Flexible continuous modification for SOTA post training quantization methods.

A Preprint

Sedova Anna
Moscow Institute of Physics and Technology

Zharikov Ilya
Moscow Institute of Physics and Technology

Abstract

Neural network quantization gives the opportunity to inference large models on resource constrained devices. Post-Training Quantization (PTQ) methods have became popular, as they are simple and fast to use. They do not require whole model retraining and use only small calibration set to calculate quantization parameters. However, these methods show significant accuracy decrease on low-bit setting. There are methods that allow to increase the accuracy of model by increasing its computational complexity. In this paper, we propose a continuous modification for these methods and find a reasonable trade-off between computational complexity and performance.

Keywords Deep Learning · Model Compression · Post-Training Quantization

1 Introduction

ToDo: Need to be extended in terms related works further.

Deep Neural Networks (DNN) are applicable to wide range of tasks nowadays. Despite showing the great performance on these tasks, state-of-the-art models require high computational resources. There is a need to run large models on power-limited devices such as smartphones. Many different methods were proposed for model compression Song Han [2015] Geoffrey Hinton [2015] In this paper, we concentrate on quantization method Steven K. Esser [2019].

The quantization is a process of mapping real numbers to the low-precision discrete values. There are two main types of quantization methods: quantization aware training (QAT) and post-training quantization (PTQ) pos. Quantization-aware training shows great results, however, it requires the whole model retraining. Hence, this method is not applicable in some real-life cases, as if training data is not available or computational resources are limited. Unlike QAT, post-training quantization usually uses only an unlabeled calibration set for setting up quantization. Current post-training quantization methods are not such efficient as quantization aware training. However, post-training quantization is a promising technique and therefore should be explored further.

The goal of post-training quantization is to find optimal quantization parameters having only small set of data. The main problem of this technique is that quantization errors of layers can be amplified by deeper layers. Quantization errors can accumulate layer by layer and lead to accuracy degradation. Quantization accuracy degradation is explored in the paper Yury Nahshan [2020] article, which explains why low-bit post-training quantization is a quite challenging task.

Most of post-training quantization methods quantize model parameters and data by minimizing the error between quantized and the original model layers outputs. The recent post-training quantization techniques [Itay Hubara, 2021, Yuhang Li, 2021] made a progress towards low-bit post-training quantization, considering previous layers errors during quantization. However, these methods leave model structure without changes and don't consider improving accuracy of quantized model by complicating its structure.

In this work, we study ways to improve quantized model accuracy by making model more complex. Paper [Xinghcao Liu, 2021] uses the idea of approximating model weights as a sum of low-precision values. Our paper suggests a modification to this method. There are two main goals of this work. Firstly, we propose a method to make post-training quantization lossless. This is relevant to situations when computational device support only low bit data types. Second approach of this paper is to find a trade-off between model complexity and quantization bits, allowing to compress model for resource constrained devices.

2 Problem statement

ToDo: Need to be slightly reformulated during theory week.

In this article, we use uniform quantization. Given value to quantize v , the maximum and minimum quantization value Q_P and Q_N and quantization step size α, quantizer computes integer representation of a data $\bar{v}$:

$$\bar{v} = \begin{cases} -Q_N, \text{if} \frac{v}{\alpha} \leq -Q_N \\ \lfloor \frac{v}{\alpha} \rceil, \text{if} \frac{v}{\alpha} \in [-Q_N, Q_P] \\ Q_P, \text{if} \frac{v}{\alpha} \geq Q_P \end{cases} .$$

To get representation of the same scale, $\bar{v}$ is multiplied by α:

$$\hat{v} = \bar{v} * \alpha.$$

Let's suppose that model has n parameters $W_1, ..., W_n$, then let's denote the model consisting of these parameters as $M(W_1, ..., W_n)$. Also let quantized model parameters be denoted as $Q(W_1, \alpha_1), ..., Q(W_n, \alpha_n)$.

The goal of our work is to quantize model M without significant performance degradation. We will achieve this by making outputs of $M(Q(W_1, \alpha_1), ..., Q(W_n, \alpha_n))$ similar to the outputs of $M(W_1, ..., W_n)$. Let's denote model M complexity as $P(M)$, model quality as $F(M)$.

Then, we want to maximize $F(M(Q(W_1, \alpha_1), ..., Q(W_n, \alpha_n)))$ for given model $M(W_1, ..., W_n)$ and some complexity limit P_0:

$$\arg \max_{\alpha_1,...,\alpha_n} \{F(M(Q(W_1, \alpha_1), ..., Q(W_n, \alpha_n))) \, , P(M(Q(W_1, \alpha_1), ..., Q(W_n, \alpha_n))) \leq P_0\}$$

3 Quantization Modification

In this section we propose modification of quantization, that allows to continiously increase the quantization quality. We will focus on quantization for convolutional layers, but the method can be applicable to weights of linear layers also.

