# SWALP: Stochastic Weight Averaging in Low-Precision Training

**Guandao Yang** [1] **Tianyi Zhang** [1] **Polina Kirichenko** [1] **Junwen Bai** [1]
**Andrew Gordon Wilson** [1] **Christopher De Sa** [1]

## Abstract

Low precision operations can provide scalability, memory savings, portability, and energy efficiency. This paper proposes SWALP, an approach to low precision training that averages low-precision SGD iterates with a modified learning rate schedule. SWALP is easy to implement and can match the performance of *full-precision* SGD even with all numbers quantized down to 8 bits, including the gradient accumulators. Additionally, we show that SWALP converges arbitrarily close to the optimal solution for quadratic objectives, and to a noise ball asymptotically smaller than low precision SGD in strongly convex settings.

## 1. Introduction

Training deep neural networks (DNNs) usually requires a large amount of computational time and power. As model sizes grow larger, training DNNs more efficiently and with less memory becomes increasingly important. This is especially the case when training on a special-purpose hardware accelerator; such ML accelerators are in development and used in industry (Jouppi et al., 2017; Burger, 2017). Many ML accelerators have limited on-chip memory, and many ML applications are memory bounded (Jouppi et al., 2017). It is desirable to fit numbers that are frequently used during the training process into the on-chip memory. One of the useful techniques for doing this is *low-precision computation*, since using fewer bits to represent numbers reduces both memory consumption and computational cost.

Training a DNN in low-precision usually results in worse performance compared to training in full precision. Many techniques have been proposed to reduce this performance gap (Zhou et al., 2016; Wu et al., 2018; Banner et al., 2018; Wang et al., 2018). One useful method is to compute forward and backward propagation with low-precision weights and accumulate gradient information in higher precision gradient accumulators (Courbariaux et al., 2015; Wu et al., 2018; Wang et al., 2018). Recently, Wang et al. (2018) showed that one could eliminate the performance gap between low-precision and high-precision training by quantizing all numbers except the gradient accumulator to 8 bits without changing the network structure, establishing the state-of-the-art result in low-precision training. Since gradient accumulators are frequently updated during training, it would be desirable to also represent and store them in low-precision (e.g. 8 bits). In this paper, we will focus on the setting where all numbers including the gradient accumulators are represented in low precision during training.

Independently from low-precision computation, *stochastic weight averaging* (SWA) (Izmailov et al., 2018a) has been recently proposed for improved generalization in deep learning. SWA takes an average of SGD iterates with a modified learning rate schedule and has been shown to lead to wider optima (Izmailov et al., 2018a). Keskar et al. (2016) also connect the width of the optimum and generalization performance. A wider optimum is especially relevant in the context of low-precision training as it is more likely to contain high-accuracy low-precision solutions. Izmailov et al. (2018a) also observed that SWA works well with a relatively high learning rate and can tolerate additional noise during training. Low-precision training, on the other hand, produces extra quantization noise during training and tends to underperform when the learning rate is low. Moreover, by averaging, one can combine weights that have been rounded up with those that been rounded down during quantization. For these reasons, we hypothesize that SWA can boost the performance of low-precision training and that performance improvement is more significant than in the case of SWA applied to full-precision training.

In this paper, we propose a principled approach using stochastic weight averaging while quantizing all numbers including the gradient accumulator and the velocity vectors during training. We prove that for quadratic objectives SWALP can converge to the optimal solution as well as to a smaller asymptotic noise ball than low-precision SGD for strongly convex objectives. Figure 1 illustrates the intuition behind SWALP. A quantized grid is only able to represent certain suboptimal solutions. By averaging we find centred

[1]Cornell University. Correspondence to: Guandao Yang <gy46@cornell.edu>, Tianyi Zhang <tz58@cornell.edu>.

*Proceedings of the $36^{th}$ International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. 

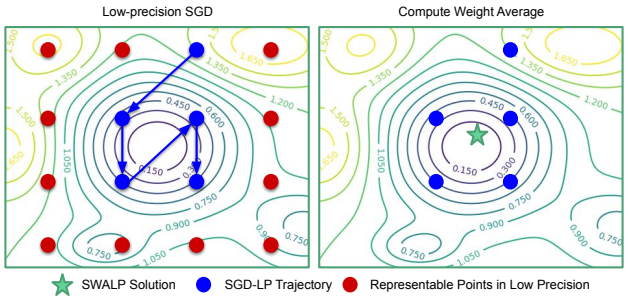


*Figure 1.* SWALP intuition. The trajectory of low-precision SGD, with a modified learning rate, over the training objective (with given contours), and the SWALP solution obtained by averaging.

solutions with better performance. Empirically, we demonstrate that training with 8-bit SWALP can match the full precision SGD baseline in deep learning tasks such as training Preactivation ResNet-164 (He et al., 2016) on CIFAR-10 and CIFAR-100 datasets (Krizhevsky & Hinton, 2009). In summary, our paper makes the following contributions:

- We propose a principled approach to using stochastic weight averaging in low-precision training (SWALP) where all numbers including the gradient accumulators are quantized. Our method is simple to implement and has little computational overhead.
- We prove that SWALP can reach the optimal solution for quadratic objectives with no loss of accuracy from the quantization. For strongly convex objectives, we prove that SWALP converges to a noise ball that is asymptotically smaller than that of low-precision SGD.
- We show that our method can significantly reduce the performance gap between low-precision and full-precision training. Our experiment results show that 8-bit SWALP can match the full-precision SGD baseline on CIFAR-10 and CIFAR-100 with both VGG-16 (Simonyan & Zisserman, 2014) and PreResNet-164.
- We provide code at `https://github.com/stevenygd/SWALP`.

