Towards a Deeper Understanding of Training Quantized Neural Networks

Hao Li [*1] **Soham De** [*1] **Zheng Xu** [1] **Christoph Studer** [2] **Hanan Samet** [1] **Tom Goldstein** [1]

Abstract

Training neural networks with coarsely quantized weights is a key step towards learning on embedded platforms that have limited computing resources, memory capacity, and power consumption. Numerous recent publications have studied methods for training quantized networks, but these studies have been purely experimental. In this work, we investigate the theory of training quantized neural networks by analyzing the convergence properties of some commonly used methods. Our main result shows that training algorithms that exploit high-precision representations have an important annealing property that purely quantized training methods lack, which explains many of the observed empirical differences between these types of algorithms.

1. Introduction

Neural networks (NNs) are an integral part of numerous state-of-the-art computer vision and natural language processing tasks. Because of their high memory requirements and computational complexity, networks are usually trained using powerful hardware. There is an increasing interest in training and deploying neural networks directly on battery-powered devices, such as cell phones or other platforms. Such low-power embedded systems are, however, memory and power limited, and in some cases lack basic support for floating-point arithmetic.

To make neural nets practical on embedded systems, many researchers have focused on training nets with *coarsely quantized* weights. For example, weights may be constrained to integer/binary values, or represented using low-precision (8 bits or less) fixed-point numbers. Quantized nets offer the potential of superior memory and computation efficiency, while achieving performance competitive with state-of-the-art high-precision nets. Quantized weights can dramatically reduce memory size and access bandwidth, increase power efficiency, exploit hardware-friendly bitwise operations, and accelerate inference throughput (Marchesi et al., 1993; Courbariaux et al., 2016; Rastegari et al., 2016).

Handling low-precision weights is difficult and motivates interest in new training methods. When learning rates are small, stochastic gradient methods make small updates to the weights. Binarization/discretization of weights after each training iteration "rounds off" these small updates and causes training to stagnate (Courbariaux et al., 2016). Thus, the naïve approach of quantizing weights using rounding yields poor results when weights are represented using a small number of bits. Other approaches include classical stochastic rounding methods (Gupta et al., 2015), as well as schemes that combine full-precision floating-point weights with discrete rounding procedures (Courbariaux et al., 2015). While some of these schemes seem to work in practice, results in this area are largely experimental, and little work has been devoted to explaining the excellent performance of some methods, the poor performance of others, and the important differences in behavior between these methods.

Contributions This paper studies quantized training methods from a theoretical perspective, with the goal of understanding the differences in behavior, and reasons for success or failure, of various methods. In particular, we show that recent powerful methods like BinaryConnect (BC) (Courbariaux et al., 2015) are capable of concentrating on minimizers and solving discrete problems up to a high level of accuracy. On the other hand, we show that classical stochastic rounding (SR) methods (Gupta et al., 2015) lack an important annealing property that is required for non-convex optimization. This explains the large differences observed in empirical performance between classical SR methods and the more powerful BC methods.

2. Related Work

The arithmetic operations of deep networks can be encoded down to 8-bit fixed-point without significant deterioration

[*]Equal contribution. Author ordering determined by coin flip. [1]Department of Computer Science, University of Maryland, College Park, MD [2]Department of Electrical and Computer Engineering, Cornell University, Ithaca, NY. Correspondence to: Hao Li <haoli@cs.umd.edu>, Soham De <sohamde@cs.umd.edu>.

Presented at the ICML 2017 Workshop on Principled Approaches to Deep Learning, Sydney, Australia, 2017.

in inference performance (Gupta et al., 2015; Lin et al., 2016a; Hwang & Sung, 2014; Lin et al., 2016b; Li et al., 2016). The most extreme scenario of quantization is binarization, in which only 1-bit (two states) is available for weight representation (Kim & Smaragdis, 2015; Courbariaux et al., 2015; 2016; Rastegari et al., 2016; Hubara et al., 2016; Baldassi et al., 2015).

Previous work on obtaining a quantized neural network (NN) can be divided into two categories: quantizing pre-trained models with or without retraining (Hwang & Sung, 2014; Anwar et al., 2015; Lin et al., 2016a; Zhu et al., 2017; Zhou et al., 2017), and training a quantized model from scratch (Gupta et al., 2015; Courbariaux et al., 2015; Rastegari et al., 2016; Courbariaux et al., 2016; Zhou et al., 2016). We focus on approaches that belong to the second category, as they can be used for both training and inference under constrained resources.

For training quantized NNs from scratch, many authors suggest maintaining a high-precision copy of the weights while feeding quantized weights into backprop (Courbariaux et al., 2015; Hubara et al., 2016; Rastegari et al., 2016; Zhou et al., 2016), which results in good empirical performance. There are limitations in using such methods on low-power devices, however, where floating-point arithmetic is not available or not desirable. Another widely used solution using only low-precision weights is *stochastic rounding* (Höhfeld & Fahlman, 1992; Gupta et al., 2015). Experiments show that networks using 16-bit fixed-point representations with stochastic rounding can deliver results nearly identical to 32-bit floating-point computations (Gupta et al., 2015), while lowering the precision down to 3-bit fixed-point often results in a significant performance degradation (Miyashita et al., 2016). Bayesian learning has also been applied to train binary networks (Soudry et al., 2014; Cheng et al., 2015). A more comprehensive review can be found in (Rastegari et al., 2016).

3. Training Quantized Neural Nets

We consider problems of the form:

$$\min_{w\in\mathcal{W}} F(w) := \frac{1}{m}\sum_{i=1}^{m} f_i(w), \tag{1}$$

where the objective function decomposes into a sum over many functions $f_i : \mathbb{R}^d \to \mathbb{R}$. Neural networks have objective functions of this form where each f_i is a non-convex loss function. When floating-point representations are available, the standard method for training neural networks is stochastic gradient descent (SGD), which on each iteration selects a function $\tilde{f}$ randomly from $\{f_1, f_2, \ldots, f_m\}$, and then computes

$$\text{SGD:}\quad w^{t+1} = w^t - \alpha_t \nabla \tilde{f}(w^t), \tag{2}$$

for some learning rate α_t. In this paper, we consider the problem of training convolutional neural networks (CNNs). In CNNs, each convolutional layer i accepts inputs x_i and generates the response corresponding to filter j with linear transformation $a_{i,j} = w_{i,j} * x_i + b_j$, where $w_{i,j}$ represent the vector of weights of filter j at layer i, and $*$ denotes the convolution operation. The neuron output is generated by applying a non-linear activation function $h(\cdot)$ to the filter response, i.e., $x_{i+1} = h(a_i)$, which is used as the input to the next layer. Convolutions are expensive; low precision weights can be used to accelerate the convolutions by replacing expensive multiplications with binary operations or dedicated high-performance computer arithmetic.

