# Learnable Lookup Table for Neural Network Quantization

Longguang Wang[1], Xiaoyu Dong[2,3], Yingqian Wang[1], Li Liu[1], Wei An[1], Yulan Guo[1*]
[1]National University of Defense Technology [2]The University of Tokyo [3]RIKEN AIP
{wanglongguang15,yulan.guo}@nudt.edu.cn

## Abstract

*Neural network quantization aims at reducing bit-widths of weights and activations for memory and computational efficiency. Since a linear quantizer (i.e., $round(\cdot)$ function) cannot well fit the bell-shaped distributions of weights and activations, many existing methods use pre-defined functions (e.g., exponential function) with learnable parameters to build the quantizer for joint optimization. However, these complicated quantizers introduce considerable computational overhead during inference since activation quantization should be conducted online. In this paper, we formulate the quantization process as a simple lookup operation and propose to learn lookup tables as quantizers. Specifically, we develop differentiable lookup tables and introduce several training strategies for optimization. Our lookup tables can be trained with the network in an end-to-end manner to fit the distributions in different layers and have very small additional computational cost. Comparison with previous methods show that quantized networks using our lookup tables achieve state-of-the-art performance on image classification, image super-resolution, and point cloud classification tasks.*

## 1. Introduction

Deep neural networks have achieved huge success in computer vision, natural language processing and many other fields. However, high computational cost and memory footprint limit their applications on edge devices. Many efforts have been made to address this problem, including efficient architecture designs [1–3], network pruning [4–6], weight decomposition [7, 8], knowledge distillation [9–11], and network quantization [12–14]. Among these approaches, network quantization can significantly reduce the computational cost and is widely applied in real-world applications (*e.g.*, inference framework like TF-lite [15]).

Neural network quantization aims at obtaining low-bit networks to reduce memory footprint and computational cost. The decrease in bit-width naturally introduces quantization errors, which in turn leads to accuracy loss. Network

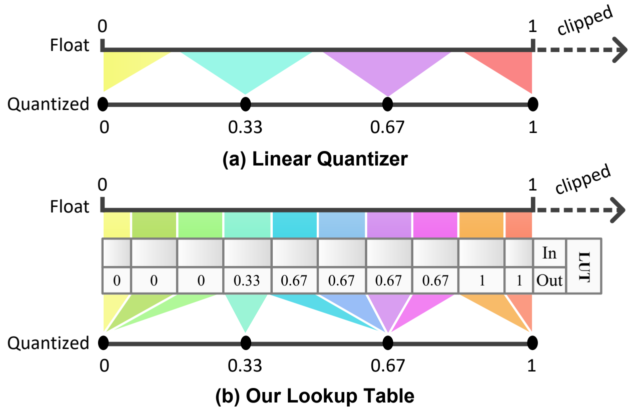

Figure 1. An illustration of a linear quantizer and our learned lookup table (LUT) for 2-bit quantization of activations.

quantization can be divided into uniform approaches and non-uniform approaches. In this paper, we focus on uniform ones since they can be directly deployed on off-the-shelf hardwares with the support of integer arithmetic [16].

Early uniform quantization methods [16–19] use fixed linear quantizers (*i.e.*, $\mathrm{round}(\cdot)$ function) to quantize float values. However, these quantizers cannot fit the bell-shaped and long-tailed distributions of weights and activations well [20]. Moreover, the distributions in different layers can vary significantly. Consequently, fixed quantizers have limited capability to adapt to various layers.

To fit various distributions of weights and activations in different layers, several works [14, 21] adopt trainable quantizers for joint optimization. These methods use a combination of pre-defined functions (*e.g.*, exponential function and sigmoid function) with learnable parameters to represent the quantization function. Since activation quantization needs to be conducted online during inference, these complicated quantizers introduce considerable computational overhead.

In this paper, we propose to use learnable lookup tables (LLTs) to map float values to quantized values for network quantization (Fig. 1). Specifically, the quantization function is formulated as a lookup table (Fig. 2(a)) and then transformed to one-hot distributions (Fig. 2(d)) for optimization. During training, we soften one-hot distributions to temperatured softmax distributions (Fig. 2(e)) to update both the lookup tables and the network in an end-to-end manner. In addition, we introduce several training strategies to promote the convergence of our lookup tables, including an exponential formulation of scale parameters and a gradient rescaling scheme. As temperatured softmax distributions converge to one-hot distributions, our lookup tables learn an adaptive mapping from continuous float values to the quantization levels. During inference, online activation quantization can be achieved by a simple lookup operation with very small computational cost. Extensive experiments on image classification, image super-resolution (SR), and point cloud classification tasks demonstrate the state-of-the-art performance of our method.

## 2. Related Work

In this section, we first briefly review several major works for network quantization. Then, we discuss typical applications of these quantization methods.

### 2.1. Network Quantization

**Uniform Quantization.** Uniform quantization [16–19] maps full-precision values to uniform quantization levels (*e.g.*, $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$ for 2-bit quantization of activations). BinaryConnect [24] was first proposed to quantize the weights using binary values $\{-1, +1\}$. Trained ternary quantization (TTQ) [25] was then introduced to use ternary values $\{-1, 0, +1\}$ to represent the weights. However, the activations in these networks are still full-precision values. To address this limitation, Rastegari *et al.* [26] used scale parameters to binarize the activations. Recently, DoReFa-Net [27] was developed to further quantize the gradients during backward propagation to speed up the training process. Most uniform quantization methods use linear quantizers (*i.e.*, $\mathrm{round}(\cdot)$ function) to uniformly quantize float values. However, these quantizers do not take the bell-shaped distributions of weights and activations into consideration and thus cannot fit them well.

To make the quantizers adapt to the distributions of weights and activations, several efforts [14, 21] were made to parameterize the quantizers for joint optimization. Jung *et al.* [14] formulated the quantizer as a combination of a transformer and a discretizer. Then, a hand-crafted non-linear function with learnable parameters was used as the transformer to be trained with the network. Yang *et al.* [21] used a linear combination of sigmoid functions with learnable biases and scales to represent the quantization function. Their reformulated quantization function was trained via continuous relaxation of the steepness for the sigmoid functions. However, since quantized activations need to be calculated online during inference, these complicated quantizers introduce considerable computational overhead.

