NUQSGD: PROVABLY COMMUNICATION-EFFICIENT DATA-PARALLEL SGD VIA NONUNIFORM QUANTIZATION

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 on clusters to perform parallel model training. Alistarh et al. (2017) describe two variants of data-parallel SGD that quantize and encode gradients to lessen communication costs. For the first variant, QSGD, they provide strong theoretical guarantees. For the second variant, which we call QSGDinf, they demonstrate impressive empirical gains for distributed training of large neural networks. Building on their work, we propose an alternative scheme for quantizing gradients and show that it yields stronger theoretical guarantees than exist for QSGD while matching the empirical performance of QSGDinf.

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., 2016; 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, Alistarh et al. (2017) are able to establish a number of theoretical guarantees 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, similar to 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. In particular, 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.

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. In the overparametrized regime of interest, these bounds decrease rapidly as the number of quantization levels increases. Combined 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 slightly stronger than the one provided by QSGD.

To study how quantization affects convergence on state-of-the-art deep models, we compare NUQSGD, QSGD, and QSGDinf, 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 expected by our theoretical results. This 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.

Summary of Contributions.

- We establish stronger theoretical guarantees for the excess variance and communication costs of our gradient quantization method than those available for QSGD's uniform quantization method.
- We then establish stronger convergence guarantees for the resulting algorithm, NUQSGD, under standard assumptions.
- 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.

1.1 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. Bernstein et al. (2018) later

provided convergence guarantees for 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.

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 and gradient sparsity.

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. Convergence guarantees for TernGrad rely on a nonstandard gradient norm assumption. As discussed, Alistarh et al. (2017) proposed QSGD, a more general stochastic quantization scheme, for which they provide both theoretical guarantees and experimental validation (although for different variants of the same algorithm). We note that their implementation was only provided in Microsoft CNTK; by contrast, here we provide a more 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.

NUQSGD uses a logarithmic quantization scheme. Such schemes have long been used 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. 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, convex, 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 $(\mathscr{S}, \Sigma, \mu)$ 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 $g : \Omega \times \mathscr{S} \to \mathbb{R}^d$ 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 $\gamma = r\sqrt{2/T}/\sigma$. Given repeated and independent access to stochastic gradients with a variance upper bound σ^2, projected SGD satisfies*

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

Minibatched (with larger batch sizes) and data-parallel SGD are two common SGD variants used in practice to reduce variance and improve computational efficiency of conventional SGD.

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 framework models

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.

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 minibatch gradient of size B. Then, this update rule is essentially a minibatched update with size BK.

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 1 (Alistarh et al. 2017, Corollary 2.2)**.** *Let f, R, and γ be as defined in Theorem 1 and let $\varepsilon > 0$. 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 $\mathbb{E}\big[f(\frac{1}{T}\sum_{t=0}^T \mathbf{w}_t)\big] - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \varepsilon$ is $T_\varepsilon = O\big(R^2 \max(\frac{2B}{K\varepsilon^2}, \frac{\beta}{\varepsilon})\big)$.*

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 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.

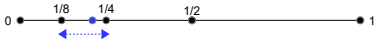

Figure 1: An example of nonuniform stochastic quantization with $s = 3$. The point between the arrows represents the value of the normalized coordinate.

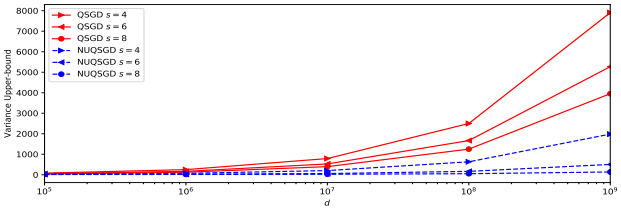

Figure 2: Variance upper bounds.

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 $\mathscr{L} = (l_0, l_1, \cdots, l_{s+1})$ denote the sequence of quantization levels, where $l_0 = 0 < l_1 < \cdots < l_{s+1} = 1$. For $r \in [0,1]$, let $s(r)$ and $p(r)$ satisfy $l_{s(r)} \leq r \leq l_{s(r)+1}$ and $r = \big(1 - p(r)\big) l_{s(r)} + p(r) l_{s(r)+1}$, respectively. Define $\tau(r) = l_{s(r)+1} - l_{s(r)}$. Note that $s(r) \in \{0, 1, \cdots, s\}$.

