Direct Quantization for Training Highly Accurate Low Bit-width Deep Neural Networks

Tuan Hoang[1] , **Thanh-Toan Do**[2] , **Tam V. Nguyen**[3] and **Ngai-Man Cheung**[1]
[1]Singapore University of Technology and Design
[2]University of Liverpool
[3]University of Dayton
nguyenanhtuan_hoang@mymail.sutd.edu.sg, thanh-toan.do@liverpool.ac.uk
tamnguyen@udayton.edu, ngaiman_cheung@sutd.edu.sg

Abstract

This paper proposes two novel techniques to train deep convolutional neural networks with low bit-width weights and activations. First, to obtain low bit-width weights, most existing methods obtain the quantized weights by performing quantization on the full-precision network weights. However, this approach would result in some mismatch: the gradient descent updates full-precision weights, but it does not update the quantized weights. To address this issue, we propose a novel method that enables direct updating of quantized weights with learnable quantization levels to minimize the cost function using gradient descent. Second, to obtain low bit-width activations, existing works consider all channels equally. However, the activation quantizers could be biased toward a few channels with high-variance. To address this issue, we propose a method to take into account the quantization errors of individual channels. With this approach, we can learn activation quantizers that minimize the quantization errors in the majority of channels. Experimental results demonstrate that our proposed method achieves state-of-the-art performance on the image classification task, using AlexNet, ResNet and MobileNetV2 architectures on CIFAR-100 and ImageNet datasets.

1 Introduction

Deep Convolutional Neural Networks (CNNs) have been playing a crucial role in the recent tremendous successes of a variety of computer vision tasks including image classification, object detection, segmentation, image retrieval, to name a few. However, unfortunately, this benefit comes at the cost of an excessive amount of memory and expensive computational resources, which impedes its application in embedded devices. Therefore, how to make CNN lightweight and practical in terms of memory and computation is an important task and has attracted a great amount of effort.

An effective approach is to use low bit-width weights and/or low bit-width activations. This approach not only can reduce the memory footprint, but it also achieves a significant gain in speed. In particular, using binary weights with full-precision (FP) activations can reduce the model size by $32\times$ and achieve $\sim 2\times$ faster during inference [Rastegari *et al.*, 2016]. When also using low bit-width activations, the speedup is significantly greater; as the most computationally expensive convolutions can be done by bitwise operations [Rastegari *et al.*, 2016]. Even though there are great improvements achieved by the existing quantization-based methods [Cai *et al.*, 2017; Hou and Kwok, 2018; Li *et al.*, 2017; Rastegari *et al.*, 2016; Wan *et al.*, 2018; Zhang *et al.*, 2018; Zhou *et al.*, 2016; Zhu *et al.*, 2017], there are still noticeable accuracy gaps between the quantized CNNs and their FP counterparts, especially in the challenging cases of 1 or 2 bit-width weights and activations.

Most of the current state-of-the-art network quantization methods [Wan *et al.*, 2018; Zhang *et al.*, 2018; Zhou *et al.*, 2016; Zhu *et al.*, 2017; Zhou *et al.*, 2018] learn the FP weights and the quantizers to quantize FP weights. The learning can be sequential or joint learning. In the sequential learning, the models firstly learn FP weights. They then learn the quantizers to quantize FP weights. In the joint learning, the models learn the FP weights and their quantizers simultaneously. However, both learning approaches cause mismatch during training, i.e., the gradient descent process is only used to update the FP weights, but not the quantized weights. To address this problem, we propose a novel method that allows gradients to update the quantized weights with learnable quantization levels directly, i.e., the quantized weights are learned without requiring FP weights. In specific, the quantized weights are learned via learning two auxiliary variables: binary weight encodings and quantization basis vectors.

Another important aspect of training a low bit-width network is the activation quantization. Earlier works use simple, hand-crafted quantizers (e.g., uniform or logarithmic quantization) [Gupta *et al.*, 2015; Lin *et al.*, 2016; Zhou *et al.*, 2016; Aojun *et al.*, 2017; Hubara *et al.*, 2017] or pre-computed quantizers fixed during network training [Cai *et al.*, 2017]. Recently, [Zhang *et al.*, 2018; Jung *et al.*, 2019] proposed to learn the quantizers for activations during training adaptively. However, all existing methods consider all activation channels of an activation map equally and do not pay attention to the fact that different channels of activations contain different amount of information. As a result, the learned quantizer could be biased toward a few channels with high variances; while losing information from a large number of lower-variance channels. To handle this problem, we propose a novel activation quantization method which tries to minimize the information loss in the majority of channels, which cumulatively have noticeable impacts on outputs, via learning channel-wise quantizers (i.e., one quantizer per channel). Our main contributions can be summarized as follows:

- Our work is the first one to propose a novel method to learn the quantized weights with learnable quantization levels directly by using gradient descent such that they directly minimize the cost function. Hence, our method could avoid the training mismatch (i.e., the quantized weights are not updated by gradients) of existing methods.
- To avoid the activation quantizers being biased toward a few high variance activation channels, we propose to learn quantizers that focus to minimize quantization errors of the majority of channels. This is obtained via learning channel-wise quantizers.
- A detailed analysis on the inference efficiency of the proposed method is provided. The proposed method also achieves the state-of-the-art image classification accuracy when comparing to other low bit-width networks on CIFAR-100 and ImageNet datasets with AlexNet, ResNet and MobileNetV2 architectures.

2 Related Works

How to improve inference efficiency of CNN for practical applications has been an active research field. Existing approaches can be roughly divided into three main categories: compact network design, parameter reduction, and network quantization. In this section, we discuss recent works on the last category, to which our work belongs.