To train neural networks using a low-precision representation of the weights, a quantization function $Q(\cdot)$ is required that converts a real-valued number w into a quantized version $\hat{w} = Q(w)$. We use the same notation for quantizing vectors, where we assume Q acts on each dimension of the vector. Different quantized optimization routines can be defined by selecting different quantizers, and also by selecting when quantization happens during optimization. The common options are:

Deterministic Rounding (R) A basic *uniform* or deterministic quantization function converts a floating point value to the closest quantized value as:

$$Q_d(w) = \text{sign}(w)\cdot\Delta\cdot\left\lfloor\frac{|w|}{\Delta}+\frac{1}{2}\right\rfloor, \tag{3}$$

where Δ denotes the quantization step or resolution, i.e., the smallest positive number that is representable. One exception to this definition is when we consider binary weights, where all weights are constrained to have two values $w \in \{-1, 1\}$ and uniform rounding becomes $Q_d(w) = \text{sign}(w)$.

The deterministic rounding SGD maintains quantized weights with updates of the form:

$$\text{Deterministic Rounding: } w_b^{t+1} = Q_d\big(w_b^t - \alpha_t \nabla \tilde{f}(w_b^t)\big), \tag{4}$$

where w_b denotes the low-precision weights, which are quantized using Q_d immediately after applying the gradient descent update. If gradient updates are significantly smaller than the quantization step, this method loses gradient information and weights may never be modified from their starting values.

Stochastic Rounding (SR) The quantization function for *stochastic rounding* is defined as:

$$Q_s(w) = \Delta\cdot\begin{cases}\lfloor\frac{w}{\Delta}\rfloor+1 & \text{for } p \le \frac{w}{\Delta} - \lfloor\frac{w}{\Delta}\rfloor,\\ \lfloor\frac{w}{\Delta}\rfloor & \text{otherwise,}\end{cases} \tag{5}$$

where $p \in [0,1]$ is produced by a uniform random number generator. This operator is non-deterministic, and rounds its argument up with probability $w/\Delta - \lfloor w/\Delta \rfloor$, and down otherwise. This quantizer satisfies the important property:

$$\mathbb{E}[Q_s(w)] = w.$$

Similar to the deterministic rounding method, the SR optimization method also maintains quantized weights with updates of the form:

$$\text{Stochastic Rounding:} \quad w_b^{t+1} = Q_s\big(w_b^t - \alpha_t \nabla \tilde{f}(w_b^t)\big). \quad (6)$$

BinaryConnect (BC) The BinaryConnect algorithm (Courbariaux et al., 2015) accumulates gradient updates using a full-precision buffer w_r, and quantizes weights only just before gradient computations. BinaryConnect uses updates of the form:

$$\text{BinaryConnect:} \quad w_r^{t+1} = w_r^t - \alpha_t \nabla \tilde{f}\big(Q(w_r^t)\big). \quad (7)$$

Either stochastic rounding Q_s or deterministic rounding Q_d can be used for quantizing the weights w_r, but in practice, Q_d is the common choice. The original BinaryConnect paper constrains the weights to be $\{-1, 1\}$, which can be generalized to $\{-\Delta, \Delta\}$. A more recent method, Binary-Weights-Net (BWN) (Rastegari et al., 2016), allows different filters to have different scales for quantization, which often results in better performance on large datasets.

Notation For the rest of the paper, we use Q to denote both Q_s and Q_d unless the situation requires this to be distinguished. We also drop the subscripts on w_r and w_b, and simply write w.

4. Analyzing the BinaryConnect (BC) Method

Here we present convergence guarantees for the BinaryConnect (BC) algorithm, with updates of the form (7), and we show that BinaryConnect has the property of concentrating its iterates on minimizers over time. To do this, we make the following assumptions that are standard for analyzing convergence rates of optimization algorithms over convex functions.

Assumption 1. *We assume that F is μ-strongly convex, i.e., it satisfies the following inequality:*

$$\langle \nabla F(w'), w - w' \rangle \leq F(w) - F(w') - \frac{\mu}{2}\|w - w'\|^2.$$

Assumption 2. *We assume that the magnitude of each ∇f_i is uniformly upper-bounded as:*

$$\|\nabla f_i(w)\|^2 \leq G^2, \quad \forall w \in \mathcal{W}, \forall i = 1, 2, \ldots, m.$$

Assumption 3. *We assume that the domain is bounded as:*

$$\|w - w^\star\| \leq D, \quad \forall w \in \mathcal{W}.$$

In this case, the rounding algorithm clips values that leave the domain. For example, in the binary case, the rounding algorithm never returns a value outside of $\{-1, 1\}$.

In addition to these standard assumptions, we also assume the following.

Assumption 4. *We assume that the Hessian satisfies*

$$\|\nabla^2 F_i(x) - \nabla^2 F_i(y)\| \leq L_2 \|x - y\|,$$

for some constant $L_2 > 0$. Equivalently, this translates to the condition:

$$\nabla F_i(y) = \nabla F_i(x) + \nabla^2 F_i(x)(y - x) + \delta,$$

where

$$\|\delta\| \leq \frac{L_2}{2}\|x - y\|^2.$$

While this is a slightly non-standard assumption, we will see below that it enables us to gain better insights into the behavior of the BinaryConnect algorithm. We now analyze the BinaryConnect algorithm.

