# Dimension-Free Bounds for Low-Precision Training

**Zheng Li**
Tsinghua University
lzlz19971997@gmail.com

**Christopher De Sa**
Cornell University
cdesa@cs.cornell.edu

## Abstract

Low-precision training is a promising way of decreasing the time and energy cost of training machine learning models. Previous work has analyzed low-precision training algorithms, such as low-precision stochastic gradient descent, and derived theoretical bounds on their convergence rates. These bounds tend to depend on the dimension of the model $d$ in that the number of bits needed to achieve a particular error bound increases as $d$ increases. In this paper, we derive new bounds for low-precision training algorithms that do not contain the dimension $d$ , which lets us better understand what affects the convergence of these algorithms as parameters scale. Our methods also generalize naturally to let us prove new convergence bounds on low-precision training with other quantization schemes, such as low-precision floating-point computation and logarithmic quantization.

## 1 Introduction

As machine learning models continue to scale to target larger problems on bigger data, the task of training these models quickly and efficiently becomes an ever-more-important problem. One promising technique for doing this is *low-precision computation*, which replaces the 32-bit or 64-bit floating point numbers that are usually used in ML computations with smaller numbers, often 8-bit or 16-bit fixed point numbers. Low-precision computation is a broadly applicable technique that has received a lot of attention, especially for deep learning, and specialized hardware accelerators have been developed to support it [2, 3, 13].

A major application for low-precision computation is the training of ML models using empirical risk minimization. This training is usually done using *stochastic gradient descent* (SGD), and most research in low-precision training has focused on low-precision versions of SGD. While most of this work is empirical [4–7, 11, 12, 14, 15, 17, 19, 21, 22], significant research has also been done in the theoretical analysis of low-precision training. This theoretical work has succeeded in proving bounds on the convergence rate of low-precision SGD and related low-precision methods in various settings, including for convex [8, 20] and non-convex objectives [1, 9, 16]. One common characteristic of these results is that the bounds tend to depend on the dimension $d$ of the model being learned (equivalently, $d$ is the number of parameters). For example, [16] gives the convergence bound

$$\mathbf{E}\left[f(\bar{w}_T) - f(w^*)\right] \leq \frac{(1+\log(T+1))\sigma_{\max}^2}{2\mu T} + \frac{\sigma_{\max}\delta\sqrt{d}}{2}, \tag{1}$$

where the objective $f$ is strongly convex with parameter $\mu$, low-precision SGD outputs $\bar{w}_T$ after $T$ iterations, $w^*$ is the true global minimizer of the objective, $\sigma_{\max}^2$ is an upper bound on the second moment of the stochastic gradient samples $\mathbf{E}\left[\left\|\nabla\tilde{f}(w)\right\|_2^2\right] \leq \sigma_{\max}^2$, and $\delta$ is the quantization step, the difference between adjacent numbers in the low-precision format. Notice that, as $T \to \infty$, this bound shows convergence down to a level of error that increases with the dimension $d$. Equivalently, in order to achieve the same level of error as $d$ increases, we would need to use more bits of quantization to make $\delta$ smaller. Similar dimension-dependent results, where either the error or the

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

number of bits needed increases with $d$, can also be seen in other work on low-precision training algorithms [1, 8, 20]. This dependence on $d$ is unsatisfying because the motivation for low-precision training is to tackle large-scale problems on big data, where $d$ can range up to $10^8$ or more for commonly used models [18]. For example, to compensate for a factor of $d = 10^8$ in (1), we could add bits to decrease the quantization step $\delta$ by a factor of $\sqrt{d}$, but this would require adding $\log_2(10^4) \approx 13$ bits, which is significant compared to the 8 or 16 bits that are commonly used in low-precision training.

In this paper, we address this problem by proving bounds on the convergence of LP-SGD [16] that do not contain dimension $d$ in the expression. Our main technique for doing so is a tight dimension-free bound on the expected quantization error of the low-precision stochastic gradients in terms of the $\ell_1$-norm. Our results are summarized in Table 3, and we make the following contributions:

- We describe conditions under which we can prove a dimension-free bound on the convergence of SGD with fixed-point, quantized iterates on both convex and non-convex problems.
- We study non-linear quantization schemes, in which the representable low-precision numbers are distributed non-uniformly. We prove dimension-free convergence bounds for SGD using logarithmic quantization [15], and we show that using logarithmic quantization can reduce the number of bits needed for LP-SPG to provably converge.
- We study quantization using low-precision floating-point numbers, and we present theoretical analyis that suggests how to assign a given number of bits to exponent and mantissa to optimize the accuracy of training algorithms. We validate our results experimentally.

## 2 Related Work

Motivated by the practical implications of faster machine learning, much work has been done on low-precision training. This work can be roughly divided into two groups. The first focuses on training deep models with low-precision weights, to be later used for faster inference. For some applications, methods of this type have achieved good results with very low-precision models: for example, binarized [5, 12, 17] and ternary networks [22] have been observed to be effective (although as is usual for deep learning they lack theoretical convergence results). However, these approaches are still typically trained with full-precision iterates: the goal is faster inference, not faster training (although faster training is often achieved as a bonus side-effect).

