

Precision Neural Network Quantization via Learnable Adaptive Modules

Wenqiang Zhou^{a,b}, Zhendong Yu^{a,b}, Xinyu Liu^{a,b}, Jiaming Yang^{a,b}, Rong Xiao^{a,b}, Tao Wang^{a,b}, Chenwei Tang^{a,b,*}, Jiancheng Lv^{a,b}

^a*College of Computer Science, Sichuan University, Chengdu, 610065, China*

^b*Engineering Research Center of Machine Learning and Industry Intelligence, Ministry of Education, Chengdu, 610065, China*

Abstract

Quantization Aware Training (QAT) is a neural network quantization technique that compresses model size and improves operational efficiency while effectively maintaining model performance. The paradigm of QAT is to introduce fake quantization operators during the training process, allowing the model to autonomously compensate for information loss caused by quantization. Making quantization parameters trainable can significantly improve the performance of QAT, but at the cost of compromising the flexibility during inference, especially when dealing with activation values with substantially different distributions. In this paper, we propose an effective learnable adaptive neural network quantization method, called Adaptive Step Size Quantization (ASQ), to resolve this conflict. Specifically, the proposed ASQ method first dynamically adjusts quantization scaling factors through a trained module capable of accommodating different activations. Then, to address the rigid resolution issue inherent in Power of Two (POT) quantization, we propose an efficient non-uniform quantization scheme. We utilize the Power Of Square root of Two (POST) as the basis for exponential quantization, effectively handling the bell-shaped distribution of neural network weights across various bit-widths while maintaining computational efficiency through a Look-Up Table method (LUT). Extensive experimental results demonstrate that the proposed ASQ method is superior to the state-of-the-art QAT approaches. Notably that the ASQ is even competitive compared to full precision baselines, with its 4-bit quantized ResNet34 model improving accuracy by 1.2%

*Corresponding author: tangchenwei@scu.edu.cn

on ImageNet.

Keywords: Model compression, Quantization aware training, Image classification

1. Introduction

In recent years, deep neural networks have achieved significant advancements, excelling in domains like computer vision [1, 2], natural language processing [3], and speech recognition [4], often surpassing human-level performance. However, as models grow in size and complexity, they pose challenges for deployment on resource-limited edge devices. To address this, researchers have developed model compression techniques that reduce model size and computational demands while preserving performance. Key compression techniques include quantization [5, 6], pruning [7, 8], knowledge distillation [9, 10], low-rank factorization [11], and the development of compact architectures [12, 13] specifically designed for efficiency.

Quantization stands out as a promising approach to model compression, offering significant advantages in reducing model size and accelerating inference through efficient integer computations [14]. These benefits make quantization particularly attractive for deployment on resource-constrained devices. However, as the bit-width used to represent model weights and activations decreases, performance degradation becomes increasingly apparent, especially in extremely low-bit scenarios, e.g., 2-bit or binary (1-bit) representations, where the representational capacity of network is severely limited [15]. Quantization-aware training (QAT) technologies have emerged as an effective quantization method to mitigate performance loss [6, 16]. The core principle of QAT involves training low-bit-width networks using stochastic gradient descent by updating full-precision weights, which are subsequently quantized to lower bit-widths. Therefore, the QAT technologies allow the network to learn to compensate for quantization errors during training, resulting in quantized models that often achieve performance much closer to their full-precision counterparts.

The researches on QAT can be roughly divided into two categories, i.e., gradient estimation and quantization parameter optimization. Among them, the straight-through estimator (STE) is the most mainstream method in the gradient estimation category [17]. Due to the rounding operation, the quantizer itself is non-differentiable. STE essentially ignores the rounding

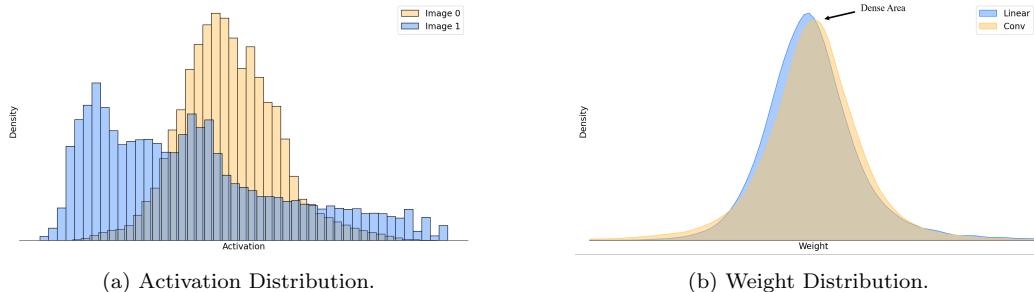


Figure 1: (a) The data distributions of different images (model inputs) exhibit significant variations. (b) The distribution of weights in a convolutional layer or linear layer of a deep neural network generally exhibits a bell-shaped curve centered around a mean close to zero.

operation and uses an identity function to approximate the gradient before and after quantization. Despite its simplicity, this approach is often highly effective. In the early stages, quantization parameters were typically determined manually [18] or calculated based on the statistical distribution of the data [19]. Although these methods significantly reduced quantization error, they were not optimal for the model’s task performance. Later, researchers proposed various methods using backpropagation with stochastic gradient descent to learn quantizer parameters that minimize task loss [5, 6, 16]. These methods make the optimization of quantizer parameters more direct. Notably, the learned step size quantization (LSQ) method [6] introduced a new gradient estimate to learn the scaling factors (also known as step size) for non-negative activations during QAT. This innovation enabled quantized models to achieve state-of-the-art performance across different bit-widths. However, the method of using trainable quantization parameters has a significant limitation: during the inference phase, each layer of the neural network can only use a fixed quantization step size. This limitation is particularly prominent when dealing with diverse activation value distributions, as illustrated in Fig.1(a).

Moreover, extensive researches indicate that the weights of deep neural networks typically follow a bell-shaped distribution, as shown in Fig.1(b), with a significant concentration of values around the mean. Applying uniform quantization to this distribution results in substantial information loss, especially for the critical values near the center. To address these challenges, researchers have proposed various non-uniform quantization methods [20, 21]

that employ different quantization levels to better capture critical features. However, these methods introduce non-linear operators, significantly reducing hardware inference efficiency. Power of two (POT) quantization [22] attempts to balance efficiency and non-uniform distribution by encoding using a logarithmic scale based on powers of two, but its rigid resolution [23] prevents it from fully leveraging the benefits of non-uniform distribution.