Weight-only Quantization. Earlier works have applied the basic form of weight quantization to directly constrains the weight values into the binary space without or with a scaling factor, i.e., $\{-1, 1\}$ in BinaryConnect [Courbariaux *et al.*, 2015] or $\{-\alpha, \alpha\}$ in Binary-Weight-Networks (BWN) [Rastegari *et al.*, 2016]. Ternary Weight Networks (TWN) [Li and Liu, 2017] proposed to quantize weights into a ternary form of $\{-\alpha, 0, \alpha\}$. Trained Ternary Quantization (TTQ) [Zhu *et al.*, 2017] generalized TWN by constraining the weights to asymmetric ternary values $\{-\alpha_n, 0, \alpha_p\}$. In comparison to the binary weights, the ternary weights can help to improve performance, while achieving a similar speedup during inference [Wan *et al.*, 2018]. Hou and Kwok proposed loss-aware quantized networks which can further improve the accuracy by considering the loss in learning the binary/ternary weights. Aojun *et al.* presented incremental network quantization (INQ) to incrementally convert a pretrained FP CNN model to a low-precision one.

Weight and Activation Quantization. Since quantization of activations can substantially reduce complexity further, by allowing the dot products to be implemented with bitwise operations (i.e., xor and $popcount$) [Tang *et al.*, 2017; Rastegari *et al.*, 2016], this approach attracts more and more attention [Rastegari *et al.*, 2016; Zhou *et al.*, 2016; Zhang *et al.*, 2018]. In early works [Tang *et al.*, 2017; Rastegari *et al.*, 2016], the authors proposed to binarize both weights and activations to $\{-1, 1\}$. Although these methods can be highly efficient in reference time, there are considerable accuracy drops compared to FP networks. To address this problem, the generalized low bit-width quantization was further studied in [Zhou *et al.*, 2016; Miyashita *et al.*, 2016]. By allowing to use more bits, this approach can provide a trade-off between accuracy and inference complexity. A popular choice of early works is the uniform quantization [Zhou *et al.*, 2016; Hubara *et al.*, 2017]. Miyashita *et al.* later proposed logarithmic quantization which can help to improve the inference efficiency via bit-shift operations. Park *et al.* used weighted-entropy to learn quantizers that are more concentrated on the values that are neither too small nor too large. Cai *et al.* proposed to pre-compute a single activation quantizer for all activation layers by fitting the probability density function of a half-wave Gaussian distribution. As opposed to the fixed, handcrafted quantization schemes of the aforementioned works, LQ-Nets [Zhang *et al.*, 2018] proposed to jointly train a quantized CNN and its associated quantizers. Jung *et al.* proposed to train quantizers with parameterized intervals, which simultaneously performs both pruning and clipping. Gu *et al.* proposed Bayesian optimized 1-bit CNNs (BONN) which incorporates the prior distributions of FP kernels and features into the Bayesian framework to construct 1-bit CNNs. Besides, recent works on designing new structures for binary networks achieve promising results. Bi-Real Net [Liu *et al.*, 2018] introduced a new variant of residual structure to preserve the real activations before the sign function. The shortcut can help to increase the representational capability of the 1-bit convolutional block. Zhuang *et al.* proposed Group-Net, in which a set of binary convolution branches can effectively replace a FP convolution.

3 Proposed Method

3.1 Learned Quantized Weights

Apart from existing network quantization works, which train the weight quantizers from the full-precision (FP) network weights either jointly or in advance [Zhou *et al.*, 2016; Wan *et al.*, 2018; Zhang *et al.*, 2018]; in this work, we propose to directly learn the quantized weights.

Specifically, for a general case of representing a network weight $\mathbf{W}_q$ of N_w elements by K_w-bits, we learn a binary weight encoding $\mathbf{S}_b^w \in \{-1, +1\}^{N_w \times K_w}$ and a quantization basis vector $\mathbf{v}^w = [v_1^w, \cdots, v_{K_w}^w] \in \mathbb{R}^{K_w}$. Consequently, the network weight can be directly replaced by $\mathbf{S}_b^w \mathbf{v}^w$ in the cost function, i.e., $\mathbf{W}_q = \mathbf{S}_b^w \mathbf{v}^w$.

$$\min_{\Theta} \mathcal{L}(\Theta), \quad \text{s.t. } \mathbf{S}_{bi}^w \in \{-1, +1\}^{N_w \times K_w} \ \forall i, \tag{1}$$

where $\Theta = \{\mathbf{S}_{bi}^w, \mathbf{v}_i^w\}_{\forall i}$ is the set of all network parameters and $\mathcal{L}$ is a cost function, e.g., cross-entropy for classification.

To handle the binary constraint on the weight encoding $\mathbf{S}_b^w$, inspired by Courbariaux *et al.*, we introduce a full-precision encoding matrix $\mathbf{S}^w$ (i.e., $\mathbf{S}^w \in \mathbb{R}^{N_w \times K_w}$) for parameter updates, and obtain $\mathbf{S}_b^w$ using sign function (i.e.,

Algorithm 1: Learned Quantized Weights (LQW). $\mathcal{L}$ is the cost function for a mini-batch.

```
Requires   : Learning rate $\eta$ and learning-rate scaling factors $\{\gamma_{\mathbf{v}}, \gamma_{\mathbf{s}}\}$
Parameters: Encoding matrix $\mathbf{S}^w$, basis vector $\mathbf{v}^w$
1 Forward propagation:
2     $\mathbf{S}_b^w = \text{sign}(\mathbf{S}^w)$
3     $\mathbf{W}_q = \mathbf{S}_b^w \mathbf{v}^w$
4 Backward propagation and Parameter update:
5     Given $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_q}$ obtained using the chain rule of gradient descent, compute $\frac{\partial \mathcal{L}}{\partial \mathbf{S}_b^w}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{v}^w}$
6     $\mathbf{v}^w \leftarrow \mathbf{v}^w - \gamma_{\mathbf{v}} \eta \frac{\partial \mathcal{L}}{\partial \mathbf{v}^w}$
7     $\mathbf{S}^w \leftarrow \text{clip}\left(\mathbf{S}^w - \gamma_{\mathbf{s}} \eta \frac{\partial \mathcal{L}}{\partial \mathbf{S}_b^w}, -1, 1\right)$          // STE
```

