Main Question:
① 为什么对 weights 的界进行 clamp 之后会减少量化误差?
② 为什么增加 information entropy of weights 就能 提升 performance?
③ 进一步地. 增加 entropy 能提升的是什么?是直接的 performance. 还是上界?
combine ④ & ①: 减少量化误差. 多大程度上能保证 性能的提升？更关键的. 保证 Binary 与FP 网络的相像 究竟是不是合理的方向?

ReCU: Reviving the Dead Weights in Binary Neural Networks

Zihan Xu[1], Mingbao Lin[1], Jianzhuang Liu[2], Jie Chen[3]
Ling Shao[4], Yue Gao[5], Yonghong Tian[6], Rongrong Ji[1,3,7*]
[1]Media Analytics and Computing Laboratory, Department of Artificial Intelligence,
School of Informatics, Xiamen University, Xiamen, China
[2]Noah's Ark Lab, Huawei Technologies   [3]Peng Cheng Laboratory, Shenzhen, China
[4]Inception Institute of Artificial Intelligence, Abu Dhabi, UAE
[5]Tsinghua University, Beijing, China   [6]Peking University, Beijing, China
[7]Institute of Artificial Intelligence, Xiamen University, Xiamen, China

arXiv:2103.12369v1 [cs.LG] 23 Mar 2021

Abstract

Binary neural networks (BNNs) have received increasing attention due to their superior reductions of computation and memory. Most existing works focus on either lessening the quantization error by minimizing the gap between the full-precision weights and their binarization or designing a gradient approximation to mitigate the gradient mismatch, while leaving the "dead weights" untouched. This leads to slow convergence when training BNNs. In this paper, for the first time, we explore the influence of "dead weights" which refer to a group of weights that are barely updated during the training of BNNs, and then introduce rectified clamp unit (ReCU) to revive the "dead weights" for updating. We prove that reviving the "dead weights" by ReCU can result in a smaller quantization error. Besides, we also take into account the information entropy of the weights, and then mathematically analyze why the weight standardization can benefit BNNs. We demonstrate the inherent contradiction between minimizing the quantization error and maximizing the information entropy, and then propose an adaptive exponential scheduler to identify the range of the "dead weights". By considering the "dead weights", our method offers not only faster BNN training, but also state-of-the-art performance on CIFAR-10 and ImageNet, compared with recent methods. Code can be available at https://github.com/z-hXu/ReCU.

1. Introduction

Deep Neural Networks (DNNs) have shown tremendous success and advanced many visual tasks [27, 42, 14, 45]. Nevertheless, this comes at the price of massive memory
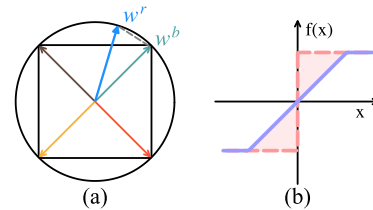
---
*Corresponding author

Figure 1: Illustration of the quantization error (a) and the gradient mismatch (b).

usage and computational burden, which poses a great challenge to the resource-constrained cutting-edge devices such as mobile phones and embedded devices. The community has proposed various approaches to solve this problem. Typical techniques include, but are not limited to, efficient architecture design [18, 23, 38], knowledge distillation [22, 25, 43], network pruning [13, 30, 34], and network quantization [52, 48, 51].

Among them, by converting the full-precision parameters and activations into low-bit forms, network quantization has offered a promising solution to yield a light and efficient version of DNNs [24, 15, 49, 47]. In the extreme case of a 1-bit representation, a binary neural network (BNN) restricts the weights and activations to only two possible values, *i.e.*, -1 and +1. In comparison with the original networks, BNNs show overwhelming superiority in reducing the model complexity by around 32× parameter compression, and 58× speedup, using the efficient XNOR and bitcount operations [6].

Despite the superiority of BNNs in memory saving and computation reduction, they suffer a drastic drop in accuracy compared with their real-valued counterparts [41, 11, 10], which greatly limits the practical deployment. There are two main reasons for the performance degradation: large quantization error in the forward propagation and gradient mismatch during backpropagation.
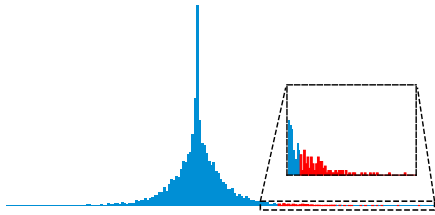
这两者关系 是独立的吗？

1

Figure 2: Illustration of the "dead weights". The red outliers in both tails of the distribution are barely updated during training BNNs and affect the convergence of the training (Layer2.0.conv2 of ResNet-20). (Best viewed in color)

Specifically, quantization error refers to the residual between the full-precision weight vector and its binarization [41, 31], as illustrated in Fig. 1(a). The representational ability of BNNs is indeed limited upon the vertices of a unit square. In contrast, the full-precision weights possess an almost unlimited representation space. Such a representation gap easily results in a large accumulated error when mapping the real-valued weights into the binary space. To solve this, existing approaches try to lessen the quantization error by introducing a scaling factor to reduce the norm difference [6, 7], or devising a rotation matrix to align the angle bias [31]. Gradient mismatch comes from the disagreement between the presumed and actual gradient functions [29] as illustrated in the pink area of Fig. 1(b). Since the quantization function in the forward propagation of BNNs has zero gradient almost everywhere, an approximate gradient function is required to enable the network to update. A typical example is the straight through estimator (STE) [3], which however leads to inaccurate optimization directions and thus hurts the stability of network training, especially in the low bitwidth [1, 5]. To mitigate this, a large collection of works have been proposed, typically by adjusting the network structures [36, 35, 9], or using gradient functions that gradually approach zero [15, 47, 31].