Theorem 1. *Given Assumptions 1-4 and learning rates given by $\alpha_t = \frac{1}{\mu(t+1)}$, the BinaryConnect algorithm, with updates of the form (7) using stochastic rounding quantization function Q_s, converges as:*

$$\mathbb{E}[F(\bar{w}^T) - F(w^\star)] \leq \frac{(1 + \log(T+1))G^2}{2\mu T} + \frac{DL_2\sqrt{d}\Delta}{2},$$

where $\bar{w}^T$ denotes the iterate average: $\bar{w}^T = \frac{1}{T}\sum_{t=1}^{T} w_t$.

Proof. From the update rule (7), we get

$$\begin{aligned} w^{t+1} &= w^t - \alpha_t \nabla \tilde{f}\big(Q(w^t)\big) \\ &= w^t - \alpha_t \nabla \tilde{f}\big(w^t + r^t\big) \\ &= w^t - \alpha_t [\nabla \tilde{f}\big(w^t\big) + \nabla^2 \tilde{f}\big(w^t\big) r^t + \hat{r}^t], \end{aligned}$$

where $\|\hat{r}^t\| \leq \frac{L_2}{2}\|r^t\|^2$ from Assumption 4. Note that in general r^t has mean zero while $\hat{r}^t$ does not.

Subtracting by the optimal $w^\star$, taking norm, and taking expectation conditioned on w^t, we get:

$$\begin{aligned} &\mathbb{E}\|w^{t+1} - w^\star\|^2 \\ &= \|w^t - w^\star\|^2 - 2\alpha_t \mathbb{E}\langle w^t - w^\star, \nabla \tilde{f}(w^t + r^t)\rangle \\ &\quad + \alpha_t^2 \mathbb{E}\|\nabla \tilde{f}(w^t + r^t)\|^2 \\ &\leq \|w^t - w^\star\|^2 - 2\alpha_t \mathbb{E}\langle w^t - w^\star, \nabla F(w^t) + \hat{r}^t\rangle + \alpha_t^2 G^2 \\ &= \|w^t - w^\star\|^2 - 2\alpha_t \mathbb{E}\langle w^t - w^\star, \nabla F(w^t)\rangle + \alpha_t^2 G^2 \\ &\quad - 2\alpha_t \mathbb{E}\langle w^t - w^\star, \hat{r}^t\rangle, \end{aligned}$$

where the inequality uses Assumptions 2 and 4. Using Assumption 3 and observing that the quantization error for

BC-SGD can always be upper-bounded as $\|r^t\| \leq \sqrt{d}\Delta$, we get:

$$\begin{aligned}-2\alpha_t\mathbb{E}\langle w^t - w^\star, \hat{r}^t\rangle &\leq 2\alpha_t D\mathbb{E}\|\hat{r}^t\| \\ &\leq 2\alpha_t D\frac{L_2}{2}\|r^t\| \\ &\leq \alpha_t D L_2\sqrt{d}\Delta.\end{aligned}$$

Thus, using Assumption 1, we get:

$$\begin{aligned}\mathbb{E}\|w^{t+1} - w^\star\|^2 \leq (1-\alpha_t\mu)\|w^t - w^\star\|^2 + \alpha_t D L_2\sqrt{d}\Delta \\ - 2\alpha_t(F(w^t) - F(w^\star)) + \alpha_t^2 G^2.\end{aligned}$$

Re-arranging the terms, and taking expectation we get:

$$\begin{aligned}\mathbb{E}(F(w^t) - F(w^\star)) \leq \left(\frac{1}{2\alpha_t} - \frac{\mu}{2}\right)\mathbb{E}\|w^t - w^\star\|^2 + \frac{\alpha_t G^2}{2} \\ - \frac{1}{2\alpha_t}\mathbb{E}\|w^{t+1} - w^\star\|^2 + \frac{DL_2\sqrt{d}\Delta}{2}.\end{aligned}$$

Assume that the stepsize decreases with the rate $\alpha_t = 1/\mu(t+1)$. Then we have:

$$\begin{aligned}\mathbb{E}(F(w^t) - F(w^\star)) \leq \frac{G^2}{2\mu(t+1)} + \frac{DL_2\sqrt{d}\Delta}{2} \\ - \frac{\mu t}{2}\mathbb{E}\|w^t - w^\star\|^2 - \frac{\mu(t+1)}{2}\mathbb{E}\|w^{t+1} - w^\star\|^2.\end{aligned}$$

Averaging over $t = 0$ to T, we get a telescoping sum on the right hand side, which yields:

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}(F(w^t) - F(w^\star)) \\ &\leq \frac{G^2}{2\mu T}\sum_{t=0}^{T}\frac{1}{t+1} + \frac{DL_2\sqrt{d}\Delta}{2} - \frac{\mu(T+1)}{2}\mathbb{E}\|w^{T+1} - w^\star\|^2 \\ &\leq \frac{(1+\log(T+1))G^2}{2\mu T} + \frac{DL_2\sqrt{d}\Delta}{2}.\end{aligned}$$

Using Jensen's inequality, we have:

$$\mathbb{E}(F(\bar{w}^T) - F(w^\star)) \leq \frac{1}{T}\sum_{t=0}^{T}\mathbb{E}(F(w^t) - F(w^\star)),$$

where $\bar{w}^T = \frac{1}{T}\sum_{t=0}^{T} w^t$, the average of the iterates. The final convergence result follows. □

Thus, we see that BinaryConnect converges until it reaches an "accuracy floor", which is determined by the quantization error Δ and L_2. Now, consider a quadratic least-squares problem, $F(w) = \frac{1}{2}\|Aw - b\|^2$. Here, the gradient of F is linear: $\nabla F(w) = Aw - b$, and the Hessian is constant. Thus, $L_2 = 0$ and we get the following corollary.

Corollary 1. *Assume that F is quadratic of the form $F(w) = \frac{1}{2}\|Aw - b\|^2$ and the learning rates are given by $\alpha_t = \frac{1}{\mu(t+1)}$. Given Assumptions 1-4, the BC algorithm with updates of the form (7) and using the stochastic rounding quantization function Q_s, yields*

$$\mathbb{E}[F(\bar{w}^T) - F(w^\star)] \leq \frac{(1+\log(T+1))G^2}{2\mu T}.$$

We see that accumulating the real-valued weights in BC allows it to converge to the *true minimizer* of quadratic losses. Furthermore, this suggests that, when the function behaves like a quadratic on the distance scale Δ, one would expect BC to effectively concentrate on minimizers.