$\mathbf{S}_b^w = \text{sign}(\mathbf{S}^w)$) only during forward and backward propagations. Furthermore, in order to update $\mathbf{S}^w$ using gradient descent, we adopt the straight-through estimator (STE) $\frac{\partial \,\text{sign}(z)}{\partial z} = 1_{|z| \leq 1}$ [Hinton, 2012], i.e., $\frac{\partial \mathcal{L}}{\partial \mathbf{S}_b^w} = \frac{\partial \mathcal{L}}{\partial \mathbf{S}^w}$ for $\mathbf{S}^w \in [-1, 1]$, to approximate the gradients propagating through the sign function. Regarding the quantization basis vectors $\mathbf{v}^w$, as they have no constraint, they can be updated normally using the standard gradient descent process. The proposed method allows the quantized weights with learnable quantization levels to be updated (via the weight encodings and the quantization basis vectors) such that they directly minimize the final cost function through the gradient descent process. Additionally, we want to emphasize that other works (e.g., LQ-Nets, TBN) achieve quantized weights from FP weights during a joint training process. More specifically, their FP weights are updated via back-propagation, while the quantized weights ($\mathbf{W}_q$) are updated to minimize the ℓ_2-loss with the learning FP weights (i.e., $\arg\min_{\mathbf{W}_q} \|\mathbf{W}_q - \mathbf{W}\|^2$). This process potentially causes training mismatch, as the gradients cannot directly affect the quantized weights.

Algorithm 1, dubbed as Learned Quantized Weights (LQW), presents the procedure of our proposed method to train a quantized network weight. Note that, to maximize the flexibility of the low bit-width CNN, while ensuring its compatibility with bitwise operations, we learn one quantization basis vector and weight encoding per filter.

3.2 Activation Channel-Wise Averaged Quantizer

Scalar Quantizer. Given a full-precision ReLU activation $\mathbf{A} \in \mathbb{R}_{\geq 0}^{B \times C \times W \times H}$ (i.e., the set of non-negative real values), our goal is to represent $\mathbf{A}$ using K_a-bits with minimized information loss. We consider $\mathbf{A}$ as a vector $\mathbf{a}$ of N_a-dimension ($N_a = BCWH$). We aim to find an optimal quantization basis vector $\mathbf{v}^a = [v_1^a, \cdots, v_{K_a}^a] \in \mathbb{R}^{K_a}$ and an optimal binary encoding $\mathbf{S}^a \in \{0, 1\}^{N_a \times K_a}$ that minimize the following quantization error:

$$\arg\min_{\mathbf{v}^a, \mathbf{S}^a} \|\mathbf{a} - \mathbf{S}^a \mathbf{v}^a\|_2^2, \quad \text{s.t. } \mathbf{S}^a \in \{0, 1\}^{N_a \times K_a}. \qquad (2)$$

This problem can be solved by alternatively updating $\mathbf{S}^a$ and $\mathbf{v}^a$: ***(i)*** Given $\mathbf{v}^a$ fixed, the optimal $\mathbf{S}^a$ can be found

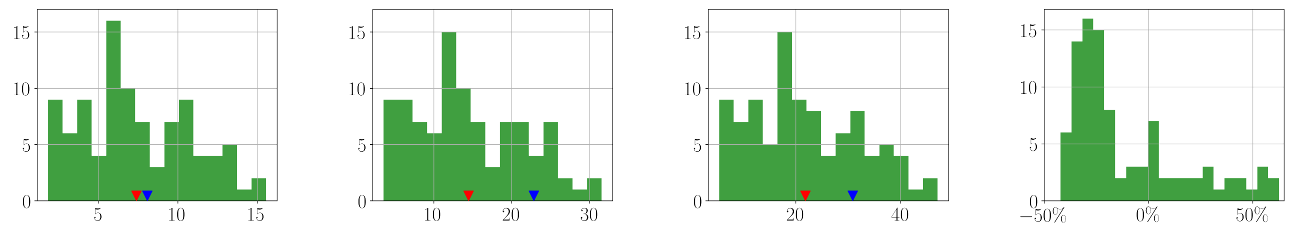
Algorithm 2: Channel-wise Averaged Quantizer (CAQ). $\mathcal{L}$ is a cost function for a mini-batch.

```
Inputs      : A mini-batch activation $\mathbf{A}$ with $C$ channels, Moving-average factor $\mu$, and number of iteration $T$.
Outputs     : The quantized activation $\mathbf{A}_q$.
Parameters: A set of quant. basis vectors $\{\mathbf{v}^{(j)}\}_{j=1}^C$.
1  if in the training stage, then
2      Forward propagation:
3      for $j = 1$ to $C$ do
4          Set $\mathbf{v}_0^{(j)} = \mathbf{v}^{(j)}$
5          for $t = 1$ to $T$ do
6              Given $\mathbf{v}_{t-1}^{(j)}$, compute $\mathbf{S}_t^{(j)}$ by looking up the index of the nearest quant. level;
7              Given $\mathbf{S}_t^{(j)}$, compute $\mathbf{v}_t^{(j)}$ by using Eq. 3.
8          $\mathbf{v}^{(j)} \leftarrow (1 - \mu)\mathbf{v}_T^{(j)} + \mu \mathbf{v}^{(j)}$
9      $\mathbf{v}^a = \frac{1}{C}\sum_{j=1}^C \mathbf{v}^{(j)}$
10     Compute $\mathbf{A}_q$ by looking up the nearest quantization level $\in \mathbf{v}^a$ for each element.
11     Backward propagation:
12     Given $\frac{\partial \mathcal{L}}{\partial \mathbf{A}_q}$ obtained using back-propagation;
13     $\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_q}$                    // STE
14 else // inference stage
15     $\mathbf{v}^a = \frac{1}{C}\sum_{j=1}^C \mathbf{v}^{(j)}$
16     Compute $\mathbf{A}_q$ by looking up the nearest quantization level $\in \mathbf{v}^a$ for each element.
```