## 2. Related Works

**Low Precision Computation.** Many works in low precision computation focus on expediting inference and reducing model size. Some compress trained models into low precision (Han et al., 2015); others train models to produce low-precision weights for inference (Hubara et al., 2016; Zhu et al., 2016; Aojun Zhou, 2017). In contrast to works that focus on inference (test) time low-precision computation, our work focuses on low-precision training. Prior work on low precision training mostly explores two directions. Some investigate different numerical representations and quantization methods (Gupta et al., 2015; Köster et al., 2017; Miyashita et al., 2016; Mellempudi et al., 2017; Wang et al., 2018); others explore building specialized layers using low-precision arithmetic (Wu et al., 2018; Rastegari et al., 2016; Zhou et al., 2016; Courbariaux et al., 2015; Banner et al., 2018). Our work is orthogonal to both directions since we improve on the learning algorithm itself.

**Stochastic Weight Averaging.** Inspired by the geometry of the loss function traversed by SGD with a modified learning rate schedule, Izmailov et al. (2018a) proposed Stochastic Weight Averaging (SWA), which performs an equally weighted average of SGD iterates with cyclical or high constant learning rates. Izmailov et al. (2018a) develop SWA for deep learning, showing improved generalization. While our work is inspired by Izmailov et al. (2018a), we focus on developing SWA for low-precision training.

**Convergence Analysis.** It is known that, due to quantization noise, low-precision SGD cannot necessarily produce solutions arbitrarily close to an optimum (Li et al., 2017). A recently developed variant of low-precision SGD, HALP (De Sa et al., 2018), has the ability to produce such arbitrarily close solutions (for general convex objectives) by dynamically scaling its low-precision numbers and using variance reduction. We will show that SWALP can also achieve arbitrarily close-to-optimal solutions (albeit only for quadratic objectives), while being computationally simpler. Li et al. (2017) analyze low-precision SGD, and even provide a convergence bound for low precision SWA. However, they use SWA only as a theoretical condition, not as a suggested algorithm. In contrast, we study SWA explicitly as a potential method that can improve low-precision training, and we use the averaging to improve the bound on the noise ball size. QSGD (Alistarh et al., 2017) studies the convergence properties of using low-precision numbers for communicating among parallel workers, and ZipML (Zhang et al., 2017) investigates the convergence properties of end-to-end quantization of the whole model. Our paper focuses on convergence properties of low-precision SWA, which we hope can be combined with these exciting prior works.

## 3. Methods

Izmailov et al. (2018a) show that SWA leads to a wider solution, works well with a high learning rate, and is robust toward training noise. These properties make SWA a good fit for low precision training, since a wider optimum is more likely to capture a representable solution in low precision, and low-precision training suffers from low learning rate and quantization noise. We will first introduce quantization methods make training low-precision (Sec 3.1), then present SWALP algorithm in Sec 3.2 and Sec 3.3.

### 3.1. Quantization

In order to use low-precision numbers during training, we define a *quantization function* $Q$, which rounds a real number to be stored in fewer bits. In this paper, we use *fixed-point quantization* with *stochastic rounding* to demonstrate the algorithm and analyze its convergence properties. Recently, many sophisticated quantization methods have been proposed and studied (Miyashita et al., 2016; Köster et al., 2017; Mellempudi et al., 2017; Courbariaux et al., 2014). We will use *block floating point* (Song et al., 2017) in our deep learning experiments (Sec 5).

**Fixed Point Quantization.** In stochastic rounding, numbers are rounded up or down at random such that $\mathbb{E}[Q(w)] = w$ for all $w$ that will not cause overflow. Explicitly, suppose we allocate $W$ bits to represent the quantized number and allocate $F$ of the $W$ bits to represent the fractional part of the number. The *quantization gap* $\delta = 2^{-F}$ represents the distance between successive representable fixed-point numbers. The upper limit of the representable numbers is $u = 2^{W-F-1} - 2^{-F}$ and the lower limit is $l = -2^{W-F-1}$. We write the quantization function as $Q_\delta : \mathbb{R} \to \mathbb{R}$ such that

$$Q_\delta(w) = \begin{cases} \text{clip}(\delta \left\lfloor \frac{w}{\delta} \right\rfloor, l, u) & \text{w.p. } \left\lceil \frac{w}{\delta} \right\rceil - \frac{w}{\delta} \\ \text{clip}(\delta \left\lceil \frac{w}{\delta} \right\rceil, l, u) & \text{w.p. } 1 - (\left\lceil \frac{w}{\delta} \right\rceil - \frac{w}{\delta}) \end{cases} \quad (1)$$

where $\text{clip}(x, l, u) = \max(\min(x, u), l)$. This quantization method has been shown to be useful for theory (Li et al., 2017) and has been observed to perform empirically better than quantization methods without stochastic rounding (Gupta et al., 2015).

**Block Floating Point (BFP) Quantization.** Floating-point numbers have individual exponents, and fixed-point numbers all share the same fixed exponent. For block floating-point numbers, all numbers within a block share the same exponent, which is allowed to vary like a floating-point exponent. Suppose we allocate $W$ bits to represent each number in the block and $F$ bits to represent the shared exponent. The shared exponent $E(\mathbf{w})$ for a block of numbers $\mathbf{w}$ is usually set to be the largest exponent in a block to avoid overflow (Song et al., 2017; Das et al., 2018). In our experiments, we simulated block floating point numbers by using the following formula to compute the shared exponent:

$$E(\mathbf{w}) = \text{clip}(\lfloor \log_2(\max_i |\mathbf{w}_i|) \rfloor, -2^{F-1}, 2^{F-1} - 1)$$

We then apply equation (1) with $\delta$ replaced by $2^{-E(\mathbf{w})+W-2}$ to quantize all numbers in $\mathbf{w}$.

For deep learning experiments, BFP is preferred over fixed-point because BFP usually has less quantization error caused by overflow and underflow when quantizing DNN models (Song et al., 2017). We will discuss how to design appropriate blocks in Sec 5, and show that choosing appropriate block design can result in better performance.

