

# Learning to Quantize Deep Networks by Optimizing Quantization Intervals with Task Loss

Sangil Jung<sup>1\*</sup> Changyong Son<sup>1\*</sup> Seohyung Lee<sup>1</sup> Jinwoo Son<sup>1</sup> Jae-Joon Han<sup>1</sup>  
 Youngjun Kwak<sup>1</sup> Sung Ju Hwang<sup>2</sup> Changkyu Choi<sup>1</sup>

<sup>1</sup>Samsung Advanced Institute of Technology (SAIT), South Korea

<sup>2</sup>Korea Advanced Institute of Science and Technology (KAIST), South Korea

## Abstract

*Reducing bit-widths of activations and weights of deep networks makes it efficient to compute and store them in memory, which is crucial in their deployments to resource-limited devices, such as mobile phones. However, decreasing bit-widths with quantization generally yields drastically degraded accuracy. To tackle this problem, we propose to **learn to quantize** activations and weights via a trainable quantizer that transforms and discretizes them. Specifically, we parameterize the quantization intervals and obtain their optimal values by directly minimizing the task loss of the network. This quantization-interval-learning (QIL) allows the quantized networks to maintain the accuracy of the full-precision (32-bit) networks with bit-width as low as 4-bit and minimize the accuracy degeneration with further bit-width reduction (i.e., 3 and 2-bit). Moreover, our quantizer can be trained on a heterogeneous dataset, and thus can be used to quantize pretrained networks without access to their training data. We demonstrate the effectiveness of our trainable quantizer on ImageNet dataset with various network architectures such as ResNet-18, -34 and AlexNet, on which it outperforms existing methods to achieve the state-of-the-art accuracy.*

## 1. Introduction

Increasing the depth and width of a convolutional neural network generally improves its accuracy [24, 8] in exchange for the increased memory and the computational cost. Such a memory- and computation- heavy network is difficult to be deployed to resource-limited devices such as mobile phones. Thus, many prior work have sought various means to reduce the model size and computational cost, including the use of separable filters [13, 10, 21], weight pruning [7] and bit-width reduction of weights or activations

[27, 22, 4, 26, 25, 30]. Our work aims to reduce bit-widths of deep networks both for weights and activations, while preserving the accuracy of the full-precision networks.

Reducing bit-width inherently includes a quantization process which maps continuous real values to discrete integers. Decrease in bit-width of deep networks naturally increases the quantization error, which in turn causes accuracy degeneration, and to preserve the accuracy of a full-precision network, we need to reduce the quantization error. For example, Cai *et al.* [3] optimize the activation quantizer by minimizing mean-squared quantization error using Lloyd’s algorithm with the assumption of half-wave Gaussian distribution of the response map, and some other work approximate the layerwise convolutional outputs [22, 16]. However, while such quantization approaches may accurately approximate the original distribution of the weights and activations, there is no guarantee that they will be beneficial toward suppressing the prediction error from increasing. To overcome this limitation, our trainable quantizer approximates neither the weight/activation values nor layerwise convolutional outputs. Instead, we quantize the weights and activations for each layer by directly minimizing the *task loss* of the networks, which helps it to preserve the accuracy of the full-precision counterpart.

Our quantizer can be viewed as a composition of a *transformer* and a *discretizer*. The transformer is a (non-linear) function from the unbounded real values to the normalized real values (i.e.,  $\mathbb{R}_{(-\infty, \infty)} \rightarrow \mathbb{R}_{[-1, 1]}$ ), and the discretizer maps the normalized values to integers (i.e.,  $\mathbb{R}_{[-1, 1]} \rightarrow \mathbb{I}$ ). Here, we propose to parameterize quantization intervals for the transformer, which allows our quantizer to focus on appropriate interval for quantization by pruning (less important) small values and clipping (rarely appeared) large values.

Since reducing bit-widths decreases the number of discrete values, this will generally cause the quantization error to increase. To maintain or increase the quantization resolution while reducing the bit-width, the interval for quanti-

\*These two authors contributed equally.

zation should be compact. On the other hand, if the quantization interval is too compact, it may remove valid values outside the interval which can degrade the performance of the network. Thus, the quantization interval should be selected as compact as possible according to the given bit-width, such that the values influencing the network accuracy are within the interval. Our trainable quantizer adaptively finds the optimal intervals for quantization that minimize the task loss.

Note that our quantizer is applied to both weights and activations for each layer. Thus, the convolution operation can be computed efficiently by utilizing bit-wise operations which is composed of logical operations (i.e., 'AND' or 'XNOR') and bitcount [27] if the bit-widths of weights and activations become low enough. The weight and activation quantizers are jointly trained with full-precision weights. Note that the weight quantizers and full-precision weights are kept and updated only in the training stage; at the inference time, we drop them and use only the quantized weights.

We demonstrate our method on the ImageNet classification dataset [23] with various network architectures such as ResNet-18, -34 and AlexNet. Compared to the existing methods on weight and activation quantization, our method achieves significantly higher accuracy, achieving the state-of-the-art results. Our quantizers are trained in end-to-end fashion without any layerwise optimization [25].

In summary, our contributions are threefold:

- We propose a trainable quantizer with *parameterized intervals* for quantization, which simultaneously performs both *pruning* and *clipping*.
- We apply our trainable quantizer to both the weights and activations of a deep network, and optimize it along with the target network weights for the *task-specific loss* in an end-to-end manner.
- We experimentally show that our quantizer achieves the state-of-the-art classification accuracies on ImageNet with extremely *low bit-width* (2, 3, and 4-bit) networks, and achieves high performance even when trained with a *heterogeneous* dataset and applied to a pretrained network.