To better fit the potential distribution of activations and weights, as well as reduce the performance degradation caused by quantization errors, we propose an effective learnable adaptive QAT method, named Adaptive Step size Quantization (ASQ). Based on a dynamic weight technique [24], the proposed ASQ is able to adaptively adjust the model’s quantization step size parameters according to the input activation values through a simple neuron module containing one or two linear layers. This module introduces negligible computational and parameter overhead while significantly enhancing the performance of the quantizer. In addition to optimizing activation quantization, we have also improved the POT quantizer for weight quantization. The rigid resolution problem inherent to POT quantization is effectively addressed by using the Power Of Square root of Two (POST) as the exponential quantization basis. This adjustment ensures that even with larger bit widths, the quantization levels do not exhibit excessively high resolution near zero. During actual inference, POST achieves computational efficiency comparable to POT through the use of a look-up table (LUT) method.

Our main contributions are summarized as follows:

- We propose a novel uniform quantization approach for activations. The proposed ASQ method dynamically adjusts the step size of the activation quantizer through a trained adaptive module, with the aim of minimizing task-specific loss.
- For weight quantization, we improved the POT method with rigid resolution issue and proposed POST, a simple yet effective non-uniform quantization method, which achieves more universal and stable quantization performance.
- We conduct extensive experiments with various networks for image classification tasks on CIFAR-10 and ImageNet datasets. The results demonstrate that ASQ significantly outperforms the state-of-the-art QAT methods.

2. Related Works

Neural network quantization refers to the process of mapping the weights or activations of a deep neural network to a finite set of values. The quantization techniques can generally be classified into two categories, i.e., *uniform quantization* and *non-uniform quantization*.

2.1. Uniform Quantization.

Uniform quantization is a widely adopted quantization approach. A representative method in this domain is LSQ [6], which introduces a learnable step size for quantization by improving the configuration of the quantizer. LSQ+ [16] extends LSQ to address the quantization of negative activation values. DoReFa-Net [25] proposes training convolutional neural networks with low-bitwidth weights and activations using low-bitwidth parameter gradients. PACT [5] employs a novel activation quantization scheme by optimizing the clipping parameters of activations during training, achieving high accuracy even with ultra-low precision weights and activations. DSQ [26] introduces an innovative differentiable soft quantization approach that gradually evolves during training to approximate standard quantization, effectively addressing the discreteness issue in low-bit quantization.

2.2. Non-Uniform Quantization.

Non-uniform quantization can better capture the distribution characteristics, potentially leading to higher task performance. However, it often requires additional computational overhead, making it challenging to efficiently deploy non-uniform quantization schemes on general-purpose hardware. QIL [20] proposes a quantizer that learns the quantization intervals by directly minimizing the task loss of the network to find the optimal quantization intervals. POT [22, 27] utilizes a logarithmic representation based on powers of two to encode weights and activations. APOT [23] introduces an additive POT quantization scheme, which optimizes the overly dense quantization levels in POT by restricting all quantization levels to the sum of powers of two. LQ-Net [21] presents a method for jointly training the quantized network and the quantizer, rather than relying on fixed handcrafted quantization schemes.

3. Method

In this section, we first provide a brief overview of the background of neural network quantization. Subsequently, we discuss the deficiencies of

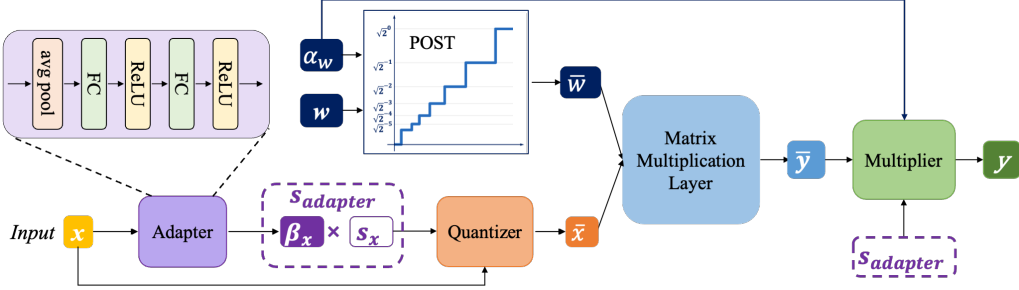


Figure 2: An overview of the proposed method. We optimized quantization for both activations and weights. For activations, a two-layer linear adapter produces an factor β , which multiplies a trainable parameter s to dynamically adjust the quantization step size. For weights, POST quantization addresses the rigid resolution issue and better fits the bell-shaped distribution.

current quantizers and detail our proposed methods. This includes a formal introduction to the uniform quantizer (ASQ) for activations, along with the formulation of related parameter gradients, and an in-depth analysis of the non-uniform quantizer (POST) for weights. Fig.2 is an overview of the proposed method.

3.1. Preliminaries

To elucidate the concepts, we will use symmetric quantization as our primary example. In uniform quantization, the processes of quantization *Quant* and dequantization *Dequant* can be succinctly expressed through the following pair of equations:

$$\text{Quant} : x_{int} = \text{clamp}(\lfloor \frac{x}{s} \rfloor, n, p), \quad (1)$$

$$\text{Dequant} : \hat{x} = x_{int} \times s, \quad (2)$$

where n and p refer to the lower and upper bounds of the quantization space, respectively. The operator $\lfloor \cdot \rfloor$ represents the rounding operation, while $\text{clamp}(\cdot)$ indicates the clipping of values to the range $[n, p]$. The variable x represents the weights or activations undergoing quantization, and s signifies the quantization step size used for value mapping. It's worth noting that the LSQ method poposed in [6] enhances model performance post-quantization by making the step size s a learnable parameter.

Non-uniform quantization can offer additional benefits, but often at the cost of increased computational complexity. POT quantization, however,

is an exception. It not only addresses the issue of non-uniform distribution of quantized parameters but also enhances the inference speed in hardware implementations. POT achieves this speed improvement by utilizing bit-shift operations instead of more computationally expensive integer arithmetic. The quantization levels Q in POT can be formally represented by the following equation:

$$Q(\alpha, b) = \alpha \times \{0, \pm 2^{-2^{b-1}+1}, \pm 2^{-2^{b-1}+2}, \dots, \pm 1\}, \quad (3)$$

where α represents the clipping threshold, and b denotes the bit-width of the fixed-point representation. For both weights and activations, values outside the range $[-\alpha, \alpha]$ are clipped, while those within this range are mapped to the quantization levels defined by $Q(\alpha, b)$. A key characteristic of POT quantization is that it provides higher resolution for values near zero. This property aligns well with the typical distribution of weights in deep neural networks, where smaller values are often more prevalent.

3.2. Adaptive Step Size Quantization for Activation

The LSQ method introduces learnable quantization step size parameters for both weights and activations, enabling the model to adaptively learn appropriate quantization mapping strategies through the stochastic gradient descent algorithm. However, since the activation is scene-sensitive, using a fixed step size to map activations with different distributions to integers may negatively impact the quality of the features, subsequently affecting the overall task performance.

