aligned}
\|\mathbf{h}\|^2 &\le \sum_{i=1}^{d} \Big(\frac{v_i}{\|\mathbf{v}\|} + \frac{1}{2^s}\Big)^2 \\
&\le 2\sum_{i=1}^{d} \Big(\frac{v_i^2}{\|\mathbf{v}\|^2} + \frac{1}{2^{2s}}\Big) = 2\big(1 + \frac{d}{2^{2s}}\big).
\end{aligned} \tag{45}$$

By Lemma 18 applied to $\mathbf{q}$ and the upper bound (45),

$$\begin{aligned}
|E| &\le -(1+o(1)) + 2\|\mathbf{h}\|_0 \\
&\quad + (1+o(1))\|\mathbf{h}\|_0 \log\log\Big(\frac{2^{2s+2}\|\mathbf{h}\|^2}{\|\mathbf{h}\|_0}\Big).
\end{aligned} \tag{46}$$

Combining (44) and (46), we obtain an upper bound on the expected code-length:

$$\mathbb{E}[|\mathrm{ENCODE}(\mathbf{v})|] \le N(\|\mathbf{h}\|_0) \tag{47}$$

where

$$N(\|\mathbf{h}\|_0) = b + 3\|\mathbf{h}\|_0 + (1+o(1))\mathbb{E}\big[\|\mathbf{h}\|_0 \log\big(\frac{d}{\|\mathbf{h}\|_0}\big)\big] \\ - (1+o(1)) + (1+o(1))\mathbb{E}\big[\|\mathbf{h}\|_0 \log\log\big(\frac{8(2^{2s}+d)}{\|\mathbf{h}\|_0}\big)\big]. \tag{48}$$

It is not difficult to show that, for all $k>0$, $g_1(x) \triangleq x\log\big(\frac{k}{x}\big)$ is concave. Note that g_1 is an increasing function up to $x = k/e$.

Defining $g_2(x) \triangleq x\log\log\big(\frac{C}{x}\big)$ and taking the second derivative, we have

$$g_2''(x) = -\big(x\ln(2)\ln(C/x)\big)^{-1}\Big(1+\big(\ln(C/x)\big)^{-1}\Big). \tag{49}$$

Hence g_2 is also concave on $x < C$. Furthermore, g_2 is increasing up to some $C/5 < x^* < C/4$. We note that $\mathbb{E}[\|\mathbf{h}\|_0] \le 2^{2s} + \sqrt{d}2^s$ following Lemma 17. By assumption $2^{2s} + \sqrt{d}2^s \le d/e$, and so, Jensen's inequality and (47) lead us to (4).

D. Proof of Theorem 13 (Expected Number of Communication Bits)

Assuming $\frac{2\hat{B}}{K\epsilon^2} > \frac{\beta}{\epsilon}$, then $T_\epsilon = O\big(\frac{2\hat{B}}{K\epsilon^2}R^2\big)$. Ignoring all but terms depending on d and s, we have $T_\epsilon = O(\hat{B}/\epsilon^2)$. Following Theorems 4 and 5 for NUQSGD, $\zeta_{\text{NUQSGD},\epsilon} = O(N_Q\epsilon_Q B/\epsilon^2)$. For QSGD, following the results of Alistarh et al. (2017), $\zeta_{\text{QSGD},\epsilon} = O(\tilde{N}_Q\tilde{\epsilon}_Q B/\epsilon^2)$ where $\tilde{N}_Q = 3(s^2+s\sqrt{d}) + (\frac{3}{2}+o(1))(s^2+s\sqrt{d})\log\Big(\frac{2(s^2+d)}{s^2+\sqrt{d}}\Big) + b$ and $\tilde{\epsilon}_Q = \min\big(\frac{d}{s^2}, \frac{\sqrt{d}}{s}\big)$.

In overparameterized networks, where $d \ge 2^{2s+1}$, we have $\epsilon_Q = 2^{-s}\sqrt{d-2^{2s}} + O(s)$ and $\tilde{\epsilon}_Q = \sqrt{d}/s$. Furthermore, for sufficiently large d, N_Q and $\tilde{N}_Q$ are given by $O\big(2^s\sqrt{d}\log\big(\frac{\sqrt{d}}{2^s}\big)\big)$ and $O\big(s\sqrt{d}\log(\sqrt{d})\big)$, respectively.