A second line of work on low-precision training, which is applied to both DNN training and non-deep-learning tasks, focuses on making various aspects of SGD low-precision, while still trying to solve the same optimization problem as the full-precision version. The most common way to do this is to make the iterates of SGD (the $w_t$ in the SGD update step $w_{t+1} = w_t - \alpha_t \nabla f_t(w_t)$) stored and computed in low-precision arithmetic [4, 8, 9, 11, 16]. This is the setting we will focus on most in this paper, because it has substantial theoretical prior work which exhibits the dimension-dependence we set out to study [1, 8, 16, 20]. The only paper we found with a bound that was not dimension-dependent was De Sa et al. [9], but in that paper the authors required that the gradient samples be 1-sparse (have only one nonzero entry), which is not a realistic assumption for most ML training tasks. In addition to quantizing the iterates, other work has studied quantizing the training set [20] and numbers used to communicate among parallel workers [1]. We expect that our results on dimension-free bounds will be complementary with these existing theoretical approaches, and we hope that they can help to explain the success of the exciting empirical work in this area.

## 3 Dimension-Free Bounds for SGD

In this section, we analyze the performance of stochastic gradient descent (SGD) using low-precision training. Though there are numerous variants of this algorithm, SGD remains the *de facto* algorithm used most for machine learning. We will start by describing SGD and how it can be made low-precision. Suppose we are trying to solve the problem

$$\text{minimize: } f(w) = \frac{1}{n}\sum_{i=1}^{n} \tilde{f}_i(w) \qquad \text{over: } w \in \mathbb{R}^d. \tag{2}$$

SGD solves this problem iteratively by repeatedly running the update step

$$w_{t+1} = w_t - \alpha \nabla \tilde{f}_{i_t}(w_t) \tag{3}$$

where $\alpha$ is the *step size*[1] or learning rate, and $i_t$ is the index of a component function chosen randomly and uniformly at each iteration from $\{1, \ldots, n\}$. To make this algorithm low-precision, we *quantize the iterates* (the vectors $w_t$) and store them in a low-precision format. The standard format to use lets us represent numbers in a set

$$\mathrm{dom}(\delta, b) = \{-\delta \cdot 2^{b-1}, -\delta \cdot (2^{b-1} - 1), \cdots, -\delta, 0, \delta, 2\delta, \cdots, \delta \cdot (2^{b-1} - 1)\}$$

with $\delta > 0$ being the *quantization gap*, the distance between adjacent representable numbers, and $b \in \mathbb{N}$ being the number of bits we use [8]. Usually, $\delta$ is a power of 2, and this scheme is called *fixed-point arithmetic*. It is straightforward to encode numbers in this set as $b$-bit signed integers, by just multiplying or dividing by $\delta$ to convert to or from the encoded format—and we can even do many arithmetic computations on these numbers directly as integers. This is sometimes called *linear quantization* because the representable points are distributed uniformly throughout their range. However, as the gradient samples will produce numbers outside this set during iteration, we need some way to map these numbers to the set of numbers that we can represent. The standard way to do this is with a *quantization function* $Q(x) : \mathbb{R} \to \mathrm{dom}(\delta, b)$. While many quantization functions have been proposed, the one typically used in theoretical analysis (which we will continue to use here) is *randomized rounding*. Randomized rounding, also known as unbiased rounding or stochastic rounding, rounds up or down at random such that $\mathbf{E}\left[Q(x)\right] = x$ whenever $x$ is within the range of representable numbers (i.e. when $-\delta \cdot 2^{b-1} \leq x \leq \delta \cdot (2^{b-1} - 1)$). When $x$ is outside that range, we quantize it to the closest representable point. When we apply $Q$ to a vector argument, it quantizes each of its components independently.

Using this quantization function, we can write the update step for low-precision SGD (LP-SGD), which is a simple quantization of (3),

$$w_{t+1} = Q(w_t - \alpha \nabla \tilde{f}_{i_t}(w_t)) \tag{4}$$

As mentioned before, one common feature of prior bounds on the convergence of LP-SGD is that they depend on the number of dimensions $d$, whereas bounds on full precision SGD under the same conditions don't. This difference is due to the fact that, when we quantize a number $w$, it increases its variance by $\mathbf{E}\left[(Q(w) - w)^2\right] \leq \delta^2/4$. Observe that this inequality is tight since it holds as an equality when $w$ is in the middle of two quantization points, e.g. $w = \delta/2$, as illustrated in Figure 1(a). When quantizing a vector $w \in \mathbb{R}^d$, the squared error can be increased by

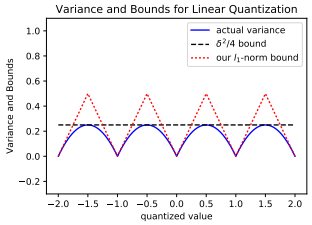
$$\mathbf{E}\left[\|Q(w) - w\|_2^2\right] = \sum_{k=1}^{d} \mathbf{E}\left[(Q(w_k) - w_k)^2\right] \leq \frac{\delta^2 d}{4}, \tag{5}$$

and this bound is again tight. This variance inequality is the source of the $d$ term in analyses of LP-SGD, and the tightness of the bound leads to the natural belief that the $d$ term is inherent, and that low-precision results are inevitably dimension-dependent.