by looking up the index of the nearest quantization level. Note that, for a K_a-bit quantization, there are 2^{K_a} quantization levels $\mathbf{q}^a = [q_i^a, \cdots, q_{2^{(K_a)}}^a] \in \mathbb{R}^{2^{(K_a)}}$ and $q_i^a = \langle \text{Dec2Bin}_{K_a}(i), \mathbf{v}^a \rangle$, where $\langle \cdot \rangle$ is a dot product and $\text{Dec2Bin}_{K_a}(i)$ is the function convert the decimal value i ($0 \leq i < 2^{K_a}$) to the equivalent K_a-bit binary vector, e.g., $\text{Dec2Bin}_3(6) = [0, 1, 1]$. ***(ii)*** Given $\mathbf{S}^a$ fixed, we have the closed-form solution for $\mathbf{v}^a$ as follows:

$$\mathbf{v}^a = (\mathbf{S}^{a\top}\mathbf{S}^a)^{-1}\mathbf{S}^{a\top}\mathbf{a}. \qquad (3)$$

Channel-Wise Averaged Quantization. The Batch Normalization layer [Ioffe and Szegedy, 2015] includes two steps: (i) normalizing and (ii) shifting and scaling. While the normalizing step tries to make each channel of $\mathbf{A}$ to have zero-mean and unit-variance, the scaling and shifting step makes the variances of channels different. The variances can be very large or very small, depending on the corresponding scaling factor of the i-th channel. Hence, when a single scalar quantizer is used for $\mathbf{A}$ as in previous works (e.g., LQ-Nets [Zhang *et al.*, 2018]) (called as baseline), few channels which have high variances can significantly affect the quantization basis values. Consequently, the resulting quantization basis vector potentially causes significant information loss in many channels with low variances, which cumulatively have noticeable impacts on outputs. Additionally, slightly less focus on large activation values might help to reduce overfit-



(a) 2-nd quant. level (b) 3-rd quant. level (c) 4-th quant. level (d) MSE relative changes

Figure 1: Fig. 1a, 1b, and 1c are respectively the distributions of the optimal 2-nd, 3-rd, and 4-th channel-wise quantization levels of the first activation layer of AlexNet with $K_a = 2$. ▼ indicates the proposed channel-wise averaged quantization levels - Mean Square Error (MSE) = 7.376 (using 100 random ImageNet images); and ▼ indicates the quantization levels when considering the whole activation as the input for a scalar quantizer (e.g., [Zhang *et al.*, 2018]) (called as baseline) - MSE = 6.664. Fig. 1d shows the histogram of the relative changes (%) in channel-wise MSE of our method compared to the baseline. We can observe that, in the baseline approach, the quantization levels tend to be biased toward large values. Although our proposed method results in higher quantization error overall (i.e., $7.376 > 6.664$), the majority of channels (68 out of 96 channels in Fig. 1d) achieve smaller quantization errors (the relative changes < 0) compared to the baseline.

ting. To address this problem, first, we propose to learn a scalar quantizer for each channel of activation $\mathbf{A}$. Then, the quantization basis values for the layer-wise quantizer can be obtained by taking the average of all quantization basis values of channel-wise quantizers, i.e., $v_i^a = \frac{1}{C}\sum_{j=1}^{C} v_i^{a(j)}$, where $v_i^{a(j)}$ is the i-th quantization basis value of j-th channel. With this proposed approach, the layer-wise quantization levels minimize the quantization errors on the majority of channels. Figure 1 provides an example to illustrate our proposed method. Besides, noticeably, only a mini-batch of input is available at each forward/backward step, we necessarily apply exponential moving average to update the channel-wise quantization level vectors to ensure stability. Algorithm 2, dubbed as Channel-wise Averaged Quantizer (CAQ), summarizes the procedure for training and inference of our proposed quantizer for activations. Note that during inference, only the channel-wise averaged quantizer is required. Hence, our method has the same time complexity as the baseline does.

4 Experiments

To evaluate the performance of the proposed methods, we conduct experiments on CIFAR-100 [Krizhevsky, 2009] and ImageNet (ILSVRC2012) [Deng *et al.*, 2009] datasets using the two representative network architectures, AlexNet [Krizhevsky *et al.*, 2012], ResNet [He *et al.*, 2016], and MobileNetV2 [Sandler *et al.*, 2018]. We adopt the standard data splits for both datasets, i.e., 50K training and 10K test images for CIFAR-100, about 1.2 million training and 50K test images for ImageNet. Additionally, we use a variant of AlexNet architecture by adding Batch Normalization layers after each convolutional layer and Fully-Connected layer and removing the Local Response layers as commonly used in recent works [Zhou *et al.*, 2016; Zhuang *et al.*, 2018].

4.1 Implementation Details

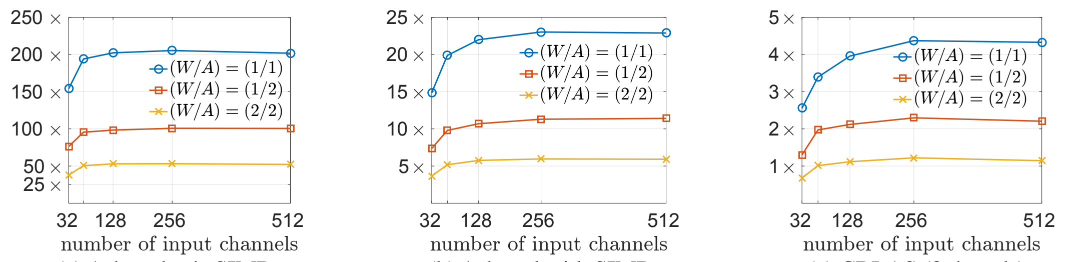
We follow the common settings in previous works [Zhou *et al.*, 2016; Zhang *et al.*, 2018]: We first resize the shorter side of the images to 256. During training, we randomly crop 227×227/224×224 patches for AlexNet/ResNet and MobileNetV2 from training images or their horizontal flips (called as basic augmentation). Following Zhang *et al.*, we adopt the basic augmentation in most of our experiments, except for the cases of ResNet on ImageNet dataset with $K_w/K_a = 2/2, 3/3$, where the augmentation strategy of the ResNet Torch implementation[1] is adopted. We report top-1 and top-5 classification accuracy using single-center crops of test images.