**Non-Uniform Quantization.** Non-uniform quantization [28–30] uses non-uniform levels (*e.g.*, $\{0, \frac{1}{4}, \frac{1}{2}, 1\}$ for 2-bit quantization of activations) to quantize the weights and activations to match their distributions. Ullrich *et al.* [28] proposed to fit a mixture of Gaussian prior models over weights and use cluster centroids as the quantization levels. Xu *et al.* [29] followed this idea and performed layer-wise clustering to quantize weights into cluster centroids. Zhang *et al.* [31] used a set of floating-point values as the basis to formulate non-uniform quantization levels. Zhou *et al.* [32] and Miyashita *et al.* [33] introduced logarithmic quantizers to represent the weights and activations using powers-of-two values. Recently, Li *et al.* [20] further proposed an additive powers-of-two quantization to represent full-precision values using a sum of powers-of-two values. Nevertheless, non-uniform quantization methods usually rely on delicate hardware to achieve acceleration [16].

### 2.2. Applications

Most network quantization methods are developed for the image classification task. Recently, increasing attention has been paid to their extended applications in many other tasks, including image SR and point cloud classification.

**Image Super-Resolution.** The powerful feature representation capability of neural networks facilitates CNN-based image SR methods [34–36] to achieve advanced performance. However, the high computational and memory cost of these methods limit their applications on mobile devices. Many efforts [37–43] have been made to improve the efficiency of SR networks. Specifically, Ma *et al.* [38] and Xin *et al.* [40] binarized SR networks to reduce the model size and speed up inference with comparable performance. Li *et al.* [41] quantized SR networks using a parameterized max value to obtain models with high accuracy and low bit. Hone *et al.* [43] introduced a distribution-aware quantization scheme to adaptively determine channel-wise dynamic ranges for SR networks to produce superior results.

**Point Cloud Classification.** Qi *et al.* [44] proposed the first deep learning based method (namely, PointNet) to directly process raw point clouds. Specifically, PointNet learns point-wise features with multi-layer perceptrons (MLPs) and obtains global features with a max-pooling layer for classification. Following PointNet, many advanced feature learners [45–47] and aggregators [48–51] have been proposed, with better performance being achieved. However, these methods rely on expensive floating-point calculation and have high computational cost. Recently, Qin *et al.* [52] made the first attempt to binarize PointNet to achieve significant speedup and memory saving.

## 3. Method

The main idea of this work is to formulate the quantization process as a differentiable lookup operation. Given

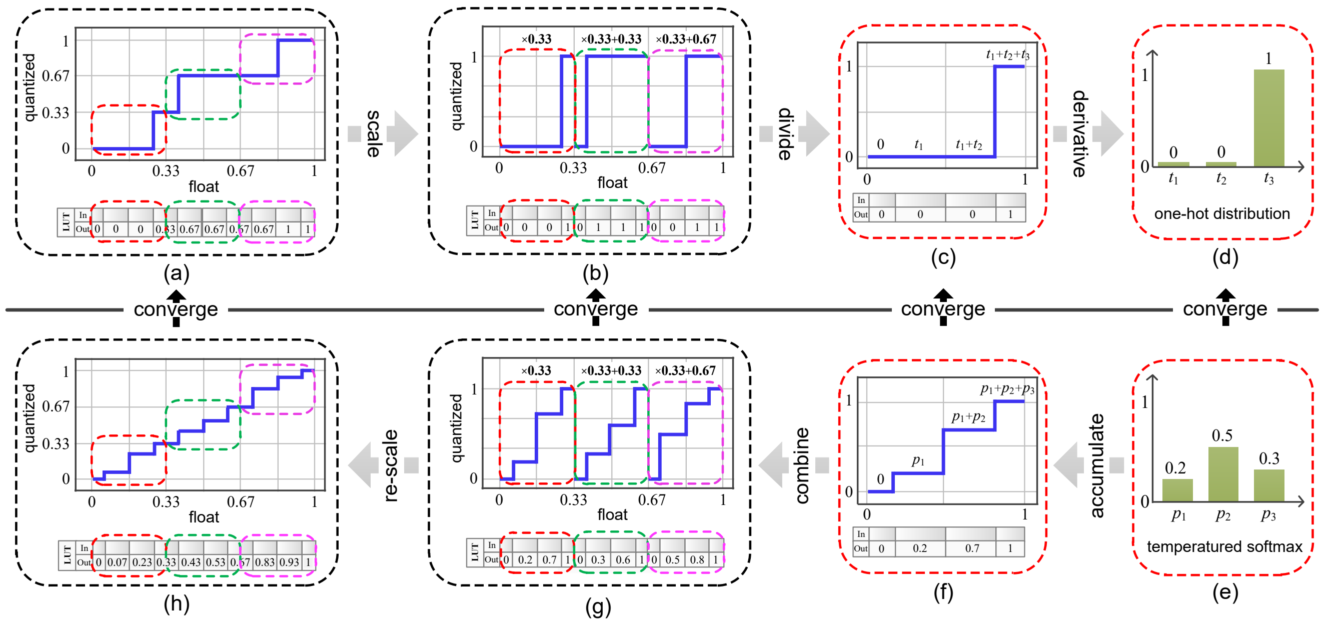

Figure 2. An illustration of lookup table construction. For simplicity, 2-bit quantization for activations is used as an example.

a network, layer-wise lookup tables are first constructed. Then, lookup operation is performed to map full-precision weights and activations to their low-bit counterparts. During training, we soften the lookup tables for joint optimization with the network. During inference, the lookup tables are hardened to obtain low-bit networks.

## 3.1. Preliminaries

Given a weight value $w$ and an activation value $a$ in a full-precision network, they are first transformed to produce $\hat{w}$ and $\hat{a}$ as

$$\begin{cases} \hat{w} = \text{clip}(w/s_w) \\ \hat{a} = \text{clip}(a/s_a) \end{cases}, \tag{1}$$

where $s_w$ and $s_a$ are trainable scale parameters, $\text{clip}(\cdot)$ clips values into $[-1,1]$ ($[0,1]$ for $a$). Then, $\hat{w}$ and $\hat{a}$ are further quantized as

$$\begin{cases} \overline{w} = s_w \cdot \mathbb{Q}(\hat{w}, Q_w) \\ \overline{a} = s_a \cdot \mathbb{Q}(\hat{a}, Q_a) \end{cases}, \tag{2}$$

where $Q_w$ and $Q_a$ are the numbers of quantization levels, $\mathbb{Q}(\cdot)$ maps float values to a set of discrete values (*e.g.*, $\{0, \frac{1}{Q_a}, \dots \frac{Q_a-1}{Q_a}, 1\}$ for $a$ and $\{-1, \dots, -\frac{1}{Q_w}, 0, \frac{1}{Q_w}, \dots, 1\}$ for $w$). For $b$-bit quantization, $Q_a = 2^b - 1$ and $Q_w = 2^{b-1} - 1$.