5. Analyzing the Stochastic Rounding Method

We can rewrite the update rule (6) of the Stochastic Rounding (SR) algorithm as:

$$w^{t+1} = w^t - \alpha_t\nabla\tilde{f}(w^t) + r^t, \tag{8}$$

where $r^t = Q_s(w^t - \alpha_t\nabla\tilde{f}(w^t)) - w^t + \alpha_t\nabla\tilde{f}(w^t)$ denotes the quantization error on the t-th iteration.

5.1. Bound on Quantization Error

We want to bound the error r^t in expectation. To this end, we present the following lemma.

Lemma 1. *The error r^t on each iteration t of the Stochastic Rounding algorithm with updates given by (8) can be bounded, in expectation, as:*

$$\mathbb{E}\left\|r^t\right\|^2 \leq \sqrt{d}\Delta\alpha_t G.$$

Proof. We want to bound the quantization error r^t. Consider the i-th entry in r^t denoted by r_i^t. Similarly, we define w_i^t and $\nabla\tilde{f}(w^t)_i$. Choose some random number $p \in [0,1]$. The stochastic rounding operation produces a value of r^t given by

$$\begin{aligned}r_i^t &= Q_s(w_i^t - \alpha_t\nabla\tilde{f}(w^t)_i) - w_i^t + \alpha_t\nabla\tilde{f}(w^t)_i \\ &= \Delta\cdot\begin{cases}\dfrac{\alpha_t\nabla\tilde{f}(w^t)_i}{\Delta} + \left\lfloor\dfrac{-\alpha_t\nabla\tilde{f}(w^t)_i}{\Delta}\right\rfloor + 1, & \text{for } p \leq q, \\ \dfrac{\alpha_t\nabla\tilde{f}(w^t)_i}{\Delta} + \left\lfloor\dfrac{-\alpha_t\nabla\tilde{f}(w^t)_i}{\Delta}\right\rfloor, & \text{otherwise.}\end{cases} \\ &= \Delta\cdot\begin{cases}-q+1, & \text{for } p \leq q, \\ -q, & \text{otherwise.}\end{cases}\end{aligned}$$

where we write $q = -\dfrac{\alpha_t\nabla\tilde{f}(w^t)_i}{\Delta} - \left\lfloor\dfrac{-\alpha_t\nabla\tilde{f}(w^t)_i}{\Delta}\right\rfloor$ and $q \in [0,1]$.

Now we have

$$\begin{aligned}\mathbb{E}_p\left[(r_i^t)^2\right] &\leq \Delta^2((-q+1)^2 q + (-q)^2(1-q)) \\ &= \Delta^2 q(1-q) \\ &\leq \Delta^2 \min\{q, 1-q\}.\end{aligned}$$

Because $\min\{q, 1-q\} \leq \left|\frac{\alpha_t \nabla \tilde{f}(w^t)_i}{\Delta}\right|$, it follows that $\mathbb{E}_p\left[(r_i^t)^2\right] \leq \Delta^2 \left|\frac{\alpha_t \nabla \tilde{f}(w^t)_i}{\Delta}\right| \leq \Delta \left|\alpha_t \nabla \tilde{f}(w^t)_i\right|$.

Summing over the index i yields

$$\begin{aligned}\mathbb{E}_p\left\|r^t\right\|_2^2 &\leq \Delta\alpha_t \left\|\nabla \tilde{f}(w^t)\right\|_1 \\ &\leq \sqrt{d}\alpha_t \Delta \left\|\nabla \tilde{f}(w^t)\right\|_2. \end{aligned} \tag{9}$$

Now, $\left(\mathbb{E}\left\|\nabla \tilde{f}(w^t)\right\|_2\right)^2 \leq \mathbb{E}\left\|\nabla \tilde{f}(w^t)\right\|_2^2 \leq G^2$. Plugging this into (9) yields

$$\mathbb{E}\left\|r^t\right\|_2^2 \leq \sqrt{d}\Delta\alpha_t G. \tag{10}$$

□

From Lemma 1, we see that the rounding error per step decreases as the learning rate α_t decreases. This is intuitive since the probability of an entry in w^{t+1} differing from w^t is small when the gradient update is small relative to Δ.

To explain the large observed differences in asymptotic behavior between the BinaryConnect and Stochastic Rounding algorithms, we now study the asymptotic, long-term behavior of the SR algorithm as the learning rate gets small. Note that Section 5 makes no convexity assumptions, and the proposed theoretical results are directly applicable to neural networks.

5.2. Asymptotic Behavior of Stochastic Rounding

Typical (continuous-valued) SGD methods have an important exploration-exploitation tradeoff. When the learning rate is large, the algorithm explores by moving quickly between states. Exploitation happens when the learning rate is small. In this case, noise averaging causes the algorithm to behave more like deterministic gradient descent, where the algorithm more greedily pursues local minimizers with lower loss function values. Thus, the distribution of iterates produced by the algorithm becomes increasingly concentrated near minimizers as the learning rate vanishes (see, e.g., the large-deviations estimates in (Lan et al., 2012)). BC maintains this property as well—indeed, we saw in Corollary 1 a class of problems for which the iterates concentrate near the minimizer for small α_t.

Here we show that the SR method lacks this important tradeoff: as the stepsize gets small and the algorithm slows down, the quality of the iterates produced by the algorithm does *not* improve, and the algorithm does *not* become progressively more likely to produce low-loss iterates.

To understand this problem conceptually, consider the simplified case of a single-variable optimization problem starting at $x^0 = 0$ with $\Delta = 1$, as depicted in Figure 1. On each iteration, the algorithm computes a stochastic approximation $\nabla \tilde{f}$ of the gradient by sampling from a distribution, which we call p. This gradient is then multiplied by the stepsize to get $\alpha \nabla \tilde{f}$. The probability of moving to the right (or left) is then roughly proportional to the magnitude of $\alpha \nabla \tilde{f}$. Note that random variable $\alpha \nabla \tilde{f}$ has distribution $p_\alpha(z) = \alpha^{-1} p(z/\alpha)$.

