Improving Low-Precision Network Quantization via Bin Regularization

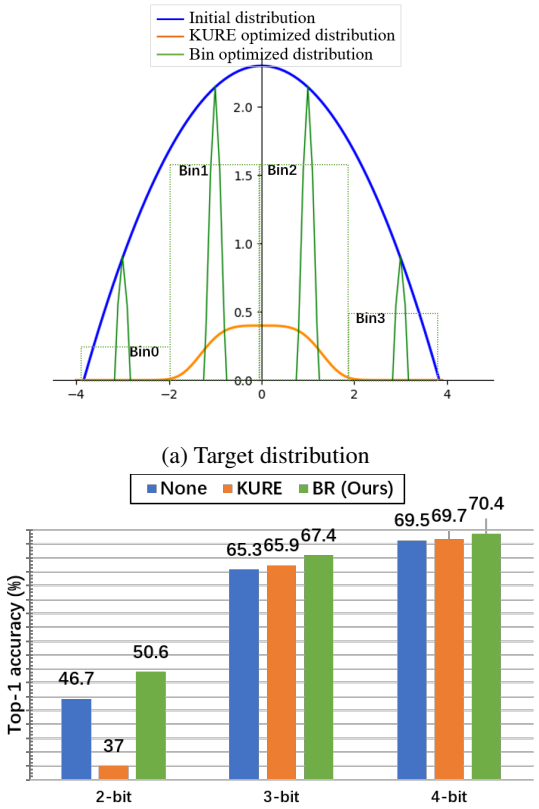
Tiantian Han Dong Li Ji Liu Lu Tian Yi Shan

Xilinx Inc., Beijing, China

{hantian, dongl, jiliu1, lutian, yishan}@xilinx.com

Abstract

Model quantization is an important mechanism for energy-efficient deployment of deep neural networks on resource-constrained devices by reducing the bit precision of weights and activations. However, it remains challenging to maintain high accuracy as bit precision decreases, especially for low-precision networks (e.g., 2-bit MobileNetV2). Existing methods have been explored to address this problem by minimizing the quantization error or mimicking the data distribution of full-precision networks. In this work, we propose a novel weight regularization algorithm for improving low-precision network quantization. Instead of constraining the overall data distribution, we separably optimize all elements in each quantization bin to be as close to the target quantized value as possible. Such bin regularization (BR) mechanism encourages the weight distribution of each quantization bin to be sharp and approximate to a Dirac delta distribution ideally. Experiments demonstrate that our method achieves consistent improvements over the state-of-the-art quantization-aware training methods for different low-precision networks. Particularly, our bin regularization improves LSQ for 2-bit MobileNetV2 and MobileNetV3-Small by 3.9% and 4.9% top-1 accuracy on ImageNet, respectively.



(a) Target distribution

(b) Top-1 accuracy of low-bit MobileNetV2

Figure 1: (a) Illustration of target distributions by different regularization methods. KURE [31] encourages the overall distribution to be uniform, while our bin regularization (BR) encourages each quantization bin distribution to be sharp. (b) Top-1 accuracy of low-bit MobileNetV2 with different regularization methods.

1. Introduction

Deep Convolutional Neural Networks (CNNs) have achieved remarkable progress in a wide range of applications including computer vision, natural language processing, speech recognition, etc. With the popularity of CNNs, there is an increasing demand for techniques to run networks efficiently on resource-constrained devices (e.g., mobile phones or FPGA). These techniques include network quantization, pruning, manual design of efficient architecture, or neural architecture search (NAS). We focus on network quantization by quantizing the networks to low bit precision in this work. It allows reduced memory footprint, faster inference time and lower power consumption when deploying CNN models on the edge devices.

Usually, the network accuracy decreases and the hardware performance increases as bit precision decreases. Recent network quantization methods have shown promising performance for 8-bit quantization by post-training quanti-

zation (PTQ). But PTQ methods tend to suffer from significant performance drop when applying for lower-bit quantization. Quantization-aware training (QAT) has become a common practice to learn a low-bit network to reduce the performance degradation from full-precision to quantized models. Prior methods attempt to minimize the quantization error or mimic the data distribution of full-precision networks. However, the final performance for the target task is still not satisfactory.

In this work, we propose a novel regularization algorithm for improving low-precision network quantization. When quantizing a full-precision network to be an n-bit network, all the floating elements of each convolution layer will be discretized into $m = 2^n$ specific values. In other words, these floating elements are grouped into m quantization bins and all the elements in each bin will be approximately represented by the same target value. We hypothesize that the quantization error will be approaching zero if all the floating elements in each bin are close enough to the target quantized value. Figure 1 (a) illustrates our idea of bin regularization. Compared to constraining the overall distribution of a certain layer to be uniform [31], our bin regularization can be viewed as a fine-grained constrain to encourage the bin distribution to be as sharp as possible. Compared to directly optimizing the overall quantization error (e.g., by L2 loss or KL loss) [8, 15, 22, 24, 35, 37], our bin regularization is expected to reduce the quantization error at the bin level. Experimental results validate the effectiveness of our method and show improved performance over the state-of-the-art methods for low-bit network quantization. Figure 1 (b) shows that our method can yield higher performance than the LSQ baseline and KURE [31] regularization method for quantized MobileNetV2 with different bit widths.

