RAPQ: Rescuing Accuracy for Power-of-Two Low-bit Post-training Quantization

Hongyi Yao, **Pu Li**, **Jian Cao**[*], **Xiangcheng Liu**, **Chenying Xie** and **Bingzhang Wang**
Peking University
yhy@stu.pku.edu.cn, spurslipu@pku.edu.cn, caojian@ss.pku.edu.cn, liuxiangcheng@stu.pku.edu.cn,
402600293@qq.com, 13919334117@163.com

Abstract

We introduce a Power-of-Two low-bit post-training quantization(PTQ) method for deep neural network that meets hardware requirements and does not call for long-time retraining. Power-of-Two quantization can convert the multiplication introduced by quantization and dequantization to bit-shift that is adopted by many efficient accelerators. However, the Power-of-Two scale factors have fewer candidate values, which leads to more rounding or clipping errors. We propose a novel Power-of-Two PTQ framework, dubbed RAPQ, which dynamically adjusts the Power-of-Two scales of the whole network instead of statically determining them layer by layer. It can theoretically trade off the rounding error and clipping error of the whole network. Meanwhile, the reconstruction method in RAPQ is based on the BN information of every unit. Extensive experiments on ImageNet prove the excellent performance of our proposed method. Without bells and whistles, RAPQ can reach accuracy of 65% and 48% on ResNet-18 and MobileNetV2 respectively with weight INT2 activation INT4. We are the first to propose the more constrained but hardware-friendly Power-of-Two quantization scheme for low-bit PTQ specially and prove that it can achieve nearly the same accuracy as SOTA PTQ method. The code[1] was released.

1 Introduction

In recent years, convolutional neural network (CNN) has been widely used in computer vision tasks. The improvement of hardware computation considerably accelerates model evolution, which produces deeper and more complex CNN models to pursue even higher accuracy. However, the deep CNN models are difficult to be deployed on resource-limited edge devices. How to reduce the model scale while maintaining the model accuracy is a trending topic in current research. In this paper, we study quantization which aims to reduce bit-width of weights and activations to enable fixed-point computation and less memory space.

Based on data usage, model quantization can be divided into three categories: (1) quantization-aware-training (QAT), (2) post-training quantization (PTQ) and (3) data-free Quantization (DFQ). QAT requires fine-tuning the model on the whole dataset, which inevitably requires a large amount of GPU resources and time cost. In contrast, PTQ demands only a small set of readily available calibration data. Although DFQ achieves quantization without dataset, its accuracy has not reached the desired level and it is difficult to apply in industrial scenarios. Therefore, this paper focuses on improving PTQ performance.

Moreover, scale factors that are constrained to the form of Power-of-Two make quantization and dequantization convert to simple bit-shift. However, compared with the float scale factors,the Power-of-Two value is essentially a discrete approximation of the float value, which will cause more rounding error or more clipping error. This significantly reduces performance of the quantized model.

We are the first to implement hardware-friendly Power-of-Two low-bit PTQ and surprisingly observe that constrained Power-of-Two PTQ can achieve nearly the same accuracy as SOTA PTQ method.

2 Related Work

2.1 Network Quantization

QAT [Krishnamoorthi, 2018] mainly adopts the STE for gradients approximation to solve the non-differentiable round problem. [Gong *et al.*, 2019] uses a differentiable quantizer to gradually approach the round function. However, QAT usually depends on the whole dataset and GPU resources to train the model. Without high cost, most models can be safely quantized to 8-bit even lower-bit by PTQ. AdaRound [Nagel *et al.*, 2020] proposed to learn the rounding way by layer reconstruction brings much improvement of accuracy. BRECQ [Li *et al.*, 2020] focused more on block reconstruction with better accuracy at 2-bit weight quantization than AdaRound.

The Power-of-Two scale factor [Miyashita *et al.*, 2016] has the advantage of reducing computation complexity. But it meanwhile produces accuracy loss. To solve this problem, [Li *et al.*, 2019] proposed an efficient method APoT for the weights and activations with bell-shaped and long-tailed dis-

[*] Corresponding author

[1] https://github.com/BillAmihom/RAPQ

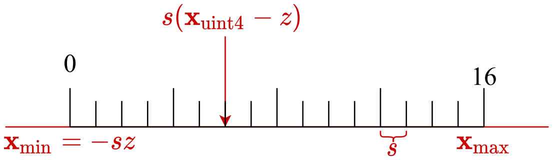


Figure 1: Visual explanation of the asymmetric uniform affine quantization grids for a bit-width of 4.

tribution in neural networks. But APoT is a non-uniform quantization QAT scheme.