Now, suppose that α is small enough that we can neglect the tails of $p_\alpha(z)$ that lie outside the interval $[-1, 1]$. The probability of transitioning from $x^0 = 0$ to $x^1 = 1$ using stochastic rounding, denoted by $T_\alpha(0, 1)$, is then

$$\begin{aligned}T_\alpha(0,1) &\approx \int_0^1 z p_\alpha(z) dz = \frac{1}{\alpha}\int_0^1 z p(z/\alpha)\, dz \\ &= \alpha \int_0^{1/\alpha} p(x) x\, dx \approx \alpha \int_0^\infty p(x) x\, dx,\end{aligned}$$

where the first approximation is because we neglected the unlikely case that $\alpha \nabla \tilde{f} > 1$, and the second approximation appears because we added a small tail probability to the estimate. These approximations get more accurate for small α. We see that, assuming the tails of p are "light" enough, we have $T_\alpha(0, 1) \sim \alpha \int_0^\infty p(x) x\, dx$ as $\alpha \to 0$. Similarly, $T_\alpha(0, -1) \sim \alpha \int_{-\infty}^0 p(x) x\, dx$ as $\alpha \to 0$.

What does this observation mean for the behavior of SR? First of all, the probability of leaving x^0 on an iteration is

$$\begin{aligned}&T_\alpha(0,-1) + T_\alpha(0,1) \\ &\qquad \approx \alpha \left[\int_0^\infty p(x) x\, dx + \int_{-\infty}^0 p(x) x\, dx\right], \end{aligned} \tag{11}$$

which vanishes for small α (as also indicated by Lemma 1). This means the algorithm slows down as the learning rate drops off, which is not surprising. However, the *conditional* probability of ending up at $x^1 = 1$ *given* that the algorithm did leave x^0 is

$$\begin{aligned}T_\alpha(0, 1 | x^1 \neq x^0) &\approx \frac{T_\alpha(0,1)}{T_\alpha(0,-1) + T_\alpha(0,1)} \\ &= \frac{\int_0^\infty p(x) x\, dx}{\int_{-\infty}^0 p(x) x\, dx + \int_0^\infty p(x) x\, dx},\end{aligned}$$

which does not depend on α. In other words, provided α is small, SR, on average, makes the same decisions/transitions with learning rate α as it does with learning rate $\alpha/10$; it just takes 10 times longer to make those

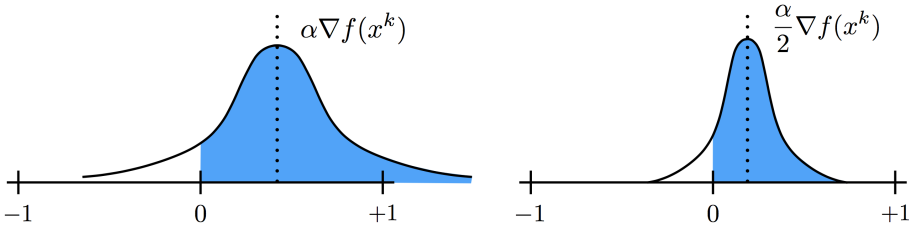

Figure 1. The SR method starts at some location x (in this case 0), adds a perturbation to x, and then rounds. As the learning rate α gets smaller, the distribution of the perturbation gets "squished" near the origin, making the algorithm less likely to move. The "squishing" effect is the same for the part of the distribution lying to the left and to the right of x, and so it does not effect the *relative* probability of moving left or right.

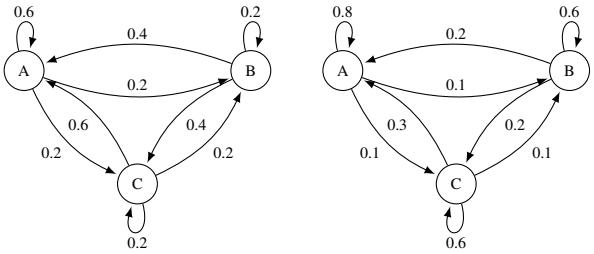

Figure 2. Markov chain example with 3 states. In the right figure, we halved each transition probability for moving between states, with the remaining probability put on the self-loop. Notice that halving all the transition probabilities would not change the equilibrium distribution, and instead would only increase the mixing time of the Markov chain.

decisions when $\alpha/10$ is used. In this situation, there is no exploitation benefit in decreasing α.

The above argument is intuitive, but also informal. To make these statements rigorous, we interpret the SR method as a Markov chain. On each iteration, SR starts at some state (iterate) x, and moves to a new state y with some transition probability $T_\alpha(x, y)$ that depends only on x and the learning rate α. For fixed α, this is clearly a Markov process with transition matrix[1] $T_\alpha(x, y)$.

The long-term behavior of this Markov process is determined by the *stationary distribution* of $T_\alpha(x, y)$. We show below that for all small α, the stationary distribution of $T_\alpha(x, y)$ is nearly invariant to α, and thus decreasing α below some threshold has virtually no effect on the long term behavior of the method. This happens because, as α shrinks, the relative probabilities of leaving a state remain the same, even though the absolute probabilities decrease (see Figure 2). In this case, there is no exploitation benefit to decreasing α.

Theorem 2. *Let $p_{x,k}$ denote the probability distribution for the kth entry in $\nabla\tilde{f}(x)$, the stochastic gradient estimate at x. Assume there is a constant C_1 such that for all x, k, and ν we have $\int_\nu^\infty p_{x,k}(z)\,dz \le \frac{C_1}{\nu^2}$, and some C_2 such that both $\int_0^{C_2} p_{x,k}(z)\,dz > 0$ and $\int_{-C_2}^0 p_{x,k}(z)\,dz > 0$. Define the matrix*

$$\tilde{U}(x,y) = \begin{cases} \int_0^\infty p_{x,k}(z)\frac{z}{\Delta}\,dz, & \text{if } x_{-k} = y_{-k},\ y_k = x_k + \Delta, \\ \int_{-\infty}^0 p_{x,k}(z)\frac{z}{\Delta}\,dz, & \text{if } x_{-k} = y_{-k},\ y_k = x_k - \Delta, \\ 0, & \text{otherwise,} \end{cases}$$

where $x_{-k} = y_{-k}$ denotes that x and y are equal except at coordinate k. Define the associated markov chain transition matrix

$$\tilde{T}_{\alpha_0} = I - \alpha_0 \cdot \mathrm{diag}(\mathbf{1}^T\tilde{U}) + \alpha_0\tilde{U}, \tag{12}$$

where α_0 is the largest constant that makes $\tilde{T}_{\alpha_0}$ nonnegative. Suppose $\tilde{T}_\alpha$ has a stationary distribution, denoted $\tilde{\pi}$. Then, for sufficiently small α, T_α has a stationary distribution π_α, and

$$\lim_{\alpha\to 0}\pi_\alpha = \tilde{\pi}.$$

Furthermore, this limiting distribution satisfies $\tilde{\pi}(x) > 0$ for any state x, and is thus not concentrated on local minimizers of f.