As a solution to the issues discussed above, we propose a general adaptive activation quantization scheme. This approach incorporates a linear layer module within the quantizer to generate dynamic parameter β . During training, this mechanism not only learns the quantization step size parameter but also dynamically adjusts its magnitude based on varying activation distributions. The corresponding formula is as follows:

$$Adapt : \quad s_a = s \times \beta, \quad (4)$$

$$Quant : \quad x_{int} = clamp(\lfloor \frac{x}{s_a} \rfloor, n, p), \quad (5)$$

$$Dequant : \quad \hat{x} = x_{int} \times s, \quad (6)$$

where β is the computed adaptive parameter, while s_a denotes the dynamically adjusted quantization step size.

3.2.1. Gradient

Following the LSQ method, we compute the gradient of s_a as follows:

$$\frac{\partial \hat{x}}{\partial s_a} = \begin{cases} -x/s + \lfloor v/s \rfloor, & \text{if } n < x/s_a < p, \\ n, & \text{if } x/s_a < n, \\ p, & \text{if } x/s_a > p. \end{cases} \quad (7)$$

The gradient update of s and β is calculated using:

$$\frac{\partial \hat{x}}{\partial s} = \frac{\partial \hat{x}}{\partial s_a} \frac{\partial s_a}{\partial s} = \frac{\partial \hat{x}}{\partial s} \beta, \quad (8)$$

$$\frac{\partial \hat{x}}{\partial \beta} = \frac{\partial \hat{x}}{\partial s_a} \frac{\partial s_a}{\partial \beta} = \frac{\partial \hat{x}}{\partial s} s. \quad (9)$$

When calculating the partial derivatives of s_a as show in Eq.(7) , STE is utilized for approximating gradient calculations.

3.2.2. Additional model overhead

The proposed ASQ can effectively improve the performance of the model after quantization. However, an unavoidable drawback is that this module introduces additional parameters and computational overhead. To verify the practical feasibility of our approach, we calculated the proportion of additional overhead introduced by the ASQ operation at different bit widths.

For a convolutional layer, the number of parameters is determined by the total number of parameters in the convolutional kernels, while for a linear (fully connected) layer, the number of parameters is determined by the weights between the input and output nodes. The formulas for calculating the parameters of convolutional and linear layers are as follows:

$$Param_{conv} = C_{in} \times C_{out} \times K_w \times K_h, \quad (10)$$

$$Param_{linear} = N_{in} \times N_{out} + N_{out}, \quad (11)$$

where C_{in} and C_{out} represent the number of input and output channels of the convolutional layer, respectively. K_w and K_h denote the width and height of the convolutional kernel, while N_{in} and N_{out} are the number of input and output nodes of the linear layer, respectively.

The number of operations required for both convolutional and linear layers can be calculated by considering the multiplications performed. For a

convolutional layer, the operations involve the number of multiplications performed by each convolutional kernel across the input feature map. Similarly, for a linear layer, the operations involve the number of multiplications and additions between the input and output nodes. The total number of operations for convolutional and linear layers is given by:

$$OPS_{conv} = C_{in} \times C_{out} \times H_{out} \times W_{out} \times K_w \times K_h, \quad (12)$$

$$OPS_{linear} = N_{in} \times N_{out}, \quad (13)$$

where H_{out} and W_{out} are the output feature map dimensions. Based on the calculated results of the parameters and operations mentioned above, we can further compare modules after specific bit-width quantization. Specifically, during the model quantization process, aside from changes in computational complexity, the storage space for parameters is also significantly reduced. In an unquantized model, each weight is typically stored using 32-bit or 16-bit floating-point numbers, whereas in a quantized model, the storage bit-width for each parameter is greatly reduced. When a model is quantized to a lower bit-width, the bit-width required to store each parameter is reduced to B bits, so the storage space after quantization is:

$$S_{quantized} = Param \times B \text{ bits}, \quad (14)$$

where $S_{quantized}$ represents the storage space required for the quantized parameters, and $Param$ denotes the amount of parameters calculated earlier.

After obtaining the number of operations for the model modules through Eq.(12) and (13), we can further calculate the computational complexity of different quantized models. Typically, the lower the quantization bit-width, the lower the equivalent OPS, which means reduced computational complexity. Assuming the number of operations for the original model is OPS, the quantized model can be compared through the following example:

$$QOPS_{8-bits} = \frac{OPS}{4}, \quad (15)$$

$$QOPS_{4-bits} = \frac{OPS}{8}, \quad (16)$$

where $QOPS$ is defined as the reference value for the computational load after quantization, assume its full-precision model is 32 bits.

Based on the above calculations, we can easily determine the additional overhead introduced by our Adapter module. Table 1 presents the data on the proportion of additional parameter storage and computational load introduced by ASQ, using the ResNet18 model as an example, after quantization at different bit-widths.

Table 1: Comparison of the proportion of additional parameters and computational overhead introduced by ASQ at different bit widths. Here, Param(%) represents the percentage of additional parameters relative to the parameter space of the quantized model, and Compute(%) represents the percentage of additional computational load relative to the computational load of the quantized model.

Network	Precision	Param(%)	Compute(%)
ResNet18	8	2.67	0.04
	4	5.34	0.07
	3	7.11	0.10
	2	10.67	0.14

From these data, it is evident that while our method (ASQ) introduces some additional overhead in terms of parameter storage and computational load, this overhead remains relatively small, even at lower bit-width quantization levels. In practical applications, especially when dealing with activation distributions that are sensitive to specific scenarios, such trade-offs are sometimes necessary to achieve optimal performance. Therefore, the additional overhead can be considered acceptable in exchange for the performance benefits it provides, making it a reasonable choice in performance-sensitive applications.

3.2.3. Training algorithm for ASQ

When training quantized DNNs using ASQ, we independently apply the ASQ quantizer to the activations of convolutional or fully connected layers. Algorithm 1 summarizes the ASQ training process for the convolutional layer as an example. The complete code and relevant details will be released after the paper is accepted.

3.3. Non-Uniform Quantization for Weight: *POST*

For weight quantization, as mentioned in section 3.1, Power of Two (POT) quantization can simultaneously improve performance and efficiency. However, when the bit-width is relatively high, it suffers from a rigid resolution

Algorithm 1 Training a convolutional layer with ASQ.

Input: full precision weights of convolutional layer W_{weight} and full precision inputs/activations A .

Parameter: the quantization step size for weights α and activations s , the bit-widths (b_w, b_a) . the weight and bias of adapter $W_{adapter}$ and $B_{adapter}$.

Output: updated parameters W_{weight} , $W_{adapter}$, $B_{adapter}$, α and s .