3 Motivation

For simplicity, we omit the analysis of ***bias*** as it can be merged into activation. In this way, the forward propagation of CNN can be expressed by Equation (1).

$$\boldsymbol{x}_{(k+1)} = \mathcal{R}\left(\boldsymbol{y}_{(k)}\right) = \mathcal{R}\left(\boldsymbol{W}_{(k)} * \boldsymbol{x}_{(k)}\right) \tag{1}$$

where $*$ represents the convolution. $\mathcal{R}(\cdot)$ is the activation function. $\boldsymbol{x}_{(k)}$ is the input of the k-th layer, $\boldsymbol{x}_{(k+1)}$ is the output activation of the kth layer, and $\boldsymbol{y}_{(k)}$ is the convolution result of the k-th layer.

The essence of quantization is to map floating-point number to low-bit fixed-point number. The quantization and dequantization process of non-uniform quantization often brings huge computing burden. So in this paper, we focus on uniform affine quantization. To quantize vector $\boldsymbol{x}$, with s denoting the scale of the floating-point number mapped to the fixed-point number, and z denoting the zero-point of shifting the number to the specified range, the process of quantization of input $\boldsymbol{x}$ can be described by Equation (2).

$$\hat{\boldsymbol{x}} = s \cdot \left[clip\left(\left\lfloor \frac{\boldsymbol{x}}{s} + z \right\rceil, n, p \right) - z \right] \tag{2}$$

where $\lfloor \cdot \rceil$ represents the rounding. The quantized variables are marked by $\hat{}$, $clip(\cdot)$ represents that input will be clipped into $[n, p]$, in the case of asymmetric quantization $n = 0$ and $p = 2^b - 1$, b is the bit-width.

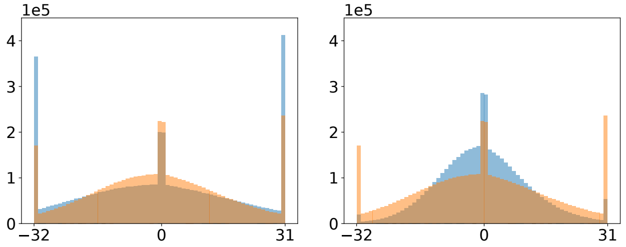
If we use a grid diagram to represent the range of fixed-point numbers, we can interpret s as the step length between two grids, and z determines the utilization interval of fixed-point numbers[Nagel *et al.*, 2021], as shown in Figure 1.

3.1 Relationship Between Scale and Rounding Error & Clipping Error

So far, there is no Power-of-Two quantization framework specifically for PTQ. But the hardware requirement for Power-of-Two scale factors usually occurs. The naive method is to first quantize the model with float scale factors, and replace them with the closest Power-of-Two scale factors. Expressed in Equation (3) is

$$\hat{\boldsymbol{x}} = s_{pow-2} \cdot \left[\text{clip}\left(\left\lfloor \frac{\boldsymbol{x}}{s_{pow-2}} + z' \right\rceil, 0, 2^{b-1} \right) - z' \right] \tag{3}$$

$$s_{pow-2} = 2^{\lfloor \log_2 s \rceil} \tag{4}$$



(a) Data quantied by $\lfloor \log_2 s \rfloor$ (b) Data quantied by $\lceil \log_2 s \rceil$

Figure 2: 6-bit symmetric uniform affine quantization for weights in 73rd layer of DarkNet53

$$z' = -\frac{x_{min}}{s_{pow-2}} \tag{5}$$

where x_{min} is the mapped smallest floating-point number in the vector $\boldsymbol{x}$. Equation (4) is the Power-of-Two scale replaced by the naive method, and Equation (5) is the zero-point updated after scale change.

The discussion on Power-of-Two scale can be divided into the following two cases:

- When $s_{pow-2} < s$, $\lfloor \log_2 s \rceil$ rounded down to $\lfloor \log_2 s \rfloor$, the grid step length decreases, resulting in more clipping error, i.e., there are more data clipped at the maximum number of fixed points, and the values of outliers all become the same value after the dequantization.
- When $s_{pow-2} > s$, $\lfloor \log_2 s \rceil$ rounded up to $\lceil \log_2 s \rceil$, the grid step length increases, resulting in more rounding error, i.e., there are more data laying at every grid, and they all become a same number after dequantization.