In this paper, we formulate the quantization process $\mathbb{Q}(\cdot)$ as a lookup operation and learn lookup tables as quantizers.

## 3.2. Lookup Table Construction

In this section, we first illustrate our motivation and then introduce the construction of our lookup tables.

**Motivation.** As shown in Fig. 1(b), a quantization function can be formulated as a lookup table that maps float values to the quantization levels, *e.g.*, $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$ for 2-bit quantization of activations[1]. However, it is challenging to directly learn lookup tables due to their non-differentiability. To address this issue, we simplify the formulation of a lookup table to make it easier to be optimized. Specifically, the quantization function in Fig. 2(a) can be considered as a concatenation of three scaled step functions ($\varepsilon(\cdot)$):

$$x_q = 0.33(\varepsilon(x - 0.28) + \varepsilon(x - 0.39) + \varepsilon(x - 0.83)). \tag{3}$$

Then, the lookup table can be divided into three binary sub-tables, with each sub-table representing one step function (Fig. 2(b)). Theoretically, a step function can be formulated as the integral of an impulse function. Consequently, for each sub-table, we could use $K$ auxiliary parameters $\{t_1, t_2, \dots, t_K\}$ to model an impulse function ($K = 3$ in Fig. 2(d) for simplicity) and then accumulate these parameters to represent the binary sub-table (Fig. 2(c)). In this way, the lookup table can be formulated as three one-hot distributions, whose optimization has been well studied in literature.

**Implementation.** During the training phase, each one-hot distribution is softened to a temperatured softmax distribution $\{p_1, p_2, \dots, p_K\}$ (Fig. 2(e)) to make it trainable:

$$p_i = \frac{\exp(g_i/\tau)}{\sum_i^K \exp(g_i/\tau)}, \tag{4}$$

where $g_i$ is a learnable parameter and $\tau$ is a temperature parameter. Then, the temperatured softmax distribution is

[1] Intuitively, float values equaling to the quantization levels (*e.g.*, $a = \frac{1}{3}$) should not be mapped to other quantized values in order not to introduce quantization error. Consequently, corresponding cells are anchored to matched quantization levels.

accumulated to construct a sub-table (Fig. 2(f)). Finally, the resultant sub-tables are re-scaled and combined to produce the lookup table (Fig. 2(h)).

In our experiments, $g_i$ is initialized as 0 such that the temperatured softmax distribution is initialized as a uniform distribution. As $\tau$ gradually decreases from a high temperature to a low one, the temperatured softmax distribution converges to a one-hot distribution. Consequently, each sub-table becomes binary and our lookup table learns an adaptive mapping from continuous float values to the quantization levels (Fig. 2(a)). The detailed temperature scheduler is presented in Sec. 3.4.

## 3.3. Differentiable Lookup Operation

With our lookup tables, lookup operation is performed to map full-precision values to low-bit ones, as shown in Fig. 1(b). Given a float value (*e.g.*, an activation value $a$), $\hat{a}$ is first calculated according to Eq. 1. Next, the lookup table is used to find the corresponding quantization level for $\hat{a}$ (*e.g.*, $\mathbb{Q}(\hat{a}, Q_a) = p_1$) to produce $\overline{a}$ based on Eq. 2. Then, $\overline{a}$ can be used as a proxy of $a$ for forward propagation to compute the task loss.

During backward propagation, the straight-through-estimator (STE) [53] is used to calculate gradients to update the lookup tables (*e.g.*, $p_1$) and full-precision values (*i.e.*, $a$):

$$\frac{\partial \overline{a}}{\partial p_1} = s_a, \quad \frac{\partial \overline{a}}{\partial a} = \begin{cases} 1, & \text{if } a < s_a \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

For the scale parameter $s_a$ in Eq. 1, the gradient formulation in [18] is used for optimization. The detailed derivation of these gradients is provided in the supplemental material. With these gradients, our lookup tables can be trained with the network in an end-to-end manner.

## 3.4. Training Strategies

In this section, we introduce several training strategies to improve the convergence of our lookup tables.

**Exponential Formulation of Scale Parameter.** The scale parameters $s_w$ and $s_a$ in Eq. 1 are positive values used to clip full-precision weights and activations into $[-1, 1]$ and $[0, 1]$, respectively. In our experiments, it is observed that these scale parameters may become negative during training, which leads to convergence issues. To address this problem, we introduce auxiliary parameters $e_w$ and $e_a$ to formulate scale parameters as:

$$\begin{cases} s_w = \exp(e_w) \\ s_a = \exp(e_a) \end{cases}. \tag{6}$$

During training, $e_w$ is initialized using the standard deviation of full-precision weights ($\ln(3\sigma_w)$) and $e_a$ is initialized using the standard deviation of activations in the first iteration ($\ln(3\sigma_a)$). It is shown in Sec. 4.4 that such an exponential formulation facilitates our network to achieve a good convergence with better performance.

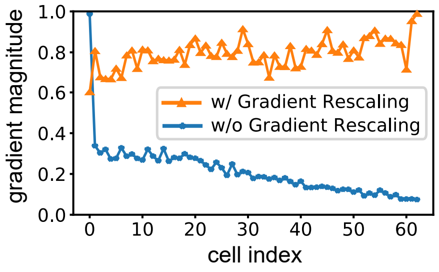

Figure 3. An illustration of gradient imbalance among different cells of our lookup table. Normalized gradient magnitudes averaged over 100 iterations are shown.

**Gradient Rescaling.** In our experiments, we observe a gradient imbalance among different cells of our lookup table. Due to the bell-shaped distributions of full-precision activations and weights, the numbers of values falling into different cells during lookup operation are quite different. Consequently, aggregated gradients of cells near 0 are much larger than those near 1 (blue curve in Fig. 3) and dominate the optimization of the lookup table. To handle this problem, gradient $g_i$ of the $i^{\text{th}}$ cell is rescaled using $\frac{N_{avg}}{\sqrt{N_i}}$, where $N_{avg}$ is the average number of float values in a cell and $N_i$ is the number of float values falling in the $i^{\text{th}}$ cell. With our gradient rescaling scheme, gradients over different cells of the lookup table are balanced, as shown in Fig. 3. It is further demonstrated in Sec. 4.4 that our gradient rescaling scheme contributes to a good convergence of the lookup table and improves the overall performance.