Definition 1. *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* $h_i(\mathbf{v}, s) = l_{s(r_i)}$ *with probability* $1 - p(r_i)$ *and* $h_i(\mathbf{v}, s) = l_{s(r_i)+1}$ *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 $\mathscr{L}$. In the following, we focus on a special case of nonuniform quantization with $\hat{\mathscr{L}} = (0, 1/2^s, \cdots, 2^{s-1}/2^s, 1)$ as the quantization 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 2, 3, 4, and 5 are provided in Appendices B, C, D, and E respectively.

Theorem 2 (Variance bound). *Let $\mathbf{v} \in \mathbb{R}^d$. The nonuniform quantization of $\mathbf{v}$ satisfies $\mathbb{E}[Q_s(\mathbf{v})] = \mathbf{v}$. Furthermore, provided that $s \leq \log(d)/2$, we have*

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

where $\varepsilon_Q = \min\{\min\{2^{-2s}(d-2^{2s}), 2^{-s}\sqrt{d-2^{2s}}\} + O(s), d/3(2^{-2s+1}+1)\}$.

The result in Theorem 2 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 $\varepsilon_Q \eta$. In the range of interest where d is sufficiently large, *i.e.,* $s = o(\log(d))$, the variance upper bound decreases with the number of quantization levels. To obtain this data-independent bound, we establish upper bounds on the number of coordinates of $\mathbf{v}$ falling into intervals defined by $\hat{\mathscr{L}}$.

Theorem 3 (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*

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

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

Theorem 3 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 1 implies the following guarantees for NUQSGD:

Theorem 4 (NUQSGD for smooth convex optimization). *Let f and R be defined as in Theorem 1, let ε_Q be defined as in Theorem 2, let $\varepsilon > 0$, $\hat{B} = (1+\varepsilon_Q)B$, and let $\gamma > 0$ be given by $\gamma^2 = 2R^2/(\hat{B}T)$. 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 $T_\varepsilon = O\big(\max\big(\frac{2\hat{B}}{K\varepsilon^2}, \frac{\beta}{\varepsilon}\big)R^2\big)$ iterations suffice to guarantee $\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=0}^{T}\mathbf{w}_t\right)\right] - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \varepsilon$. In addition, NUQSGD requires at most N_Q communication bits per iteration in expectation.*

On nonconvex problems, (weaker) convergence guarantees can be established along the lines of, e.g., (Ghadimi and Lan, 2013, Theorem 2.1).

NUQSGD vs QSGD. 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 2, 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 $T = T_{A,\varepsilon}$ 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, $\zeta_{A,\varepsilon} = T_{A,\varepsilon}\overline{N}_A$, on the expected number of bits of communicated on a run expected to achieve a suboptimality gap of at most ε.

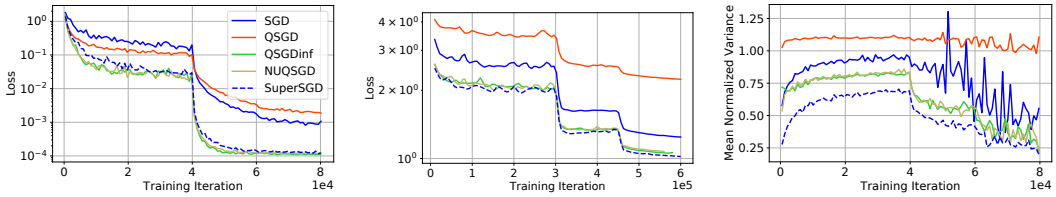

Figure 3: Training loss on CIFAR10 (left) and ImageNet (middle) for ResNet models. QSGD, QSGDinf, and NUQSGD are trained by simulating the quantization and dequantizing of the gradients from 8-GPUs. On CIFAR10, SGD refers to the single-GPU training versus on Imagenet it refers to 2-GPU setup in the original 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. (Right) 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.

Theorem 5 (Expected number of communication bits)**.** *Provided that* $s = o(\log(d))$ *and* $\frac{2\hat{B}}{K\varepsilon^2} > \frac{\beta}{\varepsilon}$, $\zeta_{\text{NUQSGD},\varepsilon} = O\big(\frac{1}{\varepsilon^2}\sqrt{d(d-2^{2s})}\log\big(\frac{\sqrt{d}}{2^s}\big)\big)$ *and* $\zeta_{\text{QSGD},\varepsilon} = O(\frac{1}{\varepsilon^2}d\log\sqrt{d})$.