- 1: Compute the quantized weights using Eq. (1) and Eq. (2): $\hat{W} \leftarrow Quantize(W, \alpha, b_w)$.
 - 2: Compute the adaptive factor β using Adapter: $\beta \leftarrow Adapter(A, W_{adapter}, B_{adapter})$.
 - 3: Compute the adaptive quantization step size for activations using Eq. (4): $s_a = s \times \beta$.
 - 4: Compute the quantized activations using Eq. (5) and Eq. (6): $\hat{A} \leftarrow Quantize(A, s_a, b_a)$.
 - 5: Compute the convolution output: $y \leftarrow \hat{W} \times \hat{A}$.
 - 6: Compute the loss \mathcal{L} and the gradients $\frac{\partial \mathcal{L}}{\partial y}$.
 - 7: Compute the gradients for the weights $\frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial W}$.
 - 8: Compute the gradients of quantization step size for weights and the adaptive quantization step size for activations: $\frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial \alpha}$ and $\frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial s_a}$ based on Eq. (7).
 - 9: Compute the gradients of quantization step size for activations and adaptive factor: $\frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial s_a} \beta$ and $\frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial s_a} s$ based on Eq. (8) and Eq. (9).
 - 10: Update W_{weight} , $W_{adapter}$, $B_{adapter}$, α and s with the corresponding gradients, respectively.
-

problem. This phenomenon manifests as an uneven distribution of quantization levels, where increasing the bit-width results in excessively fine-grained resolution near zero while leaving large gaps between larger values. Consequently, the model’s expressiveness fails to improve effectively, despite the increased bit-width. The method proposed by Additive Power Of Two [23] effectively addresses this issue. However, it increases the computational complexity compared to the POT method during the inference process. In response, we have optimized the POT approach by using the square root of 2 as the base for exponential quantization (Power Of Square root of Two).

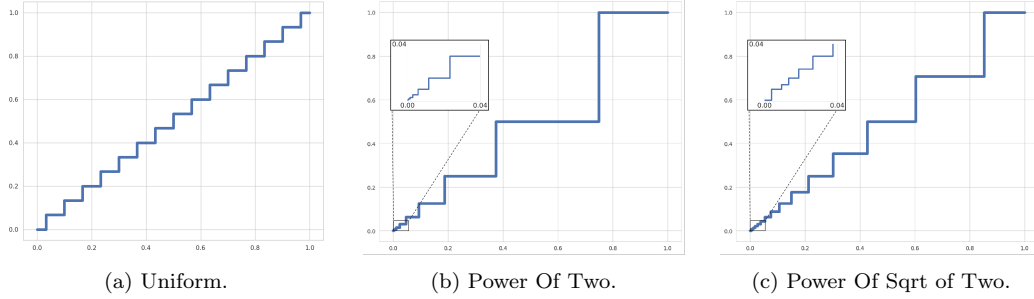


Figure 3: The Comparison of 4-bit quantization levels among Uniform, POT, and POST quantization.

The quantization levels for this approach is as follows:

$$Q(\alpha, b) = \alpha \times \{0, \pm\sqrt{2}^{-2^{b-1}+1}, \dots, \pm 1\}. \quad (17)$$

To better illustrate the effectiveness of POST, we visualized the quantization levels of uniform quantization, POT, and POST under a 4-bit quantization scenario, as shown in Fig.3. By comparing these three figures, we can clearly observe the differences in their quantization effects. Upon careful examination of the magnified portions in Fig.3(b) and Fig.3(c) within the range of 0.00 to 0.04), we can see that the POST method provides coarser quantization levels near zero compared to the POT method. This indicates that the POST scheme effectively mitigates the rigid resolution problem. Simultaneously, we have achieved computational efficiency comparable to POT (Power Of Two) by employing a Look-Up Table (LUT) method, with only a slight increase in space utilization.

3.3.1. Computation.

In hardware computation, multiplying an integer power of 2, such as 2^n , by another integer r can be efficiently achieved through simple bit-shifting, bypassing the need for a complex multiplier. A similar approach can be applied to the square root of two. Specifically, we can equate $\sqrt{2}^n$ to $2^{(\frac{n}{2})}$. When n is even, $\frac{n}{2}$ is an integer, allowing for simple bitwise shifting of r to perform the multiplication. However, when n is odd, $\frac{n}{2}$ is not an integer, rendering the shifting technique ineffective. For low-bit scenarios, where the integer space is limited, a Look-Up Table (LUT) method can effectively address this special case. For example, when quantizing a model to 3 bits, a table of size [8,4] is sufficient to handle all cases where n is odd in integer operations

within neural networks. This computation method can be represented by the following expression:

$$\sqrt{2}^{-\mathbf{W}} \mathbf{A} = \begin{cases} \mathbf{A}, & \text{if } \mathbf{W} = 0, \\ \mathbf{A} \gg \mathbf{W}/2, & \text{if } \mathbf{W} \text{ is even}, \\ LUT(\mathbf{W}, \mathbf{A}), & \text{if } \mathbf{W} \text{ is odd}. \end{cases} \quad (18)$$

Here, \mathbf{W} and \mathbf{A} represent the weights and activations in the deep neural network that have been mapped to fixed-point numbers with a specific bit-width. Compared to uniform quantization, our method achieves approximately b -fold faster multiplication operations, where b denotes the bit width used in the quantization process.

4. Experiments

In this section, we validate our proposed method on vanilla ResNet [28] and MobileNet-V2 [29] using ImageNet-ILSVRC2012 [30] and CIFAR10 [31] datasets. We also conduct an ablation study on the various components of our algorithm. All experiments were implemented using PyTorch.

4.1. Implimentation Details

4.1.1. Basic setup.

We established two quantization schemes: Scheme 1 employs pure uniform quantization, using our proposed ASQ (Adaptive Step size Quantization) for activations and a fixed trainable step size uniform quantization scheme (LSQ - Learned Step Size Quantization) for weights. Scheme 2 adopts a non-uniform quantization approach for weights: while still using ASQ for activations, it employs a POST (Power Of Square of Two) quantizer for non-uniform exponential mapping of weights. For Scheme 1, we conducted comparisons across 2, 3, 4, and 8-bit quantization. For Scheme 2, we only compared 3 and 4-bit widths. This is because 8-bit quantization results in negligible information loss, and using POST in actual inference would introduce significant additional space overhead. With 2-bit quantization, applying non-uniform symmetric quantization to weights would actually turn it into ternarisation, negating the benefits of non-uniform quantization.