To summarize our contributions, we propose a novel bin regularization algorithm for low-precision network quantization in this work. Different from prior work on constraining the overall data distribution to be uniform, our bin regularization encourages sharp distribution for each quantization bin. The proposed algorithm is easy-to-implement and compatible with the common quantization-aware training paradigm. Experiments demonstrate that our bin regularization achieves consistent performance improvements over the state-of-the-art methods for low-precision network quantization, e.g., surpassing the LSQ baseline for 2-bit MobileNetV2 and MobileNetV3-Small by 3.9% and 4.9% top-1 accuracy on ImageNet, respectively.

2. Related Work

An overview of techniques for quantizing CNNs for efficient inference has been presented in [21]. [21] introduces different quantizer designs and generally classifies quantization methods into post-training quantization (PTQ) and quantization-aware training (QAT) methods.

Post-Training Quantization. PTQ methods usually quantize a network without full training and full data [2, 9, 13, 18, 27, 38]. Some methods require a small amount of training data for calibration or fine-tuning to optimize the network parameters [18]. Other methods explore data-free quantization without access to original training data by weight equalization [27] or generative schemes [6, 36]. Although these PTQ methods have shown impressive performance for typical 8-bit quantization, they are not capable of maintaining high accuracy for very low-precision (e.g., 2-bit) networks.

Quantization-Aware Training. QAT methods generally can provide higher accuracy than PTQ methods for low-precision networks with sufficient training on original data. The basic principle is inserting quantization operations into the neural network computational graph and calculating the forward and backward propagation with simulated weights and activations. Prior methods have been explored to optimize the quantization parameters (e.g., clipping value, quantization step size) and approximate the gradients during QAT [5, 7, 12, 14, 19, 20, 41]. Another line of related work has attempted to quantize the networks for improved performance by knowledge distillation [12, 26, 29], progressive quantization [3, 20, 40, 43], learning optimal bit-widths for each layer [11, 15, 32, 34, 42] and adjusting the network architecture to adapt the quantization [24, 25]. Our method is built upon the uniform symmetric quantization framework [21] and orthogonal to those additional quantization schemes.

Regularization for Quantization. Regularization is one of the important optimization techniques to reduce the generalization error in neural network training. Recent work attempts to employ weight regularization for improving network quantization performance [1] or robustness [31]. [8] proposes mean squared quantization error (MSQE) regularization to alleviate the mismatch between high-precision backward and low-precision forward passes. [1] improves the robustness to quantization by penalizing the L1 norm of gradients. [31] proposes Kurtosis regularization (KURE) to learn uniformly distributed weight tensors for improving the robustness to quantization across different bit-widths and parameters of step size. The goal of our work is to improve the final performance of low-precision networks through QAT. Instead of controlling the overall weight distribution to be uniform-like or other shapes, we optimize each quantization bin to be as sharp as possible and approximate to a Dirac delta distribution ideally. We find such a regularization scheme is crucial for maintaining the high accuracy of low-bit quantized networks.

3. Methodology

3.1. Quantization Baseline

We follow LSQ [12] as our baseline method for quantization-aware training. LSQ adopts a symmetric uniform quantization scheme with trainable scale parameters for both weights and activations. Given the data to quantize v, the quantization scheme can be defined as:

$$\tilde{v} = \lfloor clip(\frac{v}{s}; n, p) \rceil \tag{1}$$

and the quantized values can be computed as product of the integer-scaled $\tilde{v}$ and step size s:

$$\hat{v} = \tilde{v} \times s \tag{2}$$

Here, $clip(x; n, p) = min(max(x, n), p)$ function clips all the values that exceed the representation range and $\lfloor \cdot \rceil$ indicates the round function. When quantizing data to b bits, the clipping bounds are set as $n = -2^{b-1}, p = 2^{b-1} - 1$ for signed data and $n = 0, p = 2^b - 1$ for unsigned data. The step size s is initialized to $\frac{2 \cdot \langle |v| \rangle}{\sqrt{p}}$ for training where $\langle v \rangle$ computes the average value of initial weights or activations.

In the backward pass, as $\lfloor \cdot \rceil$ is a non-differentiable function, we use straight through estimator [4] for approximation. Then the gradient $\frac{\partial \hat{v}}{\partial s}$ can be derived by the chain rule:

$$\frac{\partial \hat{v}}{\partial s} = \begin{cases} -\frac{v}{s} + \lfloor \frac{v}{s} \rceil & \text{if } n < \frac{v}{s} < p \\ n & \text{if } \frac{v}{s} \leq n \\ p & \text{if } \frac{v}{s} \geq p \end{cases} \tag{3}$$

and the gradient $\frac{\partial \hat{v}}{\partial v}$ can be computed as below:

$$\frac{\partial \hat{v}}{\partial v} = \begin{cases} 1 & \text{if } n < \frac{v}{s} < p \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

To achieve good convergence, the step size s is multiplied by a gradient scale $g = \frac{1}{\sqrt{N \cdot p}}$ to balance the gradient update magnitude and parameter magnitude of s, where N stands for the number of weights or features in a certain layer for weight step size or activation step size, respectively. Our low-precision networks are directly trained from a full-precision model to the precision of interest without time-consuming progressive training.