In this paper, we present a novel perspective to improve the effectiveness and training efficiency of BNNs. Inspired by [21], the latent weights, which refer to the real-valued weights used during backpropagation, play an important role in binarizing DNNs. We explore the real-valued weights of a given DNN and find that the weights falling into the two tails of the distribution, as shown in Fig. 2, are barely updated during the training of BNNs. We call them "dead weights" and find that they harm the optimization and slow down the training convergence of BNNs. To solve this problem, we develop a rectified clamp unit (ReCU), which aims to revive the "dead weights" by moving them towards the distribution peak in order to increase the probability of updating these weights. Through a rigorous analysis, we demonstrate that the quantization error after applying ReCU is a convex function, and thus can be further reduced. In-

stead of simply minimizing the quantization error, we consider the information entropy of the weights to increase the weight diversity of BNNs. For the first time, a systematical analysis is derived to explain why the weight standardization [39] can boost the performance of BNNs, and then a generalized weight standardization is proposed to further increase the information entropy. Combining the information entropy and the quantization error, we reveal the inherent contradiction between maximizing the former and minimizing the latter, and then propose an adaptive exponential scheduler to identify the range of the "dead weights" and balance the information entropy of the weights and the quantization error.

We conduct extensive experiments for binarizing networks including ResNet-18/20 [19] and VGG-small [49] on CIFAR-10 [26], and ResNet-18/34 [19] on ImageNet [44]. The experimental results show that ReCU achieves state-of-the-art performance, as well as faster training convergence even with the simple STE [3] as our weight gradient approximation.

To sum up, this paper makes the following contributions:

- We explore the influence of "dead weights" showing that they can adversely affect the optimization of BNNs. To the best of our knowledge, this is the first work to analyze the "dead weights" in BNNs.

- We introduce a rectified clamp unit (ReCU) to revive the "dead weights" and then provide a rigorous mathematical proof that a smaller quantization error can be derived using our ReCU.

- A mathematical analysis on why the weight standardization helps boost BNNs is provided, and the inherent contradiction between minimizing the quantization error and maximizing the information entropy in BNNs is revealed.

- Extensive experiments demonstrate that ReCU not only leads to better performance over many state-of-the-arts [12, 39, 31, 51, 15, 48, 41, 9, 46, 36, 16, 17, 8, 32], but also results in faster training convergence.

2. Related Work

As a pioneering work, Courbariaux *et al.* [11] binarizes both weights and activations with the sign function. To overcome the almost everywhere zero gradient in the sign function, they considers the STE [3] as an approximation to enable the gradient to back propagate. However, the representational ability of BNNs is very limited in a binary space, leading to a significant drop in accuracy. To mitigate the accuracy gap between BNN and its full-precision counterpart, XNOR-Net [41] introduces a scaling factor, which is obtained through the ℓ_1-norm of the weights or activations, to reduce the quantization error. XNOR-Net++ [6]

fuses the two scaling factors for quantized weights and activations into one parameter, and makes it learnable via the standard backpropagation. The rotated binary neural network (RBNN) [31] takes into account the influence of the angular bias between the binarized weight vector and its full-precision version, and then devises a bi-rotation scheme with two rotation matrices for angle alignment, which reduces the quantization error.

Other works propose to boost the performance of BNNs by devising new gradient estimation functions or designing quantization-friendly network architectures. For example, [15, 47, 31] design a continuous activation gradient that gradually approximates the sign function so as to replace the conventional STE [3]. Qin *et al.* [39] proposed an error decay estimator to minimize the information loss of gradients during backpropagation. ABC-Net [32] utilizes more binary bases for weights and activations to strengthen the model performance. ReActNet [35] constructs a strong baseline by adding parameter-free shortcuts on top of MobileNetV1 [23] and achieves 69.4% top-1 accuracy on ILSVRC-2012. Leng *et al.* [28] modelled the BNN learning as a discretization-constrained optimization problem solved by the ADMM optimizer, so as to avoid the non-differentiable quantization. In [48], an auxiliary probability matrix is made to search for the discrete quantized weights, implemented in a differentiable manner.

3. Background

In this section, we briefly review the optimization of BNNs. Given a DNN, for ease of representation, we simply denote its per-layer real-valued weights as $\mathcal{W}^r$ and the inputs as $\mathcal{A}^r$. Then, the convolutional result can be expressed as

$$Y = \mathcal{A}^r \otimes \mathcal{W}^r, \qquad (1)$$

where $\otimes$ represents the standard convolution. For simplicity, we omit the non-linear operations in this paper.

BNN aims to binarize each weight $w^r \in \mathcal{W}^r$ and each activation $a^r \in \mathcal{A}^r$ to $\{+1, -1\}$. Following XNOR-Net [41], the binarization can be achieved by the sign function,

$$x^b = \text{sign}(x^r) = \begin{cases} +1, & \text{if } x^r \geq 0, \\ -1, & \text{otherwise.} \end{cases} \qquad (2)$$

To mitigate the large quantization error in binarizing a DNN, XNOR-Net [41] further introduces two scaling factors for the weights $\mathcal{W}^b$ and activations $\mathcal{A}^b$, respectively. In this paper, following [6], we simplify these two scaling factors as one parameter, denoted as α. Then, the binary convolution operation can be formulated as

$$Y \approx (\mathcal{A}^b \circledast \mathcal{W}^b) \odot \alpha, \qquad (3)$$

where $\circledast$ represents the bit-wise operations including XNOR and POPCOUNT, and $\odot$ denotes the element-wise

multiplication. Then, the quantization error in a BNN is defined as

$$\text{QE} = \int_{-\infty}^{+\infty} f(w^r)\big(w^r - \alpha\,\text{sign}(w^r)\big)^2 \mathrm{d}w^r, \qquad (4)$$

where $f(w^r)$ is the probability density function of w^r.

To train a BNN, the forward convolution is achieved using the $\mathcal{W}^b$ and $\mathcal{A}^b$ binarized by Eq. (2), while the real-valued $\mathcal{W}^r$ and $\mathcal{A}^r$ are updated during backpropagation. However, the gradient of the sign function is zero-valued almost everywhere, which is not suitable for optimization. Instead, we use the simple STE [3] in this paper to compute the approximate gradient of the loss *w.r.t.* $w^r \in \mathcal{W}^r$, as

$$\frac{\partial \mathcal{L}}{\partial w^r} = \frac{\partial \mathcal{L}}{\partial w^b} \cdot \frac{\partial w^b}{\partial w^r} \approx \frac{\partial \mathcal{L}}{\partial w^b}, \qquad (5)$$

where $\mathcal{L}$ is the loss function.