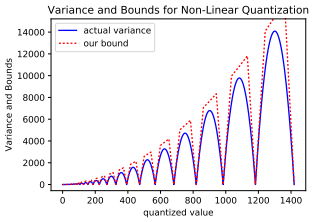
However, we propose that if we can instead bound the variance in (5) with some properties of the problem itself that is not inherently dependent on $d$, we can achieve a result that is dimension-free. One way to do this is to look at the variance graphically. Figure 1(a) plots the quantization error as a function of $w$ along with the bound in (5). Notice that the squared error looks like a series of parabolas, and the bound in (5) is tight at the top of those parabolas, but loose elsewhere. Instead, suppose we want to do the opposite and produce a bound that is tight when the error is zero (at points in $\mathrm{dom}(\delta, b)$). To do this, we observe that $\mathbf{E}\left[(Q(w) - w)^2\right] \leq \delta |w - z|$ for any $z \in \mathrm{dom}(\delta, b)$. This bound is also tight when $z$ is adjacent to $w$, and we plot it in Figure 1(a) as well. The natural vector analog of this is

$$\mathbf{E}\left[\|Q(w) - w\|_2^2\right] \leq \sum_{k=1}^{d} \delta |w_k - z_k| = \delta \|w - z\|_1, \ \forall z \in \mathrm{dom}(\delta, b)^d \tag{6}$$

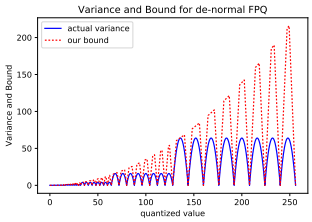
[1] Usually in SGD the step size is decreased over time, but here for simplicity we consider a constant learning rate schedule.




(a) Linear quantization error and two possible tight upper bounds




(b) Variance and bound for logarithmic quantization.




(c) Variance and bound for floating-point quantization.

Figure 1: A figure of actual quantization variance $\mathbf{E}\left[(Q(w)-w)^2\right]$ and the tight upper bound that we introduced in one dimension. We plot this bound when taking the minimum over all possible $z$.

where $\|\cdot\|_1$ denotes the $\ell_1$-norm. This is a dimension-free bound we can use to replace (5) to bound the convergence of LP-SGD and other algorithms. However, this replacement is nontrivial as our bound is now non-constant: it depends on $w$, which is a variable updated each iteration. Also, in order to bound this new $\ell_1$-norm term, we will need some new assumptions about the problem. Next, we will state these assumptions, along with the standard assumptions used in the analysis of SGD for both convex and non-convex objectives, and then we will use them to present our dimension-free bound on the convergence of SGD.

**Assumption 1.** *All the loss functions $\tilde{f}_i$ are differentiable, and their gradients are $L$-Lipschitz continuous in the sense of 2-norm, that is,*

$$\forall i \in \{1,2,\cdots,n\}, \quad \forall x,y \in \mathbb{R}^d, \quad \left\|\nabla \tilde{f}_i(x) - \nabla \tilde{f}_i(y)\right\|_2 \leq L\|x-y\|_2$$

**Assumption 2.** *All the gradients of the loss functions $\tilde{f}_i$ are $L_1$-Lipschitz continuous in the sense of 1-norm to 2-norm, that is,*

$$\forall i \in \{1,2,\cdots,n\}, \quad \forall x,y \in \mathbb{R}^d, \quad \left\|\nabla \tilde{f}_i(x) - \nabla \tilde{f}_i(y)\right\|_1 \leq L_1\|x-y\|_2$$

These two assumptions are simply expressing of Lipschitz continuity in different norms. Assumption 1 is a standard assumption in the analysis of SGD on convex objectives, and has been applied in the low-precision case as well in prior work [8]. Assumption 2 is analogous to 1, except we are bounding the $\ell_1$-norm instead of the $\ell_2$-norm. This holds naturally (with a reasonable value of $L_1$) for many problems, in particular problems for which the gradient samples are sparse.

**Assumption 3.** *The total loss function $f$ is $\mu$-strongly convex for some $\mu > 0$:*

$$\forall w, v, \quad f(w) - f(v) - \frac{\mu}{2}\|w-v\|_2^2 \geq (w-v)^T \nabla f(v)$$

This is a standard assumption that bounds the curvature of the loss function $f$, and is satisfied for many classes of convex objectives. When an objective is strongly convex and Lipschitz continuous, it is standard to say it has *condition number* $\kappa = L/\mu$, and here we extend this to say it has *L1 condition number* $\kappa_1 = L_1/\mu$. And for our analysis on the non-convex case, we don't have this assumption.

**Assumption 4.** *If the objective is convex, we assume that the gradient of each loss function is bounded by some constant near the optimal point $w^*$ in the sense of $l_1$ and $l_2$ norm, that is,*

$$\mathbf{E}\left[\left\|\nabla \tilde{f}_i(w^*)\right\|_2^2\right] \leq \sigma^2, \qquad \mathbf{E}\left[\left\|\nabla \tilde{f}_i(w^*)\right\|_1\right] \leq \sigma_1$$

*If the objective is non-convex, there is not necessarily a single optimal point, so we just assume each loss function has a global bound on its gradient: for any $w$,*

$$\forall w, \qquad \mathbf{E}\left[\left\|\nabla \tilde{f}_i(w)\right\|_2^2\right] \leq \sigma^2, \qquad \mathbf{E}\left[\left\|\nabla \tilde{f}_i(w)\right\|_1\right] \leq \sigma_1$$

This assumption constrains the gradient for each loss function at the optimal point. We know $\nabla f(w^*) = \frac{1}{n}\sum_i \tilde{\nabla} f_i(w^*) = 0$, so it is intuitive that each $\nabla \tilde{f}_i(w^*)$ can be bounded by some value.