## 2. Related Work

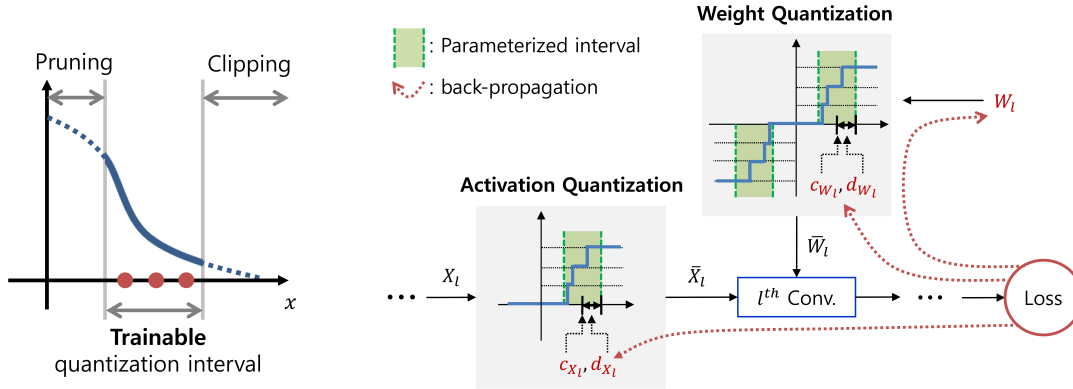
Our quantization method aims to obtain low-precision networks. Low-precision networks have two benefits: model compression and operation acceleration. Some work compresses the network by reducing bit-width of model weights, such as BinaryConnect (BC) [5], Binary-Weight-Network (BWN) [22], Ternary-Weight-Network (TWN) [16] and Trained-Ternary-Quantization (TTQ) [29]. BC

uses either deterministic or stochastic methods for binarizing the weights which are  $\{-1, +1\}$ . BWN approximates the full-precision weights as the scaled bipolar( $\{-1, +1\}$ ) weights, and finds the scale in a closed form solution. TWN utilizes the ternary( $\{-1, 0, +1\}$ ) weights with scaling factor which is trained in training phase. TTQ uses the same quantization method but the different scaling factors are learned on the positive and negative sides. These methods solely considers quantization of network weights, mostly for binary or ternary cases only, and do not consider quantization of the activations.

In order to maximally utilize the bit-wise operations for convolution, we should quantize both weights and activations. Binarized-Neural-Network(BNN) [11] binarizes the weights and activations to  $\{-1, +1\}$  in the same way as BC, and uses these binarized values for computing gradients. XNOR-Net [22] further conducts activation binarization with a scaling factor where the scaling factor is obtained in a closed form solution. DoReFa-Net [27] performs a bit-wise operation for convolution by quantizing both weights and activations with multiple-bits rather than performing bipolar quantization. They adopt various activation functions to bound the activation values. The weights are transformed by the hyperbolic tangent function and then normalized with respect to the maximum values before quantization. Half-Wave-Gaussian-Quantization (HWGQ) [3] exploits the statistics of activations and proposes variants of ReLU activation which constrain the unbounded values. Both DoReFa-Net and HWGQ use upper bounds for the activation but they are fixed during training and their quantizations are not learned as done in our model.

Several work [26, 25, 30, 4] proposed highly accurate low bit-width models by considering both weight and activation quantization. LQ-Nets [26] allows floating-point values to represent the basis of  $K$ -bit quantized value instead of the standard basis  $[1, 2, \dots, 2^{K-1}]$ , and learn the basis for weights and activations of each layer or channel by minimizing the quantization error. On the other hand, our trainable quantizer estimates the optimal quantization interval, which is learned in terms of minimizing the output task loss rather than minimizing the quantization error. Furthermore, the LQ-Nets has to use a few floating-point multiplication for computing convolution due to the floating-point basis, but our convolution can use shift operation instead of multiplication because all of our quantized values are integers.

Wang et al. [25] proposes a two-step quantization (TSQ) that decomposes activation and weight quantization steps, which achieves good performance with AlexNet and VGGNet architectures. However, TSQ adopts layerwise optimization for the weight quantization step, and thus is not applicable to ResNet architecture which includes skip connections. Contrarily, our quantizer is applicable to any types of network architectures regardless of skip connec-



(a) Quantization Interval

(b) A convolutional layer of our low bit-width network

Figure 1. Illustration of our trainable quantizer. (a) Our trainable quantization interval, which performs pruning and clipping simultaneously. (b) The  $l^{th}$  convolution layer of our low-precision network. Given bit-width, the quantized weights  $\bar{W}_l$  and activations  $\bar{X}_l$  are acquired using the parameterized intervals. The interval parameters  $(c_{w_l}, d_{w_l}, c_{x_l}, d_{x_l})$  are trained jointly with the full-precision weights  $W_l$  during backpropagation.

tion. Zhuang *et al.* [30] propose a two-stage and progressive optimization method to obtain good initialization to avoid the network to get trapped in a poor local minima. We adopt their progressive strategy, which actually improves the accuracy especially for extremely low bit-width network, i.e., 2-bit. PACT [4] proposes a parameterized clipping activation function where the clipping parameter is obtained during training. However, it does not consider pruning and the weight quantization is fixed as in DoReFa-Net. A couple of recent quantization methods use Bayesian approaches; Louizos *et al.* [18] propose Bayesian compression, which determines optimal number of bit precision per layer via the variance of the estimated posterior, but does not cluster weights. Achterhold *et al.* [1] propose a variational inference framework to learn networks that quantize well, using multi-modal quantizing priors with peaks at quantization target values. Yet neither approaches prune activations or learn the quantizer itself as done in our work.