To visualize the Power-of-Two scale factor caused clipping and rounding errors, we illustrate with specific quantization data distributions. We use 6-bit symmetric uniform affine quantization for the pre-training weights of the DarkNet53 [Redmon and Farhadi, 2018] on Hand dataset [Mittal *et al.*, 2011]. Figure 2 shows the histogram of the data distribution of the quantized weights at 73rd layer. The orange mask is the distribution of normal scale quantized data, and the blue histogram is the distribution of Power-of-Two scale quantized data. Figure 2(a) shows the data distribution using $2^{\lfloor \log_2 s \rfloor}$ quantization, and Figure 2(b) shows the data distribution using $2^{\lceil \log_2 s \rceil}$ quantization. The relationship between clipping error and rounding error is that they always trade with each other.

The selection of scale in Power-of-Two quantization is essentially a trade-off between rounding error and clipping error. The naive method simply finds a Power-of-Two scale with the smallest value difference from the original scale, which has no connection to model accuracy. Reducing the numerical difference between the Power-of-Two scale and the original scale in every layer does not necessarily decrease the task loss of the model. Besides, even using the greedy strategy to directly select the best Power-of-Two scale of a single layer often can only obtain the local optimal scale factor. So we think there should be a solider method, which adopts task loss as the criterion to choose Power-of-Two scale, theoretically trades off the clipping error and rounding error and eventually obtain the optimal solution of the whole network.

3.2 Regression Loss Function for Reconstruction

Recent excellent PTQ work AdaRound [Nagel *et al.*, 2020] performed a second-order Taylor expansion of difference between the task losses before and after quantization by two strong assumptions. It finally convert the difference to a L-2 loss of feature map before and after quantization between layers. [Li *et al.*, 2020] changed the assumptions of [Nagel *et al.*, 2020] and applied this theory to block reconstruction, eventually converting task loss to an L-2 loss minimization of feature map before and after quantization between blocks. Their crude assumptions bring crude conclusion, and the reconstructed regression loss functions all become L-2 loss functions. For L-2 loss minimization, it is easy to show that the regression value is actually the mean value of the array. The mean value is sensitive to outliers of the array, which means that using L-2 loss minimization will produce more rounding error than clipping error. If other reconstruction schemes are used instead of L-2 loss minimization, this comes back to the problem of trading off rounding and clipping errors mentioned in Sec 3.1.

4 Method

The two challenges mentioned in Sec 3.1 and Sec 3.2 have led to a collapse in model accuracy for Power-of-Two low-bit PTQ, a phenomenon that is more pronounced in light-weight networks like MobileNetV2. In this section, we propose two methods to rescue Power-of-Two PTQ from accuracy collapse. These two methods are theoretically well-founded and show significant performance improvement in practice.

4.1 Power-of-Two Scale Group

To address the first problem mentioned in Sec 3.1, we abandon the naive method of determining the Power-of-Two scale layer by layer and look for the Power-of-Two scale group of the entire network or block instead.

The goal of trading off rounding error and clipping error is to make the model more accurate, i.e., the model has a lower task loss. So we directly use the task loss as metric to evaluate performance of quantization.

$$\arg\min_{\hat{\boldsymbol{w}}} \mathbb{E}[\mathcal{L}(\hat{\boldsymbol{w}})] \qquad \hat{\boldsymbol{w}} \in \mathbb{D}_Q \tag{6}$$

where $\mathcal{L}(\cdot)$ is the loss function of the model, $\hat{\boldsymbol{w}}$ represents the quantized model weights and $\mathbb{D}_Q$ is the discrete space of fixed-point numbers. In QAT, this process is easier to optimize by stochastic gradient descent for updating quantization parameters and weights. But in case of PTQ, we can only calibrate the model with a small portion of dataset. To solve this problem, [Nagel *et al.*, 2020] degenerated the loss difference using a second-order Taylor expansion into the Equation (7).

$$\begin{aligned} &\arg\min_{\Delta \boldsymbol{w}} \mathbb{E}[\mathcal{L}(\boldsymbol{x}, \boldsymbol{tgt}, \boldsymbol{w} + \Delta \boldsymbol{w}) - \mathcal{L}(\boldsymbol{x}, \boldsymbol{tgt}, \boldsymbol{w})] \\ &\qquad \approx \mathbb{E}\left[\frac{1}{2} \Delta \boldsymbol{w}_{(k)}^{\mathrm{T}} \boldsymbol{H}(\boldsymbol{w}_{(k)}) \Delta \boldsymbol{w}_{(k)}\right] \end{aligned} \tag{7}$$