3.2. Bin Regularization

Naturally, when quantizing v to b bits, there exist 2^b distinct integers to encode the data based on Eq. 1. Accordingly, $\hat{v}$ also has at most 2^b quantized values by multiplying the step size s. In other words, the original data will be grouped into 2^b quantization bins and all the data in each bin will be quantized to an identical quantized value. From the perspective of quantization error, if full-precision values falling into a certain bin can just be quantized to the target quantized value, the quantization error for this bin will approach zero. From the perspective of data distribution, we hypothesize that a good quantizer may encourage a sharp Gaussian distribution with its mean approaching the target quantized value and variance approaching zero (i.e., a Dirac delta distribution ideally) for each quantization bin.

Motivated by this, we propose to regularize the weight distribution of each quantization bin to be as sharp as possible. To this end, we impose two constraints on the data statistics by (1) encouraging the mean of data falling into each bin to be close to the target quantized value, (2) encouraging the variance of data falling into each bin to be close to zero. Thus, our bin regularization can be formulated as:

$$\mathcal{L}_{BR} = \sum_{i=1}^{2^b} (\mathcal{L}_{mse}(\langle v_i \rangle, \hat{v_i}) + \mathcal{L}_{var}(v_i)) \tag{5}$$

where v_i denotes the data falling into the i-th bin. $\langle \cdot \rangle$ computes the average value and $\mathcal{L}_{mse}$ computes the mean squared error. $\mathcal{L}_{var}$ indicates the variance loss:

$$\mathcal{L}_{var}(v_i) = \begin{cases} 0 & \text{if } V_i \leq 1 \\ var(v_i) & \text{otherwise} \end{cases} \tag{6}$$

where V_i counts the number of elements falling into the i-th bin and $var(\cdot)$ calculates the variance.

3.3. Optimization Strategy

Following the common practice of optimizing the task loss during QAT, we adopt the cross entropy loss $\mathcal{L}_{CE}$ for ImageNet classification. Intuitively, the total quantization objective can be optimized by minimizing the linear combination of the cross entropy loss and our bin regularization loss:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \cdot \mathcal{L}_{BR} \tag{7}$$

where λ is a hyper-parameter to balance the two loss terms.

In Eq. 7, both the target quantized values $\hat{v}$ and step sizes s will be updated simultaneously during each iteration of stochastic gradient descent optimization. Since the target quantized value $\hat{v}$ in each bin is dependent on step size s (Eq. 2), regularizing the bin distribution when s is not stable is prone to incur wrong optimization directions and degraded performance. Instead of optimizing $\mathcal{L}_{CE}$ and $\mathcal{L}_{BR}$ together from the beginning of QAT, we adopt a simple two-state optimization strategy by first updating step size s only for a few epochs and then adding the bin regularization term for joint optimization. Empirically we observe that such a two-stage optimization strategy is beneficial for good convergence and performance of quantized networks. We also try a more complicated alternating training strategy between updating s and regularizing bins but find no gains for the final performance.

4. Experiments

In this section, we evaluate the proposed bin regularization method for network quantization on the large-scale ImageNet classification task. In view of the fact that high-precision models of most networks (e.g., 8-bit ResNet-50) can be quantized even without fine-tuning and obtain comparable performance to the full-precision models, we focus on the more challenging low-precision model quantization problem on multiple popular light-weight network architectures. We follow LSQ [12] as our QAT baseline method.

4.1. Experimental Setting

Dataset. The ImageNet classification dataset [10] contains 1.2M training images and 50,000 validation images of 1,000 categories. In our experiments, training images are resized to 256×256 and randomly cropped to 224×224, while validation images are center-cropped to 224×224. We use the standard top-1 and top-5 accuracies as evaluation metrics.

Networks. We evaluate our method on different mainstream light-weight networks, including ResNet18 [16], MobileNetV2 [30] and MobileNetV3-Small [17]. For ResNet18, we use the standard V1 achitecture [16]. To evaluate the performance on those efficient achitectures with depth-wise convolution, we use the popular MobileNetV2 and MobileNetV3-Small. MobileNetV2 is designed with human inductive bias and MobileNetV3 is automatically searched by NAS.

Implementation Details. For ResNet18 and MobileNetV2, we train their full-precision models from scratch with the initial learning rate of 0.1 and weight decay of 1e-4. For MobileNetV3, we use the pre-trained full-precision network [33] as initialization for the subsequent low-bit network training. For all the three types of networks, we train each low-precision model directly from the corresponding full-precision model without warmup. The initial learning rate is set to 0.01 and decayed with a cosine policy. The parameters of weight decay are set to 1e-4, 5e-5, 2.5e-5 for 4-bit, 3-bit and 2-bit networks respectively. We train each low-precision network with the same epochs of 90 in total and the same momentum of 0.9. All of our experiments are conducted on PyTorch with one Nvidia V100 GPU. For training the low-precision MobileNetV2, LSQ baseline costs 90, 90, 90 hours and our method costs 97, 111, 135 hours for 2-bit, 3-bit, 4-bit quantization, respectively.