4.1.2. Training.

Following the training scheme outlined in LSQ [6], We utilized the SGD optimizer and a cosine learning rate decay without restarts [32]. All weights in the quantized model were initialized using the official PyTorch pre-trained full-precision model. The Adapter will be initialized to produce output values close to 1. This initialization strategy is designed to ensure that the adapter does not make excessive adjustments to the quantization step size in the initial stages.

4.1.3. Architecture.

For a fair comparison, we employed vanilla ResNet and MobileNet-V2 architectures without any modifications. All convolutional and linear layers were quantized to low bit-widths, except for the first and last layers of the model, which were consistently quantized to 8-bit.

4.2. Evaluation on ImageNet

We evaluated the performance of ASQ and POST on the ResNet18/34 and MobileNet-V2 models using the ImageNet dataset. The models were trained for 90 epochs with an initial learning rate of 0.01 for the weights. The batch size was set to 512 for ResNet-18/34 and 256 for MobileNet-V2. Additionally, the weight decay was set to $1e-4$. 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 presents a comparison of the accuracy of two quantization strategies (ASQ and ASQ+POST) on ResNet models against other state-of-the-art (SOTA) models. Whether compared to uniform or non-uniform quantization, the results demonstrate a significant performance advantage for both strategies. It is worth noting that non-uniform quantization strategies show more pronounced effects at larger bit widths, as the larger discrete value set provides more flexible training space for logarithmic quantization. It can be observed that our 4-bit and 8-bit quantized networks achieve higher accuracy than the full-precision baseline network (On ResNet18, ASQ achieved a 1.0% accuracy improvement with 4-bit quantization and a 1.2% improvement with 8-bit quantization. With ASQ+POST, the 4-bit quantization improved accuracy by 1.2%. On ResNet34, ASQ improved accuracy by 0.6% with 4-bit quantization and 1.0% with 8-bit quantization, while ASQ+POST achieved a 0.8% improvement with 4-bit quantization.).

Table 2: Comparison of low precision Resnet on ImageNet2012. Methods include PACT [5], LQ-Net [21], DSQ [26], FAQ [33], QIL [20], APOT [23], LSQ [6], DAQ [34], LIMPQ [35], SEAM [36], RMQ [37]. † indicates that the method uses mixed-precision quantization. * our re-implementation. To ensure a fair comparison, we have listed the accuracy of the full-precision baseline model used by all the methods we compare. All subsequent comparison tables follow the same approach.

Network	Method	Top-1 Accuracy @ Precision					Top-5 Accuracy @ Precision				
		32	2	3	4	8	32	2	3	4	8
ResNet18	PACT	70.4	64.4(-6.0)	68.1(-2.3)	69.2(-1.2)	-	89.6	85.6(-4.0)	88.2(-1.4)	89.0(-0.6)	-
	LQ-Net	70.3	64.9(-5.4)	68.2(-2.1)	69.3(-1.0)	-	89.5	84.3(-5.2)	85.9(-3.6)	88.8(-0.7)	-
	DSQ	69.9	65.2(-4.7)	68.7(-1.2)	69.6(-0.3)	-	-	-	-	-	-
	FAQ	69.8	-	-	69.8(+0.0)	70.0(+0.2)	89.1	-	-	89.1(+0.0)	89.3(+0.2)
	QIL	70.2	65.7(-4.5)	69.2(-1.0)	70.1(-0.1)	-	-	-	-	-	-
	APOT	69.8	66.5(-3.3)	69.8(+0.0)	70.7(+0.9)	-	89.4	87.5(-1.9)	89.2(-0.2)	89.6(+0.2)	-
	LSQ*	69.8	66.3(-3.5)	69.3(-0.5)	70.3(+0.5)	70.7(+0.9)	89.1	87.0(-2.1)	89.0(-0.1)	89.5(+0.4)	89.7(+0.6)
	DAQ	69.9	66.9(-3.0)	69.6(-0.3)	70.5(+0.6)	-	-	-	-	-	-
	LIMPQ†	70.5	-	69.7(-0.8)	70.8(+0.3)	-	-	-	-	-	-
	SEAM†	70.5	-	70.0(-0.5)	70.8(+0.3)	-	-	-	-	-	-
	RMQ†	70.5	-	70.2(-0.3)	71.0(+0.5)	-	-	-	-	-	-
	ASQ(ours)	69.8	67.1(-2.7)	69.8(+0.0)	70.8(+1.0)	71.0(+1.2)	89.1	87.4(-1.7)	89.2(+0.1)	89.7(+0.6)	89.8(+0.7)
	ASQ+POST(ours)	69.8	-	70.0(+0.2)	71.0(+1.2)	-	89.1	-	89.3(+0.2)	89.9(+0.8)	-
ResNet34	LQ-Net	73.8	69.8(-4.0)	71.9(-1.9)	-	-	91.4	89.1(-2.3)	90.2(-1.1)	-	-
	DSQ	73.8	70.0(-3.8)	72.5(-1.3)	72.8(-1.0)	-	-	-	-	-	-
	FAQ	73.3	-	-	73.3(+0.0)	73.7(+0.2)	91.4	-	-	91.3(-0.1)	91.6(+0.2)
	QIL	73.7	70.6(-3.1)	73.1(-0.6)	73.7(+0.0)	-	-	-	-	-	-
	APOT	73.7	70.9(-2.8)	73.4(-0.3)	73.8(+0.1)	-	91.3	89.7(-1.6)	91.1(-0.2)	91.6(+0.3)	-
	LSQ*	73.3	70.6(-2.7)	73.1(-0.2)	73.7(+0.4)	74.1(+0.8)	91.4	89.8(-1.6)	91.3(-0.1)	91.5(+0.1)	91.8(+0.4)
	DAQ	73.3	71.0(-2.3)	73.1(-0.2)	73.7(+0.4)	-	-	-	-	-	-
	ASQ(ours)	73.3	71.3(-2.0)	73.3(+0.0)	73.9(+0.6)	74.3(+1.0)	91.4	90.2(-1.2)	91.4(+0.0)	91.6(+0.2)	91.8(+0.4)
	ASQ+POST(ours)	73.3	-	73.4(+0.1)	74.1(+0.8)	-	91.4	-	91.4(+0.0)	91.7(+0.3)	-

The 3-bit quantized network with ASQ also maintained the accuracy of the full-precision baseline model. When the bit-width was further reduced to 2, although there was a more significant drop in accuracy compared to the baseline model, our approach still outperformed other methods. These results indicate that dynamically adjusting the quantization step size of activations by training the adaptive module (Adapter), along with using a non-uniform quantization scheme, can indeed effectively improve the accuracy of quantized models. Moreover, when using our quantization scheme 2 (which uses POST quantization for weights), our model achieves better hardware performance in terms of inference speed.