As for the gradient *w.r.t.* the activations, we consider the piece-wise polynomial function [36] as follows

$$\frac{\partial \mathcal{L}}{\partial a^r} = \frac{\partial \mathcal{L}}{\partial a^b} \cdot \frac{\partial a^b}{\partial a^r} \approx \frac{\partial \mathcal{L}}{\partial a^b} \cdot \frac{\partial F(a^r)}{\partial a^r}, \qquad (6)$$

where

$$\frac{\partial F(a^r)}{\partial a^r} = \begin{cases} 2 + 2a^r, & \text{if } -1 \leq a^r < 0, \\ 2 - 2a^r, & \text{if } \quad 0 \leq a^r < 1, \\ 0, & \text{otherwise.} \end{cases} \qquad (7)$$

From now on, we drop the superscript "r" for real-valued weights for simplicity.

4. Methodology

4.1. The Dead Weights in BNNs

As pointed out in [50, 2], the latent weights $\mathcal{W}$ of a quantized network roughly follow the zero-mean Laplace distribution due to their quantization in the forward propagation. As can be seen from Fig. 2, most weights are gathered around the distribution peak (origin point), while many outliers fall into the two tails, far away from the peak. We argue that these outliers adversely affect the training of a BNN and might be the potential reason for the slow convergence when training BNNs. Specifically, in real-valued networks, weights of different magnitudes make different contributions to the network performance; in other words, it is how far each weight is from the origin that matters. In BNNs, however, there is not much distinction between weights of different magnitudes if they have the same sign since only the signs are kept in the forward inference regardless of their magnitudes. Therefore, from the perspective of optimization, though the magnitudes of the weights are updated during the backpropagation by gradient descent, the chances of changing their signs are unequal.

3

Intuitively, the signs of the weights around the distribution peak are easily changed, while it is the opposite for the outliers in the tails, which greatly limits the representational ability of BNNs and thus causes slow convergence in training. For this reason, we call these outliers "dead weights" in BNNs.

To solve this problem, in Sec. 4.2, we introduce our rectified clamp unit to revive these "dead weights" along with a rigorous proof that our clamp function leads to a smaller quantization error. In Sec. 4.3, we analyze why the weight standardization can boost the performance of BNNs, and reveal the inherent contradiction between minimizing the quantization error and maximizing the information entropy of the weights. Correspondingly, in Sec. 5, we introduce an adaptive exponential scheduler to identify the range of the "dead weights" in order to seek a balance between the quantization error and the information entropy.

4.2. Rectified Clamp Unit

To solve the aforementioned problem, we propose ReCU, which aims to move the "dead weights" towards the distribution peak to increase the probability of changing their signs. Specifically, for each real-valued weight $w \in \mathcal{W}$, ReCU is formulated as

$$\text{ReCU}(w) = \max\left(\min\left(w, Q_{(\tau)}\right), Q_{(1-\tau)}\right), \quad (8)$$

where $Q_{(\tau)}$ and $Q_{(1-\tau)}$ respectively denote the τ quantile and $1 - \tau$ quantile [53] of $\mathcal{W}$. With $0.5 < \tau \leq 1$, ReCU relocates w to $Q_{(1-\tau)}$ if it is smaller than $Q_{(1-\tau)}$, and to $Q_{(\tau)}$ if it is larger than $Q_{(\tau)}$. In this way, the "dead weights" are revived. To measure the contribution of ReCU in the quantization process, we take into account the quantization error for analysis. In what follows, we show that the weights after applying ReCU can derive a smaller quantization error.

Earlier works [50, 2] have shown that the latent weights roughly follow the zero-mean Laplace distribution, i.e., $w \sim La(0, b)$, which implies $Q_{(\tau)} + Q_{(1-\tau)} = 0$. Thus, we have

$$\int_{-\infty}^{Q_{(\tau)}} \frac{1}{2b} \exp\left(-\frac{|w|}{b}\right) \mathrm{d}w = \tau, \quad (9)$$

which results in

$$Q_{(\tau)} = -b \ln(2 - 2\tau). \quad (10)$$

However, it is difficult to know the exact value of b. Luckily, we can obtain its approximation via the maximum likelihood estimation, represented as

$$\hat{b} = \text{Mean}(|\mathcal{W}|), \quad (11)$$

where $\text{Mean}(|\cdot|)$ returns the mean of the absolute values of the inputs. Thus, $Q_{(\tau)}$ is a function of τ.
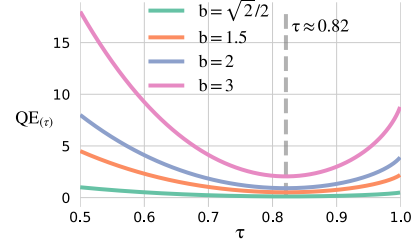
Figure 3: Visualization of the quantization error after applying ReCU. As can be seen, $\text{QE}(\tau)$ is a convex function of τ and reaches the minimum when $\tau \approx 0.82$.

After applying ReCU to w, the generalized probability density function of w can be written as follows

$$f(w) = \begin{cases} \dfrac{1}{2b} \exp\left(\dfrac{-|w|}{b}\right), & \text{if } |w| < Q_{(\tau)}, \\ 1 - \tau, & \text{if } |w| = Q_{(\tau)}, \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

To obtain the quantization error, we first compute the scaling factor in Eq. (3) using the Riemann-Stieltjes integral as

$$\begin{aligned} \alpha &= \mathbb{E}(|\text{ReCU}(\mathcal{W})|) \\ &= \int_{-Q_{(\tau)}}^{Q_{(\tau)}} |w| f(w) \mathrm{d}w + \sum_{|w| = Q_{(\tau)}} |w| f(w) \\ &= \int_0^{Q_{(\tau)}} \frac{w}{b} \exp\left(-\frac{w}{b}\right) \mathrm{d}w + 2 Q_{(\tau)}(1 - \tau) \\ &= b - (Q_{(\tau)} + b) \exp\left(-\frac{Q_{(\tau)}}{b}\right) + 2 Q_{(\tau)}(1 - \tau). \end{aligned} \quad (13)$$