Proof. Let the matrix U_α be a partial transition matrix defined by $U_\alpha(x, x) = 0$, and $U_\alpha(x, y) = T_\alpha(x, y)$ for $x \neq y$. From U_α, we can get back the full transition matrix T_α using the formula

$$T_\alpha = I - \mathrm{diag}(\mathbf{1}^T U_\alpha) + U_\alpha.$$

Note that this formula is essentially "filling in" the diagonal entries of T_α so that every column sums to 1, thus making T_α a valid stochastic matrix.

Let's bound the entries in U_α. Suppose that we begin an iteration of the stochastic rounding algorithm at some point

[1] Our analysis below does not require the state space to be finite, so $T_\alpha(x, y)$ may be a linear operator rather than a matrix. Nonetheless, we use the term "matrix" as it is standard.

x. Consider an adjacent point y that differs from x at only 1 coordinate, k, with $y_k = x_k + \Delta$. Then we have

$$\begin{aligned}
&U_\alpha(x,y)\\
&= \frac{1}{\alpha}\int_0^{\Delta} p_{x,k}(x/\alpha)\frac{x}{\Delta}\,dx + \frac{1}{\alpha}\int_{\Delta}^{2\Delta} p_{x,k}(x/\alpha)\frac{2\Delta - x}{\Delta}\,dx\\
&= \tfrac{1}{\alpha}\textstyle\int_0^{\Delta/\alpha} p_{x,k}(z)\frac{\alpha z}{\Delta}\alpha\,dz + \frac{1}{\alpha}\int_{\Delta/\alpha}^{2\Delta/\alpha} p_{x,k}(z)\frac{2\Delta-\alpha z}{\Delta}\alpha\,dz\\
&\le \alpha\int_0^{\Delta/\alpha} p_{x,k}(z)\frac{z}{\Delta}\,dz + \int_{\Delta/\alpha}^{\infty} p_{x,k}(z)\,dz\\
&= \alpha\int_0^{\infty} p_{x,k}(z)\frac{z}{\Delta}\,dz + O(\alpha^2). \qquad (13)
\end{aligned}$$

Note we have used the decay assumption:

$$\int_{\nu}^{\infty} p_{x,k}(z) \le \frac{C}{\nu^2}.$$

Likewise, if $y_k = x_k - \Delta$, then the transition probability is

$$U_\alpha(x,y) = \alpha\int_{-\infty}^{0} p_{x,k}(z)\frac{z}{\Delta}\,dz + O(\alpha^2), \qquad (14)$$

and if $y_k = x_k \pm m\Delta$ for an integer $m > 1$,

$$U_\alpha(x,y) = O(\alpha^2). \qquad (15)$$

We can approximate the behavior of U_α using the matrix $\tilde{U}(x,y)$, and we define the associated markov chain transition matrix $\tilde{T}_\alpha$ as in (12).

Let α_0 be the largest scalar such that the stochastic linear operator $\tilde{T}_{\alpha_0}$ has non-negative entries. For $\alpha < \alpha_0$, $\tilde{T}_\alpha$ has non-negative entries and column sums equal to 1; it thus defines the transition operator of a markov chain. Let $\tilde{\pi}$ denote the stationary distribution of the markov chain with transition matrix $\tilde{T}_{\alpha_0}$.

We now claim that $\tilde{\pi}$ is also the stationary distribution of $\tilde{T}_\alpha$ for all $\alpha < \alpha_0$. We verify this by noting that

$$\begin{aligned}
\tilde{T}_\alpha &= (I - \alpha\cdot\mathrm{diag}(\mathbf{1}^T\tilde{U}))\tilde{\pi} + \alpha\tilde{U}\tilde{\pi}\\
&= (1-\frac{\alpha}{\alpha_0})I + \frac{\alpha}{\alpha_0}[I - \alpha_0\cdot\mathrm{diag}(\mathbf{1}^T\tilde{U}) + \alpha_0\tilde{U}]\\
&= (1-\frac{\alpha}{\alpha_0})I + \frac{\alpha}{\alpha_0}\tilde{T}_{\alpha_0} \qquad (16)
\end{aligned}$$

and so $\tilde{T}_\alpha\tilde{\pi} = (1-\frac{\alpha}{\alpha_0})\tilde{\pi} + \frac{\alpha}{\alpha_0}\tilde{\pi} = \tilde{\pi}$.

Recall that T_α is the transition matrix for the Markov chain generated by the stochastic rounding algorithm with learning rate α. We wish to show that this markov chain is well approximated by $\tilde{T}_\alpha$. Note that

$$T_\alpha(x,y) = \prod_{k, x_k \neq y_k} T_\alpha(x, x + (y_k - x_k)\Delta e_k) \le O(\alpha^2)$$

when x, y differ at more than 1 coordinate. In other words, transitions between multiple coordinates simultaneously become vanishingly unlikely for small α. When x and y differ by exactly 1 coordinate, we know from (13) that

$$T_\alpha(x,y) = \alpha U(x,y) + O(\alpha^2).$$

These observations show that the off-diagonal elements of T_α are well approximated (up to uniform $O(\alpha^2)$ error) by the corresponding elements in αU. Since the columns of T_α sum to one, the diagonal elements are well approximated as well, and we have

$$T_\alpha = (I - \alpha\cdot\mathrm{diag}(\mathbf{1}^T U)) + \alpha U + O(\alpha^2) = \tilde{T}_\alpha + O(\alpha^2).$$

To be precise, the notation above means that

$$|T_\alpha(x,y) - \tilde{T}_\alpha(x,y)| < C\alpha^2, \qquad (17)$$

for some C that is uniform over (x,y).