Even on the compact and efficient MobileNet-V2 model, our method demonstrates improvements compared to other approaches, as shown in Table 3. Although the gains are not as pronounced as those observed on ResNet models, this still highlights the generalizability of our approach.

4.3. Evaluation on CIFAR-10

We also conducted experiments on the CIFAR-10 dataset. For these experiments, we set the initial learning rate for the weights to 0.04, with a batch size of 128. The weight decay was set to 1e-4, and momentum was set

Table 3: Comparison of low precision MobileNet-V2 on ImageNet2012. Methods include PACT [5], DSQ [26], NIPQ [38], LSQ [6]. † indicates that the method uses mixed-precision quantization. * our re-implementation.

Network	Method	Top-1 Accuracy @ Precision		
		32	3	4
MobileNet-V2	PACT	71.8	-	61.4(-10.4)
	DSQ	71.9	-	64.8(-7.1)
	NIPQ†	72.6	62.3(-10.3)	69.2(-3.4)
	LSQ*	71.9	65.2(-6.7)	69.5(-2.4)
	ASQ(ours)	71.9	65.6(-6.3)	69.8(-2.1)
	ASQ+POST(ours)	71.9	66.6(-5.3)	70.1(-1.8)

to 0.9. Each quantized model was trained for 300 epochs. For the CIFAR-10 dataset, we applied standard data augmentation techniques, including random cropping and horizontal flipping.

Table 4: Comparison of low precision ResNet on CIFAR-10. Methods include LQ-Net [21], LSQ [6], APOT [23]. * our re-implementation.

Network	Method	Top-1 Accuracy @ Precision			
		32	2	3	4
ResNet20	LQ-Net	92.1	90.2(-1.9)	91.6(-0.5)	-
	LSQ*	92.6	91.0(-1.6)	92.0(-0.6)	92.4(-0.2)
	APOT*	92.6	91.0(-1.6)	92.2(-0.4)	92.7(+0.1)
	ASQ(ours)	92.6	91.2(-1.4)	91.7(-0.9)	92.8(+0.2)
	ASQ+POST(ours)	92.6	-	92.3(-0.3)	93.0(+0.4)

Table 4 summarizes the accuracy comparison between our method and other approaches against the baseline model. For the 3-bit and 4-bit models, both quantization schemes achieved results comparable to the full-precision baseline model. Notably, our method outperformed both the uniform quantization method LSQ [6] and the non-uniform quantization methods LQ-Nets [21] and APoT [23] across all bit widths from 2 to 4.

4.4. Ablation Studies

The proposed algorithm incorporates two key techniques: an Adaptive Step Size Quantization (ASQ) for activation quantization and a Power Of Square root of Two (POST) quantization for weight quantization. To assess the impact of each component on overall performance, we conducted a thorough ablation study on the 4-bit quantized ResNet20 network using the CIFAR-10 dataset.

Table 5: Effects of different components of our method on the final performance of a 3-bit quantized ResNet-20 network. Baseline refers to the quantization strategy where both activations and weights are quantized using the learnable step size quantization [6] method.

Method	Top-1 Acc
Baseline(our implementation)	92.4
Baseline+POT(for weight)	92.5
Baseline+POST(for weight)	92.7
Baseline+ASQ(for activation)	92.8
Baseline+POST(for weight)+ASQ(for activation)	93.0
Corresponding real-valued network	92.6

As shown in Table 5, the Baseline refers to the method where both activations and weights are quantized using a trainable step size quantizer (LSQ). It can be observed that employing the POT quantizer for weights does improve quantization performance. However, the improvement is limited due to the rigid resolution issue mentioned earlier. This limitation can be effectively addressed by replacing the POT quantizer with the POST quantizer. Furthermore, optimizing the quantization of activations with the ASQ alone also significantly enhances the accuracy of the quantized model. When both the ASQ is applied to activations and the POST quantizer is used for weights, the quantized model’s accuracy improves by 0.4% compared to the real-valued model.

4.5. Visualization

To better understand the proposed ASQ (Adaptive Step size Quantization) and its benefits, we visualized the distribution of activations before and after quantization, comparing them with LSQ (Learned Step Quantization), as shown in Fig.4. Fixed quantization step sizes during inference can lead to underutilization of the quantized integer space, especially when the step size is too large for certain activations. For instance, in 2-bit quantization, the integer space can represent four values, but LSQ may ends up using only three, leading to information loss and degraded model performance. ASQ addresses this by dynamically adjusting the quantization step size to match the varying activation distributions, better utilizing the integer space and minimizing information loss, thereby improving model performance.

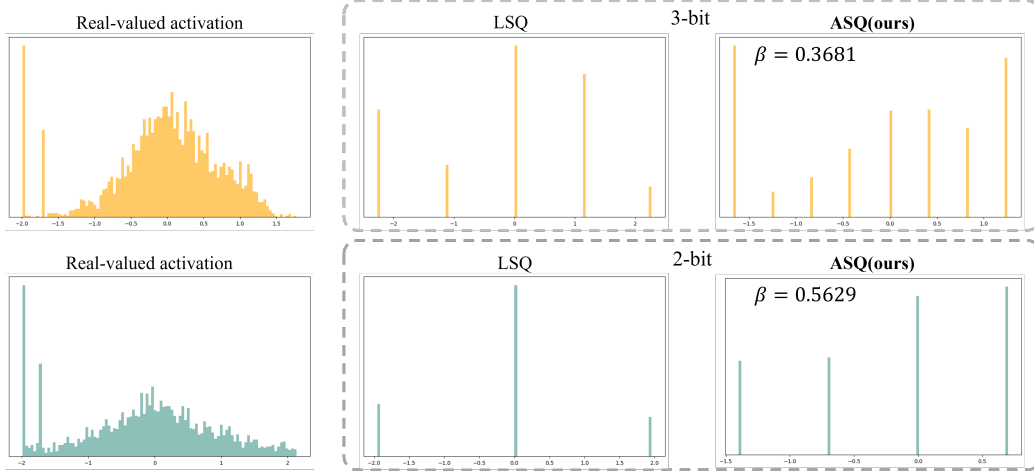


Figure 4: Compare the histograms of the activation distribution before and after quantization using ASQ and LSQ for 2-bit and 3-bit quantization. β is the output of the adaptive module used to adjust the quantization step size.