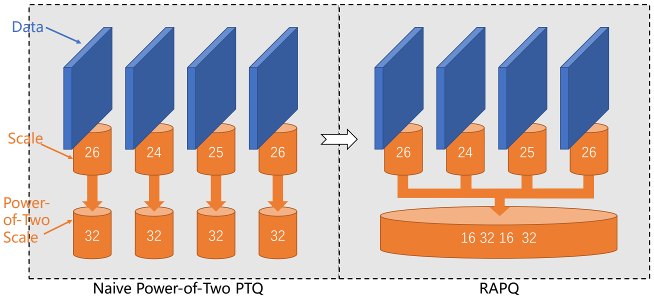


Figure 3: Method 4.1 Power-of-Two Scale Group

where $\boldsymbol{x}$ is the input to the model, $\boldsymbol{tgt}$ is the ground truth, $\boldsymbol{w}$ is the original weight of the model. $\Delta \boldsymbol{w}$ is the perturbation brought by the model quantization to the model weights.

However, the calculation of the Hessian matrix is too complex and solving this problem is not allowed by the computing resources of PTQ. Thus, they assumed that layers are mutual-independent and the second-order derivatives of pre-activation are constant diagonal matrix. In this way, they transformed the problem into an L-2 loss minimization with layer-by-layer feature map reconstruction. Solving this problem only requires focusing on the current layer and solving each subproblem as shown in Equation (8).

$$\arg\min_{\Delta W_{(k)i,:}} \mathbb{E}\left[\left(\Delta \boldsymbol{w}_{(k)i,:} x_{(k)}\right)^2\right] \tag{8}$$

[Li *et al.*, 2020] generalized this work. They ignored inter-block dependencies, and used the diagonal Fisher information matrix (FIM) instead of the pre-activation Hessian matrix [LeCun *et al.*, 2012]. Our optimization objective can be converted into a block-by-block feature map reconstruction problem, as shown in Equation (9).

$$\arg\min_{\hat{\boldsymbol{w}}} \mathbb{E}\left[\Delta \boldsymbol{y}_{(k)}^{\mathrm{T}} \operatorname{diag}\left(\left(\frac{\partial L}{\partial \boldsymbol{y}_{(k)i}}\right)^2\right) \Delta \boldsymbol{y}_{(k)}\right] \tag{9}$$

Where $\Delta \boldsymbol{y}_{(k)}$ is the change of output by quantization. The middle term is the diagonal Fisher information matrix.

According to Equation (9), if we want to calculate the optimal Power-of-Two scale and weights for the whole block, we no longer need to calculate the extremely complicated Hessian matrix, but still need to solve the NP-hard discrete optimization problem. Because the calculation of $\Delta \boldsymbol{y}_{(k)}$ requires $\hat{\boldsymbol{y}}_{(\boldsymbol{k})}$ from the discrete quantization space. Therefore, we degenerate Equation (9) to a continuous optimization problem Equation (10) which is based on soft quantization variables, in order to solve it using the back propagation algorithm.

$$\begin{aligned} &\arg\min_{\boldsymbol{U},\boldsymbol{V}} \|\Delta \boldsymbol{y}_{(k)}\|_F^2 + \lambda f_{reg}(\boldsymbol{U}) + \mu f_{reg}(\boldsymbol{V}) \\ &= \left\|\widetilde{\boldsymbol{y}}_{(k)} - \boldsymbol{y}_{(k)}\right\|_F^2 + \lambda f_{reg}(\boldsymbol{U}) + \mu f_{reg}(\boldsymbol{V}) \end{aligned} \tag{10}$$

where $\|\cdot\|_F^2$ denotes the L-2 loss and $\widetilde{\boldsymbol{y}}_{(\boldsymbol{k})}$ is the result of $\widetilde{\boldsymbol{W}}$ and input calculation. $\widetilde{\boldsymbol{W}}$ is weights of the soft quantization.

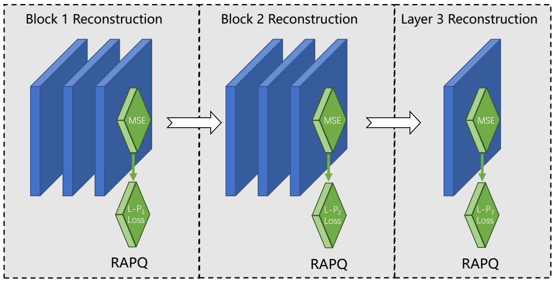


Figure 4: Method 4.2 BN based L-P Loss

The so-called soft quantization is to first replace the discrete quantized variables with continuous float variables in order to back propagate and help the model converge. With the help of differentiable regularizer $\lambda f_{reg}(\boldsymbol{U})$ and $\mu f_{reg}(\boldsymbol{V})$, the data will eventually converge or clip to a truly quantized fixed-point discrete space. Expanding $\widetilde{\boldsymbol{W}}$, we have Equation (11).