**Temperature Scheduler.** The temperature parameter $\tau$ in Eq. 4 controls the sharpness of the temperatured softmax distribution. In our experiments, we start with a high temperature and then anneal it to a low one. Specifically, an exponential annealing scheduler is empirically used to update the temperature parameter $\tau$:

$$\tau(n) = \begin{cases} \tau_0 \times (\frac{\tau_1}{\tau_0})^{\frac{n}{MN_{iter}}}, & n < MN_{iter} \\ \tau_1, & \text{otherwise} \end{cases}, \tag{7}$$

where $\tau_0$ and $\tau_1$ are the initial and final temperatures, respectively. $n$ represents the iteration index, $N_{iter}$ is the number of iterations in an epoch, and $M$ determines the rate of decline.

## 3.5. Discussion

It is demonstrated that progressive quantization (*e.g.*, 32→4→3→2 for 2-bit quantization) can improve the performance of low-bit quantized networks [14, 20]. From the perspective of progressive quantization, the training of our lookup table in Fig. 2 can be considered as an evolution from 10 quantization levels (Fig. 2(h)) to 4 quantization levels (Fig. 2(a)). Intuitively, due to the distribution difference between (1) weights and activations, (2) various intervals of float values (*e.g.*, $[0, 0.2]$ vs. $[0.8, 1]$), and (3) various layers,

Table 1. Top-1 accuracy (%) achieved on CIFAR-10. Results marked with * are copied from their corresponding papers. Since QIL and LSQ are not tested on CIFAR-10 in their papers, we used our implementation for evaluation.

| Model | Method | Bit-Width (Weight/Activation) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 32/32 | 4/4 | 3/3 | 2/2 | 2/3 | 2/4 | 2/8 | 2/32 | 4/8 | 4/32 |
| ResNet-20 | DoReFa-Net* [27] | 92.96 | 90.50 | 89.90 | 88.20 | - | - | - | - | - | - |
| | PACT* [17] | | 91.30 | 91.10 | 89.70 | - | - | - | - | - | - |
| | PACT-SAWB* [54] | | - | - | 89.23 | - | - | - | 90.73 | - | - |
| | QIL [14] | | 91.52 | 91.81 | 90.45 | 91.28 | 91.33 | 91.77 | 91.89 | 91.71 | 92.02 |
| | LSQ [18] | | 92.30 | 91.69 | 90.08 | 91.32 | 91.72 | 91.83 | 92.02 | 92.47 | 92.54 |
| | SLB* [19] | | 91.60 | - | 90.60 | - | 91.30 | 91.80 | 92.00 | 91.80 | 92.10 |
| | LLT (Ours) | | **92.71** | **92.17** | **90.63** | **91.65** | **91.80** | **92.00** | **92.15** | **92.71** | **92.74** |
| VGG-Small | DoReFa-Net* [27] | 94.10 | 88.20 | 89.90 | 90.50 | - | - | - | - | - | - |
| | QIL [14] | | 93.77 | 93.71 | 93.45 | 93.68 | 93.73 | 93.85 | 93.89 | 93.91 | 94.02 |
| | SLB* [19] | | 93.80 | - | 93.50 | - | 93.90 | 94.00 | 94.00 | 94.00 | 94.10 |
| | CPQ* [55] | | 93.23 | 93.18 | 92.51 | - | - | - | - | - | - |
| | LLT (Ours) | | **94.20** | **94.03** | **93.83** | **93.90** | **94.02** | **94.10** | **94.10** | **94.17** | **94.24** |

progressive quantization should be self-paced. However, the heuristic progressive quantization approach in existing methods [14, 20, 56] reduce quantization levels consistently for all intervals and layers without considering their difference. In contrast, the evolution of our lookup tables is more flexible and adaptive, as demonstrated in Sec. 4.4.

## 4. Experiments

### 4.1. Experiments on Image Classification

#### 4.1.1 Evaluation on CIFAR-10

**Settings.** The CIFAR-10 dataset [57] contains 50K training images and 10K test images of size $32 \times 32$ from 10 classes. We used ResNet-20 [58] and VGG-Small [59] as baseline networks for quantization. Their pre-trained full-precision models were used to initialize quantized models[2]. Following [14, 16], the first and the last layers were not quantized for fair comparison. Implementation details are provided in the supplemental material.

**Performance Evaluation.** We compare our method to several recent uniform quantization methods, including DoReFa-Net [27], PACT [17], PACT-SAWB [54], QIL [14], LSQ [18], SLB [19], and CPQ [55]. DoReFa-Net, PACT, PACT-SAWB, LSQ, and CPQ use fixed linear quantizers while QIL adopts a learnable quantizer. Note that, we focus on uniform quantization in this paper and do not include non-uniform methods (*e.g.*, LQ-Net [31], APoT [20], and LCQ [30]) for comparison. Baseline networks were quantized to different bit-widths for comparison. Comparative results are listed in Table 1.

Compared to full-precision baselines, our 4/4-bit ResNet-20, 4/4-bit and 3/3-bit VGG-Small maintain comparable accuracy. For 2/2-bit quantization, the accuracy drops by 2.33% and 0.27% for ResNet-20 and VGG-Small, respectively. Compared to previous quantization methods, quantized models using our lookup tables produce notable performance improvements. For example, our method outperforms the second best approach by 0.41% (92.71% vs. 92.30%) and 0.40% (94.20% vs. 93.80%) for 4/4-bit quantization. For 3/3-bit quantization, the accuracy of our method is 0.36% and 0.32% higher than the second best approach (*i.e.*, QIL) for ResNet-20 and VGG-Small, respectively. For quantization with different bit-widths for weights and activations, our method consistently produces better performance than other approaches. This clearly demonstrates the superiority of our lookup tables.

[2]For ResNet-20, the pre-trained full-precision model provided in [20] was employed. For VGG-Small, we trained a full-precision model for initialization.

#### 4.1.2 Evaluation on ImageNet

**Settings.** The ImageNet (ILSVRC-2012) dataset [60] includes $\sim$1.2M training images and 50K validation images from 1K classes. ResNet-18 was employed as the baseline network for quantization. We used official pre-trained model in TorchVision library for the initialization of quantized networks. Following [14, 16], the first and the last layers were not quantized for fair comparison. Implementation details are provided in the supplemental material.