**Algorithm 1** SWALP

**Require:** Initial after-warm-up weight $w_0$; learning rate $\alpha$; total number of iterations $T$; cycle length $c$; random gradient samples $\nabla \tilde{f}(w_t)$; quantization function $Q$.
  $\bar{w}_0 \leftarrow w_0$ {Accumulator for SWA (high precision)}
  $m \leftarrow 1$ {Number of models that have been averaged}
  **for** $t = 1, 2, \ldots, T$ do **do**
    $w_t \leftarrow Q(w_{t-1} - \alpha \nabla \tilde{f}_t(w_{t-1}))$ {Training with weight quantization; $w_t$ is stored in low precision}
    **if** $t \equiv 0 \pmod{c}$ **then**
      $\bar{w}_m \leftarrow (\bar{w}_{m-1} \cdot m + w_t)/(m+1)$ {Update model with weight averaging in high precision}
      $m \leftarrow m + 1$ {Increment model count}
    **end if**
  **end for**
  **return** $\bar{w}$

### 3.2. Algorithm

In the *warm-up phase*, we first run regular low-precision SGD to produce a pre-trained model $w_0$. SWALP then continues to run low-precision SGD, while periodically averaging the current model weight into an accumulator $\bar{w}$, which SWALP eventually returns. A detailed description is presented in Algorithm 1. SWALP is substantially similar to the algorithm in Izmailov et al. (2018a), except that we use a constant learning rate and low-precision SGD. The convergence analysis in Sec 4 will be all based on Algorithm 1.

State-of-the-art low-precision training approaches usually separate weights from gradient accumulators (Courbariaux et al., 2015; Zhou et al., 2016; Wu et al., 2018; Wang et al., 2018). Expensive computations in forward and backward propagation are done with the low-precision weights (e.g., 8 bits), while the gradient information is accumulated onto a higher precision copy of the weights (e.g. 16 bits). Formally, the updating step with higher precision gradient accumulator can be written as $w_{t+1} = w_t - \alpha \nabla \tilde{f}_t(Q(w_t))$, where $w_t$ is the gradient accumulator and $Q(w_t)$ is the weight. However, such an approach needs to store the high precision accumulators in low-latency memory (e.g. on-chip when running on an accelerator), which limits the memory efficiency. SWALP quantizes the gradient accumulator so that we only need to store the low-precision model in low-latency memory during training. Meanwhile, the averaged model is accessed infrequently so that it can be stored in higher-latency memory (e.g. off-chip when running on an accelerator).

Note that in Algorithm 1, we only quantize the gradient accumulator while leaving the quantization of the gradient, the layer activations, and back-propagation signals in full precision. In practice, however, it is desirable to quantize all above mentioned numbers. We make this simplification

**Algorithm 2** SWALP with all numbers quantized.

**Require:** $L$ layers DNN $\{f_1, \dots, f_L\}$; Scheduled learning rate $\alpha_t$; Momentum $\rho$; Initial weights $w_0^{(i)}, \forall l \in [1, L]$; Total iterations $T$; Warm-up iterations $S$; Cycle length $c$; Quantization functions $Q_W$, $Q_A$, $Q_G$, $Q_E$, and $Q_M$; Loss function $\mathcal{L}$; Data batch sequence $\{(x_i, y_i)\}_{i=1}^T$.
$\bar{w}_0^{(l)} \leftarrow 0, \forall l \in [1, L]$; $m \leftarrow 0$
**for** $t = 1, 2, \dots, T$ **do**
  **1. Forward Propagation:**
  $a_t^{(0)} = x_i$
  $a_t^{(l)} = Q_A(f_l(a_t^{(l-1)}, w_t^{(l)})), \forall l \in [1, L]$
  **2. Backward Propagation:**
  $e_t^{(L)} = \nabla_{a_t^{(L)}} \mathcal{L}(a_t^{(L)}, y_t)$
  $e_t^{(l-1)} = Q_E(\frac{\partial f_l(a_t^{(l)})}{\partial a_t^{(l-1)}} e_t^{(l)}), \forall l \in [1, L]$
  $g_t^{(l)} = Q_G(\frac{\partial f_l}{\partial w_t^{(l)}} e_t^{(l)}), \forall l \in [1, L]$
  **3. Low Precision SGD Update (with momentum):**
  $v_t^{(l)} \leftarrow \rho Q_M(v_{t-1}^{(l)}) + g_t^{(l)}, \forall l \in [1, L]$
  $w_t^{(l)} \leftarrow Q_W(w_{t-1} - \alpha_t v_t^{(l)}), \forall l \in [1, L]$
  **4. High Precision SWA Update:**
  **if** $t > S$ and $(t - S) \equiv 0 \pmod{c}$ **then**
    $\bar{w}_m^{(l)} \leftarrow (\bar{w}_{m-1}^{(l)} \cdot m + w_t^{(l)})/(m + 1), \forall l \in [1, L]$
    $m \leftarrow m + 1$
  **end if**
**end for**
**return** $\bar{w}$

for the convenience of the theoretical analysis in Sec 4, and following previous theoretical works in this space (Li et al., 2017; De Sa et al., 2018). We will discuss how to quantize other numbers during training in the next section.

### 3.3. Quantizing Other Numbers

In order to train DNNs with low-precision storage, we need to also quantize other numbers during training. We follow prior convention to quantize the weights, activations, back-propagation errors, and gradients signals (Wu et al., 2018; Wang et al., 2018). Since we quantized the gradient accumulators $w_t$ into low-precision, there is no need to differentiate them from model weights. To use momentum during training, we need to store the velocity tensors in low precision, so we modified the SGD update as follows:

$$v_t = \rho \cdot Q_M(v_{t-1}) + Q_G(\nabla \tilde{f}_t(w_{t-1}))$$
$$w_t = Q_W(w_{t-1} - \alpha \cdot v_t)$$

where $Q_M$, $Q_G$, and $Q_W$ are quantizers for momentum, gradients, and weights respectively. For simplicity, we set $Q_M = Q_G$ (i.e. both quantized to 8 bits). We describe the details in Algorithm 2. Our deep learning experiments will use Algorithm 2 unless specified otherwise.