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 stronger guarantees. Note that our stronger guarantees come without any assumption about the data.

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 show that NUQSGD can provide the same performance and accuracy compared to the QSGDInf heuristic, which has no theoretical compression guarantees. 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. We split our study across two axes: first, we examine the convergence of the methods and their induced variance. Second, we provide an efficient implementation of all four methods in Pytorch using the Horovod communication back-end (Sergeev and Del Balso, 2018), adapted to efficiently support quantization, and examine speedup relative to the full-precision baseline. We investigate the impact of quantization on training performance by measuring loss, variance, accuracy, and speedup for ResNet models (He et al., 2016) applied to ImageNet (Deng et al., 2009) and CIFAR10 (Krizhevsky).

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 and the number of quantization bits are set to 8192 and 4, respectively. We observe similar results in experiments with various bucket sizes and number of bits. 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 3 (left and middle), we show the training loss with 8 GPUs. We observe that NUQSGD and QSGDinf improve training loss compared to QSGD on ImageNet. We observe significant gap in training loss on CIFAR10 where the gap grows as training proceeds. We also observe similar performance gaps in test accuracy (provided in Appendix F). In particular, unlike NUQSGD, QSGD does not achieve test accuracy of full-precision SGD. Figure 3 (right) shows the mean normalized variance of the gradient (defined in Appendix F) versus training iteration 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.

Efficient Implementation and Speedup. To examine speedup behavior, we implemented all quantization methods in Horovod (Sergeev and Del Balso, 2018), a communication back-end supporting Pytorch, Tensorflow and MXNet. Doing so efficiently requires non-trivial refactoring of this framework, since it does not support communication compression—our framework will be open-sourced

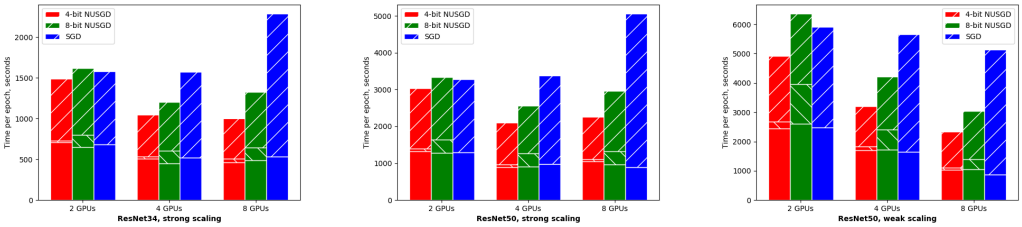

Figure 4: Scalability behavior for NUQSGD versus the full-precision baseline when training ResNet34 and ResNet50 on ImageNet. The ResNet34 graph examines *strong scaling* (left), splitting a global batch of size 256 onto the available GPUs, whereas the ResNet50 graph examines both strong scaling (middle) and *weak scaling* (right), keeping a fixed per-GPU batch size of 16. Each time bar is split into computation (bottom), encoding cost (middle), and transmission cost (top). Notice the significant negative scalability of the SGD baseline in both scenarios. By contrast, the 4bit communication-compressed implementation achieves positive scaling, while the 8bit variant stops scaling between 4 and 8 nodes due to the higher comunication and encoding costs. Further, notice the significantly better scaling behavior of the compression methods in the weak scaling experiment (right).

upon publication. Our implementation diverges slightly from the theoretical analysis. First, Horovod 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 a significant adverse effect on accuracy; since the communication impact of biases is negligible, we transmit them at full precision. Finally, for efficiency reasons, we do not employ any complex encoding of the quantized gradients, and instead directly pack the quantized values into 32-bit numbers. We implemented compression and de-compression via efficient CUDA kernels.

Our baseline is full-precision SGD (SuperSGD), using a standard all-to-all reduction pattern, which we compare against the 4bit and 8bit NUQSGD variants executing the same pattern (our QSGD/QSGDinf implementation provides virtually identical performance numbers). Figure 4 shows the execution 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. We examine both strong scaling and weak scaling behavior in the context of the two given models. 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.