Table 1: Summary of our dimension-free results compared with prior work. The values report the number of bits needed, according to the theoretical bound, for the LP-SGD [16] algorithm to achieve an expected objective gap ($f(w) - f(w^*)$) of $\epsilon$ in the convex case, and an expected gradient of $\epsilon$ in the non-convex case, when we let step size $\alpha \to 0$, epoch length $T \to \infty$. Here we let $R$ denote the radius of the range of numbers representable in the low-precision format and assume $\|w^*\|_2 = \Theta(R)$. The rest of the parameters can be found in the assumptions to be introduced later.

| OBJECTIVE CLASS | CONVEX | NON-CONVEX |
|---|---|---|
| NUMBER OF BITS NEEDED FOR | $\mathbf{E}\left[f(w) - f(w^*)\right] \leq \epsilon$ | $\mathbf{E}\left[\|\nabla f(\bar{w})\|_2^2\right] \leq \epsilon$ |
| PRIOR DIMENSION-DEPENDENT BOUND | $\log_2 \mathcal{O}(R\sigma_{\max}\sqrt{d}/\epsilon)$ | — |
| OUR DIMENSION-FREE BOUND | $\log_2 \mathcal{O}(R\sigma_1/\epsilon)$ | $\log_2 \mathcal{O}(LR\sigma_1/\epsilon)$ |
| DIMENSION-FREE WITH LOGARITHMIC QUANTIZATION | $\log_2 \mathcal{O}(\frac{R\sigma}{\epsilon} \cdot \log(1 + \frac{\sigma_1}{\sigma}))$ | $\log_2 \mathcal{O}(\frac{LR}{\sqrt{\epsilon}} \cdot \log(1 + \frac{\sigma_1}{\sqrt{\epsilon}}))$ |

In the non-convex case, however, we need a global bound on the gradient instead of just at the optimum. This is a natural assumption to make and it has been used in a lot of other work in this area. Note that this assumption only needs to hold under the expectation over all $\tilde{f}_i$.

For non-convex cases, we need the following additional assumption.

**Assumption 5.** *The variance of the gradient of each loss function is bounded by some constant $\sigma_0^2$:*

$$\forall w, \quad \mathrm{Var}(\nabla f_i(w)) = \mathbf{E}\left[\|\nabla f_i(w) - \nabla f(w)\|_2^2\right] \leqslant \sigma_0^2$$

With these assumptions, we proved the following theorems for low-precision SGD:

**Theorem 1.** *Suppose that we run LP-SGD on an objective that satisfies Assumptions 1–4, and with step size $\alpha < 1/(2\kappa^2\mu)$. After $T$ LP-SGD update steps (4), select $\bar{w}_T$ uniformly at random from $\{w_0, w_1, \ldots, w_{T-1}\}$. Then, the expected objective gap of $\bar{w}_T$ is bounded by*

$$\mathbf{E}\left[f(\bar{w}_T) - f(w^*)\right] \leq \frac{1}{2\alpha T}\|w_0 - w^*\|_2^2 + \frac{\alpha\sigma^2 + \delta\sigma_1}{2} + \frac{\delta^2\kappa_1^2\mu}{4}$$

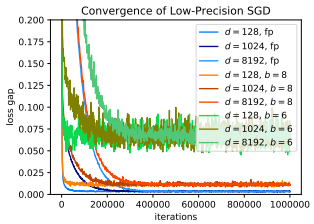
**Theorem 2.** *Suppose that we run LP-SGD on an objective that is non-convex and satisfies Assumptions 1, 4, 5, with constant step size $\alpha$. After $T$ LP-SGD update steps, select $\bar{w}_T$ uniformly at random from $\{w_0, w_1, \ldots, w_{T-1}\}$. Then the expected squared gradient norm of $\bar{w}_T$ is bounded by*

$$\mathbf{E}\left[\|\nabla f(\bar{w}_T)\|_2^2\right] \leqslant \frac{2}{2\alpha - \alpha^2 L}\frac{f(w_0) - f^*}{T} + \frac{\alpha\sigma_0^2 L + L\delta\sigma_1}{2 - \alpha L}$$

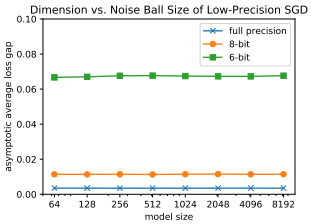
The first theorem shows a bound of the expected distance between the result we get at $T$-th iteration and the optimal value for a convex objective, and the second shows a bound of the expected gradient at $T$-th iteration, where $f^*$ is the global minimum of objective $f$. By choosing an appropriate step size we can achieve convergence at a $1/T$ rate, while the limit we converge to is only dependent on dimension-free factors. Meanwhile, as mentioned in the first section, previous work gives a dimension-dependent bound (1) for the problem, which also converges at a $1/T$ rate.[2] Therefore our result guarantees a dimension-free convergence limit without weakening the convergence rate.