In all experiments, we use Nesterov SGD with a momentum of 0.9 and mini-batch size of 256. The starting learning rate is set at 0.02/0.1/0.02 for AlexNet/ResNet/MobileNetV2 respectively. We adopt the polynomial learning rate scheduler with the annealing power of 2 and the final learning rate of 10^{-6} after 120 epochs. Regarding the weight decay, for FP weights, we set weight decay as 5×10^{-4}/5×10^{-5}/10^{-5} for AlexNet/ResNet/MobileNetV2. For the quantized weights, the weight decay has a stronger effect; hence, we cross-validate within $\{10^{-6}, 5 \times 10^{-6}, 10^{-5}, 5 \times 10^{-5}\}$ using 1% of the training set as the validation set. We scale the learning rate of the binary weight encodings as suggested by Courbariaux *et al.*. We also scale the learning rate of quantization basis vectors by a factor of $\gamma_{\mathbf{v}} = \frac{1}{50}$ to avoid unstable training. The moving-average factor μ and the number of iteration T in CAQ algorithm are fixed as 0.9 and 1, respectively. Following previous works [Rastegari *et al.*, 2016; Zhou *et al.*, 2016; Zhang *et al.*, 2018], we do not quantize the first and last layers. Since the speedup, benefited by bitwise operations, is low due to their small input-channel number or filter size [Zhou *et al.*, 2016; Rastegari *et al.*, 2016]. All networks using our proposed method are trained from scratch.

4.2 Inference Efficiency Analysis

We first show that the inner products between our quantized weight and activation vectors can be efficiently computed by bitwise operations. Let a weight vector $\mathbf{W} \in \mathbb{R}^N$ be encoded by $\mathbf{S}^w = \{\mathbf{s}_1^w, \cdots, \mathbf{s}_{K_w}^w\} \in \{-1, 1\}^{N \times K_w}$ and the learned basis vector $\mathbf{v}^w = \{v_1^w, \cdots, v_{K_w}^w\} \in \mathbb{R}^{K_w}$. Similarly, we encode an activation vector $\mathbf{A} \in \mathbb{R}^N$ by $\mathbf{S}^a = \{\mathbf{s}_1^a, \cdots, \mathbf{s}_{K_a}^a\} \in \{0, 1\}^{N \times K_a}$ and $\mathbf{v}^a = \{v_1^a, \cdots, v_{K_a}^a\} \in \mathbb{R}^{K_a}$. To utilize

[1]https://github.com/facebook/fb.resnet.torch



(a) 1 thread w/o SIMD (b) 1 thread with SIMD (c) CBLAS (8 threads)

Figure 2: The speedup of one convolution by varying input channel size at different bit-widths.

the bitwise inner products [Zhou *et al.*, 2016]; first we need to convert $\mathbf{S}^a \in \{0,1\}^{N\times K_a}$ to $\hat{\mathbf{S}}^a \in \{-1,1\}^{N\times K_a}$ by $\mathbf{S}^a_i = 0.5\hat{\mathbf{S}}^a_i + 0.5$. Then, with some manipulation, we can derive that:

$$\mathcal{Q}(\mathbf{W},\mathbf{v}^w)^\top \mathcal{Q}(\mathbf{A},\mathbf{v}^a) = Q + \sum_{i=1}^{K_w}\sum_{j=1}^{K_a} \frac{v_i^w v_j^a}{2}\left(\mathbf{s}_i^w \odot \hat{\mathbf{s}}_j^a\right), \quad (4)$$

where $Q = \sum_{i=1}^{K_w}\sum_{j=1}^{K_a} \frac{v_i^w v_j^a}{2}(\mathbf{s}_i^w \odot \mathbf{1})$, $\mathbf{1}$ is a K_a-dimension vector with all elements are 1, and $\odot$ denotes the inner product with bitwise operations, *xnor* and *popcnt*. Noting that during inference, $\mathbf{S}^w$, $\mathbf{v}^w$, and $\mathbf{v}^a$ are fixed; so Q is a scalar constant and can be pre-computed. Similar to [Zhou *et al.*, 2016; Zhang *et al.*, 2018], the computation complexity of Eq. 4 w.r.t. K_w, K_a is $\mathcal{O}(K_w K_a)$, i.e., directly linear-proportional to the bitwidths of weights and activations.

Considering a matrix multiplication (MatMul) $\widetilde{\mathbf{C}} = \widetilde{\mathbf{W}}\widetilde{\mathbf{A}}$, where $\widetilde{\mathbf{W}} \in \mathbb{R}^{n\times p}$, $\widetilde{\mathbf{A}} \in \mathbb{R}^{p\times m}$ and $\widetilde{\mathbf{C}} \in \mathbb{R}^{n\times m}$. To calculate $\widetilde{\mathbf{C}}$ with FP $\widetilde{\mathbf{W}}, \widetilde{\mathbf{A}}$, there are nmq multiply-accumulate operations (MACs) required. Following Wan *et al.*, the theoretical speedup ratio for the bitwise MatMul of our method, which requires $K_w K_a mn$ MACs and $2(K_w K_a)mn$ binary operations, is given as follows:

$$S = \frac{\gamma q}{\gamma K_w K_a + 2(K_w K_a)\lceil \frac{q}{L}\rceil}, \quad (5)$$

where L is the number of bits binary operation in one clock cycle, and $\gamma = 1.91$ is ratio between the speed of performing a L-bit binary operation and a MAC [Wan *et al.*, 2018].