E. Optimal Level for the Special Case with $s = 1$

Corollary 19 (Optimal level) *For the special case with $s = 1$, the optimal level to minimize the worst-case bound obtained from problem $\mathcal{P}_1$ is given by $l_1^* = 1/2$.*

Proof For $s = 1$, problem $\mathcal{P}_1$ is given by

$$\mathcal{P}_0 : \max_{(d_0,d_1)} \ (\tau_0^2 d_0 + \tau_1^2 d_1)/4 \\ \text{subject to } d - d_0 \le (1/l_1)^2, \\ d_0 + d_1 \le d, \\ d_0 \ge 0,\ d_1 \ge 0.$$

Note that the objective of $\mathcal{P}_0$ is monotonically increasing in (d_0, d_1). It is not difficult to verify that the optimal (d_0^*, d_1^*) is a corner point on the boundary line of the feasibility region of $\mathcal{P}_0$. Geometrical representation shows that that candidates for an optimal solution are $(d-(1/l_1)^2, (1/l_1)^2)$ and $(d, 0)$. Substituting into the objective of $\mathcal{P}_0$, the optimal value of $\mathcal{P}_0$ is given by

$$\epsilon_{LP}^* = \max\{\tau_0^2 d, \tau_0^2 d + \tau_1^2/\tau_0^2 - 1\}/4. \tag{50}$$

Finally, note that $\tau_0 = \tau_1 = 1/2$ minimizes the optimal value of $\mathcal{P}_0$ (50). ■

F. Lower Bound in the Regime of Large s

In the following theorem, we show that for exponentially spaced levels with $p = 0.5$, if s is large enough, there exists a distribution of points such that the variance is in $1/8\|\mathbf{v}\|^2$.

Theorem 20 (Lower bound for large s) *Let $d \in \mathbb{Z}^{>0}$ and let $(0, 2^{-s}, \cdots, 1/2, 1)$ denote the sequence of quantization levels. Provided $s \geq \log(\sqrt{d})$, there exists a vector $\mathbf{v} \in \mathbb{R}^d$ such that the variance of unbiased quantization of $\mathbf{v}$ is given by $\|\mathbf{v}\|^2/8$.*

Proof Consider $\mathbf{v}_0 = [r, r, \cdots, r]^T$ for $r \neq 0$. The normalized coordinates is $\hat{\mathbf{v}}_0 = [1/\sqrt{d}, \cdots, 1/\sqrt{d}]^T$. Suppose s is large enough such that $2^{j-s} = \frac{3}{4\sqrt{d}}$ for some $j = 0, \cdots, s-1$. We can compute of the variance and obtain $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] = \|\mathbf{v}\|^2/8$. ■

G. Asynchronous Variant of NUQSGD

Asynchronous parameter-server model is an important setting that provides additional flexibility for distributed training of large models. In this section, we extend our consideration to asynchronous variant of NUQSGD when a star-shaped parameter-server setting is used for training. The star machine is a master processor, while other processors serve as workers. The master aggregates stochastic gradients received from workers and updates the model parameters. We consider a consistent scenario where workers cannot read the values of model parameters during the update step. This is a valid model when computer clusters are used for distributed training (Lian et al., 2015).

On smooth and possibly nonconvex problems in a consistent parameter-server setting, we establish convergence guarantees for asynchronous variant of NUQSGD along the lines of, *e.g.,* (Lian et al., 2015, Theorem 1):

Theorem 21 (NUQSGD for asynchronous smooth nonconvex optimization) *Let $f : \Omega \to \mathbb{R}$ denote a possibly nonconvex and β-smooth function. Let $\mathbf{w}_0 \in \Omega$ denote an initial point, ϵ_Q be defined as in Theorem 4 and $f^* = \inf_{\mathbf{w}\in\Omega} f(\mathbf{w})$. Suppose K workers each compute an unbiased stochastic gradient with mini-batch size J and second-moment bound B, compress the stochastic gradient using nonuniform quantization, and send to a master processor for T iterations. Suppose the delay for model parameters used in evaluation of a stochastic gradient at each iteration is upper bounded by τ. Provided that the sequence of learning rates satisfies $\beta J\alpha_t + 2\beta^2 J^2 \tau \alpha_t \sum_{i=1}^{\tau} \alpha_{t+i} \leq 1$, for all $t = 1, 2, \cdots$, we have*

$$\sum_{t=1}^{T} \frac{\alpha_t \mathbb{E}[\|\nabla f(\mathbf{w}_t)\|^2]}{\sum_{t=1}^{T} \alpha_t} \leq \frac{2(f(\mathbf{w}_0) - f^*) + \lambda(1 + \epsilon_Q)B/K}{J \sum_{t=1}^{T} \alpha_t}$$

where $\lambda = \sum_{t=1}^{T} \left(\alpha_t^2 J\beta + 2\beta^2 J 2\alpha_t \sum_{i=t-\tau}^{t-1} \alpha_i^2\right)$.

