Learnable Companding Quantization for Accurate Low-bit Neural Networks

Kohei Yamamoto
Oki Electric Industry Co., Ltd.
yamamoto833@oki.com

Abstract

Quantizing deep neural networks is an effective method for reducing memory consumption and improving inference speed, and is thus useful for implementation in resource-constrained devices. However, it is still hard for extremely low-bit models to achieve accuracy comparable with that of full-precision models. To address this issue, we propose learnable companding quantization (LCQ) as a novel non-uniform quantization method for 2-, 3-, and 4-bit models. LCQ jointly optimizes model weights and learnable companding functions that can flexibly and non-uniformly control the quantization levels of weights and activations. We also present a new weight normalization technique that allows more stable training for quantization. Experimental results show that LCQ outperforms conventional state-of-the-art methods and narrows the gap between quantized and full-precision models for image classification and object detection tasks. Notably, the 2-bit ResNet-50 model on ImageNet achieves top-1 accuracy of 75.1% and reduces the gap to 1.7%, allowing LCQ to further exploit the potential of non-uniform quantization.

1. Introduction

Deep neural networks (DNNs) have been successfully applied to image-based tasks such as image classification and object detection, but their implementation in resource-constrained mobile or edge devices remains difficult, owing to the large number of required multiply–accumulate (MAC) operations and parameters. To mitigate this problem, various techniques for compressing DNNs while maintaining performance have been proposed, such as pruning [9], knowledge distillation [15], low-rank approximation [10], and network quantization [16]. Among these, network quantization is important as a way to effectively improve both memory consumption and inference speed. However, network quantization is known to degrade performance of the original model in proportion to the amount of bit-width reduction.

In network quantization, the weights or activations of DNNs are typically discretized by a quantization function. Although the quantization function is not differentiable, a straight-through estimator (STE) [1] can be used to approximate the gradient calculation for backpropagation. Quantization functions are divided into two types: uniform and non-uniform quantization, in which input values are respectively linearly and nonlinearly discretized. Because the weight or activation distribution is empirically dissimilar to the uniform distribution, non-uniform quantization can be expected to further reduce quantization and prediction errors than can uniform quantization via proper optimization. For example, previous works on non-uniform quantization have attempted to use fixed and logarithmic quantization levels [16,22] or learnable quantization levels that minimize quantization errors [30].

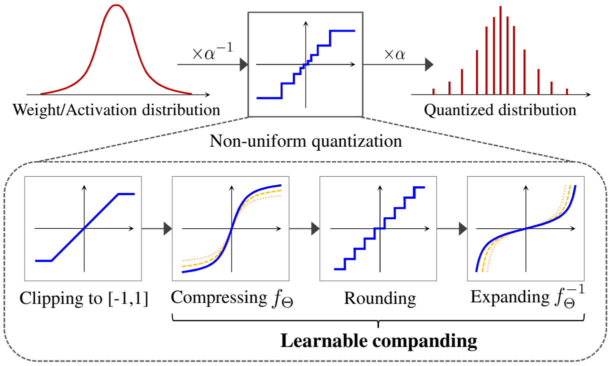


Figure 1: An overview of the proposed method. Our non-uniform quantizer quantizes weights or activations with four functions, those for clipping, compressing, rounding, and expanding. In particular, a composite function consisting of those except for the clipping function is generally called the companding function. We formulate the companding function in a learnable form with a set of parameters Θ and jointly optimize it with clipping parameter α and the other parameters in the model.

However, it is not easy to estimate effective quantization levels accurately, especially in low-bit models, where accuracy is often inferior to that of uniform quantization methods. This paper thus aims to exploit the potential of non-uniform quantizers and further bridge the accuracy gap between quantized and full-precision models.

We propose a non-uniform quantization method called learnable companding quantization (LCQ). Figure 1 shows an overview of LCQ. Our method is based on a companding (compressing and expanding) technique [21] that is widely used in telecommunication and signal processing to reduce the bit-width of input signals by nonlinearly limiting their dynamic range. We assume that companding is effective for quantization of DNNs in two aspects. The first is that the scale remains unchanged between inputs and outputs by using a nonlinear function and its inverse function, and that maintaining scale reduces quantization error and stabilizes training via backpropagation. The second is that if the companding function is differentiable, its parameters can be optimized to directly minimize task loss. Then, since the parameters are updated with a sum of the two gradients from the paths of before and after rounding, they can be trained with a large quantization influence. Specifically, we formulate a learnable piecewise linear function as a nonlinear compressing function, allowing flexible and non-uniform control of quantization levels by optimization.

We also propose a new weight normalization method that improves accuracy by restoring the standard deviation of quantized weights to its original level, and we discuss a training trick for efficient inference with lookup tables.

Our main contributions are summarized as follows:

- We propose a LCQ method as a novel non-uniform quantization method, which optimizes non-uniform quantization levels to minimize task loss by training the learnable companding function.
- We present a simple normalization method for weight quantizers called limited weight normalization (LWN) that results in stable training and better solutions.
- We evaluate LCQ on various networks for image classification and object detection tasks, and the results show promising performance on the CIFAR-10/100, ImageNet, and COCO datasets.