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. In the overparametrized regime of interest, 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.

REFERENCES

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

M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2010.

R. Bekkerman, M. Bilenko, and J. Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

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

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

A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and A. Ng. Deep learning with cots hpc systems. In *Proc. International Conference on Machine Learning (ICML)*, 2013.

T. Chilimbi, Y. Suzue J. Apacible, and K. Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014.

M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014.

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

E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Y. Petuum. Petuum: A new platform for distributed machine learning on big data. *IEEE transactions on Big Data*, 1(2):49–67, 2015.

S. Zhang, A. E. Choromanska, and Y. LeCun. Deep learning with elastic averaging SGD. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2015.

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

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

C. M. D. Sa, Ce. Zhang, K. Olukotun, and C. Ré. Taming the wild: A unified analysis of hogwild-style algorithms. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2015.

S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In *Proc. International Conference on Machine Learning (ICML)*, 2015.

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. Devin. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467, 2016.

S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv:1606.06160, 2016.

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

J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *Proc. International Conference on Machine Learning (ICML)*, 2018.

W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016.

I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016.

J. Park, S. Li, W. Wen, P. Tang, H. Li, Y. Chen, and P. Dubey. Faster CNNs with direct sparse convolutions and guided pruning. In *Proc. International Conference on Learning Representations (ICLR)*, 2017.

K. W. Cattermole. *Principles of pulse code modulation*. Iliffe, 1969.

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

H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. 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.

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

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

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

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

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

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

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

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

A ENCODING

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;
  Update N;

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

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 $\text{ERC} : \{1, 2, \cdots\} \to \{0,1\}^*$ and $\text{ERC}^{-1} : \{0,1\}^* \to \{1, 2, \cdots\}$ 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 2 (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 1. *Let $\mathbf{v} \in \mathbb{R}^d$, $\mathscr{L} = (l_0, l_1, \cdots, l_{s+1})$, and fix $s \geq 1$. The variance of $Q_s(\mathbf{v})$ for general sequence of quantization levels is given by*

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] = \|\mathbf{v}\|^2 \sum_{i=1}^{d} \tau^2(r_i) p(r_i)\big(1 - p(r_i)\big) \tag{5}$$

where $r_i = |v_i|/\|\mathbf{v}\|$ and $p(r), s(r), \tau(r)$ 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:

$$\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] = \sum_{i=1}^{d} \|\mathbf{v}\|^2 \sigma^2(r_i) \tag{6}$$

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 $\hat{\mathscr{L}} = (0, 1/2^s, \cdots, 2^{s-1}/2^s, 1)$ 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

$$h_i(\mathbf{v}, s) = \begin{cases} 0 & \text{with probability } 1 - p_1(r_i, s); \\ 2^{-s} & \text{otherwise} \end{cases} \tag{7}$$

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

$$h_i(\mathbf{v}, s) = \begin{cases} 2^{j-s} & \text{with probability } 1 - p_2(r_i, s); \\ 2^{j+1-s} & \text{otherwise} \end{cases} \tag{8}$$

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

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

$$\begin{aligned} \mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] &= \|\mathbf{v}\|^2 \tau_0^2 \sum_{i \in \mathscr{S}_0} p_1(r_i, s)(1 - p_1(r_i, s)) \\ &\quad + \|\mathbf{v}\|^2 \sum_{j=0}^{s-1} \tau_{j+1}^2 \sum_{i \in \mathscr{S}_{j+1}} p_2(r_i, s)\big(1 - p_2(r_i, s)\big) \end{aligned} \tag{9}$$

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

Substituting $\tau_0 = 2^{-s}$ and $\tau_j = 2^{j-1-s}$ for $j \in \{1, \cdots, s\}$ into (9), we have

$$\begin{aligned} \mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] &= \|\mathbf{v}\|^2 2^{-2s} \sum_{i \in \mathscr{S}_0} p_1(r_i, s)(1 - p_1(r_i, s)) \\ &\quad + \|\mathbf{v}\|^2 \sum_{j=0}^{s-1} 2^{2(j-s)} \sum_{i \in \mathscr{S}_{j+1}} p_2(r_i, s)\big(1 - p_2(r_i, s)\big) \\ &\leq \|\mathbf{v}\|^2 2^{-2s} \sum_{i \in \mathscr{S}_0} p_1(r_i, s) \\ &\quad + \|\mathbf{v}\|^2 \sum_{j=0}^{s-1} 2^{2(j-s)} \sum_{i \in \mathscr{S}_{j+1}} p_2(r_i, s) \end{aligned} \tag{10}$$