Although SWA adds minor computation and memory overhead by averaging weights, the fact that averaging could be done infrequently (i.e. once per epoch) and that the weight communication is one way (i.e. from accelerator to host) makes it possible to separate the low-precision training workload from the model averaging workload. We could use hardware specialized for low-precision training to accelerate the SGD and to occasionally ship the weights in low precision to a separate device that computes weight averaging. For instance, one could train the model in low precision on a GPU, and the averaging could be computed on a CPU once per epoch. However, in this paper, we will focus on the statistical properties of SWALP and will leave the hardware discussion to future work. Though the averaging workload (i.e. Step(4) in algorithm 2) is typically done in higher precision, we empirically show in Sec 5.1 that SWALP can achieve comparable performance when the averaging is performed with low-precision computation.

## 4. Convergence Analysis

In this section, we analyze the convergence of SWALP theoretically and compare it to SGD-LP. Specifically, we first prove that SWALP can pierce the quantization noise ball of SGD-LP and can converge to the optimal solution for quadratic objectives (Sec 4.1). Then, we generalize this theory to strongly convex objectives (Sec 4.2) where we show that SWALP converges to a noise ball with better asymptotic dependency on the number of bits compared to SGD-LP. These results are empirically verified in Sec 4.3.

### 4.1. Convergence of SWALP for Quadratic Objectives

It is known that conventional low-precision SGD cannot obtain arbitrarily accurate solutions to optimization problems since it can only represent so much. If the optimal parameter is not one of the representable low-precision numbers, then the best SGD-LP can possibly do is to output the closest representable number – and it is not even guaranteed to do this. One recent algorithm, HALP, circumvents this problem by dynamically re-centering and re-scaling the representable numbers to produce arbitrarily accurate solutions with low precision iterates (De Sa et al., 2018). In this subsection, we will demonstrate that SWALP can also achieve this property for quadratic objectives with a simple training procedure. A detailed proof is included in the appendix.

Consider the quadratic objective function $f(w) = \frac{1}{2}(w - w^*)^T A(w - w^*)$ for some symmetric matrix $A \in \mathbb{R}^{d \times d}$ and optimal solution $w^* \in \mathbb{R}^d$. Assume $A \succeq \mu I$ for some constant $\mu > 0$, the strong convexity parameter of this function. Suppose that we want to minimize $f$ using SWALP with gradient samples $\nabla \tilde{f}(w)$ that satisfy $\mathbb{E}[\nabla \tilde{f}(w)] = \nabla f(w) = A(w - w^*)$. Suppose that the variance of these samples always satisfies $\mathbb{E}[\|\nabla \tilde{f}(w) - \nabla f(w)\|_2^2] \leq \sigma^2$ for some

constant $\sigma > 0$; this is a standard assumption used in the analysis of SGD. Then we can prove the following.

**Theorem 1.** *Suppose we run SWALP under the above assumptions with cycle length $c$ and $0 < \alpha < \frac{1}{2}\|A\|_2$. The expected squared distance to the optimum of SWALP's output $\bar{w}$ is bounded by*

$$\mathbb{E}\left[\|\bar{w} - w^*\|^2\right] \leq \frac{\|w_0 - w^*\|^2}{\alpha^2\mu^2T^2} + \frac{c(\alpha^2\sigma^2 + \frac{\delta^2 d}{4})}{\alpha^2\mu^2 T}.$$

Theorem 1 shows that SWALP will converge to the optimal solution at a $O(1/T)$ rate. Since $\mathbb{E}[\|\bar{w}_T - w^*\|_2^2]$ converges to 0 regardless of what $\delta$ is, we can get an arbitrarily precise solution no matter what numerical precision we use for quantization is, as long as we train for enough iterations. This result is surprising since SWALP has the same $O(1/T)$ asymptotic convergence rate as full precision SGD, even though SWALP cannot even evaluate gradient samples at points that are arbitrarily close to the optimal solution.

### 4.2. Convergence of SWALP for General Strongly Convex Objectives

To generalize Theorem 1 from quadratic settings to strongly convex settings, we want to construct a bound that is tight with our bound in the quadratic case: one that depends on how much the objective function differs from a quadratic. A natural way to measure this is the Lipschitz constant $M$ of the second derivative of $f$, which is defined such that

$$\forall x, y \in \mathbb{R}^d, \quad \|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \leq M\|x - y\|_2$$

where $\|\cdot\|_2$ denotes the matrix induced 2-norm, and $M = 0$ only if $f$ is a quadratic function.

We prove our result in two steps. First, we bound the trajectory of low precision SGD within some distance of $w^*$ (a noise ball). Then, we leverage a method similar to the proof of Theorem 1 to analyze the dynamics of SWALP, keeping track of the effect caused by the function not being quadratic as an extra error term that we bound in terms of $M$. We give a tight bound that converges with an $O(1/T)$ rate to a noise ball with squared error proportional to $M^2$.

Let $f(w)$ be a function that is strongly convex with parameter $\mu$, Lipschitz continuous with parameter $L$, and has global minimum $w^*$. Assume that we run SWALP with gradient samples $\nabla\tilde{f}_t$ that satisfy $\mathbb{E}[\nabla\tilde{f}(w)] = \nabla f(w)$. Suppose the distance from the gradient samples to the actual gradient is bounded by some constant $G$ such that $\|\nabla\tilde{f}_t(w) - \nabla f(w)\| \leq G$ for all points $w$ that may appear in the course of execution. Similar to Sec 4.1, we assume that no overflow happens during quantization.