2. Related Works

To quantize weights and activations in training, two operations are often applied in sequence: "clipping" and "quantizing" of inputs. Quantization errors occur in each process, so various methods have been proposed to reduce them.[1]

Clipping technique. To perform quantization, clipping is first applied to constrain the value range of inputs. The simplest way to determine the clipping threshold is to use a given fixed value, but doing so does not adapt to variations in the dynamic range of the input values during training. To address this issue, Jacob *et al*. [11] proposed a method that uses as the threshold the maximum value of the input tracked by the exponential moving averaging. Choi *et al*. [2] proposed a method that treats the threshold as a learnable parameter, optimizing it to minimize the task loss along with the weights. Zhao *et al*. [31] used a simulated gradient to estimate the near-optimal threshold in every iteration. Several prior works [4, 16, 27] have proposed improved formulations of the learnable threshold approach, updating the parameter with a gradient calculated from residuals between the pre-quantized and quantized values.

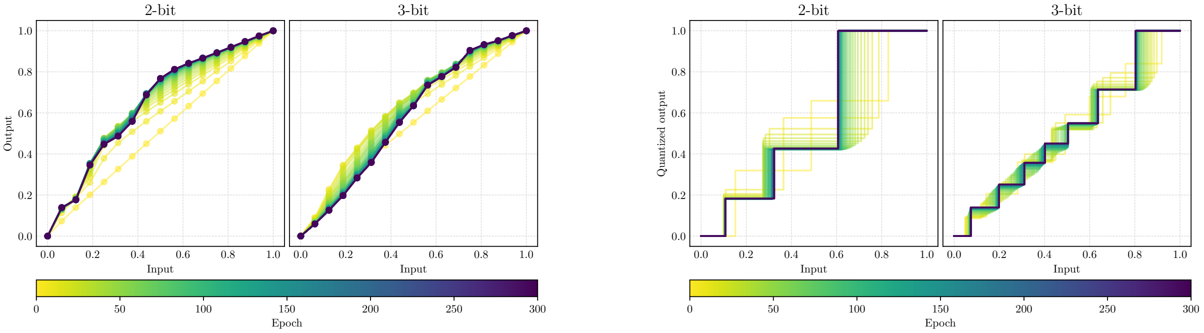
Uniform quantization. Uniform quantization maps a clipped value to one of equally spaced discrete levels. Although such mapping is performed with a nondifferentiable step function, STE [1] is often applied to approximate the gradient calculation and to enable parameter updates based on backpropagation. Gong *et al*. [5] proposed a method for mitigating gradient approximation errors incurred by using STE, representing the quantization function as a concatenation of several tanh functions and training their shape parameter to gradually converge on the step function. Li *et al*. [14] applied uniform quantization to object detection models with batch normalization folding. Zhuang *et al*. [33] proposed a progressive two-stage quantization approach. Jung *et al*. [12] introduced parameterized quantization intervals and optimized them to minimize task loss. Liu *et al*. [19] used a scheme that does not apply STE, instead using a weighted average of pre- and post-quantization values to gradually shift to the quantized values. Zhuang *et al*. [32] proposed a training scheme using an auxiliary module connected to a low-bit network, providing it with the gradient from other loss.

Non-uniform quantization. Since DNN weights and activities are empirically non-uniformly distributed, non-uniform quantization, which discretizes inputs into unequal levels, should work effectively. Han *et al*. [7] uses k-means clustering as a method of quantization to share weights. Xu *et al*. [29] applies the same clustering strategy, but by gradually sharing weights in a layer-by-layer manner. Miyashita *et al*. [22] introduced non-uniform quantization using a powers-of-two (PoT) function and showed that multiplication in DNNs can be replaced by cheaper bit-shift operations. Polino *et al*. [25] formulated quantization levels as learnable parameters and trained them with gradient descent and distillation methods. Zhang *et al*. [30] proposed parameterized bases for quantization levels and sequentially estimated an analytical solution that minimizes quantization error. Li *et al*. [16] proposed an additive PoT quantizer to solve the problem of PoT functions that map extremely low quantization levels to larger input values.

3. Method

In this section, we first provide a brief background of network quantization. We then discuss details of the proposed method, including formulation of the LCQ quantizer, the LWN method, and a training trick for efficient inference.

[1] Relations to LCQ are also explained in the supplementary material.



(a) Compressing function f_Θ

(b) Companding function g

Figure 2: Examples of the evolutionary process for the compressing (*left*) and companding (*right*) functions via training the LCQ quantizer for the ResNet-20 model on the CIFAR-10 dataset. Note that the 2- and 3-bit results of the companding function are generated using the compressing function with a corresponding number of bits.

3.1. Preliminaries

The goal of network quantization is to replace floating-point weights and activations for DNNs with lower bit-width ones to reduce memory consumption and speed up MAC operations. For uniform quantization, a standard uniform quantizer $Q_U(x;\alpha)$ quantizes an input $x \in \mathbb{R}$ as

$$Q_U(x;\alpha) = \mathrm{sgn}(x) \cdot \alpha \begin{cases} q_b\left(\frac{|x|}{\alpha}\right), & |x| < \alpha, \\ 1, & |x| \geq \alpha, \end{cases} \tag{1}$$

where $\alpha \in \mathbb{R}_{>0}$ is a clipping parameter, $q_b(\cdot)$ is a uniform quantization function, and the subscript $b \in \mathbb{N}$ is the bit-width. This clipping operation reduces quantization error by multiplying the quantized value by α again to return it to its original value range. Letting the clipped input be $v \in [0, 1)$, $q_b(v)$ can be represented as

$$q_b(v) = \frac{\lfloor s \cdot v \rceil}{s}, \tag{2}$$

where the scaling factor s becomes $s = 2^{b-1} - 1$ in the case of signed quantization or $s = 2^b - 1$ in the case of unsigned quantization, and $\lfloor \cdot \rceil$ is a rounding function. The quantization function $q_b(v)$ is not differentiable, because it contains a rounding function, but can be relaxed by STE [1] as $\partial q_b(v)/\partial v = 1$. Similarly, the gradient for input x does not vanish through the quantizer due to $\partial Q_U(x;\alpha)/\partial x = 1$ for $|x| < \alpha$. When applying this quantizer to convolutional neural networks (CNNs), the weights and activations are independently quantized, just before the convolutional operations. Then MAC operations in the convolution can be transformed to execute with integer precision, which can speed up the inference process [11].