We are now ready to show that the stationary distribution of T_α exists and approaches $\tilde{\pi}$. Re-arranging (16) gives us

$$\alpha_0\tilde{T}_\alpha + (\alpha - \alpha_0)I = \alpha\tilde{T}_{\alpha_0}.$$

Combining this with (17), we get

$$\|\alpha_0 T_\alpha + (\alpha-\alpha_0)I - \alpha\tilde{T}_{\alpha_0}\|_\infty < O(\alpha^2),$$

and so

$$\|\frac{\alpha_0}{\alpha}T_\alpha + (1-\frac{\alpha_0}{\alpha})I - \tilde{T}_{\alpha_0}\|_\infty < O(\alpha). \qquad (18)$$

From (18), we see that the matrix $\frac{\alpha_0}{\alpha}T_\alpha + (1-\frac{\alpha_0}{\alpha})I$ approaches $\tilde{T}_{\alpha_0}$. Note that $\tilde{\pi}$ is the Perron-Frobenius eigenvalue of $\tilde{T}_{\alpha_0}$, and thus has multiplicity 1. Multiplicity 1 eigenvalues/vectors of a matrix vary continuously with small perturbations to that matrix (Theorem 8, p130 of (Lax, 2007)). It follows that, for small α, $\frac{\alpha_0}{\alpha}T_\alpha + (1-\frac{\alpha_0}{\alpha})I$ has a stationary distribution, and this distribution approaches $\tilde{\pi}$. The leading eigenvector of $\frac{\alpha_0}{\alpha}T_\alpha + (1-\frac{\alpha_0}{\alpha})I$ is the same as the leading eigenvector of T_α, and it follows that T_α has a stationary distribution that approaches $\tilde{\pi}$.

Finally, note that we have assumed $\int_0^{C_2} p_{x,k}(z)\,dz > 0$ and $\int_{-C_2}^0 p_{x,k}(z)\,dz > 0$. Under this assumption, for $\alpha < \frac{1}{C_2}$, $\tilde{T}_{\alpha_0}(x,y) > 0$ whenever x, y are neighbors the differ at a single coordinate. It follows that every state in the Markov chain $\tilde{T}_{\alpha_0}$ is accessible from every other state by traversing a path of non-zero transition probabilities, and so $\tilde{\pi}(x) > 0$ for every state x. □

While the long term stationary behavior of SR is relatively insensitive to α, the convergence speed of the algorithm is not. To measure this, we consider the *mixing time* of the Markov chain. Let π_α denote the stationary distribution of a Markov chain. We say that the ϵ-mixing time of the chain is M_ϵ if M_ϵ is the smallest integer such that (Levin et al., 2009)

$$|\mathbb{P}(x^{M_\epsilon} \in A | x^0) - \pi(A)| \le \epsilon, \quad \forall x^0, \forall \text{ subsets } A \subseteq X. \qquad (19)$$

We show below that the mixing time of the Markov chain gets large for small α, which means exploration slows down, even though no exploitation gain is being realized.

Theorem 3. *Let $p_{x,k}$ satisfy the assumptions of Theorem 2. Choose some ϵ sufficiently small that there exists a* proper *subset of states $A \subset X$ with stationary probability $\pi_\alpha(A)$ greater than ϵ. Let $M_\epsilon(\alpha)$ denote the ϵ-mixing time of the chain with learning rate α. Then,*

$$\lim_{\alpha \to 0} M_\epsilon(\alpha) = \infty.$$

Proof. Given some distribution π over the states of the markov chain, and some set A of states, let $[\pi]_A = \sum_{a \in A} \pi(a)$ denote the measure of A with respect to π.

Suppose for contradiction that the mixing time of the chain remains bounded as α vanishes. Then we can find an integer M_ϵ that upper bounds the ϵ-mixing time for all α. By the assumption of the theorem, we can select some set of states A with $[\tilde{\pi}]_A > \epsilon$, and some starting state $y \notin A$. Let e be a distribution (a vector in the finite-state case) with $e_y = 1$, $e_k = 0$ for $k \neq y$. Note that $[e]_A = 0$ because $y \notin A$. Then

$$|[e]_A - [\tilde{\pi}]_A| > \epsilon.$$

Note that, as $\alpha \to 0$, we have $\|T_\alpha - \tilde{T}_\alpha\| \to 0$ and thus $\|T_\alpha^{M_\epsilon} - \tilde{T}_\alpha^{M_\epsilon}\| \to 0$. We also see from the definition of $\tilde{T}_\alpha$ in (12), $\lim_{\alpha \to 0} \tilde{T}_\alpha = I$. It follows that

$$\lim_{\alpha \to 0} |[T_\alpha^{M_\epsilon} e]_A - [\tilde{\pi}]_A| = |[e]_A - [\tilde{\pi}]_A| > \epsilon,$$

and so for some α the inequality (19) is violated. This is a contradiction because it was assumed M_ϵ is an upper bound on the mixing time. □

6. Experiments

To explore the implications of the theory above, we train two types of networks, VGG-like CNNs (Simonyan & Zisserman, 2015) and Residual networks (He et al., 2016), on CIFAR-10 and CIFAR-100. VGG-BC is a high-capacity network used for the original BC model (Courbariaux et al., 2015). We use the same architecture as in (Courbariaux et al., 2015) except using softmax and cross-entropy loss instead of SVM and squared hinge loss, respectively. ResNets-56 has 55 convolutional layers and one linear layer, and contains three stages of residual blocks where each stage has the same number of residual blocks. We also create a wide ResNet-56 (WRN-56-2) that doubles the number of filters in each residual block as in (Zagoruyko & Komodakis, 2016). We use ADAM (Kingma & Ba, 2015) as the baseline optimizer, and train with the three quantized algorithms mentioned in Section 3, i.e., R-ADAM, SR-ADAM and BC-ADAM. All methods start with the same binary weight initialization of random ± 1s. The image pre-processing and data augmentation procedures are the same as (He et al., 2016). Similar to (Rastegari et al., 2016), here we only quantize the weights in the convolutional layers, but not linear layers, during training. We set the initial learning rate to 0.01 and decrease the learning rate by a factor of 10 at epochs 82 and 122 for CIFAR-10 and CIFAR-100 (He et al., 2016). The default minibatch size is 128. Following (Courbariaux et al., 2015), we do not use weight decay during training. We implement all quantization algorithms in Torch7 (Collobert et al., 2011) and train the quantized models with NVIDIA GPUs.