4.2. Comparisons to State-of-the-Art Quantization Methods

We compare the proposed approach with previous quantization methods in Table 1, 2, 3. For ResNet18, we note that prior PTQ and QAT methods have shown promising performance in Table 1. For example, the state-of-the-art PTQ method of AdaQuant achieves 67.5% top-1 accuracy for 4-bit quantization, which largely reduces the accuracy gap between low-bit and full-precision networks. The state-of-the-art QAT method of APoT even achieves a slightly higher top-1 accuracy of 70.7% for 4-bit quantization than the full-precision model. Despite the limited room for a performance boost, our method still can achieve comparable or slightly better performance than existing methods (e.g., +0.7% over LSQ for 2-bit quantization). For MobileNetV2 in Table 2, our method consistently outperforms the LSQ baseline by 3.9%, 2.1%, 0.9% for 2-bit, 3-bit, 4-bit quantization, respectively. For this challenging network architecture, PTQ methods [18, 28] without fine-tuning usually have a significant accuracy drop compared to the full-precision model, especially for the very low bit-widths. For example, AdaQuant [18] only achieves 34.9% top-1 accu-

Method	W/A	Acc@1 (%)	Acc@5 (%)
Full-precision	32/32	70.5	89.6
PACT [7]	4/4	69.2	88.8
LQ-Nets [37]	4/4	69.3	88.8
DSQ [14]	4/4	69.6	-
QIL [20]	4/4	70.1	-
LLSQ [39]	4/4	69.8	89.1
LAPQ [28]	4/4	60.3	-
APoT [22]	4/4	70.7	**89.6**
AdaQuant [18]	4/4	67.5	-
BRECQ [23]	4/4	69.6	-
LSQ* [12]	4/4	70.5	89.5
LSQ + BR (Ours)	4/4	**70.8**	**89.6**
PACT [7]	3/3	68.1	-
LQ-Nets [37]	3/3	68.2	87.9
DSQ [14]	3/3	68.7	-
QIL [20]	3/3	69.2	-
LLSQ [39]	3/3	68.1	88.2
APoT [22]	3/3	**69.9**	**89.2**
LSQ* [12]	3/3	69.4	88.9
LSQ + BR (Ours)	3/3	**69.9**	89.1
PACT [7]	2/2	64.4	-
LQ-Nets [37]	2/2	64.9	85.9
DSQ [14]	2/2	65.2	-
QIL [20]	2/2	65.7	-
APoT [22]	2/2	**67.3**	**87.5**
LSQ* [12]	2/2	66.5	87.0
LSQ + BR (Ours)	2/2	67.2	87.3

Table 1: Performance comparisons with different bit widths on ResNet18. "W/A" represents bit widths of weights and activation. * represents our re-implementation. The meanings in other tables are the same.

Method	W/A	Acc@1 (%)	Acc@5 (%)
Full-precision	32/32	71.8	90.2
PACT [7]	4/4	61.4	83.7
DSQ [14]	4/4	64.8	-
LAPQ [28]	4/4	49.7	60.3
LLSQ [39]	4/4	67.4	88.0
AdaQuant [18, 23]	4/4	34.9	-
BRECQ [23]	4/4	66.6	-
LSQ* [12]	4/4	69.5	89.2
LSQ + BR (Ours)	4/4	**70.4**	**89.4**
LSQ* [12]	3/3	65.3	86.3
LSQ + BR (Ours)	3/3	**67.4**	**87.4**
LSQ* [12]	2/2	46.7	71.4
LSQ + BR (Ours)	2/2	**50.6**	**74.6**

Table 2: Performance comparisons with different bit widths on MobileNetV2.

Method	W/A	Acc@1 (%)	Acc@5 (%)
Full-precision	32/32	65.1	85.4
LSQ* [12]	4/4	61.0	82.6
LSQ + BR (Ours)	4/4	**61.5**	**82.8**
LSQ* [12]	3/3	52.0	76.1
LSQ + BR (Ours)	3/3	**56.0**	**78.8**
LSQ* [12]	2/2	31.4	55.5
LSQ + BR (Ours)	2/2	**36.3**	**61.0**

Table 3: Performance comparisons with different bit widths on MobileNetV3-Small.

racy, which is much worse than the full-precision model. Our method also surpasses other QAT methods by a large margin (e.g., +3% over LLSQ) and reaches the state-of-the-art performance on ImageNet. For MobileNetV3-Small, similar conclusions can be drawn based on our experiments in Table 3. Our method also provides a consistent performance improvement over the LSQ baseline (e.g., +4.9% for 2-bit quantization).