**Lemma 1.** *Under the above conditions, suppose that we run low-precision SGD with step size $\alpha = \sqrt{\frac{\delta^2 d}{G^2}}$. Assume $\delta$ is small enough so that it satisfies $(1 - 2\alpha\mu + \alpha^2 L)^2 \leq 1 - 2\alpha\mu$ and $\alpha\mu < 1$. If we run for $T$ iterations such that $T \geq \frac{2G}{\mu\delta\sqrt{d}} \log\left(\frac{\mu\|w_0 - w^*\|^2}{44G\delta\sqrt{d}}\right)$, then for some fixed constant $\chi$ that is independent of dimension and problem parameters,*

$$\mathbb{E}[\|w_T - w^*\|^4] \leq \frac{\chi^2 G^2\delta^2 d}{\mu^2}.$$

**Theorem 2.** *Suppose that we run SWALP under the above conditions, with the parameters specified in Lemma 1. Also, suppose that we first run a warm-up phase and start averaging at some point $w_0$ after enough iterations of low-precision SGD such that the bound of Lemma 1 is already guaranteed to apply for this and all subsequent iterates. Let $\bar{w}$ be the output of SWALP using cycle length $c$, and $\gamma = min(\alpha^2\mu^2c^2, 1)$. The expected squared distance to the optimum of the output of SWALP is bounded by*

$$\mathbb{E}[\|\bar{w} - w^*\|^2] \leq \frac{3\chi^2 M^2 G^2\delta^2 d}{\mu^4} + \frac{6G^2 c}{\mu^2 T} + \frac{528\sqrt{d}\delta G^3 c^2}{\gamma\mu T^2}.$$

The first term $3\chi^2 M^2 G^2\mu^{-4}\delta^2 d$ represents the squared errors caused by the noise ball, the asymptotic accuracy floor to which SWALP will converge given enough iterations. The error caused by this noise ball is proportional to $M^2$, which measures how much the objective function differs from the quadratic setting. The second and third term converge to 0 at a $O(1/T)$ rate, which shows that the whole bound will converge to the noise ball at a $O(1/T)$ rate. Our proof leverages some techniques used in prior work (Moulines & Bach, 2011). In particular, Moulines & Bach (2011) showed that one could provide a better bound in SGD using $M$, the third derivative of a strongly convex function. The proofs of our results here are provided in detail in the appendix.

To the best of our knowledge, our result in Theorem 2 is the tightest known bound for low precision SWA. Li et al. (2017) also provide results analyzing a convergence bound for LP-SGD with weight averaging, but their bound is proportional to $\delta$, whereas ours is proportional to $\delta^2$.[1] If we consider a fixed-point representation using $F$ fractional bits, then our bound is proportional $\delta^2 = 2^{-2F}$ whereas the bound from prior work proportional to $\delta = 2^{-F}$. Asymptotically, we can say that this halves the number of bits we need to decrease the noise ball by some factor, compared with the prior bound. As this prior bound also describes the convergence behaviour, we can equivalently say that the number of bits in SWALP has double the effect on the noise ball, compared with SGD-LP.

To understand whether SWALP can achieve a better bound than SGD-LP, we need to answer the following question: can SGD-LP algorithm potentially achieve a better bound than the $O(\delta)$ bound proved in Li et al. (2017)? In the following theorem, we show that this is not possible:

[1] Although their bound is stated in terms of the objective gap $f(\bar{w}_T) - f(w^*)$ whereas ours is the squared distance to the optimum, these metrics are directly comparable as they may differ by at most a factor of $\mu$: $2(f(\bar{w}_T) - f(w^*)) \geq \mu\|\bar{w}_T - w^*\|_2^2$.

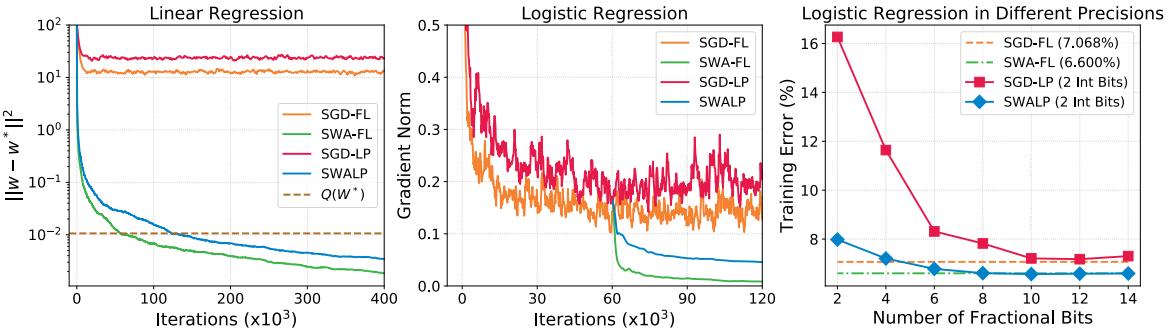


*Figure 2.* Empirical verification of two theorems with linear regression and logistic regression. (Left) SWALP converges below the quantization noise and to the optimal solution in linear regression; (Middle) SWALP can converge to a smaller noise ball than SGD-LP and SGD; (Right) SWALP requires less than half of the float bits to achieve the same performance compared to SGD-LP.

**Theorem 3.** *Consider the one-dimensional objective function* $f(x) = \frac{1}{2}x^2$ *with gradient samples* $\tilde{f}'(w) = w + \sigma u$ *where* $u \sim \mathcal{N}(0, 1)$. *Compute* $w_T$ *recursively using the quantized SGD updating step:* $w_{t+1} = Q_\delta(w_t - \alpha \tilde{f}'(w_t))$. *Then there exists a constant* $A > 0$ *such that for all step size* $\alpha > 0$, *we have* $\lim_{T\to\infty} \mathbb{E}[w_T^2] \geq \sigma\delta A$.