3.2. Learnable Companding Quantization

As Fig. 1 shows, our proposed LCQ is a non-uniform quantization method, without clipping, which mainly consists of three functions: a compressing function $f_\Theta(\cdot)$, a uniform quantization function $q_b(\cdot)$, and an expanding function $f_\Theta^{-1}(\cdot)$. A composite function formed from these three is generally called the companding function. Note that $\Theta = \{\theta_1, \theta_2, \ldots, \theta_K\}$ is a common set of parameters for both functions. Here, we denote the LCQ quantizer as

$$Q_L(x;\alpha,\Theta) = \mathrm{sgn}(x) \cdot \alpha \begin{cases} g\left(\frac{|x|}{\alpha}\right), & |x| < \alpha, \\ 1, & |x| \geq \alpha, \end{cases} \tag{3}$$

where

$$g(v) = (f_\Theta^{-1} \circ q_b \circ f_\Theta)(v) \tag{4}$$

is the companding function. We then use a learnable piecewise linear function as the compressing function $f_\Theta(\cdot)$ and train its parameters Θ to minimize task loss. The piecewise linear function is suited to fine-grained control of quantization levels, because it flexibly changes its slopes in proportion to the number of breakpoints (or intervals). For example, Fig. 2 shows the evolutionary processes of the piecewise linear function (Fig. 2a) and its companding function (Fig. 2b) at the different bit-widths. These figures show that the quantization levels and intervals of the companding function can be finely determined by changes in the slope of each interval of the piecewise linear function. In this way, we generate accurate low-bit DNNs by giving the model the capability to directly tune the quantization levels.

3.2.1 Detailed formulation

Specifically, such a piecewise linear function needs to be monotonically increasing and to satisfy the constraint of an input range of $[0, 1)$ to account for the quantization function. In our formulation, we first let the breakpoints that form the k-th interval (where $k \in \{1, 2, \ldots, K\}$) be equally spaced, meaning all interval lengths $\Delta = 1/K$. We then prepare learnable parameters $\theta_k \in \Theta$ and use the softmax function to restrict their value range to $[0,1]$, as in $\tilde{\theta}_k = \exp(\theta_k)/\sum_{i=1}^{K}\exp(\theta_i)$. We further define the slope of the k-th interval as $\gamma_k = \tilde{\theta}_k/\Delta$, the total length of the k-th interval as $d_k = k\Delta$, and the cumulative sum of the output levels as $\beta_k = \sum_{i=1}^{k}\tilde{\theta}_i$, and we set $d_0 = 0$ and $\beta_0 = 0$. With the above preparation, the piecewise linear compressing and expanding functions can be formulated as

$$f_\Theta(v) = \sum_{k=1}^{K} \left(\gamma_k(v - d_{k-1}) + \beta_{k-1}\right) \mathbb{1}_{[d_{k-1},d_k)}(v), \quad (5)$$

$$f_\Theta^{-1}(v) = \sum_{k=1}^{K} \left(\frac{v - \beta_{k-1}}{\gamma_k} + d_{k-1}\right) \mathbb{1}_{[\beta_{k-1},\beta_k)}(v), \quad (6)$$

where $\mathbb{1}_{\mathcal{C}}(v)$ is an indicator function that returns 1 if $v \in \mathcal{C}$ and 0 otherwise. We finally use a gradient descent algorithm to optimize θ_k through γ_k and β_k.

3.2.2 Backpropagation for companding

By the chain rule, the gradient of our quantizer $Q_L(\cdot)$ with respect to θ_k can be written as

$$\frac{\partial Q_L}{\partial \theta_k} = \left(\frac{\partial Q_L}{\partial \gamma_k}\frac{\partial \gamma_k}{\partial \tilde{\theta}_k} + \frac{\partial Q_L}{\partial \beta_k}\frac{\partial \beta_k}{\partial \tilde{\theta}_k}\right)\frac{\partial \tilde{\theta}_k}{\partial \theta_k}. \quad (7)$$

Here, the gradients of $Q_L(\cdot)$ with respect to γ_k and β_k should be carefully calculated, because the compressing and expanding function may use intervals with different correspondences when going through the quantization function. For simplicity, let v_q be the output of the quantization and compressing function, i.e., $v_q = (q \circ f_\Theta)(v)$. Then the gradients of the companding function $g(v)$ can be represented as

$$\frac{\partial g(v)}{\partial \gamma_k} \simeq \sum_{i=1}^{K} \left(\frac{v - d_{k-1}}{\gamma_i} I_{(k,i)} - \frac{v_q - \beta_{k-1}}{\gamma_k^2} I_{(i,k)}\right), \quad (8)$$

$$\frac{\partial g(v)}{\partial \beta_k} \simeq \sum_{i=1}^{K} \left(\frac{I_{(k,i)}}{\gamma_i} - \frac{I_{(i,k)}}{\gamma_k}\right), \quad (9)$$

where

$$I_{(i,j)} = \mathbb{1}_{[d_{i-1},d_i)}(v) \cdot \mathbb{1}_{[\beta_{j-1},\beta_j)}(v_q). \quad (10)$$