4.3. Comparisons to Different Weight Regularization Methods

We compare the proposed bin regularization approach with other weight regularization methods for quantizing MobileNetV2, including L2, KL and KURE [31] regularization. L2 and KL regularization methods aim to reduce the overall quantization error by minimizing the L2 and KL distance between the low- and full-precision weights. KURE regularization aims to improve the robustness of quantization by encouraging the weight distribution to be uniform. Table 4 and Table 5 show that our BR method can achieve superior performance compared to L2, KL and KURE regularization methods. Instead of directly minimizing the overall quantization error by L2 or KL regularization, our bin regularization minimizes the quantization error at a fine-grained level (i.e., quantization bin). The results validate the superiority of our method with improved

Method	W/A	Acc@1 (%)	Acc@5 (%)
Full-precision	32/32	71.8	90.2
LSQ* [12]	4/4	69.5	89.2
LSQ + L2	4/4	69.8	**89.4**
LSQ + KL	4/4	69.6	89.2
LSQ + KURE	4/4	69.7	89.2
LSQ + BR (Ours)	4/4	**70.4**	**89.4**
LSQ* [12]	3/3	65.3	86.3
LSQ + L2	3/3	66.5	87.1
LSQ + KL	3/3	65.6	86.5
LSQ + KURE	3/3	65.9	86.8
LSQ + BR (Ours)	3/3	**67.4**	**87.4**
LSQ* [12]	2/2	46.7	71.4
LSQ + L2	2/2	42.1	67.0
LSQ + KL	2/2	39.8	64.8
LSQ + KURE	2/2	37.0	62.0
LSQ + BR (Ours)	2/2	**50.6**	**74.6**

Table 4: Performance comparisons with other regularization methods on MobileNetV2.

Method	W/A	Acc@1 (%)	Acc@5 (%)
Full-precision	32/32	70.5	89.6
LSQ* [12]	4/4	70.5	89.5
LSQ + L2	4/4	70.3	89.5
LSQ + KL	4/4	70.2	89.4
LSQ + KURE	4/4	70.0	89.3
LSQ + BR (Ours)	4/4	**70.8**	**89.6**
LSQ* [12]	3/3	69.4	88.9
LSQ + L2	3/3	69.3	88.8
LSQ + KL	3/3	69.0	88.5
LSQ + KURE	3/3	69.1	88.6
LSQ + BR (Ours)	3/3	**69.9**	**89.1**
LSQ* [12]	2/2	66.5	87.0
LSQ + L2	2/2	66.2	86.6
LSQ + KL	2/2	65.3	86.2
LSQ + KURE	2/2	66.2	86.7
LSQ + BR (Ours)	2/2	**67.2**	**87.3**

Table 5: Comparison with other regularization methods on ResNet18.

Different Training Strategies	W/A	Acc@1 (%)	Acc@5 (%)
Full-precision	32/32	71.8	90.2
LSQ* [12]	4/4	69.5	89.2
S1	4/4	69.5	89.0
S2	4/4	**70.4**	**89.4**
S3	4/4	69.4	89.0
LSQ* [12]	3/3	65.3	86.3
S1	3/3	66.0	86.7
S2	3/3	**67.4**	**87.4**
S3	3/3	65.2	86.3
LSQ* [12]	2/2	46.7	71.4
S1	2/2	44.3	69.1
S2	2/2	**50.6**	**74.6**
S3	2/2	43.6	68.4

Table 6: Impact of different training strategies for MobileNetV2.

performance. Compared to KURE regularization, we also obtain higher performance for all the low-precision MobileNetV2 and ResNet18 models. We observe that adding KURE regularization sometimes causes inferior accuracy than the LSQ baseline (e.g., for 2-bit MobileNetV2). We analyze that KURE regularization is more suitable for the scenario where a network trained with KURE regularization can be well-quantized to lower-bit networks on the fly. That is, the goal of KURE regularization is improving the quantization robustness across different bit widths, not improving the final performance of a target low-bit network.

4.4. Ablation Study

Optimization Strategy. We test different optimization strategies for network quantization including:

- S1: Joint updating step size and regularizing quantization bins from the beginning of training.
- S2: Updating step size for a few epochs (30 in our experiments) and then adding bin regularization.
- S3: Alternating between updating step size without regularizing bins and regularizing bins without updating step size.

Table 6 shows that the S2 strategy works best for all the bit widths. As the target quantized value in each bin is dependent on step size, it is better to add this regularization term when the step size becomes stable. Alternating training strategy (S3) did not bring performance gains, which indicates that the step size needs to be optimized throughout the QAT process.

Parameter Analysis. Table 7 shows the results with different loss weight λ for 3-bit MobileNetV3-Small. We find $\lambda = 0.5$ works best and choose this value for other low-bit networks. More ablation experiments are included in the supplementary material.

λ	W/A	Acc@1 (%)	Acc@5 (%)
Full-precision	32/32	65.1	85.4
4	3/3	55.2	78.1
2	3/3	55.5	78.4
1	3/3	55.8	78.7
0.5	3/3	**56.0**	**78.8**
0.25	3/3	55.8	**78.8**
0.05	3/3	54.7	77.8

Table 7: Impact of different loss weights of λ on MobileNetV3-Small.