We propose the trainable quantization interval which performs pruning and clipping simultaneously during training, and by applying this parameterization both for weight and activation quantization we keep the classification accuracy on par with the one of the full-precision network while significantly reducing the bit-width.

### 3. Method

The trainable quantization interval has quantization range within the interval, and prunes and clips ranges out of the interval. We apply the trainable quantization intervals to both activation and weight quantization and optimize them with respect to the task loss (Fig. 1). In this section, we first review and interpret the quantization process for low bit-width networks, and then present our trainable quantizer with quantization intervals.

### 3.1. Quantization in low bit-width network

For the  $l$ -th layer of a full-precision convolutional neural network (CNN), the weight  $\mathbf{W}_l$  is convolved with the input activation  $\mathbf{X}_l$  where  $\mathbf{W}_l$  and  $\mathbf{X}_l$  are real-valued tensors. We denote the elements of  $\mathbf{W}_l$  and  $\mathbf{X}_l$  by  $w_l$  and  $x_l$ , respectively. For notational simplicity, we drop the subscript  $l$  in the following. Reducing bit-widths inherently involves a quantization process, where we obtain the quantized weight  $\bar{w} \in \bar{\mathbf{W}}$  and the quantized input activation  $\bar{x} \in \bar{\mathbf{X}}$  via quantizers,

$$\begin{aligned} Q_W &: w \xrightarrow{T_W} \hat{w} \xrightarrow{D} \bar{w} \\ Q_X &: x \xrightarrow{T_X} \hat{x} \xrightarrow{D} \bar{x}. \end{aligned} \quad (1)$$

A quantizer  $Q_\Delta (\Delta \in \{\mathbf{W}, \mathbf{X}\})$  is a composition of a transformer  $T_\Delta$  and a discretizer  $D$ . The transformer maps the weight and activation values to  $[-1, 1]$  or  $[0, 1]$ . The simplest example of the transformer is the normalization that divides the values by their maximum absolute values. Another example is a function of  $\tanh(\cdot)$  for weights and clipping for activations [27]. The discretizer maps a real-value  $\hat{v}$  in the range  $[-1, 1]$  (or  $[0, 1]$ ) to some discrete value  $\bar{v}$  as follows:

$$\bar{v} = \frac{\lceil \hat{v} \cdot q_D \rceil}{q_D} \quad (2)$$

where  $\lceil \cdot \rceil$  is the round operation and  $q_D$  is the discretization level.

In this paper, we parameterize the quantizer (transformer) to make it trainable rather than fixed. Thus the quantizers can be jointly optimized together with the neural network model weights. We can obtain the optimal  $\bar{\mathbf{W}}$  and  $\bar{\mathbf{X}}$  that directly minimize the task loss (i.e., classification loss) of the entire network (Fig. 1 (b)) rather than simply approximating the full-precision weight/activation ( $\mathbf{W} \approx \bar{\mathbf{W}}, \mathbf{X} \approx \bar{\mathbf{X}}$ ) [16, 3] or the convolutional outputs

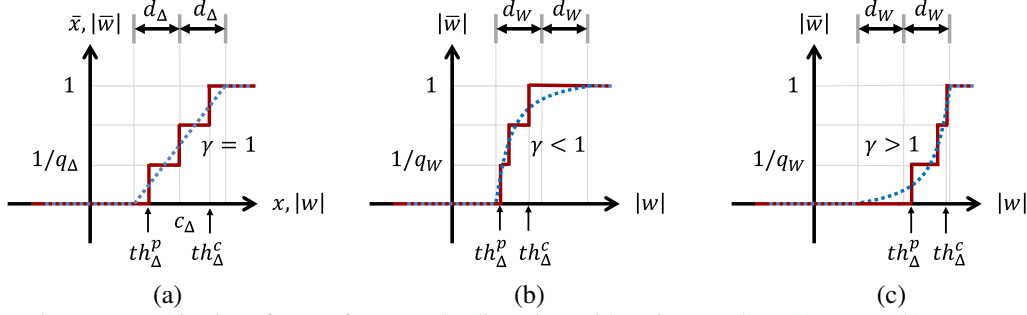


Figure 2. A quantizer as a combination of a transformer and a discretizer with various  $\gamma$  where (a)  $\gamma = 1$ , (b)  $\gamma < 1$ , and (c)  $\gamma > 1$ . The blue dotted lines indicate the transformers, and the red solid lines are their corresponding quantizers. The  $th_{\Delta}^p$  and the  $th_{\Delta}^c$  represent the pruning and clipping thresholds, respectively.

$(\mathbf{W} * \mathbf{X} \approx \bar{\mathbf{W}} * \bar{\mathbf{X}})$  [22, 25] where  $*$  denotes convolution operation.

### 3.2. Trainable quantization interval

To design the quantizers, we consider two operations: clipping and pruning (Fig. 1 (a)). The underlying idea of clipping is to limit the upper bound for quantization [27, 4]. Decreasing the upper bound increases the quantization resolution within the bound so that the accuracy of the low bit-width network can increase. On the other hand, if the upper bound is set too low, accuracy may decrease because too many values will be clipped. Thus, setting a proper clipping threshold is crucial for maintaining the performance of the networks. Pruning removes low-valued weight parameters [7]. Increasing pruning threshold helps to increase the quantization resolution and reduce the model complexity, while setting pruning threshold too high can cause the performance degradation due to the same reason as the clipping scheme does.

We define the quantization interval to consider both pruning and clipping. To estimate the interval automatically, the intervals are parameterized by  $c_{\Delta}$  and  $d_{\Delta}$  ( $\Delta \in \{\mathbf{W}, \mathbf{X}\}$ ) for each layer where  $c_{\Delta}$  and  $d_{\Delta}$  indicate the center of the interval and the distance from the center, respectively. Note that this is simply a design choice and other types of parameterization, such as parameterization with lower and upper bound, are also possible.