Obviously, α is a function of τ after replacing b with the estimation $\hat{b}$ in Eq. (11). With $Q_{(\tau)} + Q_{(1-\tau)} = 0$, the quantization error in Eq. (4) after ReCU is derived as

$$\begin{aligned} \text{QE}(\tau) &= \int_{-\infty}^{+\infty} f(w)\left(w - \alpha \operatorname{sign}(w)\right)^2 \mathrm{d}w \\ &= \int_{-Q_{(\tau)}}^{Q_{(\tau)}} f(w)\left(w - \alpha \operatorname{sign}(w)\right)^2 \mathrm{d}w \\ &\quad + \sum_{|w| = Q_{(\tau)}} f(w)\left(w - \alpha \operatorname{sign}(w)\right)^2 \\ &= \int_0^{Q_{(\tau)}} \frac{1}{b} \exp\left(-\frac{w}{b}\right)(w - \alpha)^2 \mathrm{d}w \\ &\quad + \sum_{|w| = Q_{(\tau)}} (1 - \tau)\left(w - \alpha \operatorname{sign}(w)\right)^2 \\ &= (\alpha - b)^2\left(1 + \exp\left(-\frac{Q_{(\tau)}}{b}\right)\right) + b^2 \\ &\quad - \left((b + Q_{(\tau)})^2 - 2\alpha Q_{(\tau)}\right) \exp\left(-\frac{Q_{(\tau)}}{b}\right) \\ &\quad + 2(1 - \tau)(Q_{(\tau)} - \alpha)^2. \end{aligned} \quad (14)$$

4

Similarly, setting $b = \hat{b}$ makes $\text{QE}(\tau)$ a function of τ. From Eq. (14), we have two observations: (1) As plotted in Fig. 3, $\text{QE}(\tau)$ is a convex function when $0.5 < \tau \leq 1$ and reaches the minimum when $\tau \approx 0.82$[1]. (2) When $\tau = 1$, Eq. (14) degenerates to the normal quantization error as defined in Eq. (4) where ReCU is not introduced.

However, we cannot keep $\tau = 0.82$ to pursue the least quantization error. In the next subsection, we analyze another important factor to the network performance, *i.e.*, information entropy, which requires $\tau > 0.82$ to support good performance, and reveal the contradiction between minimizing the quantization error and maximizing the information entropy. Overall, we have the following inequality

$$\text{QE}(\tau) \leq \text{QE}(1) = \text{QE}, \quad 0.82 \leq \tau \leq 1. \quad (15)$$

That is, ReCU provides a smaller quantization error than QE when $0.82 \leq \tau < 1$.

4.3. Information Entropy of Weights

The information entropy of a random variable is the average level of uncertainty in the variable's possible outcomes, which is also used as a quantitative measure to reflect the weight diversity in BNNs [31, 39, 40, 33]. Usually, the more diverse, the better the performance of a BNN. Given a probability density function $p(x)$ on domain $\mathcal{X}$, the information entropy is defined as

$$H(p) = \mathbb{E}\big(-\ln(p(x))\big) = -\int_{\mathcal{X}} p(x) \ln\big(p(x)\big)\mathrm{d}x. \quad (16)$$

Accordingly, the information entropy of $\mathcal{W}$ after applying ReCU can be computed by

$$
\begin{aligned}
H(f) &= -\int_{-Q_{(\tau)}}^{Q_{(\tau)}} f(w) \ln\big(f(w)\big)\mathrm{d}w \\
&\quad - \sum_{|w|=Q_{(\tau)}} f(w) \ln\big(f(w)\big) \\
&= -\int_{0}^{Q_{(\tau)}} \frac{1}{b} \exp(-\frac{w}{b}) \ln\big(-\frac{1}{2b}\exp(-\frac{w}{b})\big)\mathrm{d}w \\
&\quad - 2(1-\tau)\ln(1-\tau) \\
&= 2(\ln b + 1)\tau + \ln\frac{2}{b} - 1,
\end{aligned}
\quad (17)
$$

which is a function of τ by substituting $\hat{b}$ in Eq. (11) for b.

For ease of the following analysis, we visualize the information entropy *w.r.t.* varying values of b and τ in Fig. 4. Then, we have

Case 1: $b = e^{-1}$. In this situation, the information entropy $H(f)$ is fixed to $\ln 2$ (the dotted white line in Fig. 4).
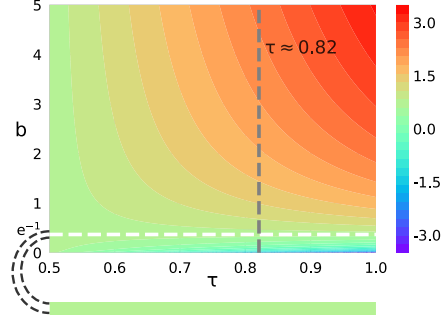
<hr>
[1]A rigorous proof is provided in the Appendix.

Figure 4: The information entropy of $\mathcal{W}$ *w.r.t.* τ and b. The dotted white line indicates a fixed value of the information entropy when $b = e^{-1}$. (Best viewed in color)

Case 2: $b < e^{-1}$. $H(f)$ is a monotonically decreasing function of τ.

Case 3: $b > e^{-1}$. $H(f)$ is a monotonically increasing function of τ.

Recall that b is estimated by the mean of the absolute values of $\mathcal{W}$ in Eq. (11). We have experimentally observed that in the cases of $b \leq e^{-1}$, the information entropy ($\leq \ln 2$) is too small to enable good performance (see Sec. 5.2.2). Thus, a larger b should be derived to overcome this problem. However, in practice, the weights $\mathcal{W}$ gradually become sparse during network training due to the widely-used ℓ_p-norm regularization in modern neural networks, making the information entropy uncontrollable, which inevitably brings a loss of diversity.