$$\begin{aligned}\widetilde{\boldsymbol{W}} = \widetilde{s}_{pow-2} \cdot \left[\text{clip}\left(\left\lfloor \frac{\boldsymbol{W}}{\widetilde{s}_{pow-2}} + z' \right\rfloor + h_1(\boldsymbol{U}), 0, 2^{b-1}\right) - z'\right] \\ \widetilde{s}_{pow-2} = 2^{\lfloor \log_2 s \rfloor + h_2(\boldsymbol{V})} \\ z' = -\frac{\boldsymbol{x}_{(\min)}}{\widetilde{s}_{pow-2}} = \frac{s \cdot z}{\widetilde{s}_{pow-2}}\end{aligned} \tag{11}$$

where $\lfloor \cdot \rfloor$is the downward rounding operation, $\widetilde{\boldsymbol{W}}$ is the weight of soft quantization, and $\widetilde{s}_{pow-2}$ is the soft Power-of-Two scale. $h_1(\cdot)$ and $h_2(\cdot)$ are differentiable functions that take values between $[0, 1]$ and serve to process the trainable tensor $\boldsymbol{U}$,$\boldsymbol{V}$ in soft quantization and map them to 0 or 1 eventually.

However, we cannot guarantee that the trainable variables $\boldsymbol{U}$ of scale and $\boldsymbol{V}$ of weight in Equation (10) converge at the same time. If $\boldsymbol{V}$ converges before $\boldsymbol{U}$ it will lead to the problem that the converged Power-of-Two scale does not match the converged weight. Therefore, we convert the nonlinear programming problem into two binary constrained optimization problems to be solved in two steps.

1. Look for the optimal solution for the Power-of-Two scale group by Equation(12).

$$\arg\min_{\boldsymbol{V}} \left\| \widetilde{\boldsymbol{y}}_{(k)} - \boldsymbol{y}_{(k)} \right\|_F^2 + \mu f_{reg}(\boldsymbol{V}) \tag{12}$$

Freeze the Power-of-Two scale after the variable $\boldsymbol{V}$ converges.

2. Look for the optimal solution to the quantized weight $\widetilde{\boldsymbol{W}}$ by Equation (13).

$$\arg\min_{\boldsymbol{U}} \left\| \widetilde{\boldsymbol{y}}_{(k)} - \boldsymbol{y}_{(k)} \right\|_F^2 + \lambda f_{reg}(\boldsymbol{U}) \tag{13}$$

After convergence of the variable $\boldsymbol{U}$, the quantized $\hat{\boldsymbol{W}}$ are stored.

Thus problem (11) is transformed into two binary constrained optimization problems. Both problems are large scale combinatorial problems, and referring to the work of [Nagel *et al.*, 2020] we solve these two problems using an efficient approximation algorithm Hopfield methods [Hopfield and Tank, 1985].

For the training functions $h_1(\cdot)$, $h_2(\cdot)$ we adopt the rectified sigmoid function, as shown in Equation (14), which is proposed in [Louizos *et al.*, 2018].

$$h\left(\boldsymbol{x}_{ij}\right) = \text{clip}\left(\text{sigmoid}\left(\boldsymbol{x}_{ij}\right)(\xi - \gamma) + \gamma, 0, 1\right) \tag{14}$$

where ξ and γ are the stretching parameters, fixed at 1.1 and -0.1, respectively. They help the rectified sigmoid function to converge more easily to the extremities 0 and 1, and are not as prone to gradient vanishing as sigmoid function. For regularizer we choose:

$$f_{reg}(\boldsymbol{x}) = \sum_{i,j} 1 - \left|2h\left(\boldsymbol{x}_{i,j}\right) - 1\right|^{\beta} \tag{15}$$

Because in this regularizer, we can achieve annealing by controlling the β. $h(x)$ can easily converge to 0 or 1 when the value of β drops to a low value.