Let us first consider the weight quantization. Because weights contain both positive and negative values, the quantizer has symmetry on both positive and negative sides. Given the interval parameters  $c_W$  and  $d_W$ , we define the transformer  $T_W$  as follows:

$$\hat{w} = \begin{cases} 0 & |w| < c_W - d_W \\ \text{sign}(w) & |w| > c_W + d_W \\ (\alpha_W |w| + \beta_W)^{\gamma} \cdot \text{sign}(w) & \text{otherwise,} \end{cases} \quad (3)$$

where  $\alpha_W = 0.5/d_W$ ,  $\beta_W = -0.5c_W/d_W + 0.5$  and the  $\gamma$  is another trainable parameter of the transformer. That is,

the quantizer is designed by the interval parameters  $c_W$ ,  $d_W$  and  $\gamma$  which are trainable. The non-linear function with  $\gamma$  considers the distribution inside the interval. The graphs in Fig. 2 show the transformers (dotted blue lines) and their corresponding quantizers (solid red lines) with various  $\gamma$ . If  $\gamma = 1$ , then the transformer is a piecewise linear function where the inside values of the interval are uniformly quantized (Fig. 2 (a)). The inside values can be non-uniformly quantized by adjusting  $\gamma$  (Fig. 2 (b, c)).  $\gamma$  could be either set to a fixed value or trained. We demonstrate the effects of  $\gamma$  in the experiment. If  $\gamma \neq 1$ , the function is complex to be calculated. However, the weight quantizers are removed after training and we use only the quantized weights for inference. For this reason, this complex non-linear function does not decrease the inference speed at all. The actual pruning threshold  $th_{\Delta}^p$  and clipping threshold  $th_{\Delta}^c$  vary according to the parameters  $c_W$ ,  $d_W$  and  $\gamma$  as shown in Fig. 2. For example, the  $th_{\Delta}^p$  and  $th_{\Delta}^c$  in the case of  $\gamma = 1$  are derived as follows:

$$\begin{aligned} th_{\Delta}^p &= c_{\Delta} - d_{\Delta} + 0.5d_{\Delta}/q_{\Delta} \\ th_{\Delta}^c &= c_{\Delta} + d_{\Delta} - 0.5d_{\Delta}/q_{\Delta}. \end{aligned} \quad (4)$$

Note that the number of quantization levels  $q_W$  for the weight can be computed as  $q_W = 2^{N_W-1} - 1$  (one-side, except 0), given the bit-width  $N_W$ . Thus, the 2-bit weights are actually ternary  $\{-1, 0, 1\}$ .

The activations fed into a convolutional layer are non-negative due to ReLU operation. For activation quantization, a value larger than  $c_X + d_X$  is clipped and mapped to 1 and we prune a value smaller than  $c_X - d_X$  to 0. The values in between are linearly mapped to  $[0, 1]$ , which means that the values are uniformly quantized in the quantization interval. Unlike the weight quantization, activation quantization should be conducted on-line during inference, thus we fix the  $\gamma$  to 1 for fast computation. Then, the transformer  $T_X$  for activation is defined as follows (Fig. 2 (a)):

$$\hat{x} = \begin{cases} 0 & x < c_X - d_X \\ 1 & x > c_X + d_X \\ \alpha_X x + \beta_X & \text{otherwise,} \end{cases} \quad (5)$$

---

**Algorithm 1** Training low bit-width network using parameterized quantizers

---

**Input:** Training data

**Output:** A low bit-width model with quantized weights  $\{\bar{w}_l\}_{l=1}^L$  and activation quantizers  $\{c_{X_l}, d_{X_l}\}_{l=1}^L$

```

1: procedure TRAINING
2:   Initialize the parameter set  $\{P_l\}_{l=1}^L$  where  $P_l = \{w_l, c_{W_l}, d_{W_l}, \gamma_l, c_{X_l}, d_{X_l}\}$ 
3:   for  $l = 1, \dots, L$  do
4:     Compute  $\bar{w}_l$  from  $w_l$  using Eq. 3 and Eq. 2
5:     Compute  $\bar{x}_l$  from  $x_l$  using Eq. 5 and Eq. 2
6:     Compute  $\bar{w}_l * \bar{x}_l$ 
       Compute the loss  $\ell$ 
7:     Compute the gradient w.r.t. the output  $\partial\ell/\partial x_{L+1}$ 
8:     for  $l = L, \dots, 1$  do
9:       Given  $\partial\ell/\partial x_{l+1}$ ,
10:      Compute the gradient of the parameters in  $P_l$ 
11:      Update the parameters in  $P_l$ 
12:      Compute  $\partial\ell/\partial x_l$ 
13: procedure DEPLOYMENT
14:   for  $l = 1, \dots, L$  do
15:     Compute  $\bar{w}_l$  from  $w_l$  using Eq. 3 and Eq. 2
16:   Deploy the low bit-width model  $\{\bar{w}_l, c_{X_l}, d_{X_l}\}_{l=1}^L$ 

```

---

where  $\alpha_X = 0.5/d_X$  and  $\beta_X = -0.5c_X/d_X + 0.5$ . Given the bit-width  $N_X$  of activation, the number of quantization levels  $q_X$  (except 0) can be computed as  $q_X = 2^{N_X} - 1$ ; i.e., the quantized values are  $\{0, 1, 2, 3\}$  for 2-bit activations.