H. NUQSGD for Decentralized Training

In a networked distributed system, decentralized algorithms are deployed when either processors cannot establish a fully connected topology due to communication barriers or the network suffers from high latency. When the network is bandwidth-limited as well, communication-efficient variants of decentralized parallel SGD is a promising solution to tackle both high latency and limited bandwidth to train deep models in a networked system (Tang et al., 2018). In this section, we show that NUQSGD can be integrated with extrapolation compression decentralized parallel SGD (ECD-PSGD) proposed in (Tang et al., 2018).[6] In decentralized optimization, we consider the following problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{K}\sum_{i=1}^{K} f_i(\mathbf{w})$$

where $f_i(\mathbf{w}) = \mathbb{E}_{\xi\sim\mathcal{D}_i}[f(\mathbf{w};\xi)]$, $\mathcal{D}_i$ is the local data distribution stored in processor i, and $f(\mathbf{w};\xi)$ is the loss of a model described by $\mathbf{w}$ on example (mini-batch) ξ.

At iteration t of D-PSGD, each computing processor computes its local stochastic gradient, *e.g.,* processor i computes $g_i(\mathbf{w}_t^{(i)})$ with $\mathbb{E}_{\xi_t^{(i)}\sim\mathcal{D}_i}[g_i(\mathbf{w}_t^{(i)})] = \nabla f_i(\mathbf{w}_t^{(i)})$ where $\mathbf{w}_t^{(i)}$ and $\xi_t^{(i)}$ are the local parameter vector and samples on processor i, respectively. Processor i then fetches its neighbours' parameter vectors and updates its local parameter vector using $\mathbf{w}_{t+1/2}^{(i)} = \sum_{j=1}^{K} W_{i,j}\mathbf{w}_t^{(j)}$ where $W \in \mathbb{R}^{K\times K}$ is a symmetric doubly stochastic matrix, *i.e.,* $W = W^T$ and $\sum_{j=1}^{K} W_{i,j} = 1$ for all i. Note that $W_{i,j} \geq 0$ in general and $W_{i,j} = 0$ means that processors i and j are not connected. Finally, processor i updates its local parameter vector using the update rule $\mathbf{w}_{t+1}^{(i)} \leftarrow \mathbf{w}_{t+1/2}^{(i)} - \alpha g_i(\mathbf{w}_t^{(i)})$. ECD-PSGD integrated with NUQSGD is described in Algorithm 3.

In this section, we make the following assumptions:

- Given W, $\rho < 1$ where ρ denotes the second largest eigenvalue of W.
- The variance of local stochastic gradients are bounded. In particular, there are σ^2, ζ^2 such that for all $\mathbf{w}$ and i:

$$\mathbb{E}[\|g_i(\mathbf{w}) - \nabla f_i(\mathbf{w})\|^2] \leq \sigma^2,$$
$$\frac{1}{K}\sum_{i=1}^{K} \|\nabla f_i(\mathbf{w}) - \nabla F(\mathbf{w})\|^2 \leq \zeta^2.$$

On smooth and possibly nonconvex problems and under assumptions above, we establish convergence guarantees for ECD-PSGD integrated with NUQSGD along the lines of, *e.g.,* (Tang et al., 2018, Corollary 4):

Theorem 22 (NUQSGD for decentralized smooth nonconvex optimization) *Let f_i denote a possibly nonconvex and β-smooth function for all i. Let ϵ_Q be defined as in*

6. Similarly, NUQSGD can be integrated with difference compressions D-PSD (DCD-PSD). However, DCD-PSD requires stringent constraints on stochastic compression scheme, which may fail under an aggressive compression. Hence, we focus on ECD-PSGD in this paper.

Input: local data, weight matrix W, initial parameter vector $\mathbf{w}_1^{(i)} = \mathbf{w}_1$, initial estimate $\tilde{\mathbf{w}}_1^{(i)} = \mathbf{w}_1$, learning rate α, and K

for $t = 1$ ***to*** T **do**
  for $i = 1$ ***to*** K **do** // each transmitter processor (in parallel)
    Compute $g_i(\mathbf{w}_t^{(i)})$; // local stochastic gradient
    Compute $\mathbf{w}_{t+1/2}^{(i)} = \sum_{j=1}^{K} W_{i,j}\tilde{\mathbf{w}}_t^{(j)}$; // neighbourhood weighted average
    Update $\mathbf{w}_{t+1}^{(i)} \leftarrow \mathbf{w}_{t+1/2}^{(i)} - \alpha g_i(\mathbf{w}_t^{(i)})$;
    Compute $\mathbf{z}_t^{(i)} = (1 - t/2)\mathbf{w}_t^{(i)} + t/2\mathbf{w}_{t+1}^{(i)}$;
    Encode $c_t^{(i)} \leftarrow \mathrm{ENCODE}(\mathbf{z}_t^{(i)})$;
    Broadcast $c_t^{(i)}$ to connected neighbours;
    Receive $c_t^{(j)}$ from connected neighbours;
    $\tilde{\mathbf{w}}_{t+1}^{(j)} \leftarrow (1 - 2/t)\tilde{\mathbf{w}}_t^{(j)} + 2/t\mathrm{DECODE}(c_t^{(j)})$; // update estimates for connected neighbours
Aggregate $\overline{\mathbf{w}}_T \leftarrow 1/K \sum_{i=1}^{K} \mathbf{w}_T^{(i)}$;

Algorithm 3: ECD-PSGD with NUQSGD.