Note that we use the STE [1] approximation for the derivative of the quantization function, and that v_q may exceed the upper bound on the value range $[0,1)$ due to rounding, but in that case an inifinitesimal value ε is subtracted from v_q to keep it within the range. The gradient of $Q_L(\cdot)$ with respect to γ_k can then be written as

$$\frac{\partial Q_L}{\partial \gamma_k} \simeq \begin{cases} \operatorname{sgn}(x) \cdot \alpha \cdot \frac{\partial g\left(\frac{|x|}{\alpha}\right)}{\partial \gamma_k}, & |x| < \alpha, \\ 0, & |x| \geq \alpha. \end{cases} \quad (11)$$

Similarly, $\partial Q_L/\partial \beta_k$ is the replacement of γ_k by β_k in Eq. (11). Since the gradient contains the clipping parameter α, the clipping effect is considered in the optimization of θ_k.

The gradient of $g(\cdot)$ with respect to the clipped input v is similarly affected by the quantization function as $\partial g(v)/\partial v = \sum_{i=1}^{K}\sum_{j=1}^{K}\gamma_i/\gamma_j \cdot I_{(i,j)}$. However, since we have empirically found that the gradient may be too large when γ_j is small and γ_i is large, and that such a gradient makes the training unstable, we use $\partial g(v)/\partial v = 1$ instead. Then the gradient of the quantizer Q_L with respect to the input x becomes $\partial Q_L/\partial x = 1$ for $|x| < \alpha$ and 0 otherwise, like the uniform quantizer. This strategy of not modifying ("straight-throughing") the gradient for inputs has been used for other non-uniform quantization methods as well [16, 30].

3.2.3 Backpropagation for clipping

We estimate the clipping parameter α based on training, as in previous works [4,16,27]. We specifically update α based on the gradient of our quantizer $Q_L(\cdot)$, represented as

$$\frac{\partial Q_L}{\partial \alpha} \simeq \operatorname{sgn}(x) \begin{cases} g\left(\frac{|x|}{\alpha}\right) - \frac{|x|}{\alpha}, & |x| < \alpha, \\ 1, & |x| \geq \alpha. \end{cases} \quad (12)$$

Note that α is jointly trained with companding parameters Θ. When parameterizing the compressing function, breakpoints related to the input interval are set to be equally spaced rather than trained, thereby preventing changes in the clipping parameter from having a significant effect on breakpoint locations, which would reduce the training efficiency. Using equal spacing is less flexible, but this can be compensated for by increasing the number of breakpoints (or intervals).

3.3. Limited Weight Normalization

Li *et al.* [16] reported that clipping parameter training for each layer is stabilized when the weights are standardized with their mean and standard deviation before applying the quantizer. There are two main reasons for this: because the weight distribution is zero-centered, satisfying the symmetry assumption in signed quantization, and because the clipping parameters are less sensitive to variations in the standard deviation.

However, considering that the quantized model is initialized with pretrained, full-precision weights to obtain good accuracy in many quantization methods [4, 12, 16], weight normalization causes a gap for the output scale of the linear layer before and after quantization, which may negatively affect training. Therefore, we also propose a method called limited weight normalization (LWN), which limits only the effective scope of normalization to the weight quantizer.

LWN can be formulated as

$$\tilde{w} = \sigma_w \cdot Q_* \left(\frac{w - \mu_w}{\sigma_w} \right), \tag{13}$$

where $w \in \mathbb{R}$ is an input weight, μ_w and σ_w are the sample mean and standard deviation of the weights, respectively, and $Q_*(\cdot)$ is any signed quantizer. Note that the gradients for μ_w and σ_w are not used to update all learnable parameters, as in [16]. The only difference between LWN and [16] is whether the standard deviation is multiplied after quantization or not. This simple method has the effect of not only restoring the standard deviation to its pre-normalized level in forward propagation, but also making the gradients for learnable quantizer parameters depend on the standard deviation in backpropagation. We have empirically observed that this method is more stable and gives better solutions.

3.4. Training for the LUT-based Inference

In general, non-uniform quantization functions, including the companding function, output floating-point values, so the multiplication between weights and activations is also performed in floating-point, which is inefficient. To speed up the operations during inference in deployment scenarios, it is often used to replace the multiplication with a memory access to a precomputed lookup table (LUT), as shown in Fig. 3. However, our method requires one LUT per convolutional or fully-connected layer, and thus incurs additional memory costs. For example, with b_w and b_a for the number of bits in the signed weights and in the unsigned activations, the number of LUT elements m becomes $m = (2^{b_w - 1} - 1)(2^{b_a} - 1)$.[2] Therefore, the additional memory cost per layer is simply $4m$ bytes for multiplication at the 32-bit floating-point precision. The memory cost of LUTs should be reduced as much as possible, because this cost relates to the memory access speed and the accumulator capacity on dedicated devices such as field-programmable gate arrays (FPGAs).