We use stochastic gradient descent for optimizing the parameters of both the weights and the quantizers. The transformers are piece-wise differentiable, and thus we can compute the gradient with respect to the interval parameters  $c_\Delta$ ,  $d_\Delta$  and  $\gamma$ . We use straight-through-estimator [2, 27] for the gradient of the discretizers.

Basically, our method trains the weight parameters jointly with quantizers. However, it is also possible to train the quantizers on a pre-trained network with full-precision weights. Surprisingly, training only the quantizer without updating weights also yields reasonably good accuracy, although its accuracy is lower than that of joint training (See Fig. 6).

We describe the pseudo-code for training and deploying our low bit-width model in Algorithm 1.

## 4. Experiment results

To demonstrate the effectiveness of our trainable quantizer, we evaluated it on the ImageNet [23] and the CIFAR-100 [14] datasets.

### 4.1. ImageNet

The ImageNet classification dataset [23] consists of 1.2M training images from 1000 general object classes and 50,000 validation images. We used various network architectures such as ResNet-18, -34 and AlexNet for evaluation.

**Implementation details** We implement our method using PyTorch with multiple GPUs. We use original ResNet architecture [9] without any structural change for ResNet-18 and -34. For AlexNet [15], we use batch-normalization layer after each convolutional layer and remove the dropout layers and the LRN layers while preserving the other factors such as the number and the sizes of filters. In all the experiments, the training images are randomly cropped and resized to  $224 \times 224$ , and horizontally flipped at random. We use no further data augmentation. For testing, we use the single center-crop of  $224 \times 224$ . We used stochastic gradient descent with the batch size of 1024 (8 GPUs), the momentum of 0.9, and the weight decay of 0.0001 for ResNet (0.0005 for AlexNet). We trained each full-precision network up to 120 epochs where the learning rate was initially set to 0.4 for ResNet-18 and -34 (0.04 for AlexNet), and is decayed by a factor of 10 at 30, 60, 85, 95, 105 epochs. We finetune low bit-width networks for up to 90 epochs where the learning rate is set to 0.04 for ResNet-18 and -34 (0.004 for AlexNet) and is divided by 10 at 20, 40, 60, 80 epochs. We set the learning rates of interval parameters to be  $100\times$  smaller than those of weight parameters. We did not quantize the first and the last layers as was done in [12, 27].

**Comparison with existing methods** We evaluate our learnable quantization method with existing methods, by quoting the reported top-1 accuracies from the original papers (Table 1). Our 5/5 and 4/4-bit models preserve the accuracy of full-precision model for all three network architectures (ResNet-18, -34 and AlexNet). For 3/3-bit, the accuracy drops only by 1% for ResNet-18 and by 0.6% for ResNet-34. If we further reduced the bit-width to 2/2-bit, the accuracy drops by 4.5% (ResNet-18) and 3.1% compared to the full-precision. Compared to the second best method (LQ-Nets [26]), our 3/3 and 2/2-bit models are around 1% more accurate for ResNet architectures. For AlexNet, our 3/3-bit model drops the top-1 accuracy only by 0.5% with respect to full-precision which beats the second best by large margin of 5.7%. For 2/2-bit, accuracy drops by 3.7% which is almost same accuracy with TSQ [25]. Note that TSQ used layerwise optimization, which makes it difficult to apply to the ResNet architecture with skip connections. However, our method is general and is applicable any types of network architecture.

Table 1. Top-1 accuracy (%) on ImageNet. Comparison with the existing methods on ResNet-18, -34 and AlexNet. The ‘FP’ represents the full-precision (32/32-bit) accuracy in our implementation.

| Method                        | ResNet-18 (FP: <b>70.2</b> ) |             |             |             | ResNet-34 (FP: <b>73.7</b> ) |             |             |             | AlexNet (FP: <b>61.8</b> ) |             |             |             |
|-------------------------------|------------------------------|-------------|-------------|-------------|------------------------------|-------------|-------------|-------------|----------------------------|-------------|-------------|-------------|
|                               | Bit-width (A/W)              |             |             |             |                              |             |             |             |                            |             |             |             |
|                               | 5/5                          | 4/4         | 3/3         | 2/2         | 5/5                          | 4/4         | 3/3         | 2/2         | 5/5                        | 4/4         | 3/3         | 2/2         |
| <b>QIL (Ours)<sup>†</sup></b> | <b>70.4</b>                  | <b>70.1</b> | <b>69.2</b> | <b>65.7</b> | <b>73.7</b>                  | <b>73.7</b> | <b>73.1</b> | <b>70.6</b> | <b>61.9</b>                | <b>62.0</b> | <b>61.3</b> | <b>58.1</b> |
| LQ-Nets [26]                  | -                            | 69.3        | 68.2        | 64.9        | -                            | -           | 71.9        | 69.8        | -                          | -           | -           | 57.4        |
| PACT [4]                      | 69.8                         | 69.2        | 68.1        | 64.4        | -                            | -           | -           | -           | 55.7                       | 55.7        | 55.6        | 55.0        |
| DoReFa-Net [27]               | 68.4                         | 68.1        | 67.5        | 62.6        | -                            | -           | -           | -           | 54.9                       | 54.9        | 55.0        | 53.6        |
| ABC-Net [17]                  | 65.0                         | -           | 61.0        | -           | 68.4                         | -           | 66.7        | -           | -                          | -           | -           | -           |
| BalancedQ [28]                | -                            | -           | -           | 59.4        | -                            | -           | -           | -           | -                          | -           | -           | 55.7        |
| TSQ <sup>†</sup> [25]         | -                            | -           | -           | -           | -                            | -           | -           | -           | -                          | -           | -           | 58.0        |
| SYQ <sup>†</sup> [6]          | -                            | -           | -           | -           | -                            | -           | -           | -           | -                          | -           | -           | 55.8        |
| Zhuang <i>et al.</i> [30]     | -                            | -           | -           | -           | -                            | -           | -           | -           | -                          | 58.1        | -           | 52.5        |
| WEQ [20]                      | -                            | -           | -           | -           | -                            | -           | -           | -           | -                          | 55.9        | 54.9        | 50.6        |