The proof is provided in the appendix. Theorem 3 shows that there exists a strongly convex objective function such that $\mathbb{E}[(w_T - w^*)^2] \geq O(\delta)$. This shows that the asymptotic lower bound for low-precision SGD with the gradient accumulator quantized at every step is $O(\delta)$, which is an asymptotically worse bound compared to the $O(\delta^2)$ bound obtained for SWALP. Therefore, SWALP achieves a better asymptotic dependency on the quantization gap $\delta$.

### 4.3. Experimental validation

**Linear regression.** We will use linear regression on a synthetic dataset to empirically verify Theorem 1. For details on how we generate the synthetic dataset, please refer to appendix. We train linear regression models using float SGD (SGD-FL), float SWA (SWA-FL), low precision SGD (SGD-LP), and SWALP. Low-precision models use 8-bit fixed point numbers with 6 fractional bits (i.e., $\delta = 2^{-6}$).

The results are displayed in Figure 2 where we plot the square distance between $w_t$ (or $\bar{w}_t$ for SWALP) and the optimal solution $w^*$. For reference, we also plot in Figure 2 the squared distance between $Q(w^*)$ and $w^*$ to illustrate the size of quantization noise. We observe that both SGD-LP and SGD-FL converge to a noise ball, and SGD-LP's noise ball is further away from $w^*$ than SGD-FL's noise ball—indicating that we are operating in a regime where quantization noise matters. SWA-FL and SWALP, on the other hand, both converge asymptotically towards the optimal solution. Notably, SWALP pierces the quantization noise ball and even outperforms $Q(w^*)$.[2] The asymptotic convergence rate of SWALP and SWA-FL appears to be the same, and both appear to follow a $O(1/T)$ convergence trajectory, which validates our results in Theorem 1.

**Logistic regression.** To empirically validate Theorem 2, we use logistic regression with L2 regularization on the MNIST dataset (LeCun et al., 1998). Following prior work (De Sa et al., 2018; Johnson & Zhang, 2013), we choose $10^{-4}$ weight decay, which makes the objective a strongly convex function with $M \neq 0$. Similarly to our linear regression experiment, we use SGD-FL, SWA-FL, SGD-LP, and SWALP to train logistic regression models. For this experiment, we measure the norm of gradient at each iteration to illustrate the convergence of the algorithm; this is a more useful metric because MNIST is sparse and poorly conditioned, and it is a metric that has been used for logistic regression on MNIST in previous work (De Sa et al., 2018). For SWALP and SGD-LP, we use 4-bit word length and 2-bit fractional length. See appendix for detailed hyper-parameters.

In Figure 2, we again observe that SGD-LP converges to a larger noise ball than SGD-FL, which is caused by the additional quantization noise of low precision training. Both SWA-FL and SWALP pierce the noise ball of SGD-FL. However, unlike SWA-FL whose gradient norm appears to converge to zero, SWALP still hits a noise ball, albeit one that is much smaller than the one from SGD. This validates Theorem 2, which predicts that SWALP will converge to a noise ball when the problem setting is strongly convex yet non-quadratic (i.e. $M \neq 0$).

Figure 2 also compares the training errors of logistic regression trained with different numbers of fractional bits, which determine $\delta$ in the theorem. Both SGD-LP and SWALP are trained with 2 integer bits and the same hyper-parameters, but we vary the number of fractional bits. SWALP recovers

[2]Note that SWALP is able to do this because it represents the averaged model in full precision.

*Table 1.* Test error (%) on CIFAR-10 and CIFAR-100 for VGG16 and PreResNet-164 trained in different quantization setting.

| | | FLOAT | | 8-BIT BIG-BLOCK | | 8-BIT SMALL-BLOCK | |
|---|---|---|---|---|---|---|---|
| DATASET | MODEL | SGD | SWA | SGDLP | SWALP | SGDLP | SWALP |
| CIFAR-10 | VGG16 | 6.81 ±0.09 | 6.51 ±0.14 | 8.23 ±0.08 | 7.36 ±0.05 | 7.61 ±0.15 | 6.70 ±0.12 |
| | PRERESNET-164 | 4.63 ±0.18 | 4.03 ±0.10 | 6.51 ±0.08 | 5.61 ±0.17 | 5.83 ±0.05 | 5.01 ±0.14 |
| CIFAR-100 | VGG16 | 27.23 ±0.17 | 25.93 ±0.21 | 30.56 ±0.67 | 28.66 ±0.17 | 29.59 ±0.32 | 26.65 ±0.29 |
| | PRERESNET-164 | 22.20 ±0.57 | 19.95 ±0.19 | 25.84 ±0.52 | 24.92 ±0.60 | 23.97 ±0.08 | 21.76 ±0.28 |

the performance of the full precision SGD model with only 4 fractional bits, while SGD-LP needs 10 bits to do so. This result validates the claim that SWALP needs only half the number of bits to achieve the same performance, which is predicted by Theorem 2 in terms of the asymptotic upper bound. Although our theory bounds the convergence in terms of the training set, this conclusion still holds when evaluated on MNIST test set (see appendix).

## 5. Experiments

In this section, we demonstrate the effectiveness of SWALP on non-convex problems in deep learning.

**Datasets.** We use the CIFAR (Krizhevsky & Hinton, 2009) and ImageNet (Russakovsky et al., 2014) datasets for our experiments. Following prior work (Izmailov et al., 2018a; Wu et al., 2018), we apply standard preprocessing and data augmentation for experiments on CIFAR datasets. Preprocessing and data augmentation for ImageNet are adapted from the public PyTorch example (Paszke et al., 2017).