3.1 Quantization parameters

In this paper, we will focus only on symmetric case of quantization, where $Q_N \approx Q_P$. Given the bit-width $n \in \mathbb{N}$ and quantization step size $\alpha \in \mathbb{R}$, $Q_P = \alpha(2^{n-1} - 1), Q_N = \alpha 2^{n-1}$. Thus formula for quantized value v can be reformulated as

$$Q_{\alpha,n}(v) = \begin{cases} -\alpha 2^{n-1}, \text{ if } v \leq -\alpha 2^{n-1} \\ \alpha \lfloor \frac{v}{\alpha} \rceil, \text{ if } v \in [-\alpha 2^{n-1}, \alpha(2^{n-1} - 1)] \\ \alpha(2^{n-1} - 1), \text{ if } v \geq \alpha(2^{n-1} - 1) \end{cases} .$$

3.2 The challenge of quantization

Quantization discussed in the previous section have two parameteres: bit-width n and quantization step size. The more bit-width, the more the range of values that quantized data $\hat{v}$ takes, and, consequently, the more the quality of quantization, but also the more computational complexity. As $n \in \mathbb{N}$, it can be increased only in a discrete way, changing computational complexity in a discrete way also. That is why current quantization methods focus on choosing quantization parameters with fixed bit-width.

The weights of model typically have distribution, close to normal 1. This means that the majority of weights is concentrated in zero-centered interval $[-u, u]$, and there are some outlier weights. Let's denote the biggest weight by module as w_{max}.

The bigger parameter α is, the bigger quantization range and the less precision of quantized values. If α is chosen in such way, that $[-\alpha 2^{n-1}, \alpha(2^{n-1} - 1)]$ interval covers all values, the α should be more, than $\frac{|w_{max}|}{2^{n-1}}$. In this case step size is big and precision of weights $\in [-u, u]$ is going to suffer. On the contrary, if α is chosen to cover only weights

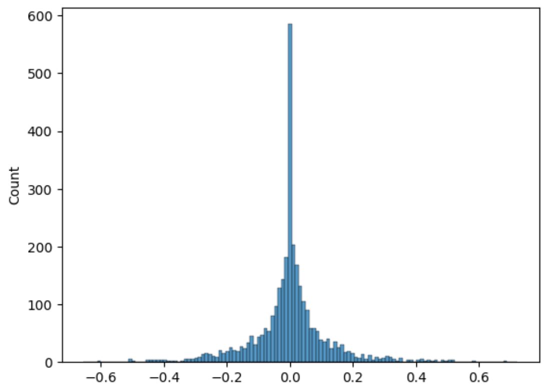

Figure 1: Weights of convolutional layer from ResNet18 distribution

$\in [-u, u]$, w_{max} and other weights than do not belong to the interval $[-u, u]$ are going to be clipped. Changing α does not change the quantization quality monotonously, and the trade-off between precision and quantization range should be found. That is why α selection is a quite challenging task.

In our paper we propose a new method that allows to continuously change the bit-width of quantization while changing α.

3.3 Proposed method of quantization

Let's suppose that we have weight of convolutional layers $W_l \in \mathbb{R}^{C_{out} \times K_h \times K_w}$ for $l = 1, ..., C_{in}$, where C_{in} and C_{out} - input and output channels number. For each l we will quantize W_l with different parameters.

The main idea of our method is to additionally quantize the weights that were clipped. Let's denote the loss of clipping from quantization with parameter α, equal to

$$\begin{cases} w - (-Q_N) = w + \alpha 2^{n-1}, \text{ if } w < -\alpha 2^{n-1} \\ 0, \text{ if } w \in [-\alpha 2^{n-1}, \alpha(2^{n-1} - 1)] \\ w - Q_P = w - \alpha(2^{n-1} - 1), \text{ if } w > \alpha(2^{n-1} - 1) \end{cases}$$

, as $C_{\alpha,n}(W)$.

For each input channel l, let's denote our quantization method of W_l as

$$\widetilde{Q_{\alpha,n}}(W_l) = Q_{\alpha,n}(W_l) + Q_{\alpha,n}(C_{\alpha,n}(W)_l).$$

This quantization method allows to increase the precision of quantization without increasing the number of clipping values. In fact, changing α continiously changes the bit-width from n to $n+1$ bits. The exact dependency between α and bit-width is calculated in the next section.