The activations cannot be quantized using adaptive rounding because they vary with different input. Thus, we can only adjust its zero-point and Pow-of-Two scale. Referring to back propagation of TQT [Jain *et al.*, 2019], when calculating the gradient, we approximate by taking $s \approx 2^{\lfloor \log_2 s \rceil}$and $\lfloor \frac{\boldsymbol{x}}{s} + z \rceil \approx \frac{\boldsymbol{x}}{s} + z$. Then Power-of-Two scale s_{pow-2} can be back-propagated by Equation (16).

$$\begin{aligned}&\nabla_{log_{\mathbf{2}} S_{pow-2}} \hat{\boldsymbol{x}} = \\ &s \cdot \ln 2 \begin{cases} \lfloor \frac{\boldsymbol{x}}{s} + z \rceil - (\frac{\boldsymbol{x}}{s} + z) & 0 \leq \lfloor \frac{\boldsymbol{x}}{s} + z \rceil \leq 2^b - 1 \\ 0 & \lfloor \frac{\boldsymbol{x}}{s} + z \rceil < 0 \\ 2^{b-1} & \lfloor \frac{\boldsymbol{x}}{s} + z \rceil > 2^b - 1 \end{cases}\end{aligned} \tag{16}$$

4.2 BN-based L-P Loss

To address the second problem mentioned in Sec 3.2, we introduce the minimum L-P loss problem, as shown in Equation (17), to measure the difference between the feature maps before and after quantization.

$$\arg\min_f \mathcal{L}(f) = \arg\min_f \sum_{i=1}^{n} \left|\boldsymbol{x}_i - f(a, b, c, \cdots)\right|^P \tag{17}$$

where $P \in [1, +\infty)$.There is the following conclusions:

- L-1 loss is $Median$ regression. It is not sensitive to outliers since $Median$ simply has an equal number of positive and negative deviations.
- L-2 loss is $Mean$ regression, which is more sensitive to outliers since it has a positive and negative deviation of the sum of 0.
- L-$+\infty$ loss is $Midrange\ Number$ regression, highly sensitive to outliers, with a maximum positive deviation and a minimum negative deviation summed to 0.

From these three special cases it is easy to see a proven theory that the sensitivity of the regression value of L-P loss to outliers increases as the P-value increases.

It is not reasonable to use one L-P loss for all of layers because the data distribution of each layer activation is very different. For example, when quantizing the last layer, we should keep his outlier characteristics because these outliers are about to become the most important scores to predict the classification; when quantizing the middle layer, if the regression is insensitive to outliers, we can let the data distribution before and after quantization more similarly.

As shown in Equation (18), the batch nomlization (BN) [Ioffe and Szegedy, 2015] is widely available in today's neural networks.

$$\widetilde{y}_{(k)} = \gamma \cdot \frac{y_{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta \tag{18}$$

where $y_{(k)}$ is the original activation of the layer, and $\widetilde{y}_{(k)}$ is the activation after BN. μ_B,σ_B^2 are the Mean and Variance of the mini-batch while training. When the model stops training, μ_B and σ_B^2 are replaced by $\mu_{running}$ and $\sigma_{running}^2$,which are the running Mean and running Variance of the statistics in training. γ and $\sigma_{running}^2$ reflect, to some extent, the statistical Variance of the data when the model was previously trained. The Variance is used to measure the degree of deviation between a random variable and its mathematical expectation and can reflect the degree of deviation of the data.

γ reflects the Variance information of the data after BN, while $\sigma_{running}^2$ reflects the Variance information of the data before BN. In practical hardware inference, the BN-layer is fused in the convolutional layer, so we need the Variance information after BN. Taking block as the unit, the layers that do not belong to block take each layer as a single unit. We take the parameter γ of BN information of the last layer of each unit, and find the average γ of all channels as the deviation degree flag of that layer. We define the formula for calculating the P-value of k-th layer BN-based L-P loss:

$$P_{(k)} = 1 + \alpha \cdot \text{sigmoid}\left(\frac{1}{N_k}\sum_{i=1}^{N_k}\gamma_{(k)j} - \beta\right) \tag{19}$$

where α, β are two adjustable parameters,$\alpha \in (0, 1]$, $\beta \in \mathbb{R}$. In order to satisfy the L-P norm definition, $P > 1$ is necessary, but the model is difficult to converge and prone to large clipping errors when $P > 2$. Therefore, we introduce such a differentiable function with a value domain of $[1, 2]$. The larger P-value is, the harder model converges. But when the number of iterations is sufficient, we can increase the disparity of P-values by turning up the α. When the model has extreme γ values, β is needed to help reduce the problem of gradient vanishing of the sigmoid function. For ease of calculation, we usually keep the P-value to 1 2 decimal places.