However, we found that Eq. 5 may not accurately reflect the actual speedup ratio of MatMul using bitwise operations in comparison with using FP. First, Eq. 5 over-simplifies the bitwise MatMul by only including *xnor* and *popcount* operations. In fact, additional operations are required, including *sum* to accumulate the *popcount* results, operations to convert from integer (*popcount* results) to FP numbers to multiply with quantization basis values. Second, Eq. 5 under-utilizes the CPU capability with FP computation. The modern CPU can perform 4 MACs in a single clock by using Single Instruction Multiple Data (SIMD) instructions. Multi-threading can also be utilized to further speed up computation by dividing the workload into multiple separate cores. Third, data transferring operations (e.g., loading, storing) is not considered. Using quantized weights and activations or SIMD can help to reduce the number of loading/storing operations. Finally, the additional time for quantizing activations should be considered in computing the speedup ratio.

FP		Bitwise	
W/o SIMD	With SIMD	2 / 2	1 / 2
3254±28	809±19	523±5	475±8

Table 1: The average ± std of inference time (ms) for 1 image using AlexNet trained on CIFAR-100 dataset under FP and bit-wise conv. implementations. The experiments was conducted on a CPU(i7-6700HQ@2.60GHz) with single-thread using 10 random images.

We provide a CPU implementation of the bitwise MatMul (Eq. 4) to measure its actual CPU speedup in comparison with FP MatMul. We fix $n = 256$ (i.e., the number of output channels C_o), $m = 14 \times 14 \times 100$ (i.e., an 100-sample mini-batch of 14×14 size), and 3×3 kernel. Noting that $p = 3 \times 3 \times C_i$, where C_i is the number of input channels. The experiment is conducted on a quad-core CPU (i7-6700HQ@2.60GHz) with $L = 64$. In Figure 2, we present the empirical speedup ratio of the bitwise MatMul in compared with the FP MatMul (with different implementations of FP MatMul) as the number of input channels C_i varies.

Firstly, from Eq. 5, we can compute the theoretical speedup for $K_w/K_a = 1/1$ is $\sim 60\times$. However, our CPU implementation of bitwise MatMul, with all overheads including quantization and memory allocation, can achieve up to $\sim 200\times$ speedup (Figure 2a) (when SIMD is not utilized for FP MatMul). When SIMD is utilized for FP MatMul, bitwise MatMul can achieve about $\sim 23\times$ speedup for $K_w/K_a = 1/1$ (Figure 2b). Additionally, when multi-threading (8 threads) is also considered for both FP (CBLAS library) and bitwise MatMul, bitwise MatMul can gain $\sim 4.3\times$ speedup for $K_w/K_a = 1/1$. While for $K_w/K_a = 2/2$, we obverse only a small speedup for bitwise MatMul (for $C_i > 64$) (Figure 2c). Furthermore, the empirical speedup ratios confirm that the complexity of our bitwise MatMul is proportional to the bit-widths of weights and activations.

Additionally, we presented in Table 1 the inference time per image using bitwise convolution for the quantized AlexNet (for both weight and activation) models. We also include the inference time using FP convolution operation for comparison. We can see that when using AlexNet with $K_w/K_a = 1/2$, our proposed method with bitwise convolution can speed up $\times 1.7$ and $\times 6.8$ in comparison with FP convolution with and without using SIMD, respectively.

4.3 Ablation Analysis

In this section, we first conduct experiments to analyze the effect of our proposals, i.e., ***(i)*** directly learning the quantized weights (LQW) and ***(ii)*** the channel-wise average quantizer

Weight	Activation	Top-1
Baseline	Baseline	69.2
Baseline	**CAQ**	69.6
LQW	Baseline	69.7
LQW	**CAQ**	69.9

Table 2: Top-1 accuracy (%) of AlexNet on CIFAR-100 dataset using different methods (LQW, CAQ, and baseline [Zhang *et al.*, 2018]) with $K_w/K_a = 2/2$.

Bit-widths	Top-1 Accuracy (% - mean±std)		
(K^w/K^a)	$T=1$	$T=2$	$T=3$
1 / 2	69.42±0.14	69.59±0.16	69.55±0.12
2 / 2	69.88±0.11	70.03±0.13	69.82±0.09

Table 3: Accuracy w.r.t. numbers of CAQ iteration T

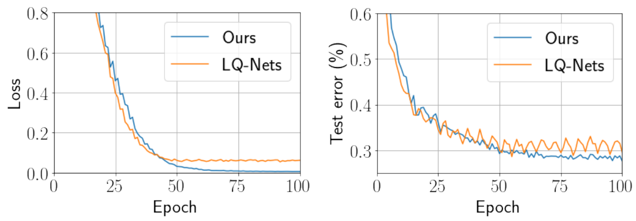
(CAQ) for activations. For easy comparison, we consider the learned quantizer proposed by Zhang *et al.*, which is used for both weights and activations, as the baseline. Table 2 shows the classification accuracy of AlexNet on CIFAR-100 with different combinations of baseline method and our proposals for $K_w/K_a = 2/2$. The experimental results demonstrate the effectiveness of our proposals, i.e., LQW and CAQ, in training networks with low bit-width weights and activations. Additionally, LQW helps achieve a slightly higher improvement gain compared to CAQ (+0.1%).

We additionally conduct an experiment on CIFAR-100 using AlexNet with different numbers of CAQ iteration T. We repeat the experiment 5 times and report the results (mean±std) in Table 3. We observer that using $T = 2, 3$ does not result in any significant difference on Top-1 accuracy compared to $T = 1$. This is mainly because, for each mini-batch, CAQ starts from proper initial quantization basis values, which are the good results of the last mini-batch.