Let the input data of the layer be $X \in \mathbb{R}^{N \times C_{in} \times H \times W}$. Then quantized output data is calculated as

$$X_{N_{out}} = Q_{\beta,n}(X[N_i, l])\widetilde{Q_{\alpha,n}}(W_l[N_{out}]) = Q_{\beta,n}(X[N_i, l])Q_{\alpha,n}(W_l) + Q_{\beta,n}(X[N_i, l])Q_{\alpha,n}(C_{\alpha,n}(W)_l).$$

Notice that $C_{\alpha,n}(W)$ contains some zeros. The number of zeros depends on α parameter: the more α is, the more often zeros occur in result, the more computationally easier the operation of multiplication $Q_{\beta,n}(X[N_i, l])$ with $Q_{\alpha,n}(C_{\alpha,n}(W)_l)$.

Varying α allows to continiously change the complexity and the bit-width of quantized weights. This allows to find trade-off between quantization complexity and its precision. Proposed method can be integrated into current post-training quantization methods [Itay Hubara, 2021, Xiuying Wei, 2022].

3.4 The bit-width of proposed quantization method

Lets denote the weight as w_p, and the lowest as w_n. There are three main cases of where w_p and w_n can be located. Based on them, different cases of bit-width of positive and negative numbers occur.

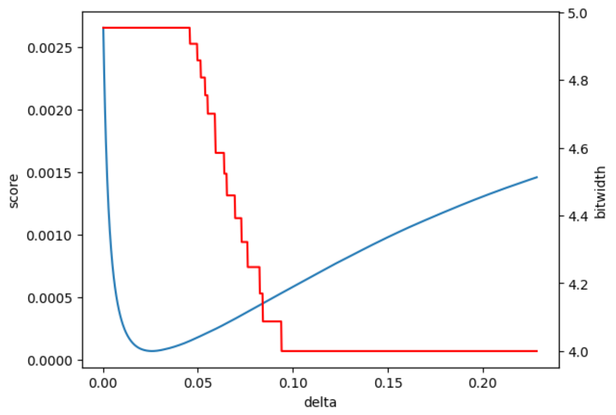

Figure 2: Dependepnce of quantizations bit-width and score from α parameter for one of ResNet18 convolutional layers.

The first case is if $w_p \in [-\alpha 2^{n-1}, \alpha(2^{n-1}-1)]$. Then all the positive weights $\in [-\alpha 2^{n-1}, \alpha(2^{n-1}-1)]$. In this case, $C_{\alpha,n}(w) = 0$ for all $w \geq 0$ and positive weights shouldn't be additionally quantized. As a result, positive weights are only mapped into 2^{n-1}-sized set of values. Same with w_n

The second case is if $w_p \geq 2 * \alpha(2^{n-1}-1)$. In this case some positive weights are additionally quantized. The quantization range becomes wider, and its upper bound is $\alpha 2^n$, as the maximum possible value of $Q_{\alpha,n}(w)$ equals to $\alpha(2^{n-1}-1)$ and the maximum possible value of $Q_{\alpha,n}(C_{\alpha,n}(w))$ also equals to $\alpha 2^{n-1} - 1$ (this value reaches at $w = w_p$, as $w_p - \alpha 2^n \geq \alpha 2^n$). As the quantization step size equals to α, the positive weights are mapped into $\frac{\alpha(2^n-2)}{\alpha} + 1 = 2^n - 1$ values in this case. Same, if $w_p \leq -2 * \alpha 2^n$, negative weights are mapped into 2^n values.

The third case is the case when $\alpha(2^{n-1}-1) < w_p < \alpha(2^n - 2)$. In this case some positive weights are additionally quantized, as in the previous case. The difference is that the maximum possible value after quantization can be less than $\alpha 2^n - 2$. The maximum possible value can be counted as $(\alpha 2^{n-1} - 1) + \alpha\lfloor\frac{w_p - \alpha(2^{n-1}-1)}{\alpha}\rceil + 1 = \alpha(2^{n-1}-1) + \alpha\lfloor\frac{w_p}{\alpha}\rceil - \alpha(2^{n-1}-1) = \alpha\lfloor\frac{w_p}{\alpha}\rceil + 1$, as the the maximum possible value of $Q_{\alpha,n}(C_{\alpha,n}(w))$ equals to $\alpha\lfloor\frac{w_p - \alpha(2^{n-1}-1)}{\alpha}\rceil$. In this case, the positive weights are mapped into $\lfloor\frac{w_p}{\alpha}\rceil + 1$ values. Same, if $-\alpha 2^{n-1} > w_n > -\alpha 2^n$, negative weights are mapped into $\lfloor\frac{-w_n}{\alpha}\rceil$ values.