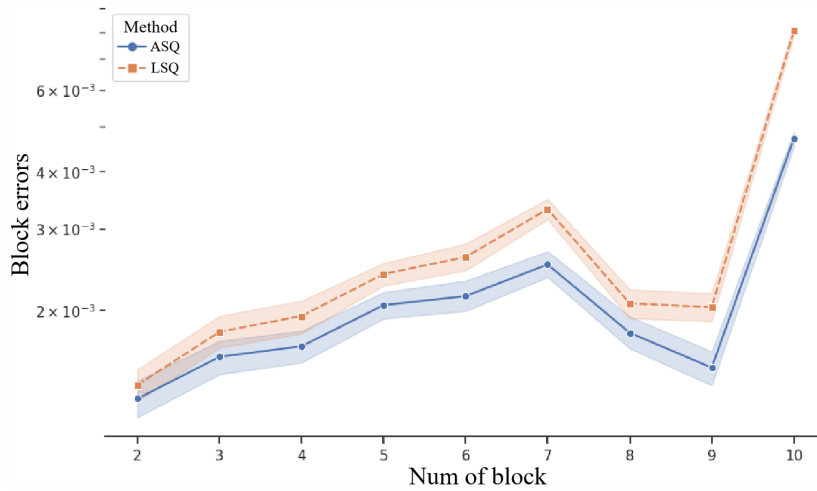


Figure 5: The output error (error accumulation) in blocks 2-9 of ResNet20 using ASQ and LSQ quantizers respectively.

Additionally, as illustrated in Fig.5, we compared the impact of LSQ and ASQ on the outputs of blocks 2 through 9 in ResNet20 by calculating the L2 norm between full-precision and 3-bit quantized activations. This analysis highlights the errors introduced by each block and the cumulative error trend across the network. The results demonstrate ASQ’s effectiveness in reducing

quantization-induced errors at each stage, mitigating error accumulation, and preserving model accuracy even under aggressive quantization, underscoring its superiority over LSQ.

Fig.6 presents a detailed examination of the quantization error encountered during actual inference in a ResNet20 model, specifically due to the rounding and clipping operations within LSQ and ASQ quantization techniques. The error is quantified as the L2 norm, measuring the deviation between activation values before and after the quantization process.

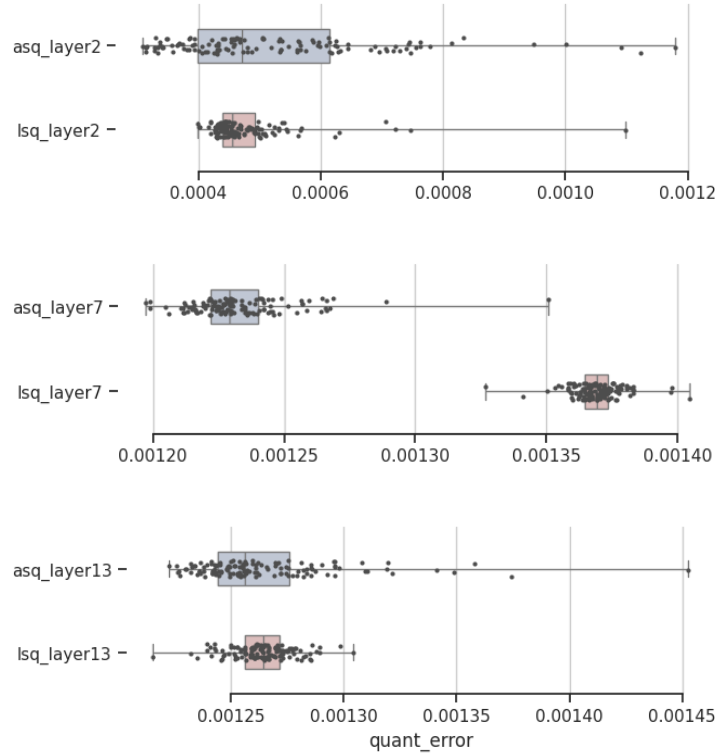


Figure 6: Quantization error of ASQ and LSQ quantizers for Activations in the 2nd, 7th, and 13th layers of ResNet20.

An intriguing observation emerges from these results: the adaptive parameters introduced by ASQ, while designed to optimize quantization, do not universally result in a reduction of quantization error. In fact, in several instances, the quantization error under ASQ surpasses that of the fixed-step size quantizer, LSQ. This is particularly notable in certain layers where ASQ’s

error not only exceeds LSQ’s but does so with a significantly higher variance.

This behavior underscores a crucial insight: a smaller quantization error does not inherently guarantee enhanced model performance. The adaptive nature of ASQ provides it with a unique ability to adjust dynamically, potentially compensating for the inherent information loss that quantization introduces. Consequently, even in scenarios where ASQ exhibits a larger or more variable quantization error, it can still maintain, or even improve, the model’s overall accuracy. This adaptability is particularly beneficial in complex models, where maintaining a balance between quantization error and model performance is critical.

These findings suggest that the effectiveness of a quantization method cannot be solely judged by the magnitude of the quantization error it produces. Instead, one must consider how well the method can preserve the integrity of the model’s performance amidst these quantization challenges.

5. Conclusion

In this paper, we propose a novel methods, called ASQ, to optimize low-bit-width quantization for improved accuracy. Recognizing the sensitivity of low-bit quantizers to activation distribution changes, the poposed ASQ method introduces a dynamic weight design to create adaptive quantization step sizes, effectively adjusting to varying input distributions. The POST for weight introduced in ASQ addresses the rigid resolution issues of POT quantization while maintaining comparable inference speed with minimal memory overhead. Extensive experiments on image classification tasks show that the proposed ASQ significantly enhances quantized model performance, surpassing or matching traditional quantization methods. Ablation studies and visual analyses further confirm that ASQ effectively mitigates information loss due to quantization errors, underscoring the advantages of our techniques.

6. Acknowledgement

This work is supported by the Natural Science Foundation of Sichuan under Grant 24NSFSC3404 and 2023NSFSC0474, the National Major Scientific Instruments and Equipments Development Project of National Natural Science Foundation of China under Grant 62427820, the Fundamental Research Funds for the Central Universities under Grant 1082204112364, the Tianfu Yongxing Laboratory Organized Research Project Funding under

Grant 2023CXXM14, and the Science Fund for Creative Research Groups of Sichuan Province Natural Science Foundation under Grant 2024NSFTD0035.