It is important to note that, because the dimension-dependent bound in (5) was tight, we should not expect our new result to improve upon the previous theory in all cases. In the worst case, $\kappa_1 = \sqrt{d} \cdot \kappa$ and similarly $\sigma_1 = \sqrt{d} \cdot \sigma$; this follows from the fact that for vectors in $\mathbb{R}^d$, the norms are related by the inequality $\|x\|_1 \leq \sqrt{d} \cdot \|x\|_2$. Substituting this into our result produces a dimension-dependent bound again. This illustrates the importance of introducing the new parameters $\kappa_1$ and $\sigma_1$ and requiring that they be bounded; if we could not express our bound in terms of these parameters, the best we could do here is recover a dimension-dependent bound. This means that we achieve the standard result of a dimension-dependent bound in the worst case, but in other cases our results are strictly better.

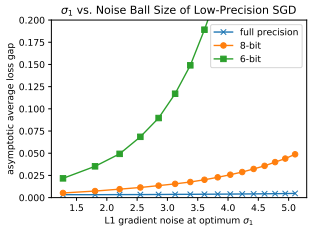
[2]Previous work (1) used a decaying step size while ours uses a constant step size to achieve a better result.




(a) The convergence of training loss gap $f(w) - f(w^*)$ for LP-SGD, showing the effect of different model sizes and precision.




(b) The size of the noise ball is not significantly affected by model size $d$.




(c) The size of the noise ball does depend on $\sigma_1$, especially when precision is low.

Figure 2: (a) Convergence of full-precision (fp) SGD and LP-SGD; (b)(c) Plots of the asymptotic loss gap from (a) as a function of model size $d$ and $\sigma_1$.

**Experiments** Next, we validate our theoretical results experimentally on convex problems. To do this, we analyzed how the size of the noise floor of convergence of SGD and LP-SGD varies as the dimension is changed for a class of synthetic problems. Importantly, we needed to pick a class of problems for which the parameters $L$, $L_1$, $\mu$, $\sigma$, and $\sigma_1$, did not change as we changed the dimension $d$. To do this, we chose a class of synthetic linear regression models with loss components sampled independently and identically as

$$\tilde{f}_i(w) = \frac{1}{2}(\tilde{x}^T w - \tilde{y})^2$$

where $\tilde{x}$ is a sparse vector sampled to have $s$ nonzero entries each of which is sampled uniformly from $\{-1, 1\}$, and $\tilde{y}$ is sampled from $\mathcal{N}(\tilde{x}^T w^*, \beta^2)$ for some variance parameter $\beta$. Importantly, the nonzero entries of $\tilde{x}$ were chosen non-uniformly such that $\Pr[\tilde{x}_i \neq 0] = p_i$ for some probabilities $p_i$ which decrease as $i$ increases; this lets us ensure that $\mu$ remains constant as $d$ is increased. For simplicity, we sampled a fresh loss component of this form at each SGD iteration, which is sometimes called the *online* setting. It is straightforward to derive that for this problem

$$\mu = p_d \qquad L = s \qquad L_1 = s\sqrt{s} \qquad \sigma^2 = \beta^2 s \qquad \sigma_1 = \sqrt{2s/\pi}\sigma.$$

We set $\alpha = 0.01$, $\beta = 0.2$, $p_1 = 0.9$, $p_d = 0.001$, and $s = 16$, we chose each entry of $w^*$ uniformly from $[-1/2, 1/2]$, and we set $\delta$ such that the low-precision numbers would range from $-1$ to 1. We set these parameters so that our model satisfies Assumptions 1 to 4 while maintaining the smallest values of the parameters $L$, $L_1$, $\sigma$, and $\sigma_1$; choosing a different set of parameters will lead to a looser theoretical bound. Figure 2(a) shows the convergence of SGD and LP-SGD as the dimension $d$ is changed, for both 8-bit and 6-bit quantization. Notice that while changing $d$ has an effect on the initial convergence rate for both SGD and LP-SGD, it has no effect on the noise ball size, the eventual loss gap that the algorithm converges to. Figure 2(b) measures this noise ball size more explicitly as the dimension is changed: it reports the loss gap averaged across the second half of the iterates. Notice that as the dimension $d$ is changed, the average loss gap is almost unchanged, even for very low-precision methods for which the precision *does* significantly affect the size of the noise ball. This validates our dimension-free bounds, and shows that they can describe the actual dependence on $d$ in at least one case.

Figure 2(c) validates our results in the opposite way: it looks at how this gap changes as our new parameters $\sigma_1$ and $L_1$ change while $d$, $\mu$, and $\sigma$ are kept fixed. To do this, we fixed $d = 1024$ and changed $s$ across a range, setting $\beta = 0.8/\sqrt{s}$, which keeps $\sigma^2$ constant as $s$ is changed: this has the effect of changing $\sigma_1$ (and, as a side effect, $L_1$ and $L$). We can see from figure 2(c) that changing $\sigma_1$ in this way has a much greater effect on LP-SGD than on SGD. This validates our theoretical results, and suggests that $\sigma_1$ and $L_1$ can effectively determine the effect of low-precision compute on SGD.