**Performance Evaluation.** We compare our method to several recent uniform quantization methods, including DoReFa-Net [27], ABC-Net [61], PACT [17], DSQ [16], QIL [14], and CPQ [55]. Comparative results are presented in Table 2.

It can be observed that our 4-bit model performs favorably against the full-precision baseline (70.4% vs. 69.8%). Moreover, our LLT outperforms other quantization methods by notable margins for different bit-widths. For example, our method produces 0.3%/0.7% higher Top-1/Top-5 accuracy than the second best approach for 3-bit quantization.

### 4.2. Experiments on Image Super-Resolution

**Settings.** We used 800 training images in DIV2K [62] as the training set and included four benchmark datasets (Set5 [63], Set14 [64], B100 [65], and Urban100 [66]) for evaluation. EDSR [67] and RDN [34] were used as baselines to obtain quantized SR networks. Pre-trained full-precision

Table 2. Top-1/Top-5 accuracy (%) achieved on ImageNet. Results marked with * are copied from their corresponding papers.

| Model | Method | Top-1 Accuracy | | | | Top-5 Accuracy | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 32-bit | 4-bit | 3-bit | 2-bit | 32-bit | 4-bit | 3-bit | 2-bit |
| ResNet-18 | DoReFa-Net* [27] | 69.8 | 68.1 | 67.5 | 62.6 | 89.1 | - | - | - |
| | ABC-Net* [61] | | - | 61.0 | - | | - | 83.2 | - |
| | PACT* [17] | | 69.2 | 68.1 | 64.4 | | 89.0 | 88.2 | 85.6 |
| | DSQ* [16] | | 69.6 | 68.7 | 65.2 | | - | - | - |
| | QIL* [14] | | 70.1 | 69.2 | 65.7 | | - | - | - |
| | CPQ* [55] | | 69.6 | 67.2 | - | | 89.0 | 87.4 | - |
| | LLT (Ours) | | **70.4** | **69.5** | **66.0** | | **89.6** | **88.9** | **86.2** |

Table 3. PSNR results achieved on four benchmarks for $\times 4$ SR. Results marked with * and † are copied from [41] and [43].

| Model | Dataset | Baseline | DoReFa-Net* [27] | | PACT* [17] | | LSQ [18] | | PAMS* [41] | | DAQ† [43] | | LLT (Ours) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 8-bit | 4-bit | 8-bit | 4-bit | 8-bit | 4-bit | 8-bit | 4-bit | 8-bit | 4-bit | 8-bit | 4-bit |
| EDSR | Set5 | 32.46 | 30.19 | 29.57 | 31.52 | 31.39 | 32.34 | 32.27 | 32.12 | 31.59 | - | 32.34 | **32.43** | **32.40** |
| | Set14 | 28.77 | 27.30 | 26.82 | 28.18 | 28.10 | 28.66 | 28.60 | 28.59 | 28.20 | - | 28.69 | **28.77** | **28.74** |
| | B100 | 27.69 | 26.77 | 26.47 | 27.29 | 27.25 | 27.65 | 27.63 | 27.57 | 27.32 | - | 27.61 | **27.71** | **27.70** |
| | Urban100 | 26.54 | 24.22 | 23.75 | 25.25 | 25.15 | 26.49 | 26.34 | 26.02 | 25.32 | - | 26.33 | **26.60** | **26.51** |
| RDN | Set5 | 32.32 | - | - | - | - | 32.27 | 32.20 | 32.34 | 30.44 | - | 31.96 | **32.37** | **32.26** |
| | Set14 | 28.71 | - | - | - | - | 28.66 | 28.63 | 28.72 | 27.54 | - | 28.38 | **28.73** | **28.70** |
| | B100 | 27.67 | - | - | - | - | 27.63 | 27.60 | 27.64 | 26.87 | - | 27.38 | **27.68** | **27.66** |
| | Urban100 | 26.35 | - | - | - | - | 26.23 | 26.20 | **26.37** | 24.52 | - | 25.73 | 26.33 | **26.29** |

models were used for initialization[3]. Following [41, 43], only backbone blocks were quantized for fair comparison. Implementation details are provided in the supplemental material.

**Performance Evaluation.** We use our method and five recent quantization methods (DoReFa-Net [27], PACT [17], LSQ [18], PAMS [41], and DAQ [43]) to quantize baseline SR networks. DoReFa-Net, PACT and LSQ are three generic quantization methods, while PAMS and DAQ are two approaches specially developed for SR networks.

As shown in Table 3, our 8-bit and 4-bit models perform favorably against their full-precision baselines. For example, our 4-bit EDSR model achieves comparable PSNR performance to the baseline (32.40/28.74/27.70/26.51 vs. 32.46/28.77/27.69/26.54). Compared to other quantization methods, quantized models using our lookup tables produce higher PSNR results. For 4-bit quantization of RDN, our LLT outperforms DAQ with significant PSNR improvements (27.66/26.29 vs. 27.38/25.73 on B100/Urban100).

From Fig. 4, we can further see that the quantized EDSR model using our lookup tables produces results with better visual quality and clearer details. Specifically, our 4-bit EDSR faithfully recovers the grids in the first row while other models suffer notable distorted artifacts.

## 4.3. Experiments on Point Cloud Classification

**Settings.** We used ModelNet40 [68] to evaluate our method on point cloud classification. This dataset contains 12311 meshed CAD models from 40 categories. PointNet [44] and PointNet++ [48] were adopted as baselines for quantization. Their pre-trained full-precision models were used for initialization. Following [52], the first and the last layers were not quantized. Implementation details are provided in the supplemental material.

[3]For EDSR, official pre-trained model was employed. For RDN, since official models are implemented in Torch while our method is implemented in PyTorch, we trained a full-precision model for initialization using our implementation.