Regularization	W/A	MSE-QE	Mean Bin Loss	Acc@1 (%)
None	4/4	9.3e-05	1.6e-03	61.0
BR (Ours)	4/4	**2.0e-05**	**2.0e-04**	**61.5**
None	3/3	1.1e-03	7.3e-03	52.0
BR (Ours)	3/3	**7.4e-05**	**5.0e-04**	**56.0**
None	2/2	7.4e-03	2.9e-03	31.4
BR (Ours)	2/2	**5.0e-04**	**1.7e-03**	**36.3**

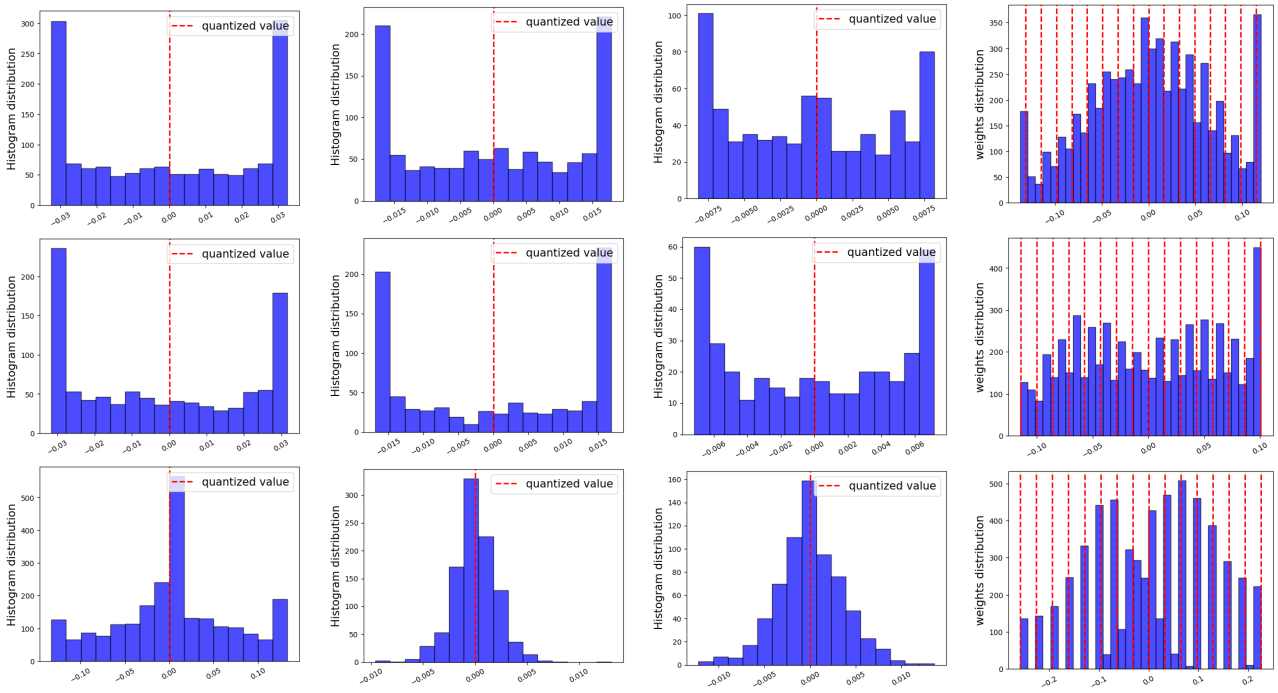
Table 8: Analysis of quantization error on MobileNetV3-Small. "None" means the LSQ baseline.

4.5. Analysis of Quantization Error

We seek to understand the relationship between our method and MSE quantization error (MSE-QE). Table 8 compares the LSQ baseline and our method in terms of quantization error, our bin loss and final performance using MobileNetV3. Although our method does not explicitly optimize the overall quantization error, we find that the error still drops after regularizing each bin.

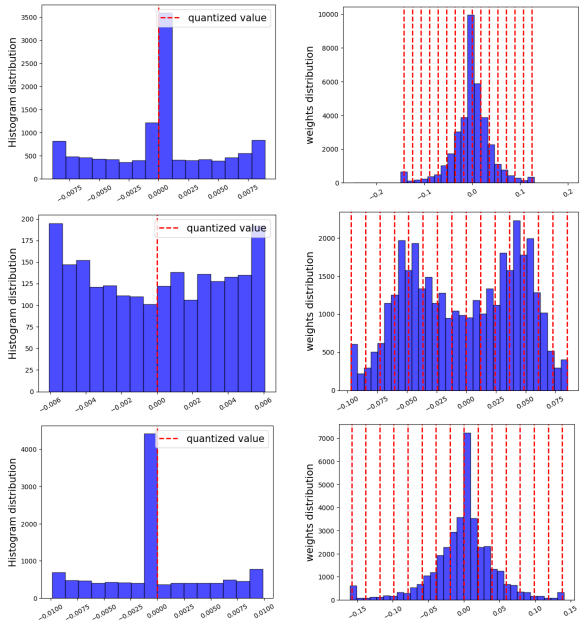
4.6. Visualization of Bin Distribution

Figures 2 and 3 show samples of bin distribution and global distribution derived by the LSQ baseline, KURE regularization and our bin regularization methods on MobileNetV2 and ResNet18, respectively. The LSQ baseline neither explicitly constrain the overall nor the bin distributions. KURE regularization encourages the overall distribution to be uniform, while our method encourages the bin distribution to be sharp. The weight values falling into a certain bin are expected to concentrate around the target quantized value (red dash line in the figures). Similar conclusions can be drawn for different bit widths.