## 4 Non-linear Quantization

Up till now, most theoretical work in the area of low-precision machine learning has been on linear quantization, where the distance between adjacent quantization points is a constant value $\delta$. Another option is *non-linear quantization* (NLQ), in which we quantize to a set of points that are non-uniformly

distributed. This approach has been shown to be effective for accelerating deep learning in some settings [15]. In general, we can quantize to a set of points

$$D = \{-q_n, \cdots, -q_1, q_0, q_1, \cdots, q_{n-1}\},$$

and, just like with linear quantization, we can still use a quantization function $Q(w)$ with randomized rounding that rounds up or down to a number in $D$ in such a way that $\mathbf{E}[Q(w)] = w$ for $w \in [-q_n, q_{n-1}]$. When we consider the quantization variance here, the natural dimension-dependent bound would be

$$\mathbf{E}\left[\|Q(w) - w\|_2^2\right] \leq \frac{d}{4} \max_i (q_i - q_{i-1})^2.$$

This is still a tight bound since it holds with equality for a number in the middle of two adjacent quantization points. However, when applied in the analysis of LP-SGD, this bound induces poor performance and often under-represents the actual result.

Here we discuss a specific NLQ method and use it to introduce a tight bound on the quantization variance. This method has been previously studied as *logarithmic quantization* or $\mu-$law quantization, and is defined recursively by

$$q_0 = 0, \qquad q_{i+1} - q_i = \delta + \zeta q_i \tag{7}$$

where $\delta > 0$ and $\zeta > 0$ are fixed parameters. Note that this includes linear quantization as a special case by setting $\zeta = 0$. It turns out that we can prove a tight dimension-free bound on the quantization variance of this scheme. First, we introduce the following definition.

**Definition 1.** An unbiased quantization function $Q$ satisfies the dimension-free variance bound with parameters $\delta$, $\zeta$, and $\eta$ if for all $w \in [-q_n, q_{n-1}]$ and all $z \in D$,

$$\mathbf{E}\left[\|Q(w) - w\|_2^2\right] \leq \delta \|w - z\|_1 + \zeta \|z\|_2 \cdot \|w - z\|_2 + \eta \|w - z\|_2^2.$$

We can prove that our logarithmic quantization scheme satisfies this bound.

**Lemma 1.** *The logarithmic quantization scheme (7) satisfies the dimension-free variance bound with parameters $\delta$, $\zeta$, and $\eta = \frac{\zeta^2}{4(\zeta+1)} < \frac{\zeta}{4}$.*

Notice that this bound becomes identical to the linear quantization bound (6) when $\zeta = 0$, so this result is a strict generalization of our results from the linear quantization case. With this setup, we can apply NLQ to the low-precision training algorithms we have studied earlier in this paper.

**Theorem 3.** *Suppose that we run LP-SGD on a convex objective that satisfies Assumptions 1–4, and using a quantization scheme that satisfies the dimension-free variance bound 1. If $\zeta < \frac{1}{\kappa}$, then*

$$\mathbf{E}\left[(f(\bar{w}_T) - f(w^*))\right] \leq \frac{\|w_0 - w^*\|_2^2}{2\alpha T} + \frac{(1+\eta)\alpha\sigma^2 + \delta\sigma_1 + \zeta\sigma\|w^*\|_2}{2} + \frac{(\delta L_1 + \zeta L\|w^*\|_2 + \zeta\sigma)^2}{4\mu}$$

For non-convex objectives, we need to assume a bound for the iterates we deal with, that is,

**Assumption 6.** *The scale of the iterates is bounded by some constant $R_0$, i.e. $\forall t,\ \|w_t\|_2 \leqslant R_0$.*

**Theorem 4.** *Suppose that we run LP-SGD on a non-convex objective thatsatisfies previous assumptions 1,4– 6, with constant step size $\alpha < \frac{1}{2(\eta+1)L}$ and using a quantization scheme that satisfies the dimension-free variance bound 1, then*

$$\mathbf{E}\left[\|\nabla f(\bar{w}_T)\|_2^2\right] \leqslant \frac{2(f(w_0) - f^*)}{\alpha T} + \alpha\sigma_0^2 L + L\delta\sigma_1 + \frac{1}{2}(L\zeta R_0)^2$$

If we fix the representable range $R$ (the largest-magnitude values representable in the low-precision format) and choose our quantization parameters optimally, we get the result that the number of bits we need to achieve objective gap or expected gradient $\epsilon$ is $\log_2 \mathcal{O}((R\sigma/\varepsilon) \cdot \log(1 + \sigma_1/\sigma))$ and $\log_2 \mathcal{O}(LR/\sqrt{\epsilon} \cdot \log(1 + \sigma_1/\sqrt{\epsilon}))$ (as is shown in table 3). These bounds are notable because even in the worst case where we do not have a bound on $\sigma_1$ and must use $\sigma_1 \leq \sqrt{d} \cdot \sigma$, which recovers the dimension term, the bounds still manage to "hide" it within a log term. This greatly decreases the effect of the dimension, and suggests that NLQ may be a promising technique to use for low-precision training at scale. Also note that, although the first bound holds only when $\zeta < \frac{1}{\kappa} = \frac{\mu}{L}$, which to some extent limits the acceleration of the strides in logarithmic quantization, the bound $\frac{\mu}{L}$ is independent of $\sigma$ and $\sigma_1$, thus this effect of "pushing " $\sigma_1$ into a $\log$ term is independent of the setting of $\zeta$.