Thus, it is necessary to maintain b at a relatively high value (*Case 3*) in a controllable manner to retain the information entropy. The previous work [39] maximizes the information entropy by centralizing and standardizing the weights in each forward propagation as follows

$$\mathcal{W}' = \frac{\mathcal{W} - \mathbb{E}(\mathcal{W})}{\sigma(\mathcal{W})}, \quad (18)$$

where $\sigma(\cdot)$ denotes the standard deviation. In what

However, we experimentally find that it is the standardization, but not the centralization, that contributes to the performance improvement. The reason comes from the fact that $\mathbb{E}(\mathcal{W}) \approx 0$ in most cases [20, 39]. This motivates us to generalize Eq. (18) by simply standardizing the weights $\mathcal{W}$ as

$$\mathcal{W}' = \frac{\mathcal{W}}{K}, \quad (19)$$

where $K > 0$ is a given constant. Then, the mean of the absolute values of the weights after standardization is

$$b' = \text{Mean}(|\mathcal{W}'|) = \frac{b}{K}. \quad (20)$$

It is easy to see that $\sigma(\mathcal{W}) = \sqrt{2}b$ due to the Laplace distribution. Therefore, by setting $K = \sigma(\mathcal{W})$ as in [39], b'

becomes

$$b' = \frac{b}{\sqrt{2}b} = \frac{\sqrt{2}}{2} > e^{-1}, \qquad (21)$$

which increases the information entropy and explains why dividing $\mathcal{W}$ by the standard deviation can result in better performance when training BNNs [39]. To the best of our knowledge, this is the first time that a mathematical explanation is provided. Nevertheless, according to Fig. 4, the information entropy can be increased with a larger b. Thus, we further define $K = \sigma(\mathcal{W})/(\sqrt{2}b^\star)$ where $b^\star$ is a predefined constant, and the standardized weights $\mathcal{W}'$ become

$$\mathcal{W}' = \frac{\mathcal{W}}{\sigma(\mathcal{W})/(\sqrt{2}b^\star)}. \qquad (22)$$

It is easy to see that

$$b' = b^\star. \qquad (23)$$

The innovation behind this analysis lies in that our standardization transforms the uncontrolled information entropy to an adjustable one by manually setting $b^\star$ based on the premise of $b^\star > e^{-1}$, and therefore generalizes the information gain of Eq. (18) by [39]. Thus, by standardizing the weights using Eq. (22) before applying ReCU, the information entropy can be increased in the learning of a BNN.

Nevertheless, the information increase from enlarging b is still very limited (see Fig. 4). In contrast, the increase of τ leads to more information gain, with an unexpected increase in quantization error when $\tau > 0.82$, as analyzed in Sec. 4.2. Thus, there exists an inherent contradiction between minimizing the quantization error and maximizing the information entropy in BNNs. To the best of our knowledge, we are the first to find this contradiction. In Sec. 5.2.1, we propose an exponential scheduler for adapting τ along the network training, so as to seek a balance between the information entropy and the quantization error.

Training Procedures. Given a DNN with its per-layer real-valued weights $\mathcal{W}$ and the inputs $\mathcal{A}$, in the forward propagation, we first standardize $\mathcal{W}$ and revive the "dead weights" using Eq. (22) and ReCU of Eq. (8), respectively. Then, we compute the scaling factor α using Eq. (13), and binarize the inputs and the revived weights using the sign function of Eq. (2). Finally, we complete the binary convolution using Eq. (3) for the forward propagation. During backpropagation, we derive the gradients *w.r.t.* $\mathcal{W}$ and $\mathcal{A}$ using Eq. (5) and Eq. (6), respectively, and update $\mathcal{W}$ using the stochastic gradient descent (SGD) described in Sec. 5.1.

5. Experiments

In this section, we evaluate ReCU on the two widely-adopted CIFAR-10 [26] and ILSVRC-2012 [44] datasets, and then compare it to several state-of-the-art methods [39, 36, 31, 48, 15].

Table 1: Top-1 accuracy of ResNet-20 *w.r.t.* different values of τ on CIFAR-10.

τ	mean $\pm$ std (%)	τ	mean $\pm$ std (%)
1.00	85.44 ± 0.13	0.90	86.47 ± 0.23
0.98	86.40 ± 0.10	0.85	86.21 ± 0.29
0.96	86.48 ± 0.16	0.82	85.19 ± 0.18
0.94	$\mathbf{86.67 \pm 0.34}$	0.80	82.44 ± 0.22
0.92	86.45 ± 0.31	0.78	80.01 ± 0.16

5.1. Implementation

Network Structures. On CIFAR-10, we evaluate ReCU with ResNet-18/20 [19] and VGG-Small [49]. Following the compared methods, we binarize all convolutional and fully-connected layers except the first and the last ones. For ResNet-18/20, we adopt the double skip connections as proposed in [36] for fair comparison.

On ILSVRC-2012, we choose to binarize ResNet-18/34. Following [4], the downsampling layers are not quantized. Similarly, the double skip connections [36] are added.

Training Details. Our network is trained from scratch without depending on a pre-trained model. For all experiments, we use SGD for optimization with a momentum of 0.9 and the weight-decay is set to 5e-4. The initial learning rate is 0.1 and then adjusted by the cosine scheduler [37]. We follow the data augmentation strategies in [19] which include random crops and horizontal flips.

5.2. Ablation Study

In this section, we discuss the hyperparameter settings of ReCU, including $b^\star$ and τ. Recall that $b^\star$ affects the information entropy, while τ affects both the quantization error and the information entropy. Each experiment is run three times and we report the mean top-1 accuracy (mean $\pm$ std) of ResNet-20 for parametric analyses.

5.2.1 Effect of τ for ReCU

In Sec. 4.2, we demonstrate that the quantization error with ReCU is a convex function of τ and becomes the minimum when $\tau \approx 0.82$, while the information entropy is a monotonically increasing function of τ if $b^\star > e^{-1}$. Thus, a balance needs to be reached between the quantization error and the information entropy. To this end, following [39] (Eq. (18)), we set $b^\star = \sqrt{2}/2$ for our analyses.

We first consider setting τ to a fixed value for the whole training process. As shown in Tab. 1, when $\tau = 0.94$, the network reaches the best performance. It is worth noting that a significant drop in accuracy occurs when $\tau < 0.82$. This is understandable since it suffers both a large quantization error and small information entropy. Another observation is that ReCU does not obtain the best accuracy when

Table 2: Top-1 accuracy of ResNet-20 *w.r.t.* τ calculated by our exponential scheduler on CIFAR-10.