To reduce the memory cost of LUTs, we apply the b'-bit uniform quantization function $q_{b'}(\cdot)$ immediately after the companding function $g(\cdot)$, where $b' \in \{t \in \mathbb{N} \mid t > b\}$ is the other bit-width (below, we call this the **"outer" bit-width** for clarity). Although this re-quantization introduces an extra quantization error, the effect on the accuracy is almost negligible if the outer bit-width is sufficiently larger than the original bit-width b, considering that quantization-aware training tends to quantize to around 8 bits with nearly no degradation in accuracy [11]. Specifically, instead of Eq. (3) for training, we use a slightly modified version of the quantizer $Q_L(\cdot)$,

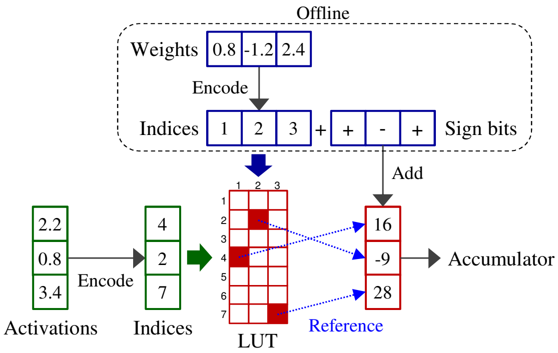


Figure 3: An example of the memory access with a LUT for inference (3-bit case). The weights can be pre-converted to the encoded low-bit indices. The LUT has precomputed multiplicative values for all output combinations generated by the weight and activation quantizers, and does not include zeros because they can simply be skipped.

$$Q'_L(x; \alpha, \Theta) = \operatorname{sgn}(x) \cdot \alpha \begin{cases} (q_{b'} \circ g) \left(\frac{|x|}{\alpha} \right), & |x| < \alpha, \\ 1, & |x| \geq \alpha. \end{cases} \tag{14}$$

Note that since STE [1] is used for $q_{b'}(\cdot)$, all the backpropagation formulas in Sec. 3.2 can also be applied to the quantizer $Q'_L(\cdot)$ as-is.[3] Since all the scalar multiplications (e.g., by the clipping parameter α, the scaling factor s and the standard deviation σ_w in LWN) after re-quantization can be moved after convolution at the inference time, the convolution can be performed with integer precision. For the integer multiplication in the convolution, the bit-width of LUT elements is equal to the sum of the outer bit-width of activations and the one of weights. Therefore, the more the outer bit-width is reduced, the more the bit-width of the LUT elements is reduced. For example, letting the outer bit-widths for weights and activations be b'_w and b'_a, as before, the memory cost of a LUT can be represented as $2^{-3}(b'_w + b'_a)m$ bytes. The memory cost for $b'_w = b'_a = 8$ is two times smaller than the one for the 32-bit floating-point case. The effect of changing the outer bit-width on accuracy is evaluated in the ablation study in Sec. 4.5.

4. Experiments

This section evaluates the effectiveness of our method in comparison with conventional state-of-the-art uniform and non-uniform quantization methods using various models, such as ResNet [8], MobileNet-V2 [26], and Reti-

[2] Note that the sign bit is reduced because it can be applied afterwards, and the number of zeros is also reduced.

[3] The training algorithm for a convolutional layer using $Q'_L(\cdot)$ is summarized in the supplementary material.

naNet [17].[4] We then report the results of ablation studies.

To evaluate our method, we used the CIFAR-10/100 [13] and ImageNet (ILSVRC-2012) [3] datasets for image classification tasks and the MS COCO [18] dataset for an object detection task. The CIFAR-10/100 dataset contains 50k training images and 10k test images, with 10/100 classes. The ImageNet dataset contains 1.2M training images and 50k test images, with 1,000 classes. For the COCO dataset with 80 object classes, following [17, 32] we used the *trainval35k* split (115k training images) and *minival* split (5k test images). All experiments were implemented using PyTorch [24] and Cupy [23], and for the object detection task we also used Detectron2 [28].

4.1. Implimention Details

Unless otherwise specified, we used the following settings in all experiments.

Basic settings. We used signed and unsigned quantizers for weights and activations, respectively. Note that a 2-bit signed quantization for weights implies ternarization. We instead applied the uniform quantizer $Q_U(\cdot)$ only for 2-bit weights (not for 2-bit activations), because ternarization eliminates the effect of companding. Although quantization of the first and last layers significantly impacts accuracy, as in [16] we applied 8-bit quantization to both for further efficiency. For both weights and activations, the number of intervals in the companding function and the outer bit-widths were set to $K = 16$ and $b' = 8$, respectively.

Optimization. We used the stochastic gradient descent algorithm with a Nesterov momentum of 0.9 and cosine learning rate decay without restart [20]. All weights in the quantized model were initialized with pretrained weights at full precision, and we did not use the progressive initialization strategy [12, 33]. The clipping parameters for weights and activations were initialized as 3.0 and 8.0, respectively, and all companding parameters were initialized as 0; this is equivalent to the uniform quantization setting. All gradients with respect to the clipping function for the weights were not zeroed out by applying STE [1], as in [16].

Architecture. Our unsigned activation quantizer can place a learnable upper bound on the inputs, but not a learnable lower bound. However, a lower bound can be applied by adding a learnable bias term just before the unsigned quantization. Implementationally, the bias term can be introduced by using batch normalization. As in [4], therefore, we used ResNet [8] with pre-activation as a target architecture satisfying this condition. We also used the same configuration as that of ResNet for the inverted residual blocks of MobileNet-V2 [26].

4.2. Evaluation on CIFAR-10

We performed experiments for ResNet-20/56 on the CIFAR-10 dataset, training the quantized models over 300 epochs with an initial learning rate of 0.04 for the weights and 0.02 for the clipping and companding parameters, and with a mini-batch size of 128. The weight decay was set to 10^{-4}. We adopted standard data augmentation techniques, namely random crop and horizontal flip.