5 Experiments

It is widely recognized by peers that for CNN-CV tasks, excellent results of quantization method on classification task models are the basis for awesome results of other kind of task models. So we have conducted extensive experiments based on ImageNet [Russakovsky *et al.*, 2015] dataset to demonstrate the superiority of our method. We randomly pick a total of 1024 images for PTQ calibration in each experiment. To be fair, we optimized each model with 80,000 weight iterations and 5000 activation iterations in order to fully converge. At this point, the parameter α corresponding to Equation (19) is set to 0.9 and β is set to 1. Although it is not listed in the table, we have confirmed that we can achieve better results than the data in the table if we fine-tune the α,β in Equation (19) for different models. The optimization of the Power-of-Two scale group will be done during the weight warm-up process. This section is divided into four parts. The first part is an ablation study to demonstrate the effectiveness of our two methods. The second part is a comparison with Power-of-Two quantization work, which demonstrates that our work achieves SOTA. The third part is a comparison with SOTA PTQ work, which demonstrates that we can still achieve a performance close to that of other unconstrained quantization work while satisfying the constraint of hardware-friendly property. The fourth part is set for the time-limited scenario. In order to obtain better quantization results quickly, then the L-P loss and number of iterations setting in the fourth part can be used.

Algorithm 1 : RAPQ Power-of-Two Quantization

Input: Calibration dataset, Pretrained FP model
Parameter: I_s, I_w, I_a iterations
Output:Quantized Model

```
Init s and z by MSE, $[P_i]$ by Equation(19)
while $i = 1, 2, \cdots$ ,N-th do
    if weight optimization in warm-up then
        $I_s$ iterations to find $s_{pow-2}$ group of weight
    else
        Freeze $s_{pow-2}$. $I_w$ iterations to optimize $\hat{W}$
    end if
end while
while $i = 1, 2, \cdots$ ,N-th do
    $I_a$ iterations to find $s_{pow-2}$ group of activation
end while
return Power-of-Two Quantized Model
```

5.1 Ablation Study

It is generally accepted that MobileNetV2 [Sandler *et al.*, 2018] is one of the most difficult lightweight networks to quantize because its weights are extremely susceptible to perturbations. To show the superiority of our method, we conduct ImageNet experiments using MobileNetV2. As shown in Table 3, we perform ablation study limited with weight IN2 activation INT4(W2/A4). It is easy to see that Power-of-Two scale group(Po2 SG) method rescues the accuracy of MobileNetV2. BN-based L-P loss can further improve their accuracy.

5.2 Comparison with SOTA Power-of-Two

As shown in Table 1, we compared our experiment results with TQT and APoT. TQT uses a uniform affine QAT scheme and APoT uses a non-uniform affine QAT scheme while we use a uniform affine PTQ scheme. Both TQT and APoT need the whole dataset while we need only 1024 of it. They spend more than 10 times as long as we do to quantize the model. APoT has achieved better results with W4/A4 than we have

Methods	**Bits (W/A)**	**ResNet-18**	**ResNet-50**	**MobileNetV2**	**RegNet-600MF**	**RegNet-3.2GF**
Full Prec.	32/32	71.08	77.00	72.49	73.71	78.36
TQT [Jain *et al.*, 2019] (QAT)	4/8	67.90*	74.40	47.16	-	-
APoT [Li *et al.*, 2019] (QAT)	4/4	**70.70**	**76.60**	-	-	-
RAPQ (Ours)	4/4	69.28	74.64	**64.48**	**69.59**	**74.25**
RAPQ (Ours)	2/4	65.32	69.71	**48.12**	**61.48**	**69.49**

Table 1: Comparison of RAPQ and SOTA Power-of-Two quantization work on ImageNet

Methods	**Bits (W/A)**	**ResNet-18**	**ResNet-50**	**MobileNetV2**	**RegNet-600MF**	**RegNet-3.2GF**
Full Prec.	32/32	71.08	77.00	72.49	73.71	78.36
ACIQ [Banner *et al.*, 2019]	4/4	67.00	73.80	-	-	-
ZeroQ [Cai *et al.*, 2020]*	4/4	20.80	4.27	25.24	27.95	12.38
AdaRound [Nagel *et al.*, 2020]*	4/4	68.45	74.51	63.94	-	-
AdaQuant [Hubara *et al.*, 2020]	4/4	**69.60**	**75.90**	44.52*	-	-
Bit-Split [Wang *et al.*, 2020]	4/4	67.56	73.71	-	-	-
Brecq [Li *et al.*, 2020]	4/4	**69.60**	75.05	**66.57**	68.33	74.21
RAPQ (Ours)	4/4	69.28	74.64	64.48	**69.59**	**74.25**
ZeroQ [Cai *et al.*, 2020]*	2/4	0.10	0.11	0.13	0.07	0.06
AdaRound [Nagel *et al.*, 2020]*	2/4	64.14	68.40	41.52	59.27	65.33
AdaQuant [Hubara *et al.*, 2020]*	2/4	0.16	0.19	0.08	0.10	0.11
Brecq [Li *et al.*, 2020]	2/4	64.80	**70.29**	**53.34**	59.31	67.15
RAPQ (Ours)	2/4	**65.32**	69.71	48.12	**61.48**	**69.40**