Theorem 4. Suppose that Algorithm 3 is executed for T iterations with a learning rate $\alpha < (12\beta/(1-\rho) + \sigma\sqrt{T/K} + \zeta^{2/3}T^{1/3})^{-1}$ on K processors, each with access to independent and local stochastic gradients with variance bound σ^2. Then we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\Big[\Big\|\sum_{i=1}^{K}\frac{\nabla f_i(\sum_{i=1}^{K}\mathbf{w}_t^{(i)}/K)}{K}\Big\|^2\Big] \lesssim \frac{\sigma(1+(1+\epsilon_Q)R^2\log(T)/K)}{\sqrt{KT}} + \frac{(1+\epsilon_Q)R^2\log(T)}{T} + \zeta^{2/3}(1+(1+\epsilon_Q)R^2\log(T)/K)/T^{2/3} + 1/T$$

where $R^2 = \max_{1\le i\le K}\max_{1\le t\le T}\|\mathbf{z}_t^{(i)}\|^2$.

I. NUQSGD vs QSGDinf

In this section, we show that there exist vectors for which the variance of quantization under NUQSGD is guaranteed to be smaller than that under QSGDinf. Intuitively, with the same communication budget, NUQSGD is guaranteed to outperform QSGDinf in terms of variance for dense vectors with a unique dominant coordinate.

Theorem 23 (NUQSGD vs QSGDinf) *Let $\mathbf{v} \in \mathbb{R}^d$ be such that $v_i = 1$ and $v_j = \Theta(1/d)$ for $j \neq i$. Provided that d and s are large enough to ensure $\frac{K_1}{(d-1)\sqrt{1+K_2^2/(d-1)}} < 2^{-s}$ and $(1+K_1^2/(d-1))K_1(0.25K_1/(d-1)+2^{-s}) < K_2(1/s - K_1/(d-1))$ for some $K_2 < K_1 = o(d)$, NUQSGD is guaranteed to have smaller variance than QSGDinf.*

Proof Our proof argument is based on establishing a lower bound on the variance of QSGDinf and an upper bound on the variance of NUQSGD. Denote the variance quantization of $\mathbf{v}$ using QSGDinf and NUQSGD by $\Delta_{\mathrm{Qinf}} \triangleq \mathbb{E}[\|Q_s^{\mathrm{Qinf}}(\mathbf{v}) - \mathbf{v}\|^2]$ and $\Delta_{\mathrm{NUQ}} \triangleq \mathbb{E}[\|Q_s^{\mathrm{NUQ}}(\mathbf{v}) - \mathbf{v}\|^2]$, respectively. Note that $\|\mathbf{v}\|_\infty = 1$. A lower bound on Δ_{Qinf} is given by

$$\Delta_{\mathrm{Qinf}} \ge K_2(1/s - K_1/(d-1)). \tag{51}$$

We can bound the Eulidean norm of $\mathbf{v}$ as:

$$\sqrt{1+K_2^2/(d-1)} \leq \|\mathbf{v}\| \leq \sqrt{1+K_1^2/(d-1)}.$$

Note that the assumption $\frac{K_1}{(d-1)\sqrt{1+K_2^2/(d-1)}} < 2^{-s}$ implies that the normalized coordinates $v_j/\|\mathbf{v}\| \in [0, 2^{-s}]$ for $j \neq i$. Then an upper bound on Δ_{NUQ} is given by

$$\begin{aligned}\Delta_{\text{NUQ}} &\leq \Big(1+\frac{K_1^2}{d-1}\Big)\Big(0.5\Big(1-\frac{1}{\sqrt{1+K_1^2/(d-1)}}\Big)+\frac{K_1 2^{-s}}{\sqrt{1+K_2^2/(d-1)}}\Big) \\ &\leq \Big(1+\frac{K_1^2}{d-1}\Big)K_1(0.25K_1/(d-1)+2^{-s})\end{aligned} \tag{52}$$

where we used $\sqrt{1+K_1^2/(d-1)} \geq 1+0.5K_1^2/(d-1)$ in the second inequality. ■

J. Additional Experiments

In this section, we present further experimental results in a similar setting to Section 5.

We first measure the variance and normalized variance at fixed snapshots during training by evaluating multiple gradient estimates using each quantization method (discussed in Section 5). All methods are evaluated on the same trajectory traversed by the single-GPU SGD. These plots answer this specific question: What would the variance of the first gradient estimate be if one were to train using SGD for any number of iterations then continue the optimization using another method? The entire future trajectory may change by taking a single good or bad step. We can study the variance along any trajectory. However, the trajectory of SGD is particularly interesting because it covers a subset of points in the parameter space that is likely to be traversed by any first-order optimizer. For multi-dimensional parameter space, we average the variance of each dimension.