τ_s/τ_e	mean $\pm$ std (%)	τ_s/τ_e	mean $\pm$ std (%)
0.80 / 1.00	86.68 ± 0.14	0.80 / 0.99	86.82 ± 0.15
0.85 / 1.00	86.76 ± 0.16	**0.85 / 0.99**	**86.89 $\pm$ 0.19**
0.90 / 1.00	86.72 ± 0.10	0.90 / 0.99	86.85 ± 0.10

Table 3: Top-1 accuracy of ResNet-20 *w.r.t.* different values of $b^\star$ for weight standardization on CIFAR-10. "w/o" denotes binarization without our standardization.

$b^\star$	mean $\pm$ std (%)	$b^\star$	mean $\pm$ std (%)
w/o	86.16 ± 0.11	**2**	**87.31 $\pm$ 0.17**
0.2	86.55 ± 0.16	3	87.15 ± 0.15
$\sqrt{2}/2$	86.89 ± 0.19	4	87.09 ± 0.17
1	87.10 ± 0.20	5	87.08 ± 0.12

Figure 5: Training and validation accuracy curves with and without ReCU (ResNet-20 on CIFAR-10).

$\tau = 0.82$. This is because though the quantization error reaches the minimum when $\tau = 0.82$ as shown in Fig. 3, the small information entropy cannot support a good performance. In summary, when $0.85 \leq \tau \leq 1.00$, we can seek a balance between the quantization error and the information entropy.

Despite its good performance when using a fixed value of τ, we find that ReCU increases the variance of the performance when $0.85 \leq \tau \leq 0.94$ while keeping it stable when $0.96 \leq \tau \leq 1.00$ as shown in Tab. 1. To solve this, we further propose an exponential scheduler for adapting τ along the network training. Our motivation lies in that τ should start with a value falling within $[0.85, 0.94]$ to pursue a good accuracy, and then gradually go to the interval $[0.96, 1.00]$ to stabilize the variance of performance. Based on this, given an initial τ_s and an end threshold τ_e, τ_i at the i-th training epoch is calculated as follows

$$\tau_i = \frac{\tau_e - \tau_s}{e - 1}e^{i/I} + \frac{e \cdot \tau_s - \tau_e}{e - 1}, \qquad (24)$$

where I denotes the total number of training epochs.

Tab. 2 shows that ReCU obtains better performance of 86.89% with $\tau_s = 0.85$ and $\tau_e = 0.99$. Besides, it can well overcome the large variance with a fixed τ.

5.2.2 Effect of $b^\star$ for Weight Standardization

Tab. 3 displays the results *w.r.t.* different values of $b^\star$. we use Eq. (24) for adapting τ with $\tau_s = 0.85$ and $\tau_e = 0.99$. The experiments are conducted under three settings for a comprehensive analysis, including training the BNN without our standardization, $b^\star = 0.2 < e^{-1}$, and $b^\star > e^{-1}$.

As can be observed, without our standardization, the binarized ResNet-20 shows a poor top-1 accuracy of 86.16%. For a detailed analysis, during network training, the ℓ_p-norm regularization in the current neural network sparsifies the network parameters, which reduces the information entropy as discussed in Sec. 4.3.

With our standardization in hand, the information entropy can be manually controlled by adjusting $b^\star$. In Tab. 3, with a small $b^\star = 0.2 < e^{-1}$, despite the better performance of 86.55%, the improvement is very limited. As discussed in Sec. 4.3, the information entropy is still too small to enable good performance when $b^\star \leq e^{-1}$.

We further set $b^\star > e^{-1}$. As can be seen, the network reaches a maximum mean top-1 accuracy of 87.31% when $b^\star = 2$, a significant improvement over the model without our standardization. We also observe that as $b^\star$ continues to increase, the performance starts to remain stable, which supports our claim in Sec. 4.3 that the improvement from enlarging $b^\star$ is limited.

5.3. Training Convergence

As discussed in Sec. 4.1, the "dead weights" introduces an obstacle to the training convergence of BNNs. To verify the effectiveness of ReCU in overcoming this problem, we compare the convergence ability when training a BNN with and without ReCU. Fig. 5 shows that the curves equipped with ReCU acquire a significantly faster training convergence, especially for the validation set. This is because, ReCU revives the "dead weights" with the τ quantile and $1 - \tau$ quantile of the weights, making it easier for them to change their signs, leading to faster training convergence.

5.4. Comparison with SOTA Methods

To quantitatively evaluate the effectiveness of the proposed ReCU, we conduct extensive experiments on CIFAR-10 [26] and ImageNet [44]. We also compare it with a number of state-of-the-art methods to demonstrate the advantages of ReCU in boosting the performance of BNNs. In the following experiments, we use Eq. (24) for adapting τ with $\tau_s = 0.85$ and $\tau_e = 0.99$. Besides, $b^\star$ is set to 2.

5.4.1 CIFAR-10

For ResNet-18, we compare ReCU with RAD [12], RBNN [31] and IR-Net [39]. For ResNet-20, the com-

Table 4: Performance comparison with the state-of-the-arts on CIFAR-10. W/A denotes the bit length of the weights and activations. FP is short for full precision.

Network	Method	W/A	Top-1
ResNet-18	FP	32/32	94.8%
	RAD [12]	1/1	90.5%
	IR-Net [39]	1/1	91.5%
	RBNN [31]	1/1	92.2%
	ReCU (Ours)	1/1	**92.8**%
ResNet-20	FP	32/32	92.1%
	DoReFa [51]	1/1	79.3%
	DSQ [15]	1/1	84.1%
	SLB [48]	1/1	85.5%
	IR-Net [39]	1/1	86.5%
	ReCU (Ours)	1/1	**87.4**%
VGG-small	FP	32/32	94.1%
	XNOR-Net [41]	1/1	89.8%
	BNN [11]	1/1	89.9%
	RAD [12]	1/1	90.0%
	DoReFa [51]	1/1	90.2%
	IR-Net [39]	1/1	90.4%
	RBNN [31]	1/1	91.3%
	DSQ [15]	1/1	91.7%
	SLB [48]	1/1	92.0%
	ReCU(Ours)	1/1	**92.2**%

Table 5: Performance comparison with the state-of-the-arts on ImageNet. W/A denotes the bit length of the weights and activations. FP is short for full precision.