Table 2. The top-1 accuracy (%) of low bit-width networks on ResNet-18 with direct and progressive finetuning. The 5/5-bit network was finetuned from full-precision network.

| Initialization | Bit-width (A/W) |      |      |      |      |
|----------------|-----------------|------|------|------|------|
|                | 32/32           | 5/5  | 4/4  | 3/3  | 2/2  |
| Direct         | 70.2            | 70.4 | 69.9 | 68.7 | 56.0 |
| Progressive    | -               | 70.1 | 69.2 | 65.7 | -    |

Table 3. Joint training vs. Quantizer only. The top-1 accuracy (%) with ResNet-18.

| Initialization | Bit-width (A/W) |      |      |      |      |
|----------------|-----------------|------|------|------|------|
|                | 32/32           | 5/5  | 4/4  | 3/3  | 2/2  |
| Joint training | 70.2            | 70.4 | 70.1 | 69.2 | 65.7 |
| Quantizer only | -               | 69.4 | 68.0 | 62.0 | 20.9 |

**Initialization** Good initialization of parameters improves the accuracy of trained low bit-width networks [17, 19]. We adopt finetuning approach for good initialization. In [17], they progressively conduct the quantization from higher bit-widths to lower bit-widths for better initialization. We compare the results of direct finetuning of full-precision network with progressively finetuning from 1-bit higher bit-width network (Table 2). For progressive finetuning, we sequentially train the 4/4, 3/3, and 2/2-bit networks (i.e., FP  $\rightarrow$  5/5  $\rightarrow$  4/4  $\rightarrow$  3/3  $\rightarrow$  2/2 for 2/2-bit network). Generally, the accuracies of progressive finetuning are higher than those of direct finetuning. The progressive finetuning is crucial for 2/2-bit network (9.7% point accuracy improvement), but has marginal impact on 4/4 and 3/3-bit networks (only 0.2% and 0.5% point improvements, respectively).

**Joint training vs. Quantizer only** The weight parameters can be optimized jointly with quantization parameters

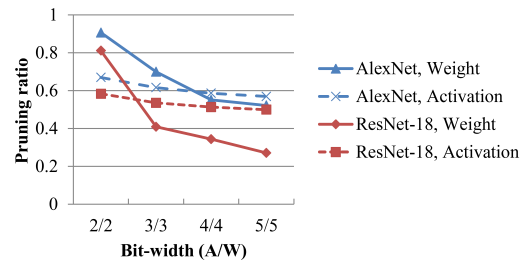


Figure 3. Average pruning ratio of weights and activations on AlexNet and ResNet-18 with various bit-widths

or we can optimize only the quantizers while keeping the weight parameters fixed. Table 3 shows the top-1 accuracy with ResNet-18 network on the both cases. Both the cases utilize the progressive finetuning. The joint training of quantizer and weights works better than training the quantizer only, which is consistent with our intuition. The joint training shows graceful performance degradation while reducing bit-width, compared to training of only the quantizers. Nevertheless, the accuracy of the quantizers is quite promising with 4-bit or higher bit-width. For example, the accuracy drop with 4/4-bit model is only 2.1% (70.1%  $\rightarrow$  68.0%).

**Pruning ratio** In order to see the pruning effect of our quantizer, we compute the pruning ratio which is the number of zeros over the total number of weights or activations. Fig. 3 shows the average pruning ratios of the entire network for ResNet-18 and AlexNet. As expected, the pruning ratio increases as the bit-width decreases. If the bit-width is high, the quantization interval can be relaxed due to the high quantization resolution. However, as the bit-width de-

<sup>†</sup> With this mark in Table 1 and 6, the 2-bit of weights is ternary  $\{-1, 0, 1\}$ , otherwise it is 4-level.

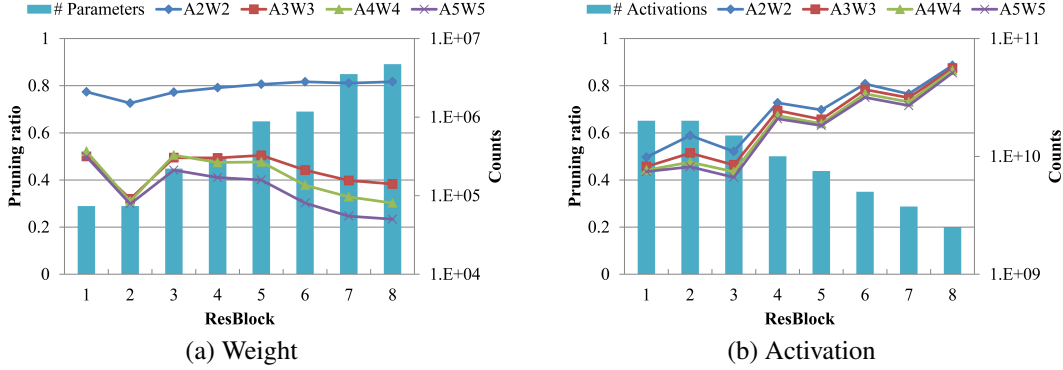


Figure 4. Blockwise pruning ratio of (a) weight and (b) activation for each ResBlock with ResNet-18. The bar graph shows the number of weights or activations for each ResBlock on a log scale.