Table 1: Top-1 accuracy (%) for the 2/3/4-bit ResNet on the CIFAR-10 dataset.

Model	Method	W2/A2	W3/A3	W4/A4
ResNet-20 (FP: 93.4)	PACT [2]	89.7	91.1	91.7
	LQ-Nets [30]	90.2	91.6	-
	APoT [16]	91.0	92.2	92.3
	LCQ (Ours)	**91.8**	**92.8**	**93.2**
ResNet-56 (FP: 94.5)	APoT [16]	92.9	93.9	94.0
	LCQ (Ours)	**93.5**	**94.6**	**94.7**

Table 1 compares the accuracy of the proposed and conventional methods at three bit-widths for the CIFAR-10 dataset. In the table, for example, "W2/A2" indicates the case where the weights and activations are both quantized to 2 bits, "FP" indicates accuracy in the full precision case, and "-" indicates no reported result. For ResNet-20, our LCQ shows better performance than do the uniform quantization method PACT [2] and the non-uniform quantization methods LQ-Nets [30] and APoT [16] at all bit-widths from 2 to 4. Figure 2b shows examples of trained quantization levels for the 2/3-bit ResNet-20. As shown in the 3-bit case in the figure, quantization levels for an input of around 0.4 are relatively dense and indicate an important value range for loss reduction. The LCQ results were also better for ResNet-56 than was the APoT method. Although APoT has fine-grained quantization levels due to the powers-of-two combination, unlike LCQ, the levels are not learnable.

4.3. Evaluation on ImageNet

We evaluated the performance of LCQ for ResNet-18/34/50 and MobileNet-V2 on the ImageNet dataset. With an initial learning rate of 0.1 for the weights and an initial learning rate of 0.01 for the clipping and companding parameters, the models were trained over 120 epochs with a mini-batch size of 1024 for ResNet-18/34 and 512 for both ResNet-50 and MobileNet-V2. In addition, we applied a warm-up method [6] for the first 5 epochs and increased the learning rate linearly from 10^{-4} to the initial value. The weight decay was set to 4×10^{-5}. The training images were resized, cropped to 224 × 224 pixels and randomly flipped horizontally. The test images were center-cropped to 224 × 224 pixels.

Table 2 compares quantization performance with state-

[4] A discussion of the validity of the comparison between LCQ and uniform quantization methods is provided in the supplementary material.

Table 2: Top-1 accuracy (%) for the 2/3/4-bit ResNet on the ImageNet dataset.

Model	Method	W2/A2	W3/A3	W4/A4
ResNet-18 (FP: 70.4)	LQ-Nets [30]	64.9	68.2	69.3
	DSQ [5]	65.2	68.7	69.6
	QIL [12]	65.7	69.2	70.1
	APoT [16]	67.3	69.9	70.7
	LSQ [4]	67.6	70.2	71.1
	LCQ (Ours)	**68.9**	**70.6**	**71.5**
ResNet-34 (FP: 74.2)	LQ-Nets [30]	68.8	71.9	-
	DSQ [5]	70.0	72.5	72.8
	QIL [12]	70.6	73.1	73.7
	APoT [16]	70.9	73.4	73.8
	LSQ [4]	71.6	73.4	74.1
	LCQ (Ours)	**72.7**	**74.0**	**74.3**
ResNet-50 (FP: 76.8)	PTG. [33]	70.0	-	75.7
	LQ-Nets [30]	71.5	74.2	75.1
	APoT [16]	73.4	75.8	76.6
	LSQ [4]	73.7	75.8	**76.7**
	Auxi [32]	73.8	-	-
	LCQ (Ours)	**75.1**	**76.3**	76.6

of-the-art conventional methods. For almost all models, LCQ outperformed the conventional uniform [4, 5, 12, 32, 33] and non-uniform [16, 30] methods on the ImageNet validation set. Accuracy improvements over the conventional methods at 2 bits were particularly remarkable, with a maximum improvement of 1.3% points for both ResNet-18 and ResNet-50. These results suggest that fine-tuning of the quantization levels by the companding function improved accuracy in relatively large datasets.

Table 3: Accuracy (%) of the 4-bit MobileNet-V2 on the ImageNet dataset.

Model	Method	Top-1 acc.	Top-5 acc.
MobileNet-V2 (FP: 71.9)	DSQ [5]	64.8	-
	LLSQ [31]	67.4	88.0
	LCQ (Ours)	**70.8**	**89.7**

As Table 3 shows, we observed that LCQ achieved relatively good accuracy even for a compact and efficient architecture, 4-bit MobileNet-V2 (W4/A4). Due to the low redundancy, the accuracy difference from the full-precision model was more than 1% point and was not as close as in the case of the ResNet models in Table 2.

4.4. Evaluation on COCO

We used RetinaNet [17] with ResNet as the backbone to evaluate the proposed method on the COCO dataset. An initial learning rate of 0.005 was used for the weights and 0.001 for both the clipping and companding parameters. The weight decay was set to 10^{-4} and the warm-up method [6], which increases the learning rate linearly from 0 to the initial value, was used for first 1k iterations. The batch size was set to 16 and the models were trained over 90k iterations. We resized both training and test images to have 800 pixels on shorter edges, randomly flipping the training images horizontally as data augmentation. Following the observation in [32], the prediction head of RetinaNet was not shared between features of different resolutions, except for the final layers for regression and classification. In addition, we inserted the batch normalization just before all convolutional layers for both the feature pyramid network (FPN) and the prediction heads, and synchronously updated all batch statistics during training. We did not quantize only the last layers. All other settings were in accordance with the original settings in [17].