4.4 Comparison With State of The Art

In this section, we compare the performance of our proposed method with existing methods including XNOR-Net [Rastegari *et al.*, 2016], DoReFa-Net [Zhou *et al.*, 2016], HWQG [Cai *et al.*, 2017], PQ+TS+Guided [Zhuang *et al.*, 2018], TBN [Wan *et al.*, 2018], LQ-Nets [Zhang *et al.*, 2018], Bi-Real Net [Liu *et al.*, 2018], QIL [Jung *et al.*, 2019], DSQ [Gong *et al.*, 2019], BONN [Gu *et al.*, 2019], RCBN [Liu *et al.*, 2019], and Group-Net [Zhuang *et al.*, 2019]. Noting that, our method is generic and can be used with any bit-width and architecture.

Comparison on CIFAR-100. Table 4 presents the top-1/top-5 classification accuracy on CIFAR-100 dataset of different network quantization methods on AlexNet, ResNet-18, and MobileNetV2. In all compared settings for AlexNet and ResNet-18, our method consistently outperforms the state-of-the-art method LQ-Nets by large margins, i.e., $\geq$ 0.7% for $K_w/K_a = 1/2$ and $\geq$ 0.6% for $K_w/K_a = 2/2$ in term of top-1 accuracy. Regarding the efficient architectures MobileNetV2, at bit-widths of $K_w/K_a = 3/3$, there is only a small accuracy drop compared with FP model. Even at lower bit-width $K_w/K_a = 2/2$, $K_w/K_a = 1/2$, our proposed method still achieves good performance for MobileNetV2, which is already very efficient. This confirms the effectiveness of our proposed method. Furthermore, we can observe



(a) Cross-entropy loss (b) Test error

Figure 3: The cross-entropy loss (a) and test error (b) ResNet-18 on CIFAR-100 dataset with $K_w/K_a = 1/2$.

from Figure 3a that the training loss on CIFAR-100 of LQ-Nets with ResNet-18 architecture is unable to approach 0 as our proposed method does. LQ-Nets also suffers from unstable test error (Figure 3b). This result is potentially due to the mismatch in the training process of LQ-Nets, as the gradient descent does not directly update the quantized weights. Consequently, the quantized weights are unable to obtain the desirable changes. This demonstrates the necessity of direct updating the quantized weights using gradients as our proposed method can do.

Comparison on ImageNet. The classification results on ImageNet dataset for AlexNet and ResNet-18/34 are presented in Table 5. Our proposed method (LQW + CAQ) outperforms the compared methods, e.g., DoReFa-Net, TBN, HORQ, HWGQ, DSQ, in the majority of settings for both AlexNet and ResNet-18/34. Our method outperforms LQ-Nets in term of Top-1 accuracy by clear margins in the majority of experiments (i.e., $\geq$ 0.4%); except for ResNet-34 with $K_w/K_a = 1/2$, where our method achieves comparable performance with LQ-Nets. For $K_w/K_a = 1/1$, our method gains 0.5% and 0.2% Top-1 improvement over Bi-Real Net for ResNet-18 and ResNet-34 respectively. Our method still under-performs BONN and RBCN for ResNet-18. However, BONN is designed specifically for 1-bit CNN, it is nontrivial to adopt this method to higher bit-widths for better trade-off between the computational speed and accuracy, as can be done easily in our proposed method. Besides, in BONN, in addition to quantization loss and the essential cross-entropy loss, BONN also has an additional feature loss to enhance the intra-class compactness which improves the classification accuracy. For our work, because we aimed to learn the quantized network without any modification in the network architecture, in addition to cross-entropy, we only introduce the quantization loss. Regarding RBCN, this method requires joint training of a FP model, a quantized model, and discriminators for distillation from the FP model to the a quantized model. Consequently, this method requires significantly high computational cost and memory. In comparison to QIL, our method achieves comparable performance for $K_w/K_a = 2/2$. Specifically, our method has 0.3% Top-1 accuracy lower than QIL for AlexNet, while achieves 0.3% higher than QIL for ResNet-18. For ResNet-34, the accuracy of our method is marginally lower than QIL's (i.e., -0.1% top-1 accuracy). We note that QIL usually achieves the best performance with trainable but complex quantizer functions. It is unsure if the computational complexity of QIL during inference can be linearly proportional to the bit-widths of weights

Methods	Bit-widths (K_w/K_a)	AlexNet		ResNet-18		MobileNetV2	
		Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Full-precision (FP)		71.2	91.3	74.0	92.7	72.9	92.1
LQ-Nets	3 / 3	70.9	91.3	72.9	92.2	-	-
LQW + CAQ	3 / 3	**71.3**	**91.3**	**73.0**	**92.3**	**71.5**	**91.6**
PQ+TS+Guided	2 / 2	64.6	87.8	-	-	-	-
DoReFa-Net	2 / 2	63.4	87.5	65.1	88.0	56.8	85.2
LQ-Nets	2 / 2	69.2	91.2	70.8	91.3	-	-
LQW + CAQ	2 / 2	**69.9**	**91.3**	**72.1**	**92.3**	**65.5**	**89.6**
LQ-Nets	1 / 2	68.7	90.5	70.4	91.2	-	-
LQW + CAQ	1 / 2	**69.3**	**91.2**	**72.1**	**91.6**	**63.0**	**88.27**

"-" indicates that the results are not provided.

Table 4: Top-1/Top-5 accuracy (%) of AlexNet, ResNet-18, and MobileNetV2 on CIFAR-100 dataset under different bit-widths.