We first note that $\sum_{i\in\mathscr{S}_0} p_1(r_i,s) \le d$ and $\sum_{i\in\mathscr{S}_{j+1}} p_2(r_i,s) \le d$ 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

$$\sum_{i\in\mathscr{S}_0} p_1\big(r_i,s\big) \le \min\{d_0, 2^s\sqrt{d_0}\} \tag{11}$$

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

$$\sum_{i\in\mathscr{S}_{j+1}} p_2\big(r_i,s\big) \le \min\{d_{j+1}, 2^{(s-j)}\sqrt{d_{j+1}}\}. \tag{12}$$

Substituting the upper bounds in (11) and (12) into (10), an upper bound on the variance of $Q_s(\mathbf{v})$ is given by

$$\begin{aligned}\mathbb{E}[\|Q_s(\mathbf{v})-\mathbf{v}\|^2] \le \min\{2^{-2s}d_0, 2^{-s}\sqrt{d_0}\}\|\mathbf{v}\|^2 \\ + \sum_{j=0}^{s-1} \min\{2^{2(j-s)}d_{j+1}, 2^{j-s}\sqrt{d_{j+1}}\}\|\mathbf{v}\|^2.\end{aligned} \tag{13}$$

The upper bound in (13) 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 2. *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

$$(d-d_0)2^{-2s} \le \sum_{i\notin\mathscr{S}_0} r_i^2 \le 1. \tag{14}$$

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

$$\begin{aligned}\mathbb{E}[\|Q_s(\mathbf{v})\|_0] &\le d-d_0+\sum_{i\in\mathscr{S}_0} r_i 2^s \\ &\le 2^{2s}+\sqrt{d_0}2^s.\end{aligned} \tag{15}$$

□

Using a similar argument as in the proof of Lemma 2, we have

$$d-d_0-d_1-\cdots-d_j \le 2^{2(s-j)} \tag{16}$$

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

$$\begin{aligned} b_0 &\le d_0 \\ b_1 &\le d_1+d_0 \\ &\vdots \quad \vdots \\ b_{s-1} &\le d_0+\cdots+d_{s-1}. \end{aligned} \tag{17}$$

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

We define

$$\begin{aligned} \tilde{d}_0 &\triangleq b_0 = d-2^{2s} \\ \tilde{d}_1 &\triangleq b_1-b_0 = 3\cdot 2^{2(s-1)} \\ &\vdots \quad \vdots \\ \tilde{d}_{s-1} &\triangleq b_{s-1}-b_{s-2} = 12 \\ \tilde{d}_s &\triangleq d-\tilde{d}_0-\tilde{d}_1-\cdots-\tilde{d}_{s-1} = 4. \end{aligned} \tag{18}$$

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 (13) 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 (13), which gives (3) and completes the proof.

C Proof of Theorem 3 (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

$$|R| \leq \|\mathbf{h}\|_0 + (1 + o(1))\|\mathbf{h}\|_0 \log\Big(\frac{d}{\|\mathbf{h}\|_0}\Big). \tag{19}$$

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

Lemma 3. *Fix a vector* $\mathbf{q}$ *such that* $\|\mathbf{q}\|_p^p \leq P$, *let* $i_1 < i_2 < \ldots i_{\|\mathbf{q}\|_0}$ *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*

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

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

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

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

$$\begin{aligned}\|\mathbf{h}\|^2 &\leq \sum_{i=1}^{d}\Big(\frac{v_i}{\|\mathbf{v}\|}+\frac{1}{2^s}\Big)^2 \\ &\leq 2\sum_{i=1}^{d}\Big(\frac{v_i^2}{\|\mathbf{v}\|^2}+\frac{1}{2^{2s}}\Big) = 2\big(1+\frac{d}{2^{2s}}\big).\end{aligned} \tag{20}$$