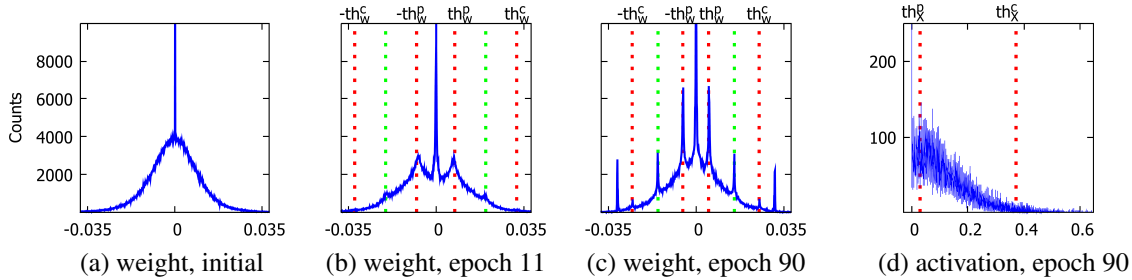


Figure 5. Weight and activation distributions at the 3rd layer of the 3/3-bit AlexNet. The 3/3-bit network is finetuned from the pretrained full-precision network. The figures show the distributions of (a) initial weights, (b) weights at epoch 11, (c) weights at epoch 90 and (d) activations at epoch 90. The  $th_{\Delta}^p$  and the  $th_{\Delta}^c$  represent the pruning and the clipping thresholds, respectively.

creases, more compact interval should be found to maintain the accuracy, and as a result, the pruning ratio increases. For 2/2-bit network, 91% and 81% of weights are pruned on average for AlexNet and ResNet-18, respectively (Fig. 3). The AlexNet is more pruned than the ResNet-18 because the AlexNet has fully-connected layers which have 18~64 times larger parameters than the convolutional layers and the fully-connected layers are more likely pruned. The activations are less affected by the bit-width. Fig. 4 shows the blockwise pruning ratio of ResNet-18. We compute the pruning ratio for each ResBlock which consists of two convolutional layers and a skip connection. For activations, the upper layers are more likely to be pruned, which may be because more abstraction occurs at higher layers.

**Training  $\gamma$**  For weight quantization, we designed the transformer with  $\gamma$  (Eq. 3) which considers the distribution inside the interval. We investigated the effects of training  $\gamma$ . Table 4 shows the top-1 accuracies according to the various  $\gamma$  for 3/3 and 2/2-bit AlexNet. We report both the trainable  $\gamma$  and fixed  $\gamma$ . For 3/3-bit model, the trainable  $\gamma$  does not affect the model accuracy; i.e., 61.4% with trainable  $\gamma$  and 61.3% with  $\gamma = 1$ . However, the trainable  $\gamma$  is effective for 2/2-bit model which improves the top-1 accuracy by 0.9% compared with  $\gamma = 1$  while the accuracies of the fixed  $\gamma$  of 0.5 and 0.75 are similar with  $\gamma = 1$ . The fixed  $\gamma$  of 1.5 de-

| Bit-width (A/W) | Fixed $\gamma$ |      |      |      | Trainable $\gamma$ |
|-----------------|----------------|------|------|------|--------------------|
|                 | 0.5            | 0.75 | 1.0  | 1.5  |                    |
| 3/3             | 61.0           | 61.3 | 61.3 | 61.0 | 61.4               |
| 2/2             | 57.3           | 57.2 | 57.2 | 56.8 | 58.1               |

|                    | AlexNet     | ResNet-18   | ResNet-34   |
|--------------------|-------------|-------------|-------------|
| Fixed $\gamma = 1$ | 57.2        | 65.7        | 70.6        |
| Trainable $\gamma$ | 58.1 (+0.9) | 66.1 (+0.4) | 70.6 (+0.0) |

grades the performance. We also evaluated trainable  $\gamma$  with various network models for 2/2-bit (Table 5). The trainable  $\gamma$  is less effective for ResNet-18 and -34.

**Weight quantization** To demonstrate the effectiveness of our method on weight quantization, we only quantize the weights and leave the activations as full-precision. We measure the accuracy on ResNet-18 with various bit-widths for weights and compare it to the other existing methods (Table 6). The accuracies of the networks quantized with our method are slightly higher than those obtained using the second-best method (LQ-Nets [26]).

Table 6. Accuracy with ResNet-18 on ImageNet. The weights are quantized to low bits and the activations remain at 32 bits. The TTQ-B and TWN-B used the ResNet-18B [16] where the number of filters in each block is  $1.5\times$  of the ResNet-18.

| Bit-width<br>(A/W) | Method                       | Accuracy (%) |             |
|--------------------|------------------------------|--------------|-------------|
|                    |                              | Top-1        | Top-5       |
| 32/32              | FP-ours                      | 70.2         | 89.6        |
| 32/4               | <b>QIL (Ours)</b>            | <b>70.3</b>  | <b>89.5</b> |
|                    | LQ-Nets [26]                 | 70.0         | 89.1        |
| 32/3               | <b>QIL (Ours)</b>            | <b>69.9</b>  | <b>89.3</b> |
|                    | LQ-Nets [26]                 | 69.3         | 88.8        |
| 32/2               | <b>QIL<sup>†</sup>(Ours)</b> | <b>68.1</b>  | <b>88.3</b> |
|                    | LQ-Nets [26]                 | 68.0         | 88.0        |
|                    | TTQ-B <sup>†</sup> [29]      | 66.6         | 87.2        |
|                    | TWN-B <sup>†</sup> [16]      | 65.3         | 86.2        |
|                    | TWN <sup>†</sup> [16]        | 61.8         | 84.2        |

**Distributions of weights and activations** Fig. 5 shows the distributions of weights and activations with different epochs. Since we train both weights and quantizers, the distributions of weights and quantization intervals are changing during training. Note that the peaks of the weight distribution appear at the transition of each quantized values. If the objective is to minimize the quantization errors, this distribution is not desirable. Since our objective is to minimize the task loss, during training the loss become different only when the quantized level of weights are different. Therefore the weight values move toward the transition boundaries. We also plot the activation distribution with the pruning and the clipping thresholds. (Fig. 5 (d)).