Methods	Bit-widths (K_w/K_a)	AlexNet		ResNet-18		ResNet-34	
		Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Full-precision (FP)		61.8	83.5	70.3	89.5	73.8	91.4
XNOR-Net	1 / 1	44.2	69.2	51.2	73.2	-	-
Bi-Real Net	1 / 1	-	-	56.4	79.5	62.2	83.9
BONN	1 / 1	-	-	59.3	**81.6**	-	-
RBCN	1 / 1	-	-	**59.5**	**81.6**	-	-
LQW + CAQ	1 / 1	**48.3**	**72.7**	56.9	79.8	**62.4**	**84.0**
DoReFa-Net	1 / 2	49.8	-	53.4	-	-	-
TBN	1 / 2	49.7	74.2	55.6	79.0	58.2	81.0
LQ-Nets	1 / 2	55.7	78.8	62.6	84.3	**66.6**	86.9
LQW + CAQ	1 / 2	**56.4**	**79.3**	**63.1**	**84.7**	66.5	**87.0**
DoReFa-Net	2 / 2	48.3	71.6	57.6	80.8	-	-
HWGQ	2 / 2	52.7	76.3	59.6	82.2	-	-
LQ-Nets	2 / 2	57.4	80.1	64.9	85.9	69.8	89.1
DSQ	2 / 2	-	-	65.2	-	70.0	-
QIL	2 / 2	**58.1**	-	65.7	-	**70.6**	-
LQW + CAQ	2 / 2	57.8	**80.4**	**66.0**	**86.2**	70.5	**89.4**
Group-Net	4 bases‡	-	-	66.3	86.6	-	-

‡ The complexity of 4-base Group-Net for a convolution is comparable to ours at $K_w/K_a = 2/2$.
"-" indicates that the results are not provided.

Table 5: Top-1/Top-5 classification accuracy (%) of AlexNet and ResNet-18/34 on ImageNet dataset under different bit-widths.

and activations (no efficiency analysis is provided by Jung *et al.*). In our work, we carefully design the method to ensure the complexity linearly proportional to the bit-widths. Additionally, in comparison with Group-Net (4 bases) on ResNet-18, our method has 0.3% lower in top-1 accuracy. Nevertheless, Group-Net requires additional computational costs for the skip-connection in every binary convolution and for computing the soft connection between binary groups.

5 Conclusion

In this paper, we have presented a novel method to train CNN with low bit-width weights and activations. Firstly, we proposed to directly learn the quantized weights with learnable quantization levels using gradient descent, instead of learning to quantize full-precision learned weights as proposed in existing works. Secondly, by considering the quantization errors of all activation channels, our proposed quantizer for activations minimizes the information loss in the majority of channels, instead of being biased toward a few high-variance channels. The experiment results demonstrate the effectiveness of our proposed method in training low bit-width CNN to achieve the competitive classification accuracy for various network architectures.

References

[Aojun *et al.*, 2017] Zhou Aojun, Yao Anbang, Guo Yiwen, Xu Lin, and Chen Yurong. Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights. In *ICLR*, 2017.

[Cai *et al.*, 2017] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep Learning with Low Precision by Half-wave Gaussian Quantization. In *CVPR*, pages 5918–5926, 2017.

[Courbariaux *et al.*, 2015] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In *NIPS*, pages 3123–3131. 2015.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, pages 248–255, 2009.

[Gong *et al.*, 2019] 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 *ICCV*, pages 4852–4861, 2019.

[Gu *et al.*, 2019] Jiaxin Gu, Junhe Zhao, Xiaolong Jiang, Baochang Zhang, Liu Jianzhuang, Guodong Guo, and Rongrong Ji. Bayesian Optimized 1-bit CNNs. In *ICCV*, pages 4909–4917, 2019.

[Gupta *et al.*, 2015] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep Learning with Limited Numerical Precision. In *ICML*, page 1737–1746, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.

[Hinton, 2012] Geoffreye Hinton. Neural networks for machine learning. *Coursera, video lectures*, 2012.

[Hou and Kwok, 2018] Lu Hou and James T. Kwok. Loss-aware weight quantization of deep networks. In *ICLR*, 2018.

[Hubara *et al.*, 2017] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *JMLR*, 18(1):6869–6898, 2017.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, page 448–456, 2015.

[Jung *et al.*, 2019] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *CVPR*, pages 4350–4359, 2019.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

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

[Li and Liu, 2017] Fengfu Li and Bin Liu. Ternary weight networks. In *ICLR*, 2017.

[Li *et al.*, 2017] Zefan Li, Bingbing Ni, Wenjun Zhang, Xiaokang Yang, and Wen Gao. Performance Guaranteed Network Acceleration via High-Order Residual Quantization. In *ICCV*, pages 2584–2592, 2017.

[Lin *et al.*, 2016] Darryl D. Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed Point Quantization of Deep Convolutional Networks. In *ICML*, page 2849–2858, 2016.

[Liu *et al.*, 2018] 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 *ECCV*, pages 747–763, 2018.

[Liu *et al.*, 2019] Chunlei Liu, Wenrui Ding, Xin Xia, Yuan Hu, Baochang Zhang, Jianzhuang Liu, Bohan Zhuang, and Guodong Guo. Rectified Binary Convolutional Networks for Enhancing the Performance of 1-bit DCNNs. In *IJCAI*, pages 854–860, 2019.

[Miyashita *et al.*, 2016] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional Neural Networks using Logarithmic Data Representation. *CoRR*, abs/1603.01025, 2016.

[Park *et al.*, 2017] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In *CVPR*, pages 5456–5464, 2017.

[Rastegari *et al.*, 2016] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV*, pages 525–542, 2016.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[Tang *et al.*, 2017] Wei Tang, Gang Hua, and Liang Wang. How to Train a Compact Binary Neural Network with High Accuracy? In *AAAI*, pages 2625–2631, 2017.

[Wan *et al.*, 2018] 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 *ECCV*, pages 322–339, 2018.

[Zhang *et al.*, 2018] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks. In *ECCV*, pages 373–390, 2018.

[Zhou *et al.*, 2016] 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. *CoRR*, abs/1606.06160, 2016.

[Zhou *et al.*, 2018] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. In *AAAI*, pages 4596–4604, 2018.

[Zhu *et al.*, 2017] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained Ternary Quantization. In *ICLR*, 2017.

[Zhuang *et al.*, 2018] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian D. Reid. Towards Effective Low-Bitwidth Convolutional Neural Networks. In *CVPR*, pages 7920–7928, 2018.

[Zhuang *et al.*, 2019] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian D. Reid. Structured Binary Neural Networks for Accurate Image Classification and Semantic Segmentation. In *CVPR*, pages 413–422, 2019.