**Floating point.** Next, we look at another type of non-linear quantization that is of great practical use: *floating-point quantization (FPQ)*. Here, the quantization points are simply floating-point numbers with some fixed number of exponential bits $b_e$ and mantissa bits $b_m$. Floating-point numbers are represented in the form

$$(-1)^{\text{sign bit}} \cdot 2^{\text{exponent}-\text{bias}} \cdot (1.m_1 m_2 m_3 \dots m_{b_m}) \tag{8}$$

where "exponent" is a $b_e$-bit unsigned number, the $m_i$ are the $b_m$ bits of the mantissa, and "bias" is a term that sets the range of the representable numbers by determining the range of the exponent. In standard floating point numbers, the exponent ranges from $[-2^{b_e-1}+2, 2^{b_e-1}-1]$, which corresponds to a bias of $2^{b_e-1}-1$. To make our results more general, we also consider non-standard bias by defining a *scaling factor* $s = 2^{-(\text{bias}-\text{standard bias})}$; the standard bias setting corresponds to $s = 1$. We also consider the case of *denormal* floating point numbers, which tries to address underflow by replacing the 1 in (8) with a 0 for the smallest exponent value. Under these conditions, we can prove that floating-point quantization satisfies the bound in Definition 1.

**Lemma 2.** *The FPQ scheme using randomized rounding satisfies the dimension-free variance bound with parameters $\delta_{normal}, \zeta$, and $\eta$ for normal FPQ and $\delta_{denormal}, \zeta$, and $\eta$ for denormal FPQ where*

$$\delta_{normal} = \frac{4s}{2^{2^{b_e-1}}}, \qquad \delta_{denormal} = \frac{8s}{2^{2^{b_e-1}+b_m}}, \qquad \zeta = 2^{-b_m}, \qquad \eta = \frac{\zeta^2}{4(\zeta+1)}.$$

This bound can be immediately combined with Theorem 3 to produce dimension-free bounds on the convergence rate of low-precision floating-point SGD. If we are given a fixed number of total bits $b = b_e + b_m$, we can minimize this upper bound on the objective gap or the expected gradient to try to predict the best way to allocate our bits between the exponent and the mantissa.

**Theorem 5.** *When using normal FPQ for a convex objective, given $b$ total bits, the optimal number of exponential bits $b_e$ such that the asymptotic upper bound on the objective gap given by Theorem 3 is minimized is in the interval between:*

$$\log_2\left[2\log_2\left(\frac{2(\ln 2)s\sigma_1}{\sigma\left\|w^*\right\|_2}\right)+2b\right] \qquad \textit{and} \qquad \log_2\left[2\log_2\left(\frac{2(\ln 2)sL_1}{L\left\|w^*\right\|_2+\sigma}\right)+2b\right].$$

**Theorem 6.** *When using denormal FPQ for a convex objective, given $b$ total bits, the optimal number of exponential bits $b_e$ such that the asymptotic upper bound on the objective gap, as $T \to \infty$ and $\alpha \to 0$, given by Theorem 3 is minimized is in the interval between:*

$$\log_2\left[1-\frac{2}{\ln 2}W\left(\frac{e\sigma\left\|w^*\right\|_2}{8s\sigma_1}\right)\right] \qquad \textit{and} \qquad \log_2\left[1-\frac{2}{\ln 2}W\left(\frac{e(L\left\|w^*\right\|_2+\sigma)}{8sL_1}\right)\right]$$

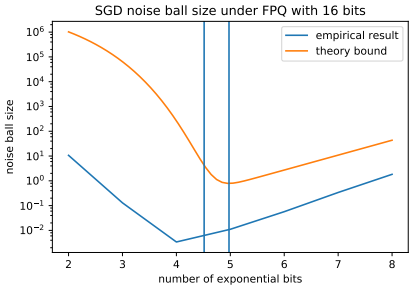
*where $e$ denotes the base of the natural logarithm and $W$ stands for the* Lambert W *function. In cases where neither of these two values exists, the noise ball size increases as $b_e$, thus $b_e = 2$ would be the optimal setting, which is equivalent to linear quantization.*

**Theorem 7.** *When using normal FPQ for a non-convex objective, given $b$ total bits, the optimal number of exponential bits $b_e$ such that the asymptotic upper bound on the gradient, as $T \to \infty$ and $\alpha \to 0$, given by Theorem 4 is minimized, at:*

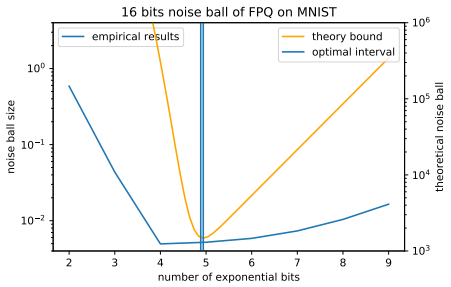
$$\log_2\left[\frac{2}{\ln 2}W\left(\frac{(\ln 2)^2 s\sigma_1 2^{2b}}{LR_0^2}\right)\right]$$

These theorems give us an idea of where the optimal setting of $b_e$ lies such that the theoretical asymptotic error or the expected gradient is minimized. When using normal FPQ, this optimal assignment of $b_e$ is $\mathcal{O}(\log(b))$, and for denormal FPQ the result is independent of $b$. Also, we found that for de-normal FPQ used in non-convex objectives, the optimal setting of $b_e$ is the solution to a transcendental equation, which may not exist. This suggests that once the total number of bits grows past a threshold, we should assign most of or all the extra bits to the mantissa.