#### 4.2. CIFAR-100

In this experiment, we train a low bit-width network from a pre-trained full-precision network without the original training dataset. The motivation of this experiment is to validate whether it is possible to train the low bit-width model when the original training dataset is not given. To demonstrate this scenario, we used CIFAR-100 [14], which consists of 100 classes containing 500 training and 100 validation images for each class where 100 classes are grouped into 20 superclasses (5 classes for each superclass). We divide the dataset into two disjoint groups  $A$  (4 superclasses, 20 classes) and  $B$  (16 superclasses, 80 classes) where  $A$  is the original training set and  $B$  is used as a heterogeneous dataset for training low bit-width model (4/4-bit model in this experiment). The  $B$  is further divided into four subsets  $B_{10}$ ,  $B_{20}$ ,  $B_{40}$  and  $B_{80}$  ( $B_{10} \subset B_{20} \subset B_{40} \subset B_{80} = B$ ) with 10, 20, 40 and 80 classes, respectively. First, we train the full-precision networks with  $A$ , then finetune the low bit-width networks with  $B$  by minimizing the mean-squared-errors between the outputs of the full-

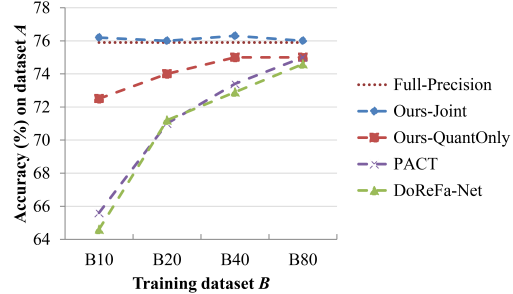


Figure 6. The accuracy on CIFAR-100 for low bit-width model training with heterogeneous dataset.

precision model and the low bit-width model. We report the testing accuracy of  $A$  for evaluation (Fig. 6). We compare our method with other existing methods such as DoReFa-Net [27] and PACT [4]. We carefully re-implemented these methods with pyTorch. Our joint training of weights and quantizers preserve the full-precision accuracy with all types of  $B$ . If we train the quantizer only with fixed weights, then the accuracies are lower than those of the joint training. Our method achieves better accuracy compared to DoReFa-Net and PACT. As expected, as the size of dataset increases ( $B_{10} \rightarrow B_{20} \rightarrow B_{40} \rightarrow B_{80}$ ), the accuracy improves. These are impressive results, since they show that we can quantize any given low bit-width networks can without access to the original data they are trained on.

#### 5. Conclusion

We proposed a novel trainable quantizer with parameterized quantization intervals for training low bit-width networks. Our trainable quantizer performs simultaneous pruning and clipping for both weights and activations, while maintaining the accuracy of the full-precision network by learning appropriate quantization intervals. Instead of minimizing the quantization error with respect to the weights/activations of the full-precision networks as done in previous work, we train the quantization parameters jointly with the weights by directly minimizing the task loss. As a result, we achieved very promising results on the large scale ImageNet classification dataset. The 4-bit networks obtained using our method preserve the accuracies of the full-precision networks with various architectures, 3-bit networks yield comparable accuracy to the full-precision networks, and the 2-bit networks suffers from minimal accuracy loss. Our quantizer also achieves good quantization performance that outperforms the existing methods even when trained on a heterogeneous dataset, which makes it highly practical in situations where we have pretrained networks without access to the original training data. Future work may include more accurate parameterization of the quantization intervals with piecewise linear functions and use of Bayesian approaches.



## References

- [1] Jan Achterhold, Jan Mathias Koehler, Anke Schmeink, and Tim Genewein. Variational network quantization. 2018.
- [2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- [3] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. *arXiv preprint arXiv:1702.00953*, 2017.
- [4] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.
- [5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.
- [6] Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip HW Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4300–4309, 2018.
- [7] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- [8] 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*, pages 770–778, 2016.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.
- [10] 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.
- [11] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.
- [12] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016.
- [13] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
- [14] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Cite-seer, 2009.
- [15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [16] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.
- [17] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017.
- [18] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298, 2017.
- [19] Jeffrey L McKinstry, Steven K Esser, Rathinakumar Appuswamy, Deepika Bablani, John V Arthur, Izzet B Yildiz, and Dharmendra S Modha. Discovering low-precision networks close to full-precision networks for efficient embedded inference. *arXiv preprint arXiv:1809.04191*, 2018.
- [20] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [21] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters: improve semantic segmentation by global convolutional network. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1743–1751. IEEE, 2017.
- [22] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.
- [23] 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*, 115(3):211–252, 2015.
- [24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [25] Peisong Wang, Qinghao Hu, Yifan Zhang, Chunjie Zhang, Yang Liu, Jian Cheng, et al. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4376–4384, 2018.
- [26] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. *arXiv preprint arXiv:1807.10029*, 2018.
- [27] 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.
- [28] Shu-Chang Zhou, Yu-Zhi Wang, He Wen, Qin-Yao He, and Yu-Heng Zou. Balanced quantization: An effective and efficient approach to quantized neural networks. *Journal of Computer Science and Technology*, 32(4):667–682, 2017.
- [29] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

- [30] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. *In other words*, 2:2, 2018.