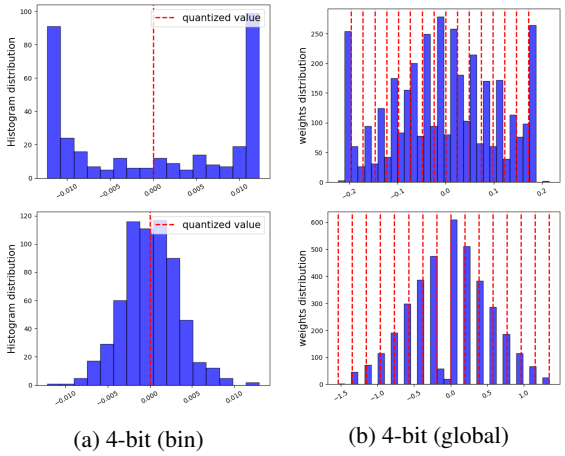
(a) 2-bit (bin) (b) 3-bit (bin) (c) 4-bit (bin) (d) 4-bit (global)

Figure 2: Bin distribution with different bit widths (a, b, c) and 4-bit global distribution (d) on `layer14` of MobileNetV2. Row 1∼3 represent the original LSQ, LSQ+KURE and LSQ+BR (Ours), respectively.



(a) 4-bit (bin) (b) 4-bit (global)

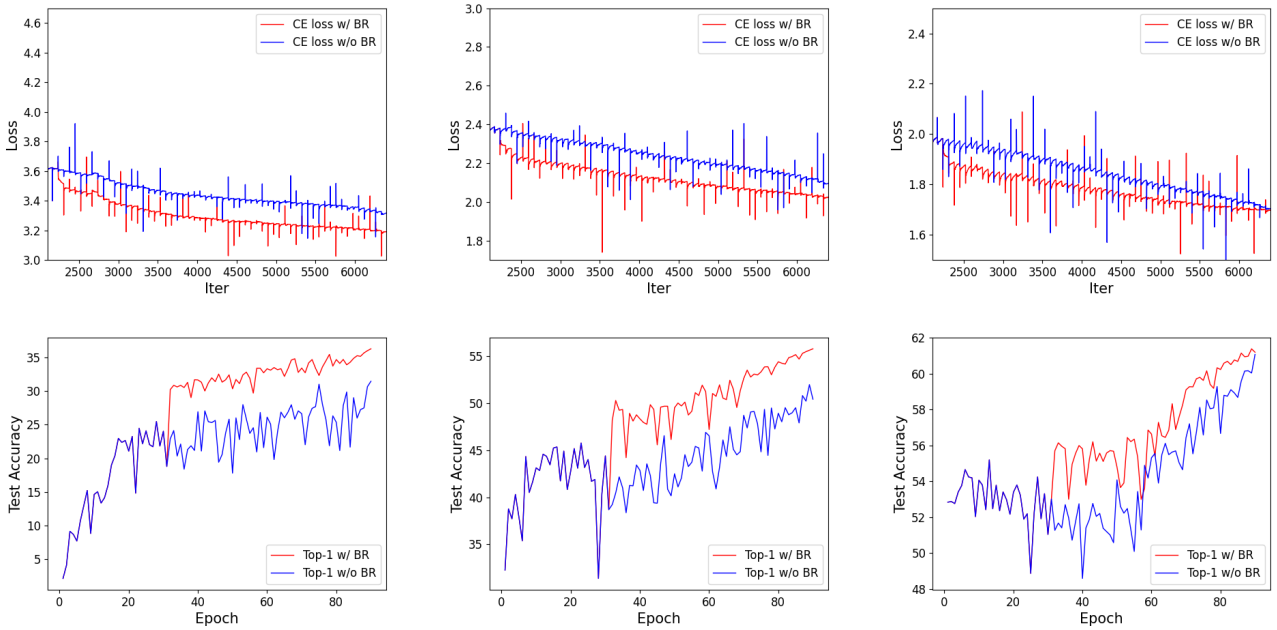
Figure 3: Bin distribution and global distribution with 4 bits on `layer2` of ResNet18. Row 1∼3 represent the original LSQ, LSQ+KURE and LSQ+BR (Ours), respectively.



(a) 4-bit (bin) (b) 4-bit (global)

Figure 4: Bin distribution and global distribution with 4 bits on `layer14` of MobileNetV3-Small. The first row represents the original LSQ and second row represents LSQ+BR (Ours).

Figure 4 shows another set of examples on 4-bit MobileNetV3-Small. We observe similar weight distribution with other types of networks after applying our regularization method, which further reinforces the intuition of our idea.



(a) 2-bit (b) 3-bit (c) 4-bit

Figure 5: Loss curves and validation accuracies with QAT for different bit widths on MobileNetV3-Small.

Method	W/A	Acc@1 (%)	Acc@5 (%)
Full-precision	32/32	71.8	90.2
LSQ [12]	4/4	69.5	89.2
LSQ [12] + BR (Ours)	4/4	**70.4**	**89.4**
TQT [19]	4/4	63.8	85.2
TQT [19]+L2	4/4	65.8	86.7
TQT [19]+BR(Ours)	4/4	**67.5**	**87.7**

Table 9: Performance comparisons with other QAT methods on MobileNetV2.

4.7. Training Convergence

Figure 5 presents the loss and accuracy curves during training the low-precision MobileNetV2. With our bin regularization method, the task loss (i.e., CE loss) can be further decreased and the final accuracy can be improved.