Figure 11 (left), shows the variance of the gradient estimates on the trajectory of single-GPU SGD on CIFAR10. We observe that QSGD has particularly high variance, while QSGDinf and NUQSGD have lower variance than single-GPU SGD.

We also propose another measure of stochasticity, normalized variance, that is the variance normalized by the norm of the gradient. The mean normalized variance can be expressed as

$$\frac{\mathbb{E}_i[V_A[\partial l(\mathbf{w};\mathbf{z})/\partial w_i]]}{\mathbb{E}_i[\mathbb{E}_A[(\partial l(\mathbf{w};\mathbf{z})/\partial w_i)^2]]}$$

where $l(\mathbf{w};\mathbf{z})$ denotes the loss of the model parametrized by $\mathbf{w}$ on sample $\mathbf{z}$ and subscript A refers to randomness in the algorithm, *e.g.,* randomness in sampling and quantization. Normalized variance can be interpreted as the inverse of Signal to Noise Ratio (SNR) for each dimension. We argue that the noise in optimization is more troubling when it is significantly larger than the gradient. For sources of noise such as quantization that stay constant during training, their negative impact might only be observed when the norm of the gradient becomes small.

Figure 11 (right) shows the mean normalized variance of the gradient versus training iteration. Observe that the normalized variance for QSGD stays relatively constant while

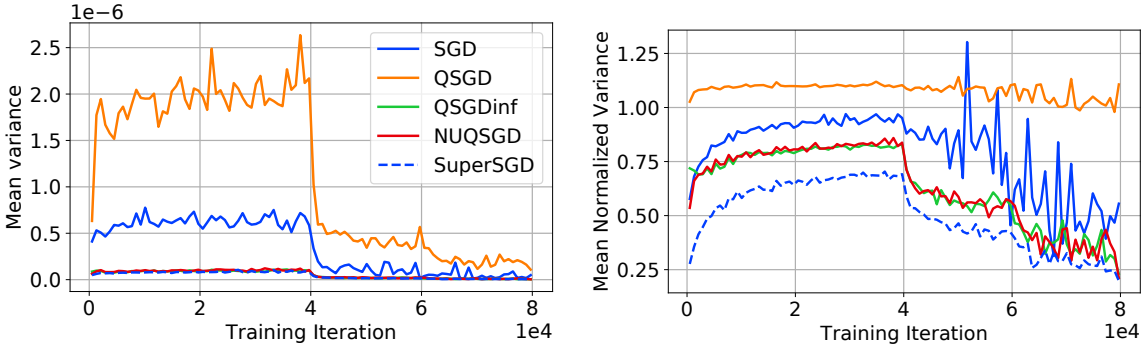

Figure 11: Estimated variance (left) and normalized variance (right) on CIFAR10 on the trajectory of single-GPU SGD. Variance is measured for fixed model snapshots during training. Notice that the variance for NUQSGD and QSGDinf is lower than SGD for almost all the training and it decreases after the learning rate drops. All methods except SGD simulate training using 8 GPUs. SuperSGD applies no quantization to the gradients and represents the lowest variance we could hope to achieve.

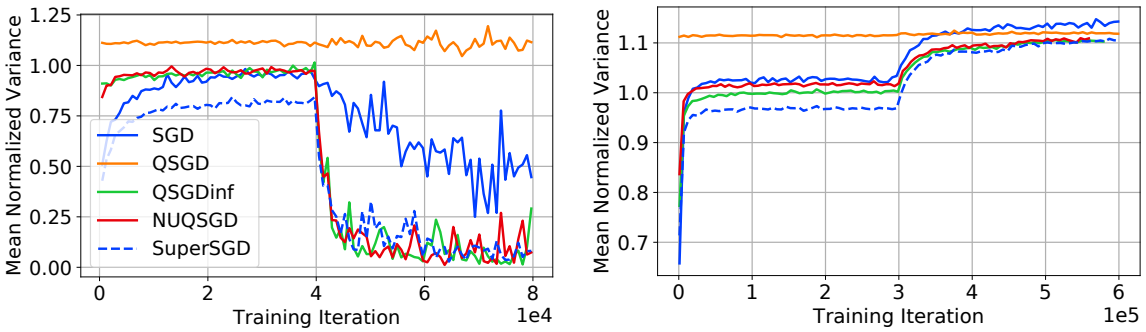

Figure 12: Estimated normalized variance on CIFAR10 (left) and ImageNet (right). For different methods, the variance is measured on their own trajectories. Note that the normalized variance of NUQSGD and QSGDinf is lower than SGD for almost the entire training. It decreases on CIFAR10 after the learning rate drops and does not grow as much as SGD on ImageNet. Since the variance depends on the optimization trajectory, these curves are not directly comparable. Rather the general trend should be studied.

the unnormalized variance of QSGD drops after the learning rate drops. It shows that the quantization noise of QSGD can cause slower convergence at the end of the training than at the beginning.