Network	Method	W/A	Top-1	Top-5
ResNet-18	FP	32/32	69.6%	89.2%
	BNN [11]	1/1	42.2%	67.1%
	XNOR-Net [41]	1/1	51.2%	73.2%
	DoReFa [51]	1/2	53.4%	-
	TBN [46]	1/2	55.6%	79.0%
	Bi-Real Net [36]	1/1	56.4%	79.5%
	PDNN [16]	1/1	57.3%	80.0%
	IR-Net [39]	1/1	58.1%	80.0%
	DoReFa [51]	1/4	59.2%	-
	BONN [17]	1/1	59.3%	81.6%
	HWGQ [8]	1/2	59.6%	82.2%
	RBNN [31]	1/1	59.9%	81.9%
	ReCU(Ours)	1/1	**61.0**%	**82.6**%
ResNet-34	FP	32/32	73.3%	91.3%
	ABC-Net [32]	1/1	52.4%	76.5%
	Bi-Real Net [36]	1/1	62.2%	83.9%
	IR-Net [39]	1/1	62.9%	84.1%
	RBNN [31]	1/1	63.1%	84.4%
	ReCU(Ours)	1/1	**65.1**%	**85.8**%

pared methods include SLB [48], DSQ [15], DoReFa [51], and IR-Net [39]. For VGG-Small, we compare ReCU with XNOR-Net [41], DoReFa [51], BNN [11], SLB [48], IR-Net [39], RAD [12], DSQ [15], and RBNN [31]. The experimental results are shown in Tab. 4. As can be observed, ReCU shows the best performance in all the networks. Specifically, with ResNet-18, ReCU obtains 0.6% performance increase over the recent RBNN. Also, it yields 0.9% performance gain over IR-Net in binarizing ResNet-20. Lastly, it retains a top-1 accuracy of 92.2% when binarizing VGG-small, which is better than the search-based SLB result of 92.0%.

5.4.2 ImageNet

Tab. 5 displays the performance comparison in binarizing ResNet-18/34. For ResNet-18, we compare ReCU with BNN [11], Bi-Real Net [36], XNOR-Net [41], SLB [48], DoReFa [51], IR-Net [39], RBNN [31], PDNN [16] and BONN [17]. For ResNet-34, ABC-Net [32], Bi-Real Net [36], IR-Net [39] and RBNN [31] are compared. From

Tab. 5, we can see that ReCU takes the leading place in both the top-1 and top-5 accuracies. Specifically, it obtains a better performance of 61.0% in top-1 and 82.6% in top-5 compared to RBNN's 59.9% top-1 accuracy and 81.9% top-5 accuracy with ResNet-18. The advantage in ResNet-34 is even more obvious where ReCU obtains 2.0% and 1.4% performance improvements in top-1 and top-5 accuracy, respectively.

Thus, Tab. 4 and Tab. 5 verify the correctness of exploring the "dead weights" and the effectiveness of ReCU in reviving them, which greatly benefits the performance of BNNs.

6. Conclusion

In this paper, we present a novel rectified clamp unit (ReCU) to revive the "dead weights" when training BNNs. We first analyze how the "dead weights" block the optimization of BNNs and slow down the training convergence. Then, ReCU is applied to increase the probability of changing the signs of the "dead weights" on the premise of a rigorous proof that ReCU can lead to a smaller quantization error. Besides, we analyze why the weight standardization can increase the information entropy of the weights, and thus benefit the BNN performance. The inherent con-

tradiction between minimizing the quantization error and maximizing the information entropy is revealed for the first time. Correspondingly, an adaptive exponential scheduler is proposed to seek a balance between the quantization error and the information entropy. Experimental results demonstrate that, by reviving the "dead weights", ReCU leads to not only faster training convergence but also state-of-the-art performance.

7. Acknowledge

References

[1] Milad Alizadeh, Javier Fernández-Marqués, Nicholas D Lane, and Yarin Gal. An empirical study of binary neural networks' optimisation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2

[2] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 7950–7958, 2019. 3, 4

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2, 3

[4] Joseph Bethge, Haojin Yang, Marvin Bornstein, and Christoph Meinel. Back to simplicity: How to train accurate bnns from scratch? *arXiv preprint arXiv:1906.08637*, 2019. 6

[5] Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. Bats: Binary architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[6] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2019. 1, 2, 3

[7] Adrian Bulat, Georgios Tzimiropoulos, Jean Kossaifi, and Maja Pantic. Improved training of binary networks for human pose estimation and image recognition. *arXiv preprint arXiv:1904.05868*, 2019. 2

[8] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5918–5926, 2017. 2, 8

[9] Hanlin Chen, Baochang Zhang Li'an Zhuo, Xiawu Zheng, Jianzhuang Liu, David Doermann, and Rongrong Ji. Binarized neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020. 2

[10] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 3123–3131, 2015. 1

[11] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016. 1, 2, 8

[12] Ruizhou Ding, Ting-Wu Chin, Zeye Liu, and Diana Marculescu. Regularizing activation distribution for training binarized deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11400–11409, 2019. 2, 7, 8

[13] Xiaohan Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, Ji Liu, et al. Global sparse momentum sgd for pruning very deep neural networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 6382–6394, 2019. 1

[14] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88:303–338, 2010. 1

[15] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4851–4860, 2019. 1, 2, 3, 6, 8

[16] Jiaxin Gu, Ce Li, Baochang Zhang, Jungong Han, Xianbin Cao, Jianzhuang Liu, and David Doermann. Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 8344–8351, 2019. 2, 8

[17] Jiaxin Gu, Junhe Zhao, Xiaolong Jiang, Baochang Zhang, Jianzhuang Liu, Guodong Guo, and Rongrong Ji. Bayesian optimized 1-bit cnns. pages 4909–4917, 2019. 2, 8