4.8. Hardware-Friendly Implementation

Most of the existing quantization methods adopt some relaxations in order to maintain the high accuracy of quantized networks (e.g., the first and last layers are not quantized or quantized to 8 bits). We follow [19, 21] to re-implement a hardware-friendly quantization baseline by power-of-2 step size, quantizing the first and last layer with the same low bit-widths, quantizing both inputs and outputs of element-wise addition modules, quantizing bias to 8 bits, merging BN to the previous linear layer during training. Table 9 shows that our bin regularization still can improve significantly over the hardware-friendly quantization baseline (e.g., +3.7% over TQT for 4-bit MobileNetV2). We believe that our work can promote the applications of network quantization for different hardware platforms and accelerators in practical industrial scenarios.

5. Conclusion

In this work, we address the neural network quantization problem, especially for low-precision networks. Instead of constraining the overall data distribution, we propose to encourage each quantization bin to be as sharp as possible. By constraining the data falling into each bin to be close to the target quantized value, we can reduce the quantization error at a fine-grained level. We evaluate our method on multiple popular network architectures and achieve the state-of-the-art accuracy on ImageNet classification for 4-bit, 3-bit, 2-bit quantization. We believe that the proposed idea can inspire new insights for the network quantization problem and promote efficient model deployment on resource-limited devices in practical applications.

References

[1] Milad Alizadeh, Arash Behboodi, Mart van Baalen, Christos Louizos, Tijmen Blankevoort, and Max Welling. Gradient _1

regularization for quantization robustness. In *ICLR*, 2020. 2

[2] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment. In *NeurIPS*, 2019. 2

[3] Chaim Baskin, Natan Liss, Yoav Chai, Evgenii Zheltonozhskii, Eli Schwartz, Raja Giryes, Avi Mendelson, and Alexander M Bronstein. Nice: Noise injection and clamping estimation for neural network quantization. In *ICLR*, 2019. 2

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

[5] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *CVPR Workshops*, 2020. 2

[6] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *CVPR*, 2020. 2

[7] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 2, 4, 5

[8] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Learning sparse low-precision neural networks with learnable regularization. *IEEE Access*, 8:96963–96974, 2020. 2

[9] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, 2019. 2

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 4

[11] Ahmed Elthakeb, Prannoy Pilligundla, FatemehSadat Mireshghallah, Amir Yazdanbakhsh, Sicuan Gao, and Hadi Esmaeilzadeh. Releq: An automatic reinforcement learning approach for deep quantization of neural networks. In *NeurIPS Workshop*, 2019. 2

[12] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *ICLR*, 2020. 2, 3, 4, 5, 6, 8

[13] Alexander Finkelstein, Uri Almog, and Mark Grobman. Fighting quantization bias with bias. In *CVPR Workshop*, 2019. 2

[14] 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*, 2019. 2, 4, 5

[15] Hai Victor Habi, Roy H Jennings, and Arnon Netzer. Hmq: Hardware friendly mixed precision quantization block for cnns. In *ECCV*, 2020. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[17] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019. 4

[18] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*, 2020. 2, 4, 5

[19] Sambhav R Jain, Albert Gural, Michael Wu, and Chris H Dick. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. In *MLSys*, 2020. 2, 8

[20] 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*, 2019. 2, 4

[21] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 2, 8

[22] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *ICLR*, 2020. 2, 4

[23] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *ICLR*, 2021. 4, 5

[24] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *ECCV*, 2020. 2

[25] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*, 2018. 2

[26] Asit Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *ICLR*, 2018. 2

[27] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *ICCV*, 2019. 2

[28] Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson. Loss aware post-training quantization. *arXiv preprint arXiv:1911.07190*, 2019. 4, 5

[29] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *ICLR*, 2018. 2

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 4

[31] Moran Shkolnik, Brian Chmiel, Ron Banner, Gil Shomron, Yuri Nahshan, Alex Bronstein, and Uri Weiser. Robust quantization: One model to rule them all. In *NeurIPS*, 2020. 1, 2, 5

[32] Mart van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. Bayesian bits: Unifying quantization and pruning. In *NeurIPS*, 2020. 2

[33] Kuan Wang. pytorch-mobilenet-v3. https://github.com/kuan-wang/pytorch-mobilenet-v3, 2019. 4

[34] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *CVPR*, 2019. 2

[35] Ying Wang, Yadong Lu, and Tijmen Blankevoort. Differentiable joint pruning and quantization for hardware efficiency. In *ECCV*, 2020. 2

[36] Shoukai Xu, Haokun Li, Bohan Zhuang, Jing Liu, Jiezhang Cao, Chuangrun Liang, and Mingkui Tan. Generative low-bitwidth data free quantization. In *ECCV*, 2020. 2

[37] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *ECCV*, 2018. 2, 4

[38] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *ICML*, 2019. 2

[39] Xiandong Zhao, Ying Wang, Xuyi Cai, Cheng Liu, and Lei Zhang. Linear symmetric quantization of neural networks for low-precision integer hardware. In *ICLR*, 2019. 4, 5

[40] 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

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

[42] Xiaotian Zhu, Wengang Zhou, and Houqiang Li. Adaptive layerwise quantization for deep neural network compression. In *ICME*, 2018. 2

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