In Figure 12, we show the mean normalized variance of the gradient versus training iteration on CIFAR10 and ImageNet. For different methods, the variance is measured on their own trajectories. Since the variance depends on the optimization trajectory, these curves are not directly comparable. Rather the general trend should be studied.

Comparison with DGC. In Figures 15 and 16, we make a comparison with DGC (Lin et al., 2018).

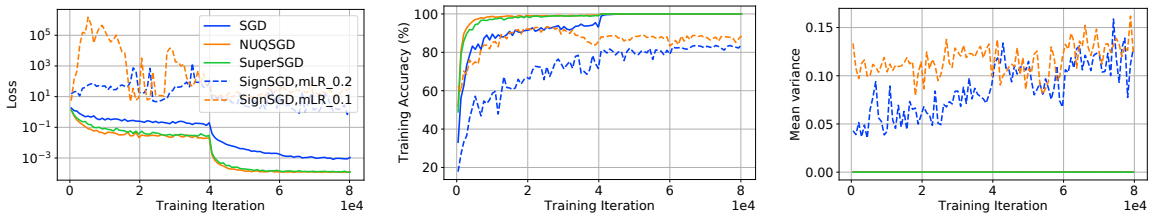

Figure 13: Comparison with SignSGD on CIFAR10. Training loss (left), training accuracy (middle), and estimated variance (right) for training on 8-GPUs. Setting is similar to Figure 4. SignSGD diverges with the standard learning rate for ResNets. We multiply the standard learning rate schedule by the constant mLR.

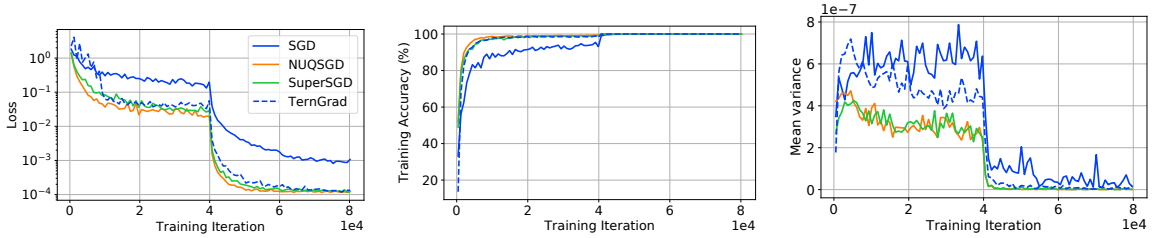

Figure 14: Comparison with TernGrad on CIFAR10. Training loss (left), training accuracy (middle), and estimated variance (right) for training on 8-GPUs. Setting is similar to Figure 4. The performance of TernGrad is inferior to NUQSGD.

ResNet152 Weak Scaling. In Figure 17, we present the weak scaling results for ResNet152/ImageNet. Each of the GPUs receives a batch of size 8, and we therefore scale up the global batch size by the number of nodes. The results exhibit the same superior scaling behavior for NUQSGD relative to the uncompressed baseline.

EF-SignSGD Convergence. In Figure 18, we present a performance comparison for NUQSGD variants (bucket size 512) and a convergent variant of EF-SignSGD with significant levels of parameter tuning for convergence. We believe this to be the first experiment to show convergence of the latter method at ImageNet scale, as the original paper only considers the CIFAR dataset. For convergence, we have tuned the choice of scaling factor and the granularity at which quantization is applied (bucket size). We have also considered learning rate tuning, but that did not appear to prevent divergence in the early stages of training for this model. We did not attempt warm start, since that would significantly decrease the practicality of the algorithm. We have found that bucket size 64 is the highest at which the algorithm will still converge on this model and dataset, and found 1-bit SGD scaling (Seide et al., 2014), which consists of taking sums over positives and over negatives for each bucket, to yield good results. The experiments are executed on a machine with 8 NVIDIA Titan X GPUs, and batch size 256, and can be found in Figure 18. Under these hyperparameter values the EF-SignSGD algorithm sends 128 bits per each bucket of 64 values (32 for each scaling factor, and 64 for the signs), doubling its baseline communication cost. Moreover, the GPU implementation is not as efficient, as error feedback must be computed and updated at every step, and there is less parallelism to leverage inside each bucket. This explains the