Overall, positive weights are mapped into $clip(\lfloor\frac{w_p}{\alpha}\rceil + 1, 2^{n-1} - 1, 2^n - 2)$ values, negative are mapped into $clip(\lfloor\frac{-w_n}{\alpha}\rceil, 2^{n-1}, 2^n)$ values. Consequently, all values are mapped into $clip(\lfloor\frac{w_p}{\alpha}\rceil + 1, 2^{n-1}, 2^n - 1) + clip(\lfloor\frac{-w_n}{\alpha}\rceil, 2^{n-1}, 2^n)$ values, so the quantization's bit-width equals to

$$\log_2(clip(\lfloor\frac{w_p}{\alpha}\rceil + 1, 2^{n-1}, 2^n - 1) + clip(\lfloor\frac{-w_n}{\alpha}\rceil, 2^{n-1}, 2^n)) \quad (1)$$

and varies from n to $\approx n + 1$.

4 Experiments

In this section, we conduct two experiments to conduct the effectiveness of proposed method. First of all, we analyze the quantization error for CNN layers weights. Secondly, we integrate proposed method into existing quantization methods.

Implementation details As a base of our code we use QDrop Xiuying Wei [2022] implementation. Original QDrop code is available at https://github.com/wimh966/QDrop. To achieve quantization effect, equivalent to proposed one, we change the number of quantizer levels.

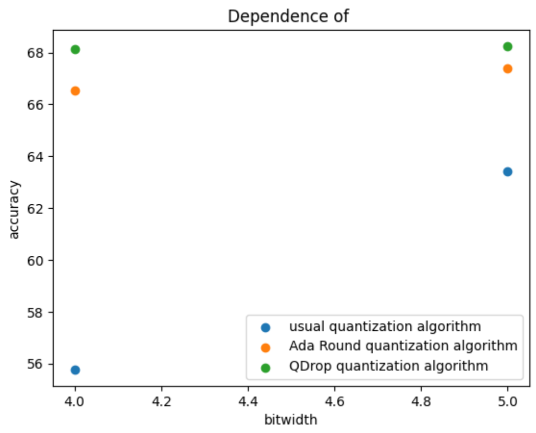

Figure 3: Accuracy of model depending on bit-width of CNN layers

4.1 Loss of quantized weights

We study the effect of additional quantization on model weights. As a score for weight quantization quality we use euclidian distance between usual and quantized weights. The plot 2 shows the dependency between quantization scale factor α and score of ResNet18 weight quantization. On the same plot dependency between α and quantization bitwidth is shown. The quantization bitwidth is calculated using the formula from 3.4

4.2 Integration to existing methods

We study the effect of proposed method by integrating it into three different quantization methods: classical quantization approach pos, AdaRound Markus Nagel and Blankevoort [2020] and QDrop Xiuying Wei [2022].

We quantize ResNet18 and Resnet50 trained on ImageNet dataset.

To study the effectiveness of proposed method, we use it in three different quantization methods: classic quantization, AdaRound and QDrop. The effect is shown on 3 and on 4.2

Accuracy of quantized model on ImageNet			
Model name	Method name	Without additional quantization	With additional quantization
Resnet18	Classical Approach		
Resnet18	AdaRound		
Resnet18	QDrop		

References

John Tran William Dally Song Han, Jeff Pool. Learning both weights and connections for efficient neural network. 2015.

Jeff Dean Geoffrey Hinton, Oriol Vinyals. Distilling the knowledge in a neural network. 2015.

Deepika Bablani Rathinakumar Appuswamy Dharmendra S. Modha Steven K. Esser , Jeffrey L. McKinstry. Learned step size quantization. 2019.

Chaim Baskin Evgenii Zheltonozhskii Ron Banner Alex M. Bronstein Avi Mendelson Yury Nahshan, Brian Chmiel. Loss aware post-training quantization. 2020.

Yair Hanani Ron Banner Daniel Soudry Itay Hubara, Yury Nahshan. Accurate post training quantization with small calibration sets. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021.

Xu Tan Yang Yang Peng Hu Qi Zhang Fengwei Yu Wei Wang Shi Gu Yuhang Li, Ruihao Gong. Brecq: Pushing the limit of post-training quantization by block reconstruction. University of Electronic Science and Technology of China, SenseTime Research, 2021.

Dengyong Zhou Qiang Liu Xinghcao Liu, Mao Ye1. Post-training quantization with multiple points: Mixed precision without mixed precision. 2021.

Yuhang Li Xianglong Liu Fengwei Yu Xiuying Wei, Ruihao Gong. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. 2022.

Mart Van Baalen Christos Louizos Markus Nagel, Rana Ali Amjad and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. 2020.