Table 4: APs for the 4-bit RetinaNet on the COCO dataset.

Backbone	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	FP	33.2	52.3	34.8	18.7	35.6	43.7
	FQN [14]	28.6	46.9	29.9	14.9	31.2	38.7
	Auxi [32]	31.9	50.4	33.7	16.5	34.6	**42.3**
	APoT [16]	32.4	51.2	34.0	18.4	34.6	42.2
	LCQ (Ours)	**32.7**	**51.7**	**34.2**	**18.6**	**35.2**	**42.3**
ResNet-34	FP	37.2	57.0	39.4	21.4	40.4	48.9
	FQN [14]	31.3	50.4	33.3	16.1	34.4	41.6
	Auxi [32]	34.7	53.7	36.9	19.3	38.0	45.9
	LCQ (Ours)	**36.4**	**55.9**	**38.7**	**21.2**	**40.0**	**46.6**
ResNet-50	FP	38.3	58.3	40.9	21.5	42.4	49.5
	FQN [14]	32.5	51.5	34.7	17.3	35.6	42.6
	Auxi [32]	36.1	55.8	38.9	**21.2**	39.9	46.3
	LCQ (Ours)	**37.1**	**57.0**	**39.6**	**21.2**	**40.8**	**47.1**

Table 4 compares the COCO average precision (AP) metrics for the 4-bit RetinaNet (W4/A4) with the different backbones. "FP" indicates the full precision case. LCQ showed more favorable results than did the conventional methods for all the 4-bit models, especially for ResNet-34, which differed by 1.7% points in AP over Auxi [32] with the uniform quantization.

Table 5: APs for the 3-bit RetinaNet on the COCO dataset.

Backbone	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	FP	33.2	52.3	34.8	18.7	35.6	43.7
	PACT [2]	25.3	41.8	26.0	13.0	26.8	34.6
	APoT [16]	31.2	50.1	32.8	**18.0**	33.5	**40.6**
	LCQ (Ours)	**31.3**	**50.2**	**33.1**	17.6	**33.8**	40.4
ResNet-34	FP	37.2	57.0	39.4	21.4	40.4	48.9
	APoT [16]	35.2	54.9	37.1	19.7	**39.1**	**45.3**
	LCQ (Ours)	**35.5**	**55.3**	**37.6**	**20.5**	39.0	45.0
ResNet-50	FP	38.3	58.3	40.9	21.5	42.4	49.5
	APoT [16]	**36.1**	56.0	**38.7**	21.2	**40.4**	44.9
	LCQ (Ours)	**36.1**	**56.2**	38.4	**21.7**	39.9	**46.1**

Table 5 shows the results of the 3-bit RetinaNet (W3/A3), where we observed that LCQ achieved comparable APs to those of the non-uniform quantization method,

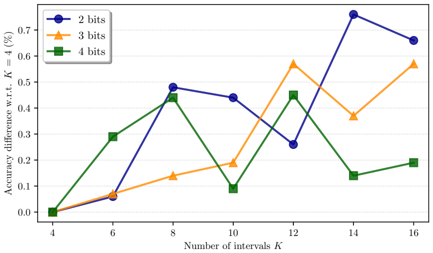


Figure 4: Relation between accuracy and number of intervals K in LCQ for the ResNet-20 model on CIFAR-100.

APoT [16] (our implementation).

These results show that LCQ can quantize the convolutional layers with less performance degradation even for the FPN architecture and heads connected to the regression and classification layers.

4.5. Ablation Studies

Number of intervals. The piecewise linear function in the companding function requires the number of intervals K as a predefined hyperparameter. We tested the relation between number of intervals and prediction accuracy in the ResNet-20 model for the CIFAR-100 dataset under the same experimental conditions as in Sec. 4.2. Figure 4 shows the relative accuracy difference with respect to $K = 4$ for each different number of bits when the number of intervals is increased from 4 to 16. The accuracy with 2, 3, and 4 bits for $K = 4$ was 65.2, 67.4, and 67.6, respectively. For numbers of bits, accuracy tends to improve as the number of intervals increases, and we found that accuracy tends to significantly improve with fewer bits. Since the companding function provides more flexibility in controlling quantization levels as the number of intervals increases, we infer that this flexibility is related to accuracy.

Table 6: Comparison of Top-1 accuracy (%) w.r.t. LWN.

Model & Data	Method	W2/A2	W3/A3	W4/A4
ResNet-20 on CIFAR-10	LCQ w/o LWN	91.4	92.6	93.1
	LCQ w/ LWN	**91.8**	**92.7**	**93.2**
ResNet-18 on ImageNet	LCQ w/o LWN	68.6	70.4	**71.5**
	LCQ w/ LWN	**68.9**	**70.5**	**71.5**

Effect of LWN. We used ResNet-18/20 and the CIFAR-10 and ImageNet datasets to investigate the effectiveness of LWN on accuracy. Table 6 shows the results for different bit-widths. Note that when LWN was not used, we instead applied the conventional method [16], which only applies standardization to pre-quantized weights. We observed that the results show a relatively large improvement at 2 bits and a minor improvement at 3- and 4-bits. Thus, LWN is a simple yet reliable method that can contribute to accuracy.