Table 4. Overall accuracy (%) achieved on ModelNet40.

| Model | Method | Bit-Width | | | |
|---|---|---|---|---|---|
| | | 32 | 4 | 3 | 2 |
| PointNet | DoReFa-Net [27] | 90.8 | 89.4 | 88.3 | 80.3 |
| | PACT [17] | | 89.4 | 87.9 | 80.8 |
| | QIL [14] | | 89.7 | 88.6 | 82.8 |
| | LSQ [18] | | 90.0 | 88.6 | 84.7 |
| | LLT (Ours) | | **90.7** | **89.9** | **87.6** |
| PointNet++ | DoReFa-Net [27] | 92.8 | 92.2 | 92.2 | 89.2 |
| | PACT [17] | | 92.3 | 92.2 | 88.9 |
| | QIL [14] | | 92.6 | 92.1 | 88.7 |
| | LSQ [18] | | 92.6 | 92.1 | 90.1 |
| | LLT (Ours) | | **92.8** | **92.4** | **92.3** |

**Performance Evaluation.** We compare our method with four uniform quantization methods, including DoReFa-Net [27], PACT [17], QIL [14], and LSQ [18]. Results achieved by quantized baseline networks using different methods are shown in Table 4.

Compared to full-precision baselines, our 4-bit quantized models produce on-par accuracy (*e.g.*, 90.7 vs. 90.8 for PointNet). Moreover, our method outperforms other quantization approaches with notable margins. For example, for 2-bit quantization of PointNet, the accuracy of our LLT is 2.9% higher than the second best approach (*i.e.*, LSQ).

## 4.4. Model Analyses

In this section, we first use ResNet-20 as the baseline and conduct ablation study on CIFAR-10 to demonstrate the effectiveness of our design choices. Then, we conduct experiments to analyze our lookup tables.

**Exponential Formulation of Scale Parameter.** Since scale parameters are vulnerable to sign reversal, an expo-

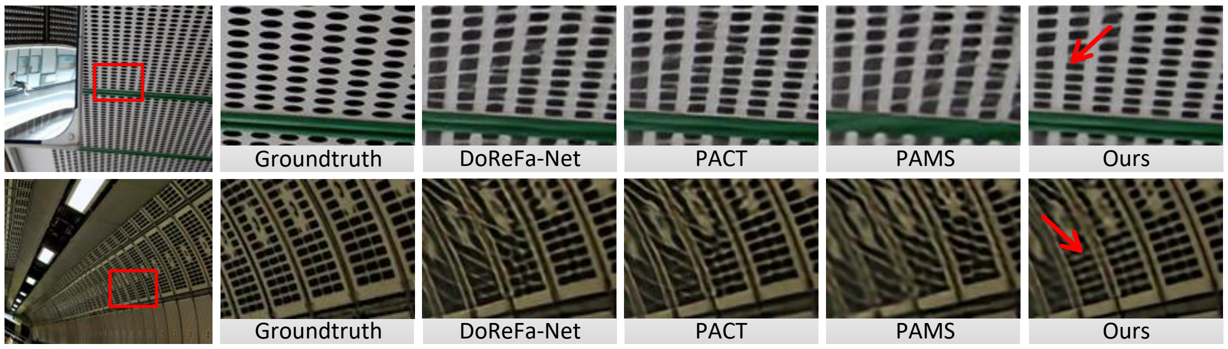

Figure 4. Visual results produced by 4-bit quantized EDSR models for $\times 4$ SR.

Table 5. Top-1 accuracy (%) achieved on CIFAR-10 with different settings. ResNet-20 is used as the baseline.

| Model | Exponential Formulation | Gradient Rescaling | Bit-Width (Weight/Activation) | | | |
|---|---|---|---|---|---|---|
| | | | 32/32 | 4/4 | 3/3 | 2/2 |
| 0 | ✗ | ✗ | 92.96 | 92.44 | 92.02 | 89.81 |
| 1 | ✓ | ✗ | | 92.42 | 92.08 | 90.10 |
| 2 | ✗ | ✓ | | 92.65 | 92.08 | 90.14 |
| 3 (Ours) | ✓ | ✓ | | **92.71** | **92.17** | **90.63** |

nential formulation is introduced for stable convergence. To demonstrate its effectiveness, we developed a network variant (model 1 in Table 5) by replacing the scale parameters in model 0 with an exponential formulation (Eq. 6). Since the sign reversal of scale parameters largely affects the convergence of the network, model 0 suffers relatively low accuracy, especially for 2/2-bit quantization. With our exponential formulation, a good convergence can be achieved such that better performance can be obtained (90.10 vs. 89.81).

Figure 5 further plots the curves of scale parameters in models 0 and 1 during training. It can be observed that the scale parameter in model 0 has a violent fluctuation and encounters sign reversals at epoch 50 and 70. Due to the convergence problem caused by the sign reversal, model 0 suffers limited accuracy. With our exponential formulation, the training of scale parameters in model 1 is more stable such that better performance can be obtained.

**Gradient Rescaling.** Gradient rescaling scheme is introduced to handle the gradient imbalance among different cells of our lookup tables. To demonstrate its effectiveness, we developed model 2 by employing the gradient rescaling scheme. It can be observed from Table 5 that our gradient rescaling scheme facilitates model 2 to obtain higher accuracy than model 0 (92.65/92.08/90.14 vs. 92.44/92.02/89.81). Without the gradient rescaling scheme, the gradient imbalance among different cells of our lookup tables hinders a good convergence. As a result, the performance of model 0 is limited. With our rescaling scheme, gradients over different cells of our lookup tables can be balanced (Fig. 3) such that a good convergence can be achieved for superior performance.

We further use models 1 and 3 to investigate the effects of our gradient rescaling scheme to the convergence

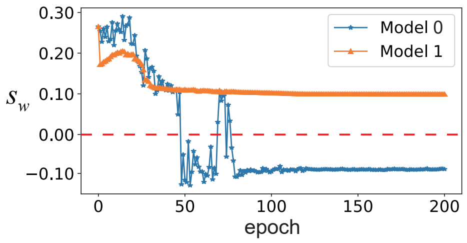

Figure 5. Evolution of scale parameter $s_w$ during training.

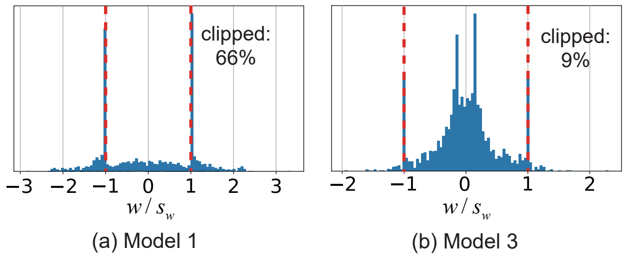

Figure 6. Histograms of weights in models 1 and 3 for 2/2-bit quantization.

of lookup tables. Specifically, histograms of weights in these models are illustrated in Fig. 6. Without our rescaling scheme, gradients over cells near 0 dominate the training of our lookup tables, as analyzed in Sec. 3.4. Consequently, the lookup tables focus too much on small values and clip plenty of large values (Fig. 6(a)). Since these clipped values cannot be updated during training, inferior performance is produced. With our rescaling scheme, model 3 can balance the gradients over different cells to achieve a good convergence, as shown in Fig. 6(b). Therefore, higher accuracy can be obtained (92.71/92.17/90.63 vs. 92.42/42.08/90.10).