By Lemma 3 applied to $\mathbf{q}$ and the upper bound (20),

$$\begin{aligned}|E| \leq &-(1+o(1))+2\|\mathbf{h}\|_0 \\ &+(1+o(1))\|\mathbf{h}\|_0 \log\log\Big(\frac{2^{2s+2}\|\mathbf{h}\|^2}{\|\mathbf{h}\|_0}\Big).\end{aligned} \tag{21}$$

Combining (19) and (21), we obtain an upper bound on the expected code-length:

$$\mathbb{E}[|\text{ENCODE}(\mathbf{v})|] \leq N(\|\mathbf{h}\|_0) \tag{22}$$

where

$$\begin{aligned}N(\|\mathbf{h}\|_0) = b+3\|\mathbf{h}\|_0 &+(1+o(1))\mathbb{E}\big[\|\mathbf{h}\|_0\log\big(\frac{d}{\|\mathbf{h}\|_0}\big)\big] \\ &-(1+o(1))+(1+o(1))\mathbb{E}\big[\|\mathbf{h}\|_0\log\log\big(\frac{8(2^{2s}+d)}{\|\mathbf{h}\|_0}\big)\big].\end{aligned} \tag{23}$$

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

$$g_2''(x) = -\big(x\ln(2)\ln(C/x)\big)^{-1}\Big(1+\big(\ln(C/x)\big)^{-1}\Big). \tag{24}$$

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 $\mathbb{E}[\|\mathbf{h}\|_0] \leq 2^{2s}+\sqrt{d}2^s$ following Lemma 2. By assumption $2^{2s}+\sqrt{d}2^s \leq d/e$, and so, Jensen's inequality and (22) lead us to (4).

D Proof of Theorem 4 (NUQSGD for Smooth Convex Optimization)

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

$$\mathbb{E}[\|\hat{g}(\mathbf{w})-\nabla f(\mathbf{w})\|^2] \leq \mathbb{E}[\|g(\mathbf{w})-\nabla f(\mathbf{w})\|^2]+\mathbb{E}[\|\hat{g}(\mathbf{w})-g(\mathbf{w})\|^2]. \tag{25}$$

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

E Proof of Theorem 5 (Expected Number of Communication Bits)

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

In overparameterized networks, where $d \geq 2^{2s+1}$, we have $\varepsilon_Q = 2^{-s}\sqrt{d-2^{2s}}+O(s)$ and $\tilde{\varepsilon}_Q = \sqrt{d}/s$. 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.

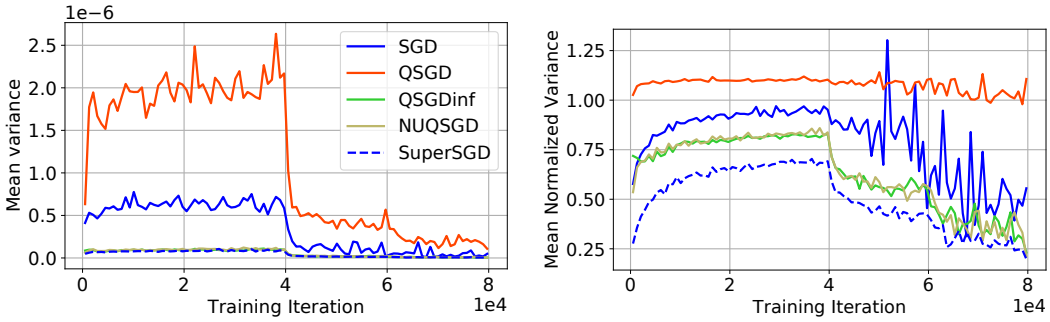

Figure 5: Estimated variance (left) and normalized variance (right) on CIFAR10 on the trajectory of single-GPU SGD. Variance is measured for fixed model snapshots during training. Notice that the variance for NUQSGD and QSGDinf is lower than SGD for almost all the training and it decreases after the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