**Experiments** For FPQ, we ran experiments on two different data sets. First, we ran LP-SGD on the same synthetic data set that we used for linear regression. Here we used normal FPQ with 20 bits in total, and we get the result in Figure 3(a). In this diagram, we plotted the empirical noise ball size, its theoretical upper bound, and the optimal interval for $b_e$ as Theorem 5 predicts. As the figure




(a) Training noise ball size of SGD using 16 bits normal FPQ on synthetic data set




(b) Training noise ball size of SGD using 16 bits normal FPQ on MNIST

Figure 3: Plots of noise ball size vs. $b_e$ when running SGD with 16 bits FPQ on synthetic data set and MNIST. Note the use of two $y$-axes in Figure 3(b) to make the series fit in one figure.

shows, our theorem accurately predicts the optimal setting of exponential bits, which is 5 in this case, to minimize both the theoretical upper bound and the actual empirical result of the noise ball size, despite the theoretical upper bound being loose.

Second, we ran LP-SGD on the MNIST dataset [10]. To set up the experiment, we normalized the MNIST data to be in $[0, 1]$ by dividing by $255$, then subtracted out the mean for each features. We ran multiclass logistic regression using an L2 regularization constant of $10^{-4}$ and a step size of $\alpha = 10^{-4}$, running for $500$ total epochs (passes through the dataset) to be sure we converged. For this task, our (measured) problem parameters were $L = 37.41$, $L_1 = 685.27$, $\sigma = 2.38$, $\sigma_1 = 29.11$, and $d = 784$. In Figure 3(b), we plotted the observed loss gap, averaged across the last ten epochs, for LP-SGD using various 16-bit floating point formats. We also plot our theoretical bound on the loss gap, and the predicted optimal number of exponential bits to use based on that bound. Our results show that even though our bound is very loose for this task, it still predicts the right number of bits to use with reasonable accuracy. This experiment also validates the use of IEEE standard half-precision floating-point numbers, which have 5 exponential bits, for this sort of task.

## 5 Conclusion

In this paper, we present dimension-free bounds on the convergence of SGD when applied to low-precision training. We point out the conditions under which such bounds hold, for both convex and non-convex objectives. We further extend our results to non-linear methods of quantization: logarithmic quantization and floating point quantization. We analyze the performance of SGD under logarithmic quantization and demonstrate that NLQ is a promising method for reducing the number of bits required in low-precision training. We also presented ways in which our theory could be used to suggest how to allocate bits between exponent and mantissa when FPQ is used. We hope that our work will encourage further investigation of non-linear quantization techniques.

## References


[1] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1707–1718, 2017.

[2] Doug Burger. Microsoft unveils project brainwave for real-time ai. *Microsoft Research, Microsoft*, 22, 2017.

[3] Adrian M Caulfield, Eric S Chung, Andrew Putnam, Hari Angepat, Daniel Firestone, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, et al. Configurable clouds. *IEEE Micro*, 37(3):52–61, 2017.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[6] Dipankar Das, Naveen Mellempudi, Dheevatsa Mudigere, Dhiraj Kalamkar, Sasikanth Avancha, Kunal Banerjee, Srinivas Sridharan, Karthik Vaidyanathan, Bharat Kaul, Evangelos Georganas, et al. Mixed precision training of convolutional neural networks using integer operations. *arXiv preprint arXiv:1802.00930*, 2018.

[7] Christopher De Sa, Matthew Feldman, Christopher Ré, and Kunle Olukotun. Understanding and optimizing asynchronous low-precision stochastic gradient descent. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 561–574. ACM, 2017.

[8] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R Aberger, Kunle Olukotun, and Christopher Ré. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

[9] Christopher M De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of hogwild-style algorithms. In *Advances in neural information processing systems*, pages 2674–2682, 2015.

[10] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[11] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.

[12] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[13] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12. ACM, 2017.

[14] Urs Köster, Tristan Webb, Xin Wang, Marcel Nassar, Arjun K Bansal, William Constable, Oguz Elibol, Scott Gray, Stewart Hall, Luke Hornof, et al. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1742–1752, 2017.

[15] Edward H Lee, Daisuke Miyashita, Elaina Chai, Boris Murmann, and S Simon Wong. Lognet: Energy-efficient neural networks using logarithmic computation. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5900–5904. IEEE, 2017.

[16] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein. Training quantized nets: A deeper understanding. In *Advances in Neural Information Processing Systems*, pages 5813–5823, 2017.

[17] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*, 2018.

[20] 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 *International Conference on Machine Learning*, pages 4035–4043, 2017.

[21] 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 preprint arXiv:1606.06160*, 2016.

[22] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.