Overall, with both exponential formulation of scale parameters and gradient rescaling scheme, model 3 achieves the best performance.

**Evolution of Lookup Tables.** As analyzed above, a good convergence of lookup tables is critical to the performance of quantized networks. Therefore, we illustrate our lookup tables at different epochs in Fig. 7 to study their evolution. More results are provided in the supplemental material.

We have three observations: ***First***, the evolution of the lookup table for activations is faster than that for weights. At epoch 10, the lookup table of weights remains almost

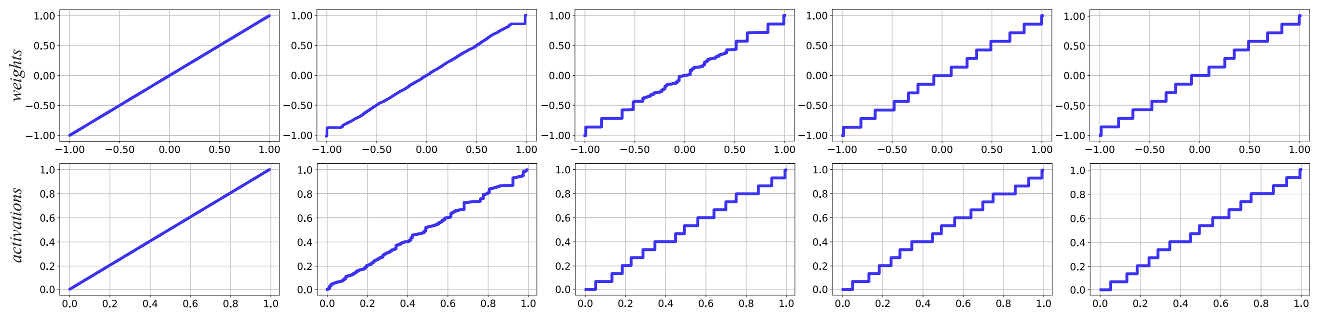

Figure 7. Evolution of our lookup tables during training for 4/4-bit quantization.

unchanged for cells near 0. In contrast, the changes of the lookup table for activations are larger. We suppose that since the distributions of activations are usually less steeper than weights, the lookup tables for activations are easier to be optimized. ***Second***, for the lookup table of weights, cells near 1 converge faster than those near 0 (Fig. 7(b)). Since the number of full-precision values near 1 is much smaller than that of values near 0, cells near 1 are easier to be optimized. ***Third***, the evolution of lookup tables starts from shallow layers and then gradually goes to deeper layers (please see Fig. I in the supplemental material). Overall, the evolution of a lookup table can be considered as a self-paced and adaptive progressive quantization process. At the beginning, our lookup tables are not limited to 4-bit quantization and adaptively assign more quantization levels (Fig. 7(b)). Then, our lookup tables progressively converge to a mapping from float values to 4-bit quantization levels (Fig. 7(e)).

Unlike previous methods [14, 20] that perform heuristic progressive quantization (*e.g.*, 32→4→3→2), our lookup tables can adaptively reduce quantization levels to fit the distributions in different layers. As a result, our LLT is well compatible to different network architectures (*e.g.*, ResNets, EDSR, and PointNet) and achieves state-of-the-art performance on a wide range of tasks (*i.e.*, image classification, image SR, and point cloud classification).

**Effect of Granularity $K$.** The granularity of lookup tables determines the number of initial quantization levels (Fig. 2(h)) and the degree of freedom during optimization. We conduct experiments to study the effect of different granularities. Table 6 compares the accuracy of ResNet-20 trained using lookup tables with different granularities. When $K = 1$, our lookup tables become fixed and degrade to vanilla $\mathrm{round}(\cdot)$ function. Consequently, this variant suffers inferior accuracy. With larger granularity, the degree of freedom for our lookup tables is increased such that better performance is achieved. Since granularity larger than 9 does not provide further notable improvement, $K$ is set to 9 as the default setting.

**Efficiency.** We used 4-bit ResNet-18 to evaluate the efficiency of our lookup tables. Intel i9-9900K and Kirin 810 are used as platforms of CPU and mobile processor. It can be observed from Table 7 that the additional memory cost of our lookup tables is very small (1.25KB). Moreover, activation quantization using our lookup operation is much faster than QIL and QNet. This is because, complicated quantizers in QIL and QNet introduce considerable computational overhead during inference. In contrast, the additional computational cost of our lookup tables is very small, which helps to achieve better inference efficiency.

Table 6. Top-1 accuracy (%) achieved on CIFAR-10 with different granularities. ResNet-20 is used as the baseline.

| Model | Granularity ($K$) | Bit-width (W/A) | | |
|---|---|---|---|---|
| | | 4/4 | 3/3 | 2/2 |
| ResNet-20 (full-precision: 92.96)) | 1 | 92.37 | 91.95 | 90.42 |
| | 5 | 92.60 | 92.14 | 90.58 |
| | 9 | 92.71 | **92.17** | **90.63** |
| | 13 | **92.72** | 92.15 | 90.62 |

Table 7. Results achieved by 4-bit ResNet-18 on ImageNet. Additional memory consumption of our lookup tables is shown in brackets. Running time for activation quantization is presented.

| Method | Model Size | Time | | Acc |
|---|---|---|---|---|
| | | CPU | Mobile | |
| QIL [14] | 7553.7KB | 90ms | 120ms | 70.1 |
| QNet [21] | 7553.7KB | 85ms | 95ms | 69.7 |
| LLT (Ours) | (+1.25KB) 7554.9KB | **20ms** | **40ms** | **70.4** |

## 5. Conclusion

In this paper, we formulate the quantization process as a lookup operation and propose to learn lookup tables for network quantization. Specifically, we develop differentiable lookup tables and introduce several training strategies for optimization. Our lookup tables can be trained with the network in an end-to-end manner for flexible adaption to the distributions of weights and activations. Moreover, our lookup tables have very small computational and memory cost. Experiments on image classification, image SR, and point cloud classification tasks demonstrate that our method is generic and achieves state-of-the-art performance.