[18] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1580–1589, 2020. 1

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2, 6

[20] Zhezhi He and Deliang Fan. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11438–11446, 2019. 5

[21] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[22] Geoffrey E. Hinton, Oriol Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[23] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 3

[24] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018. 1

[25] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 2760–2769, 2018. 1

[26] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009. 2, 6, 7

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[28] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 3

[29] Darryl D Lin and Sachin S Talathi. Overcoming challenges in fixed point training of deep convolutional networks. In *Proceedings of the International Conference on Machine Learning Workshops (ICML)*, 2016. 2

[30] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 2181–2191, 2017. 1

[31] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 3, 5, 6, 7, 8

[32] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2, 3, 8

[33] Chunlei Liu, Wenrui Ding, Xin Xia, Baochang Zhang, Jiaxin Gu, Jianzhuang Liu, Rongrong Ji, and David Doermann. Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2699, 2019. 5

[34] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 3296–3305, 2019. 1

[35] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2, 3

[36] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 722–737, 2018. 2, 3, 6, 8

[37] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with restarts. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 6

[38] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018. 1

[39] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2247–2256, 2020. 2, 3, 5, 6, 7, 8

[40] Vishnu Raj, Nancy Nayak, and Sheetal Kalyani. Understanding learning dynamics of binary neural networks via information bottleneck. *arXiv preprint arXiv:2006.07522*, 2020. 5

[41] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 3, 8

[42] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1

[43] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 1

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 6, 7

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 1

[46] Diwen Wan, Fumin Shen, Li Liu, Fan Zhu, Jie Qin, Ling Shao, and Heng Tao Shen. Tbn: Convolutional neural network with ternary inputs and binary weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 315–332, 2018. 2, 8

[47] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quan-

tization networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7308–7316, 2019. 1, 2, 3

[48] Zhaohui Yang, Yunhe Wang, Kai Han, Chunjing Xu, Chao Xu, Dacheng Tao, and Chang Xu. Searching for low-bit weights in quantized neural networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 3, 6, 8

[49] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 6

[50] Kai Zhong, Tianchen Zhao, Xuefei Ning, Shulin Zeng, Kaiyuan Guo, Yu Wang, and Huazhong Yang. Towards lower bit multiplication for convolutional neural network training. *arXiv preprint arXiv:2006.02804*, 2020. 3, 4

[51] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 1, 2, 8

[52] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7920–7928, 2018. 1

[53] Daniel Zwillinger. *CRC standard mathematical tables and formulae*. CRC press, 2002. 4

Appendix

Convexity and Minimum of $\mathrm{QE}(\tau)$.

We first revisit the formulations for the τ quantile $Q_{(\tau)}$ and scaling factor α in the following

$$Q_{(\tau)} = -b\ln(2 - 2\tau). \tag{10}$$

$$\alpha = b - (Q_{(\tau)} + b)\exp(-\frac{Q_{(\tau)}}{b}) + 2Q_{(\tau)}(1 - \tau). \tag{13}$$

Combining Eq. (10) and Eq. (13), we can rewrite α as

$$\alpha = b(2\tau - 1). \tag{A1}$$

Recall that the quantization error under our framework is given as

$$\begin{aligned}
\mathrm{QE}(\tau) = {} & (\alpha - b)^2\big(1 + \exp(-\frac{Q_{(\tau)}}{b})\big) + b^2 \\
& - \big((b + Q_{(\tau)})^2 - 2\alpha Q_{(\tau)}\big)\exp(-\frac{Q_{(\tau)}}{b}) \\
& + 2(1 - \tau)(Q_{(\tau)} - \alpha)^2.
\end{aligned} \tag{14}$$

By putting Eq. (10) and Eq. (A1) into Eq. (14), we reformulate $\mathrm{QE}(\tau)$ as

$$\begin{aligned}
& \mathrm{QE}(\tau) \\
& = b^2\big(-16\tau^3 + 44\tau^2 - 40\tau - 4(\tau - 1)\ln(2 - 2\tau) + 13\big).
\end{aligned} \tag{A2}$$

According to Eq. (A2), the derivative of $\mathrm{QE}(\tau)$ *w.r.t.* τ can be derived as

$$\begin{aligned}
\frac{\partial\,\mathrm{QE}(\tau)}{\partial\tau} & = -4b^2(12\tau^2 - 22\tau + \ln(2 - 2\tau) + 11) \\
& = 4b^2 G(\tau),
\end{aligned} \tag{A3}$$

where

$$G(\tau) = -12\tau^2 + 22\tau - \ln(2 - 2\tau) - 11. \tag{A4}$$

Note that b is estimated via the maximum likelihood estimation as

$$\hat{b} = \mathrm{Mean}(|\mathcal{W}|), \tag{11}$$

which indicates $b \neq 0$. We can know that

$$\frac{\partial\,\mathrm{QE}(\tau)}{\partial\tau} = 0 \iff G(\tau) = 0. \tag{A5}$$

Thus, the extreme value of $\mathrm{QE}(\tau)$ is irrelevant to b. Further, we yield the derivative of $G(\tau)$ *w.r.t.* τ as

$$\frac{\partial G(\tau)}{\partial\tau} = -24\tau + \frac{1}{1 - \tau} + 22. \tag{A6}$$

From Eq. (A6), it is easy to know that $\dfrac{\partial G(\tau)}{\partial\tau} > 0$ if $\tau \leq 1$. Therefore, $G(\tau)$ is monotonically increasing when $\tau \leq 1$. By solving $G(\tau) = 0$, we have $\tau \approx 0.82$. That means when $0.82 < \tau \leq 1$, $G(\tau) > 0$, while when $0.5 < x < 0.82$, $G(\tau) < 0$. That is to say, $\dfrac{\partial QE(\tau)}{\partial\tau} < 0$ when $0.5 < \tau < 0.82$, and $\dfrac{\partial\,\mathrm{QE}(\tau)}{\partial\tau} > 0$ when $0.82 < \tau \leq 1$. Thus $\mathrm{QE}(\tau)$ is a convex function *w.r.t.* $\tau \in (0.5, 1]$ and reaches the minimum when $\tau \approx 0.82$, which completes the proof. ∎