Table 2: Comparison of RAPQ and SOTA PTQ work on ImageNet

Model	MobileNetV2
Naive Power-of-Two	2.64
Po2 SG	46.48
Po2 SG + BN-based L-P loss	48.12

Table 3: Ablation study

Model	W2/A4 Acc	W4/A4 Acc
MobileNetV2	46.68	62.55
ResNet-18	64.76	69.26
ResNet-50	69.20	74.53
RegNet-600MF	60.86	69.51
RegNet-3.2GF	68.47	74.39

Table 4: The experiment results of RAPQ Quick Mode on ImageNet

on ResNet [He *et al.*, 2016], but it introduces weight normalization to smooth the learning process of clipping range in weight. It's impossible to incorporate this technique with BN folding so that it can only reproduce in academic setting[Li *et al.*, 2021]. We are the first in the Power-of-Two quantization work to achieve quantization for MobileNetV2 with W2/A4.

5.3 Comparison with SOTA PTQ

As Table 2 shows, we compare our hardware-constrained PTQ work with the PTQ work without constraint. Our experiment results on RegNet [Radosavovic *et al.*, 2020] are better than those of SOTA PTQ without hardware constraint, thanks mainly to Method 4.2, because the performance of Power-of-Two scale is worse than that of ordinary scale.

5.4 Quick Mode

Many application scenarios are not extreme in terms of accuracy requirement, and they value shorter quantization time. For such scenarios, we introduce a Quick Mode with only 20,000 weight iterations and 1,000 activation iterations. Correspondingly, the parameter α in Equation (19) is set to 0.1 and β is set to 1. This scheme takes only 10 minutes to quantize Resnet18 with Intel i9-10980XE + Nvidia RTX3090. As shown in Table 4, although it takes very short time, it also has a notable accuracy performance.

6 Conclusion

In this paper, we propose RAPQ, a Power-of-Two low-bit post-training quantization framework. At first, we analyze the reasons for the accuracy collapse of Power-of-Two PTQ, i.e., the failure to theoretically trade off rounding error and clipping error and the rough setting of the regression loss while reconstructing. For the first reason we propose a method for finding Power-of-Two scale group of CNN model. For the second reason we propose a method to formulate the regression loss based on the BN information of each unit . The experiments show that our work not only reaches SOTA in the field of Power-of-Two quantization, but also does not fall short of other unconstrained quantization methods. When quantizing MobileNetV2 with W2/A4, our work can achieve an accuracy of 48%, which was not achieved by all previous work on Power-of-Two quantization (including QAT).

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant No. 2018YFE0203801).

References

[Banner *et al.*, 2019] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, 2019.

[Cai *et al.*, 2020] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020.

[Gong *et al.*, 2019] 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, 2019.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hopfield and Tank, 1985] John J Hopfield and David W Tank. “neural” computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

[Hubara *et al.*, 2020] 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.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[Jain *et al.*, 2019] 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. *arXiv preprint arXiv:1903.08066*, 2019.

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

[LeCun *et al.*, 2012] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[Li *et al.*, 2019] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2019.

[Li *et al.*, 2020] 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 *International Conference on Learning Representations*, 2020.

[Li *et al.*, 2021] Yuhang Li, Mingzhu Shen, Jian Ma, Yan Ren, Mingxin Zhao, Qi Zhang, Ruihao Gong, Fengwei Yu, and Junjie Yan. Mqbench: Towards reproducible and deployable model quantization benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[Louizos *et al.*, 2018] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *International Conference on Learning Representations (ICLR)*, 2018.

[Mittal *et al.*, 2011] Arpit Mittal, Andrew Zisserman, and Philip HS Torr. Hand detection using multiple proposals. In *Bmvc*, volume 2, page 5, 2011.

[Miyashita *et al.*, 2016] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[Nagel *et al.*, 2020] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.

[Nagel *et al.*, 2021] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

[Radosavovic *et al.*, 2020] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[Redmon and Farhadi, 2018] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[Wang *et al.*, 2020] Peisong Wang, Qiang Chen, Xiangyu He, and Jian Cheng. Towards accurate post-training network quantization via bit-split and stitching. In *Proc. 37nd Int. Conf. Mach. Learn.(ICML)*, 2020.