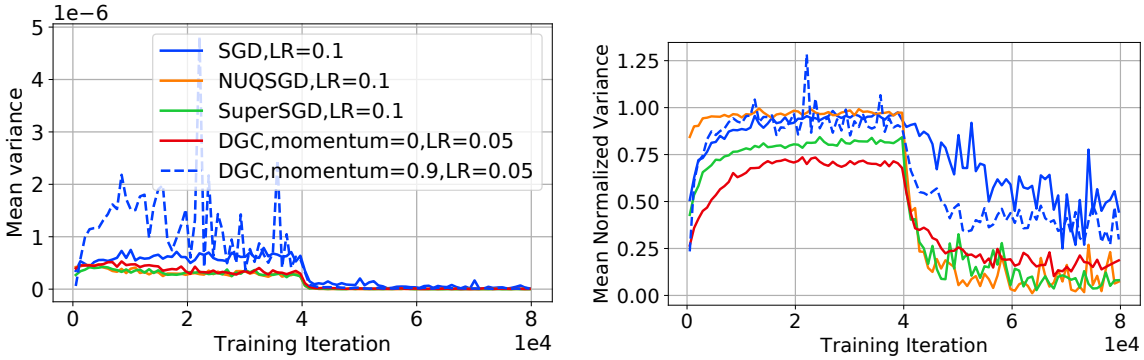

Figure 15: Estimated variance (left) and normalized variance (right) on CIFAR10 for ResNet110. We set the ratio of compression for DGC to be roughly the same as NUQSGD. In particular, we compare compression methods at compression ratio = 4/32, *i.e.,* NUQSGD with 4 bits and DGC at 12.5% compression. At this rate, both methods will have approximately the same communication cost, *i.e.,* comparison in simulation is representative of real-time performance. We tune the learning rate and momentum for DGC and show its best performance.

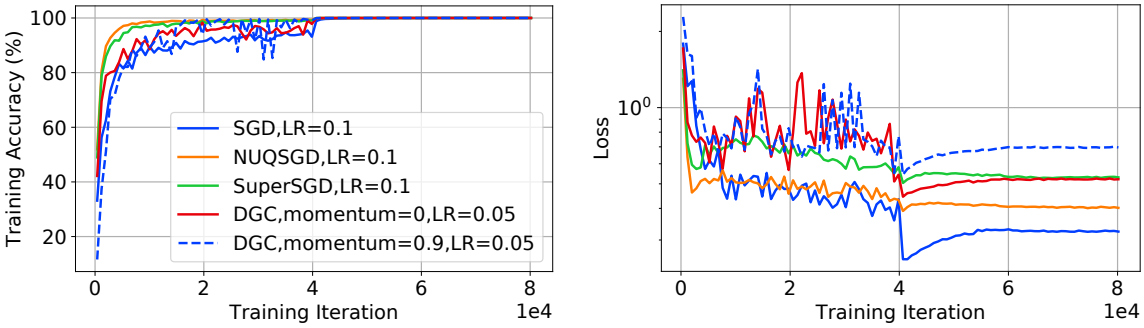

Figure 16: Training accuracy (left) and validation loss (right) on CIFAR10 for ResNet110. We set the ratio of compression for DGC to be roughly the same as NUQSGD. In particular, we compare compression methods at compression ratio = 4/32, *i.e.,* NUQSGD with 4 bits and DGC at 12.5% compression. At this rate, both methods will have approximately the same communication cost, *i.e.,* comparison in simulation is representative of real-time performance. We tune the learning rate and momentum for DGC and show its best performance.

fact that the end-to-end performance is in fact close to that of the 8-bit NUQSGD variant, and inferior to 4-bit NUQSGD.

Comparison under Small Mini-batch Size. In Figures 19, 20, and 21, we show the results when we train ResNet110 on CIFAR10 with mini-batch size 32 over 8 GPUs.

The number of quantization bits is set to 4. We observe a significant gap between the variance and accuracy of QSGD with those of QSGDinf and NUQSGD. QSGDinf and NUQSGD perform similarly and slightly outperform TernGrad in this setting.

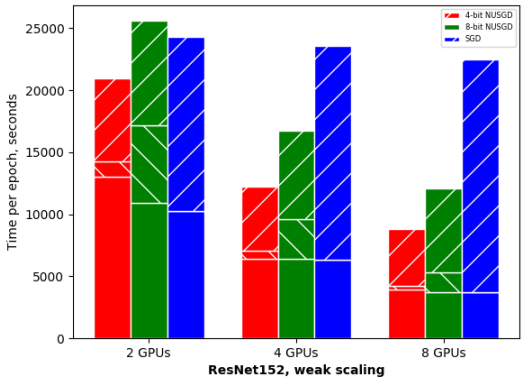

Figure 17: Scalability behavior for NUQSGD versus the full-precision baseline when training ResNet152 on ImageNet.