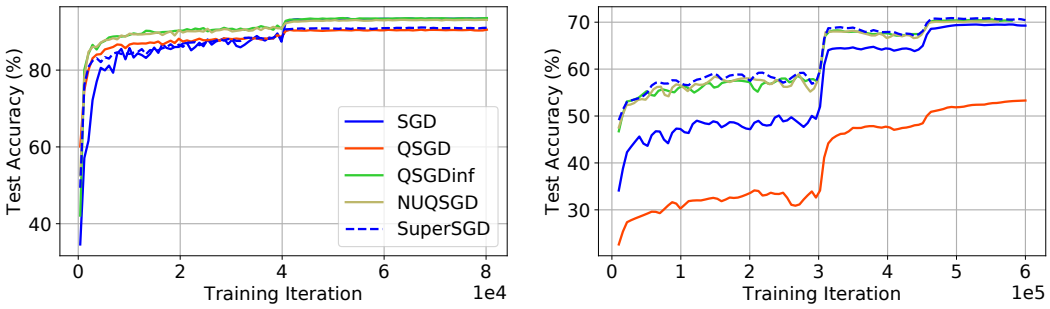

Figure 6: Accuracy on the hold-out set on CIFAR10 (left) and on ImageNet (right) for training ResNet models from random initialization until convergence. For CIFAR10, the hold-out set is the test set and for ImageNet, the hold-out set is the validation set.

F Additional Experiments

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

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 (discussed in Section 5). 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.

We also measure the variance and normalized variance at fixed snapshots during training by evaluating multiple gradient estimates using each quantization method. 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 5 (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 NUQSGDinf have lower variance than single-GPU SGD.

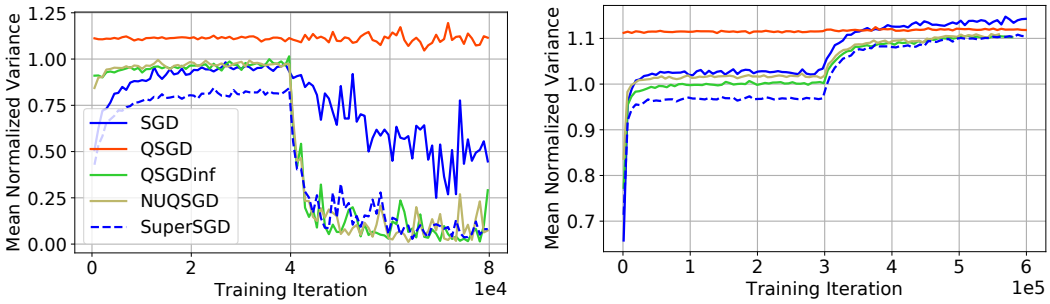

Figure 7: Estimated normalized variance on CIFAR10 (left) and ImageNet (right). For different methods, the variance is measured on their own trajectories. Note that the normalized variance of NUQSGD and QSGDinf is lower than SGD for almost the entire training. It decreases on CIFAR10 after the learning 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.

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 5 (right) shows the mean normalized variance of the gradient versus training iteration. Observe that the normalized variance for QSGD stays relatively constant while 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 7, 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.

ResNet152 Weak Scaling. In Figure 8, 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.

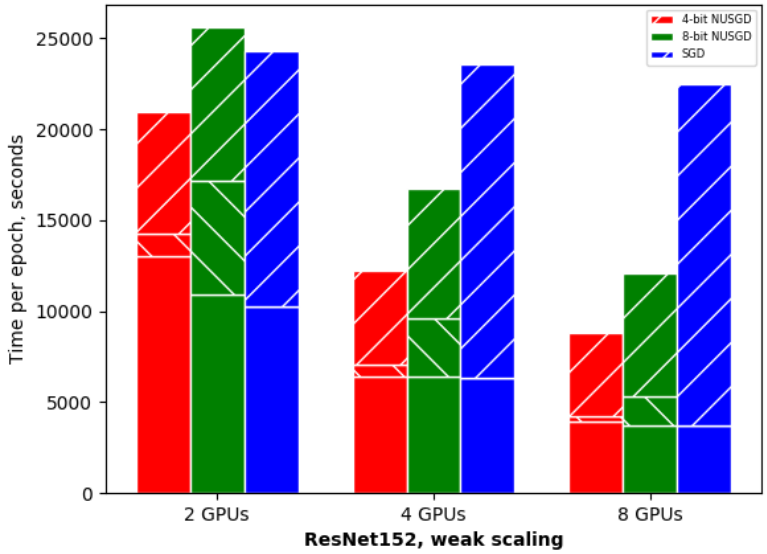

Figure 8: Scalability behavior for NUQSGD versus the full-precision baseline when training ResNet152 on ImageNet.