References

- [1] Z. Zang, C. Lin, C. Tang, T. Wang, J. Lv, Zero-shot aerial object detection with visual description regularization, *Proceedings of the AAAI Conference on Artificial Intelligence* 38 (7) (2024) 6926–6934. doi:10.1609/aaai.v38i7.28518.
URL <https://ojs.aaai.org/index.php/AAAI/article/view/28518>
- [2] C. Tang, Z. He, Y. Li, J. Lv, Zero-shot learning via structure-aligned generative adversarial network, *IEEE Transactions on Neural Networks and Learning Systems* 33 (11) (2022) 6749–6762. doi:10.1109/TNNLS.2021.3083367.
- [3] A. Vaswani, Attention is all you need, *arXiv preprint arXiv:1706.03762* (2017).
- [4] R. D. Peacocke, D. H. Graf, An introduction to speech and speaker recognition, in: *Readings in human–computer interaction*, Elsevier, 1995, pp. 546–553.
- [5] J. Choi, Z. Wang, S. Venkataramani, P. I. Chuang, V. Srinivasan, K. Gopalakrishnan, PACT: parameterized clipping activation for quantized neural networks, *CoRR* abs/1805.06085 (2018). arXiv:1805.06085.
URL <http://arxiv.org/abs/1805.06085>
- [6] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, D. S. Modha, Learned step size quantization, *CoRR* abs/1902.08153 (2019). arXiv:1902.08153.
URL <http://arxiv.org/abs/1902.08153>
- [7] T. Gale, E. Elsen, S. Hooker, The state of sparsity in deep neural networks, *CoRR* abs/1902.09574 (2019). arXiv:1902.09574.
URL <http://arxiv.org/abs/1902.09574>
- [8] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, J. Gutttag, What is the state of neural network pruning?, in: I. Dhillon, D. Papailiopoulos, V. Sze

(Eds.), Proceedings of Machine Learning and Systems, Vol. 2, 2020, pp. 129–146.

- [9] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, Z. Dai, Variational information distillation for knowledge transfer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
- [10] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, J. Kautz, Dreaming to distill: Data-free knowledge transfer via deepinversion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
- [11] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran, Low-rank matrix factorization for deep neural network training with high-dimensional output targets, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6655–6659. doi:10.1109/ICASSP.2013.6638949.
- [12] T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: A survey, Journal of Machine Learning Research 20 (55) (2019) 1–21. URL <http://jmlr.org/papers/v20/18-598.html>
- [13] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, H. Adam, Searching for mobilenetv3, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.
- [14] Y. Li, S. Xu, M. Lin, X. Cao, C. Liu, X. Sun, B. Zhang, Bi-vit: Pushing the limit of vision transformer quantization, Proceedings of the AAAI Conference on Artificial Intelligence 38 (4) (2024) 3243–3251. doi:10.1609/aaai.v38i4.28109. URL <https://ojs.aaai.org/index.php/AAAI/article/view/28109>
- [15] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, K. Keutzer, A survey of quantization methods for efficient neural network inference, in: Low-Power Computer Vision, Chapman and Hall/CRC, 2022, pp. 291–326.

- [16] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, N. Kwak, Lsq+: Improving low-bit quantization through learnable offsets and better initialization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2020.
- [17] Y. Bengio, N. Léonard, A. C. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation, CoRR abs/1308.3432 (2013). [arXiv:1308.3432](https://arxiv.org/abs/1308.3432).
URL <http://arxiv.org/abs/1308.3432>
- [18] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, et al., From the cover: Convolutional networks for fast, energy-efficient neuromorphic computing, Proceedings of the National Academy of Sciences of the United States of America 113 (41) (2016) 11441.
- [19] J. L. McKinstry, S. K. Esser, R. Appuswamy, D. Bablani, J. V. Arthur, I. B. Yildiz, D. S. Modha, Discovering low-precision networks close to full-precision networks for efficient embedded inference, CoRR abs/1809.04191 (2018). [arXiv:1809.04191](https://arxiv.org/abs/1809.04191).
URL <http://arxiv.org/abs/1809.04191>
- [20] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, C. Choi, Learning to quantize deep networks by optimizing quantization intervals with task loss, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
- [21] D. Zhang, J. Yang, D. Ye, G. Hua, Lq-nets: Learned quantization for highly accurate and compact deep neural networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.
- [22] D. Miyashita, E. H. Lee, B. Murmann, Convolutional neural networks using logarithmic data representation, CoRR abs/1603.01025 (2016). [arXiv:1603.01025](https://arxiv.org/abs/1603.01025).
URL <http://arxiv.org/abs/1603.01025>
- [23] Y. Li, X. Dong, W. Wang, Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks, arXiv preprint [arXiv:1909.13144](https://arxiv.org/abs/1909.13144) (2019).

- [24] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, Z. Liu, Dynamic convolution: Attention over convolution kernels, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
- [25] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, Y. Zou, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, CoRR abs/1606.06160 (2016). [arXiv:1606.06160](https://arxiv.org/abs/1606.06160).
URL <http://arxiv.org/abs/1606.06160>
- [26] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, J. Yan, Differentiable soft quantization: Bridging full-precision and low-bit neural networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.
- [27] A. Zhou, A. Yao, Y. Guo, L. Xu, Y. Chen, Incremental network quantization: Towards lossless cnns with low-precision weights, CoRR abs/1702.03044 (2017). [arXiv:1702.03044](https://arxiv.org/abs/1702.03044).
URL <http://arxiv.org/abs/1702.03044>
- [28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
- [29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
- [30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.
- [31] A. Krizhevsky, et al., Learning multiple layers of features from tiny images (2009).
- [32] I. Loshchilov, F. Hutter, Sgdr: Stochastic gradient descent with warm restarts, arXiv preprint [arXiv:1608.03983](https://arxiv.org/abs/1608.03983) (2016).

- [33] J. L. McKinstry, S. K. Esser, R. Appuswamy, D. Bablani, J. V. Arthur, I. B. Yildiz, D. S. Modha, Discovering low-precision networks close to full-precision networks for efficient inference, in: 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS), 2019, pp. 6–9. doi:10.1109/EMC2-NIPS53020.2019.00009.
- [34] D. Kim, J. Lee, B. Ham, Distance-aware quantization, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 5271–5280.
- [35] C. Tang, K. Ouyang, Z. Wang, Y. Zhu, W. Ji, Y. Wang, W. Zhu, Mixed-precision neural network quantization via learned layer-wise importance, in: S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, T. Hassner (Eds.), Computer Vision – ECCV 2022, Springer Nature Switzerland, Cham, 2022, pp. 259–275.
- [36] C. Tang, K. Ouyang, Z. Chai, Y. Bai, Y. Meng, Z. Wang, W. Zhu, Seam: Searching transferable mixed-precision quantization policy through large margin regularization, in: Proceedings of the 31st ACM International Conference on Multimedia, MM ’23, Association for Computing Machinery, New York, NY, USA, 2023, p. 7971–7980. doi:10.1145/3581783.3611975.
URL <https://doi.org/10.1145/3581783.3611975>
- [37] C. Tang, Y. Meng, J. Jiang, S. Xie, R. Lu, X. Ma, Z. Wang, W. Zhu, Retraining-free model quantization via one-shot weight-coupling learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 15855–15865.
- [38] J. Shin, J. So, S. Park, S. Kang, S. Yoo, E. Park, Nipq: Noise proxy-based integrated pseudo-quantization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 3852–3861.