Table 1. Test error after training with binarized initial weights.

	CIFAR-10			CIFAR-100
	VGG-BC	ResNet-56	WRN-56-2	ResNet-56
ADAM	7.12	8.10	6.62	33.98
R-ADAM	21.88	33.56	27.90	68.39
BC-ADAM	8.21	8.83	7.17	35.34
SR-ADAM	20.56	26.49	21.58	58.06

Results The overall results are summarized in Table 1. The binary model trained by BC-ADAM has comparable performance to the full-precision model trained by ADAM. SR-ADAM outperforms R-ADAM, which verifies the effectiveness of Stochastic Rounding. There is a performance gap between SR-ADAM and BC-ADAM across all models and datasets. This is consistent with our theoretical results in Sections 4 and 5, and keeping track of the real-valued weights to quantize the binary weights in BC-ADAM seems to really help empirically.

7. Conclusion

The training of quantized neural nets is essential for deploying machine learning models on embedded and ubiquitous devices. In this work, we provided a theoretical analysis to better understand the BinaryConnect (BC) and Stochastic Rounding (SR) methods for training binary nets. We showed that BC algorithms have the desirable property of being able to concentrate on minimizers. For general non-convex problems, however, we proved that SR differs from conventional stochastic methods in that it is unable to exploit greedy local search. Experiments confirm these findings, and show that the mathematical properties of SR are indeed observable (and very important) in practice.

Acknowledgments T. Goldstein was supported in part by the US National Science Foundation (NSF) under grant CCF-1535902, by the US Office of Naval Research under grant N00014-17-1-2078, and by the Sloan Foundation. C. Studer was supported in part by Xilinx, Inc. and by the US NSF under grants ECCS-1408006, CCF-1535897, and CAREER CCF-1652065. H. Samet was supported in part by the US NSF under grant IIS-13-20791.

References

Anwar, Sajid, Hwang, Kyuyeon, and Sung, Wonyong. Fixed point optimization of deep convolutional neural networks for object recognition. In *ICASSP*. IEEE, 2015.

Baldassi, Carlo, Ingrosso, Alessandro, Lucibello, Carlo, Saglietti, Luca, and Zecchina, Riccardo. Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses. *Physical review letters*, 115(12):128101, 2015.

Cheng, Zhiyong, Soudry, Daniel, Mao, Zexi, and Lan, Zhenzhong. Training binary multilayer neural networks for image classification using expectation backpropagation. *arXiv preprint arXiv:1503.03562*, 2015.

Collobert, Ronan, Kavukcuoglu, Koray, and Farabet, Clément. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

Courbariaux, Matthieu, Bengio, Yoshua, and David, Jean-Pierre. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 2015.

Courbariaux, Matthieu, Hubara, Itay, Soudry, Daniel, El-Yaniv, Ran, and Bengio, Yoshua. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

Gupta, Suyog, Agrawal, Ankur, Gopalakrishnan, Kailash, and Narayanan, Pritish. Deep learning with limited numerical precision. In *ICML*, 2015.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Höhfeld, Markus and Fahlman, Scott E. Probabilistic rounding in neural network learning with limited precision. *Neurocomputing*, 4(6):291–299, 1992.

Hubara, Itay, Courbariaux, Matthieu, Soudry, Daniel, El-Yaniv, Ran, and Bengio, Yoshua. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016.

Hwang, Kyuyeon and Sung, Wonyong. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In *IEEE Workshop on Signal Processing Systems (SiPS)*, 2014.

Kim, Minje and Smaragdis, Paris. Bitwise neural networks. In *ICML Workshop on Resource-Efficient Machine Learning*, 2015.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *ICLR*, 2015.

Lan, Guanghui, Nemirovski, Arkadi, and Shapiro, Alexander. Validation analysis of mirror descent stochastic approximation method. *Mathematical programming*, 134(2):425–458, 2012.

Lax, P.D. *Linear Algebra and Its Applications*. Number v. 10 in Linear algebra and its applications. Wiley, 2007. ISBN 9780471751564. URL https://books.google.com/books?id=e7FJM6aqZD8C.

Levin, David Asher, Peres, Yuval, and Wilmer, Elizabeth Lee. *Markov chains and mixing times*. American Mathematical Soc., 2009.

Li, Fengfu, Zhang, Bo, and Liu, Bin. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Lin, Darryl, Talathi, Sachin, and Annapureddy, Sreekanth. Fixed point quantization of deep convolutional networks. In *ICML*, 2016a.

Lin, Zhouhan, Courbariaux, Matthieu, Memisevic, Roland, and Bengio, Yoshua. Neural networks with few multiplications. *ICLR*, 2016b.

Marchesi, Michele, Orlandi, Gianni, Piazza, Francesco, and Uncini, Aurelio. Fast neural networks without multipliers. *IEEE Transactions on Neural Networks*, 4(1):53–62, 1993.

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

Rastegari, Mohammad, Ordonez, Vicente, Redmon, Joseph, and Farhadi, Ali. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *ECCV*, 2016.

Simonyan, Karen and Zisserman, Andrew. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

Soudry, Daniel, Hubara, Itay, and Meir, Ron. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NIPS*, 2014.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhou, Aojun, Yao, Anbang, Guo, Yiwen, Xu, Lin, and Chen, Yurong. Incremental network quantization: Towards lossless cnns with low-precision weights. *ICLR*, 2017.

Zhou, Shuchang, Wu, Yuxin, Ni, Zekun, Zhou, Xinyu, Wen, He, and Zou, Yuheng. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Zhu, Chenzhuo, Han, Song, Mao, Huizi, and Dally, William J. Trained ternary quantization. *ICLR*, 2017.