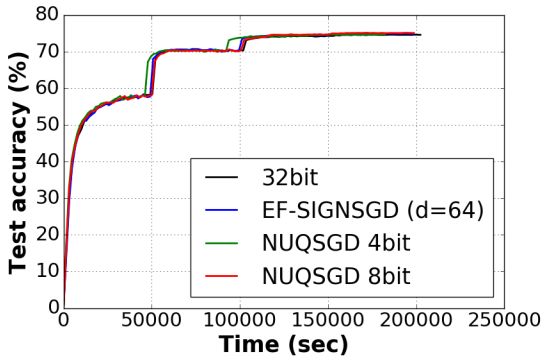

Figure 18: End-to-end training time for ResNet50/ImageNet for NUQSGD and EF-SignSGD versus the SGD baseline.

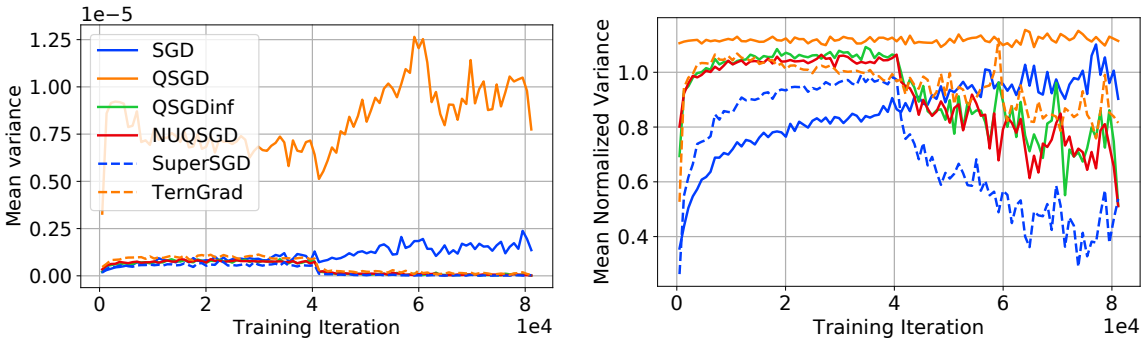

Figure 19: Estimated variance (left) and normalized variance (right) on CIFAR10 for ResNet110 with mini-batch size 32 over 8 GPUs. The number of quantization bits is set to 4.

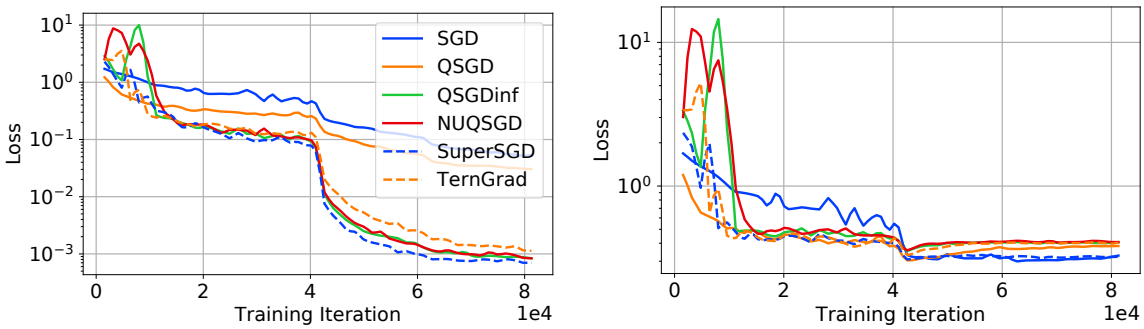

Figure 20: Training loss (left) and validation loss (right) on CIFAR10 for ResNet110 with mini-batch size 32 over 8 GPUs. The number of quantization bits is set to 4.

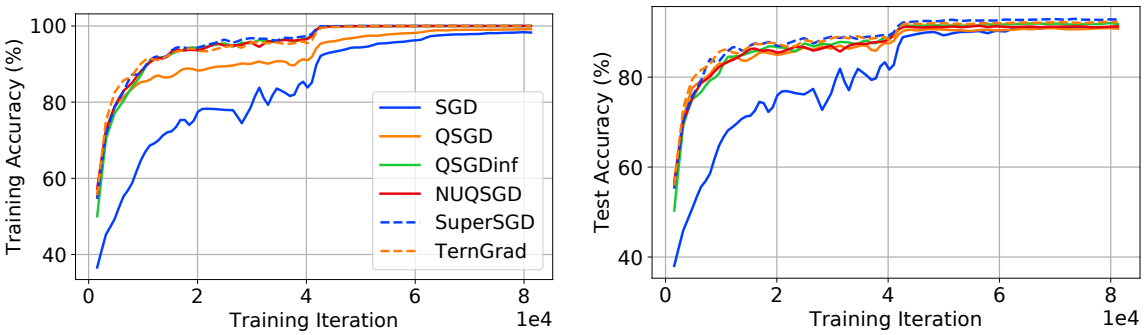

Figure 21: Training accuracy (left) and test accuracy (right) on CIFAR10 for ResNet110 with mini-batch size 32 over 8 GPUs. The number of quantization bits is set to 4.