**Acknowledgments:** This work was partially supported by the National Key Research and Development Program of China (No. 2021YFB3100800) and the National Natural Science Foundation of China (No. U20A20185, 61972435).

## References

[1] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pages 116–131, 2018. 1

[2] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 1

[3] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114, 2019. 1

[4] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI*, pages 2234–2240, 2018. 1

[5] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*, pages 4340–4349, 2019. 1

[6] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *CVPR*, pages 1529–1538, 2020. 1

[7] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *NeurIPS*, 2015. 1

[8] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *CVPR*, pages 7370–7379, 2017. 1

[9] Junho Yim, Donggyu Joo, Ji-Hoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, pages 7130–7138, 2017. 1

[10] Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. Knowledge distillation meets self-supervision. In *ECCV*, 2020. 1

[11] Yifan Liu, Changyong Shu, Jingdong Wang, and Chunhua Shen. Structured knowledge distillation for dense prediction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. 1

[12] Shijie Cao, Lingxiao Ma, Wencong Xiao, Chen Zhang, Yunxin Liu, Lintao Zhang, Lanshun Nie, and Zhi Yang. Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization. In *CVPR*, pages 11216–11225, 2019. 1

[13] Peisong Wang, Qinghao Hu, Yifan Zhang, Chunjie Zhang, Yang Liu, and Jian Cheng. Two-step quantization for low-bit neural networks. In *CVPR*, pages 4376–4384, 2018. 1

[14] 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. 1, 2, 4, 5, 6, 8

[15] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283, 2016. 1

[16] 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. 1, 2, 5, 6

[17] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv*, 2018. 1, 2, 5, 6

[18] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *ICLR*, 2019. 1, 2, 4, 5, 6

[19] Zhaohui Yang, Yunhe Wang, Kai Han, Chunjing Xu, Chao Xu, Dacheng Tao, and Chang Xu. Searching for low-bit weights in quantized neural networks. In *NeurIPS*, 2020. 1, 2, 5

[20] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *ICLR*, 2019. 1, 2, 4, 5, 8

[21] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *CVPR*, pages 7308–7316, 2019. 1, 2, 8

[22] 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.

[23] Fabien Cardinaux, Stefan Uhlich, Kazuki Yoshiyama, Javier Alonso García, Lukas Mauch, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Iteratively training look-up tables for network quantization. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):860–870, 2020.

[24] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, 2015. 2

[25] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *ICLR*, 2017. 2

[26] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542, 2016. 2

[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*, 2016. 2, 5, 6

[28] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. In *ICLR*, 2017. 2

[29] Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. Deep neural network compression with single and multiple level quantization. In *AAAI*, volume 32, 2018. 2

[30] Kohei Yamamoto. Learnable companding quantization for accurate low-bit neural networks. In *CVPR*, pages 5029–5038, 2021. 2, 5

[31] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *ECCV*, pages 365–382, 2018. 2, 5

[32] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *ICLR*, 2017. 2

[33] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv*, 2016. 2

[34] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, pages 2472–2481, 2018. 2, 5

[35] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *CVPR*, 2019. 2

[36] Yiqun Mei, Yuchen Fan, Yuqian Zhou, Lichao Huang, Thomas S Huang, and Honghui Shi. Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining. In *CVPR*, pages 5690–5699, 2020. 2

[37] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACM MM*, 2019. 2

[38] Yinglan Ma, Hongyu Xiong, Zhe Hu, and Lizhuang Ma. Efficient super resolution using binarized neural network. In *CVPRW*, pages 0–0, 2019. 2

[39] Wonkyung Lee, Junghyup Lee, Dohyung Kim, and Bumsub Ham. Learning with privileged information for efficient image super-resolution. In *ECCV*, 2020. 2

[40] Jingwei Xin, Nannan Wang, Xinrui Jiang, Jie Li, Heng Huang, and Xinbo Gao. Binarized neural network for single image super resolution. In *ECCV*, pages 91–107. Springer, 2020. 2

[41] Huixia Li, Chenqian Yan, Shaohui Lin, Xiawu Zheng, Yuchao Li, Baochang Zhang, Fan Yang, and Rongrong Ji. Pams: Quantized super-resolution via parameterized max scale. In *ECCV*, 2020. 2, 6

[42] Longguang Wang, Xiaoyu Dong, Yingqian Wang, Xinyi Ying, Zaiping Lin, Wei An, and Yulan Guo. Exploring sparsity in image super-resolution for efficient inference. In *CVPR*, 2021. 2

[43] Cheeun Hong, Heewon Kim, Junghun Oh, and Kyoung Mu Lee. Daq: Distribution-aware quantization for deep image super-resolution networks. *arXiv*, 2020. 2, 6

[44] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 2, 6

[45] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, volume 31, pages 820–830, 2018. 2

[46] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *ICCV*, pages 1607–1616, 2019. 2

[47] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, pages 6411–6420, 2019. 2

[48] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2, 6

[49] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *CVPR*, pages 2626–2635, 2018. 2

[50] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, pages 4548–4557, 2018. 2

[51] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, pages 10296–10305, 2019. 2

[52] Haotong Qin, Zhongang Cai, Mingyuan Zhang, Yifu Ding, Haiyu Zhao, Shuai Yi, Xianglong Liu, and Hao Su. Bipointnet: Binary neural network for point clouds. In *ICLR*, 2021. 2, 6

[53] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv*, 2013. 4

[54] Jungwook Choi, Swagath Venkataramani, Vijayalakshmi Srinivasan, Kailash Gopalakrishnan, Zhuo Wang, and Pierce Chuang. Accurate and efficient 2-bit quantized neural networks. In *MLSys*, 2019. 5

[55] Jung Hyun Lee, Jihun Yun, Sung Ju Hwang, and Eunho Yang. Cluster-promoting quantization with bit-drop for minimizing network quantization loss. In *ICCV*, pages 5370–5379, 2021. 5, 6

[56] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *CVPR*, pages 7920–7928, 2018. 5

[57] A. Krizhevsky and G. E. Hinton. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[59] 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. 5

[60] 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. 5

[61] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *NeurIPS*, volume 30, 2017. 5, 6

[62] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, pages 1122–1131, 2017. 5

[63] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, pages 1–10, 2012. 5

[64] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, volume 6920, pages 711–730, 2010. 5

[65] David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 5

[66] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015. 5

[67] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR*, 2017. 5

[68] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. 6