**Architectures.** We use the VGG-16 (Simonyan & Zisserman, 2014) and Pre-activation ResNet-164 (He et al., 2016) on CIFAR datasets as in Izmailov et al. (2018a;b). For ImageNet experiments, we use ResNet-18 (He et al., 2015b).

**Block Design.** Song et al. (2017) shows that appropriate block assignments are essential to achieve good performance with BFP. In our experiment, we will test two block assignments: *Big-block* and *Small-block*. The *Big-block* design puts all numbers within the same tensor into the same block. For example, the activation of a convolution layer may have shape $(B, C, W, H)$, and the Big-block design assigns one shared exponent for $B \times C \times W \times H$ numbers in this tensor. The *Small-block* design will follow Song et al. (2017) and Zhou et al. (2016) except that we assign only one exponent for the following tensors: 1) bias in convolution and fully connected layers; 2) the learned scaling parameter in batch normalization layers, and 3) the learned shift parameter in batch normalization layers. We empirically found that with these modifications, we can reduce memory consumption while regularizing the model. Storing our VGG16 network in 32-bit float requires 53.33 MB memory, while using 8-bit Small-block BFP with 8-bit shared exponents reduces this memory requirement to 14.59 MB. Moreover, Small-block design uses only 5.2 KB more memory compared to the Big-block design, which is a negligible overhead. In Sec 5.1, we will compare the performance between these two block assignment methods.

### 5.1. CIFAR Experiments

We demonstrate how SWALP (Algorithm 2) is applied to train DNNs in image classification tasks on CIFAR-10 and CIFAR-100. To examine how the block design of BFP affects performance, we train each network with both Big-block and Small-block BFP. We use the reported hyper-parameters in Izmailov et al. (2018a), for full-precision SGD, SWA, and low-precision SGD runs. SWALP's hyper-parameters are obtained from grid search on a validation set. Please see the Appendix for more detail.

Table 1 shows results for different combinations of architecture, dataset, and quantization method. First, the Small-block model outperforms all the Big-block models by a large margin. SWALP also consistently outperforms SGD-LP across architectures and datasets, showing the promise of SWA for low precision training in deep learning. Although the performance of SWALP does not match that of full precision SWA, the performance improvement of SWALP over SGD-LP is larger than that of SWA over SGD. For example, for VGG16 trained with 8-bit Small-block BFP on CIFAR100, applying SWALP improves the SGD-LP performance by $2.94\%$ whereas the corresponding full-precision improvement is only $1.3\%$. Notably, the performance of SWALP for VGG16 and PreResNet-164 trained with Small-block BFP can match that of full-precision SGD. On CIFAR-100 dataset, SWALP with Small-block BFP even outperforms the full-precision SGD baseline by $0.58\%$ with VGG-16 and by $0.44\%$ with PreResNet-164.

**Averaging in Different Frequency.** Both Theorem 1 and Theorem 2 show that averaging more frequently leads to faster convergence, but changing $c$ does not affect the final convergence results. In this section, we empirically study the effect of $c$ using VGG-16 and CIFAR-100. Previously, all runs compute the weight average once per epoch, following the convention from Izmailov et al. (2018a). We compare such default averaging frequency with higher fre-

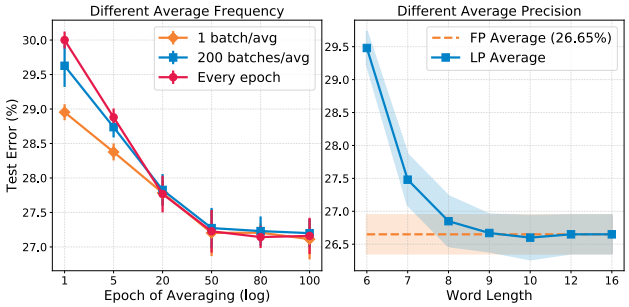


*Figure 3.* CIFAR-100 classification test error (%). **Left:** Different averaging frequency. **Right:** Different averaging precision.

quencies including averaging every batch and every 200 batches. We keep the quantization method (Small-block BFP) and all other hyper-parameters unchanged.

The left panel of Figure 3 show that averaging more frequently leads to faster convergence. For example, averaging every batch achieves an error rate of $28.85\%$ after one epoch, which is much lower than averaging only once per epoch (i.e. $30.00\%$). However, the performance gap between high and low averaging frequency quickly disappears as we apply SWALP for more epochs. After 20 epochs, we observe almost no difference in performance between averaging frequencies. This result suggests that the averaging frequency does not affect final performance after sufficiently many epochs. We also observe that the test error keeps decreasing even after 20 epochs for all averaging frequencies, so averaging for more epochs is necessary for convergence.

**Averaging in Different Precision.** We study the effect of averaging in different precisions while keeping quantization and other hyper-parameters unchanged. The weight averaging is computed with low-precision operations as follows: $\bar{w}_{m+1} \leftarrow Q_{\text{SWA}}((\bar{w}_m \cdot m + w_t)/(m+1))$, where $Q_{\text{SWA}}$ is a BFP quantizer with word length $W_{\text{SWA}}$. In this experiment, we vary $W_{\text{SWA}}$ from 6 to 16 bits. The averaging updates are first computed in high precision before quantizing down to $W_{\text{SWA}}$ bits. During inference, we will quantize the activation into $W_{\text{SWA}}$-bit Small-block BFP.