Table 7: Comparison of the different outer bit-widths.

Model & Data	b'_w/b'_a	Acc. (%)	LUT size (bytes)
ResNet-20 on CIFAR-10 ($b_w = b_a = 3$)	8 / 8	92.8	42.0
	6 / 6	92.8	31.5
	4 / 4	92.6	21.0
ResNet-18 on ImageNet ($b_w = b_a = 3$)	8 / 8	70.6	42.0
	6 / 6	70.5	31.5
	4 / 4	70.4	21.0

Outer bit-widths. Table 7 shows the top-1 accuracy and the memory cost of LUTs per layer for the 3-bit ResNet-18/20 model and the CIFAR-10 and ImageNet datasets for the outer bit-widths (b'_w for weights and b'_a for activations) described in Sec. 3.4. We observed that accuracy for CIFAR-10 remained the same from 8 to 6 bits, but degraded at 4 bits. In contrast, accuracy for ImageNet showed a clearer decrease compared with that for CIFAR-10 and tended to decrease linearly as the number of outer bit-widths decreased. However, there is still an advantage for both cases, especially at 4 bits, as accuracy degradation is as small as 0.2% points and the memory cost of a LUT is half that of the 8-bit case.

5. Conclusion

We proposed LCQ as a non-uniform quantization method that can optimize quantization levels via a learnable companding function. We formulated the companding function so that quantization levels can be flexibly and non-uniformly controlled by training. We also found that we can stabilize quantization training by limiting the effective scope of normalization to only the weight quantizer (LWN). In addition, we reduced the memory cost of the LUTs required for the efficient inference by applying the requantization technique. Various experiments involving image classification and object detection tasks for extremely low-bit models showed that LCQ achieved performance better than or comparable to conventional uniform and non-uniform quantization methods. We also conducted three ablation studies. The results showed that there is a likely proportional relationship between the number of intervals in the companding function and its accuracy, that LWN contributes to accuracy, and that accuracy can be maintained to some extent by reducing the outer bit-widths related to the LUT size. While we showed that non-uniform quantization has strong potential, fast inference on resource-constrained devices requires an efficient hardware accelerator in practice, so we plan to tackle this problem in future works.

6. Acknowledgement

This paper is partly based on results obtained from a project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] Yoshua Bengio, N. Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv*, abs/1308.3432, 2013. 1, 2, 3, 4, 5, 6

[2] Jungwook Choi, Z. Wang, Swagath Venkataramani, P. Chuang, V. Srinivasan, and K. Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *ArXiv*, abs/1805.06085, 2018. 2, 6, 7

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 6

[4] Steven K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. Modha. Learned step size quantization. *International Conference on Learning Representations (ICLR)*, 2020. 2, 4, 6, 7

[5] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, P. Hu, J. Lin, Fengwei Yu, and J. Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. *International Conference on Computer Vision (ICCV)*, 2019. 2, 7

[6] Priya Goyal, P. Dollár, Ross B. Girshick, P. Noordhuis, L. Wesolowski, Aapo Kyrola, Andrew Tulloch, Y. Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, abs/1706.02677, 2017. 6, 7

[7] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5, 6

[9] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[10] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán. Low-rank compression of neural nets: Learning the rank of each layer. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[11] B. Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 3, 5

[12] Sangil Jung, Changyong Son, Seohyung Lee, JinWoo Son, J. Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 4, 6, 7

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical report*, 2009. 6

[14] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 7

[15] T. Li, J. Li, Z. Liu, and C. Zhang. Few sample knowledge distillation for efficient network compression. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[16] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 4, 5, 6, 7, 8

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *International Conference on Computer Vision (ICCV)*, 2017. 6, 7

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. *European Conference on Computer Vision (ECCV)*, 2014. 6

[19] Zhi Gang Liu and Matthew Mattina. Learning low-precision neural networks without straight-through estimator(ste). *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2019. 2

[20] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations (ICLR)*, 2017. 6

[21] R. C. Mathes and S. B. Wright. The compandor an aid against static in radio telephony. *The Bell System Technical Journal*, 1934. 2

[22] D. Miyashita, E. G. Lee, and B. Murmann. Convolutional neural networks using logarithmic data representation. *ArXiv*, abs/1603.01025, 2016. 1, 2

[23] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman Loomis. Cupy: A numpy-compatible library for nvidia gpu calculations. *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017. 6

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, 2019. 6

[25] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *International Conference on Learning Representations (ICLR)*, 2018. 2

[26] Mark Sandler, A. Howard, Menglong Zhu, A. Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5, 6

[27] S. Uhlich, L. Mauch, F. Cardinaux, K. Yoshiyama, J. A. Garcia, S. Tiedemann, T. Kemp, and A. Nakamura. Mixed pre-

cision dnns: All you need is a good parametrization. *International Conference on Learning Representations (ICLR)*, 2020. 2, 4

[28] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. 2019. 6

[29] Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. Deep neural network compression with single and multiple level quantization. *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 2

[30] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 4, 6, 7

[31] Xiandong Zhao, Ying Wang, Xuyi Cai, Cheng Liu, and Lei Zhang. Linear symmetric quantization of neural networks for low-precision integer hardware. *International Conference on Learning Representations (ICLR)*, 2020. 2, 7

[32] Bohan Zhuang, Lingqiao Liu, Mingkui Tan, Chunhua Shen, and Ian Reid. Training quantized neural networks with a full-precision auxiliary module. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 6, 7

[33] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6, 7