We report the results in the right panel of Figure 3 of training a VGG-16 model with 8-bit Small-block BFP. We observe that the averaged weights can be computed in 9-bits with essentially no performance decrease compared to averaging in full precision. When we use 8-bits BFP numbers to store the averaged model during training, there is a minor performance loss compared to full precision averaging. That being said, the error rate ($26.85\%$) is still lower than those of the SGD-LP baseline ($29.59\%$) and the SGD-FP baseline($27.23\%$). Averaging in lower than 8-bit precision tends to substantially hurt performance. This suggests that in order to fully realize the benefits of SWALP, one needs to compute the weight averaging in a slightly higher precision (i.e. for 8-bit weights, we need to compute the average in 9-bit). These results suggest that we could replace step (4) of Algorithm 2 with this quantized averaging step to eliminate high-precision storage during training. With such modifications, SWALP can produce a low-precision model that performs comparably to a *full-precision* SGD model without any high precision storage.

*Table 2.* ImageNet experiment results with ResNet-18. 90+$X$ epochs of SWA (or SWALP) means running weight averaging for $X$ epochs starting at 90 epoch.

| Model | Epochs | Top-1 Error (%) |
|---|---|---|
| SGD | 90 | 30.49 |
| SWA | 90+10 | 29.74 |
| SDGLP | 90 | 36.56 |
| SWALP | 90+10 | 34.89 |
| SWALP | 90+30 | 34.34 |
| SWALP † | 90+30 | 34.18 |

† Averaging 50 times per epoch.

### 5.2. ImageNet Experiments

We further evaluate SWALP's performance for a large scale image classification task, by training a ResNet-18 on ImageNet. We obtain the results for SGD and SGD-LP using hyper-parameters suggested by He et al. (2015b). For all low-precision experiments, we use Small-block BFP. Please see Appendix for the details on hyper-parameters. We present the results in Table 2.

ImageNet contains substantially more information than CIFAR, and is more sensitive to hyper-parameter tuning; consequently, there is a greater drop in performance for ImageNet when using low precision computations: from $30.49\%$ with SGD to $36.56\%$ with SGD-LP. Although in preliminary experiments, SWALP does not entirely close this larger performance gap, achieving $34.89\%$ error after 10 epochs, it still leads to a substantial improvement in accuracy over SGD-LP. The performance gain with SWALP over SGD-LP is also greater than for SWA over SGD: after 10 epochs of weight averaging, there is a $1.67\%$ improvement in low precision compared to $0.84\%$ in full precision.

## 6. Conclusion

We have proposed SWALP, a convenient approach to low-precision training that outperforms low-precision SGD and is competitive with full-precision SGD, even when trained in 8 bits. SWALP is based on averaging SGD iterates in low precision, motivated by the intuition that averaging could reduce the quantization noise introduced by stochastic rounding. In the future, it would be exciting to explicitly consider loss geometry in building low precision solutions.

## Acknowledgements

Polina Kirichenko and Andrew Gordon Wilson were supported by NSF IIS-1563887, an Amazon Research Award, and Facebook Research Award. We thank Google Cloud Platform Research Credits program for providing computational resources.

## References

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1707–1718, 2017.

Aojun Zhou, Anbang Yao, Y. G. L. X. Y. C. Incremental network quantization: Towards lossless cnns with low-precision weights. In *International Conference on Learning Representations,ICLR2017*, 2017.

Banner, R., Hubara, I., Hoffer, E., and Soudry, D. Scalable methods for 8-bit training of neural networks. *arXiv preprint arXiv:1805.11046*, 2018.

Burger, D. Microsoft unveils Project Brainwave for real-time AI. https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/, 2017. Accessed: 2018-02-08.

Courbariaux, M., Bengio, Y., and David, J.-P. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pp. 3123–3131, 2015.

Das, D., Mellempudi, N., Mudigere, D., Kalamkar, D. D., Avancha, S., Banerjee, K., Sridharan, S., Vaidyanathan, K., Kaul, B., Georganas, E., Heinecke, A., Dubey, P., Corbal, J., Shustrov, N., Dubtsov, R., Fomenko, E., and Pirogov, V. O. Mixed precision training of convolutional neural networks using integer operations. *ICLR*, 2018.

De Sa, C., Leszczynski, M., Zhang, J., Marzoev, A., Aberger, C. R., Olukotun, K., and Ré, C. High-accuracy low-precision training. *CoRR*, abs/1803.03383, 2018. URL http://arxiv.org/abs/1803.03383.

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

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. URL http://arxiv.org/abs/1510.00149.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015b. URL http://arxiv.org/abs/1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4107–4115. 2016.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. 2018a.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization – released code. 2018b. URL https://github.com/timgaripov/swa.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12. ACM, 2017.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016. URL http://arxiv.org/abs/1609.04836.

Köster, U., Webb, T., Wang, X., Nassar, M., Bansal, A. K., Constable, W., Elibol, O., Gray, S., Hall, S., Hornof, L., et al. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 1742–1752, 2017.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, H., De, S., Xu, Z., Studer, C., Samet, H., and Goldstein, T. Training quantized nets: A deeper understanding. In *Advances in Neural Information Processing Systems*, pp. 5813–5823, 2017.

Mellempudi, N., Kundu, A., Das, D., Mudigere, D., and Kaul, B. Mixed low-precision deep learning inference using dynamic fixed point. *arXiv preprint arXiv:1701.08978*, 2017.

Miyashita, D., Lee, E. H., and Murmann, B. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

Moulines, E. and Bach, F. R. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pp. 451–459, 2011.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. 2017.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Song, Z., Liu, Z., Wang, C., and Wang, D. Computation error analysis of block floating point arithmetic oriented convolution neural network accelerator design. *arXiv preprint arXiv:1709.07776*, 2017.

Wang, N., Choi, J., Brand, D., Chen, C.-Y., and Gopalakrishnan, K. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems*, pp. 7686–7695, 2018.

Wu, S., Li, G., Chen, F., and Shi, L. Training and inference with integers in deep neural networks. *ICLR*, 2018.

Zhang, H., Li, J., Kara, K., Alistarh, D., Liu, J., and Zhang, C. ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 4035–4043, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

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

Zhu, C., Han, S., Mao, H., and Dally, W. J. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.