https://doi.org/10.26599/AIR.2024.9150036 CAAI Artificial Intelligence Research

Minimize Quantization Output Error with Bias Compensation

Cheng Gong[1], Haoshuai Zheng[2], Mengting Hu[1], Zheng Lin[3], Deng-Ping Fan[2, 4], Yuzhi Zhang[1, 5], and Tao Li[2, 5] ✉

ABSTRACT

Quantization is a promising method that reduces memory usage and computational intensity of Deep Neural Networks (DNNs), but it often leads to significant output error that hinder model deployment. In this paper, we propose Bias Compensation (BC) to minimize the output error, thus realizing ultra-low-precision quantization without model fine-tuning. Instead of optimizing the non-convex quantization process as in most previous methods, the proposed BC bypasses the step to directly minimize the quantizing output error by identifying a bias vector for compensation. We have established that the minimization of output error through BC is a convex problem and provides an efficient strategy to procure optimal solutions associated with minimal output error, without the need for training or fine-tuning. We conduct extensive experiments on Vision Transformer Models (ViTs) and Large Language Models (LLMs), and the results show that our method notably reduces quantization output error, thereby permitting ultra-low-precision post-training quantization and enhancing the task performance of models. Especially, BC improves the accuracy of ViT-B* with 4-bit PTQ4ViT by 36.89% on the ImageNet-1K task, and decreases the perplexity of OPT-350M with 3-bit GPTQ by 5.97 on WikiText-2. Our codes are publicly available at https://github.com/GongCheng1919/bias-compensation.

KEYWORDS

quantization; minimizing output error; bias compensation; convex optimization; large language model quantization

Quantization is one of the most promising directions as it directly reduces the memory footprint and computing intensity of Deep Neural Networks (DNNs). It replaces the high-precision floating point weights with low-precision ones, thus saving the memory footprints and computing loads in model inference. The number space of low-precision weights is much smaller than that of float ones, which inevitably results in numerical errors and affects the output of quantized layers.

Tremendous achievements have been achieved in minimizing the output errors of quantized layers, and they generally fall into two categories: Quantization Aware Training (QAT) and Post-Training Quantization (PTQ). QAT fine-tunes the quantized DNNs on the whole dataset or trains them from scratch to align the outputs of float models and quantized ones. It achieves ultra-low-precision (2–3 bits) quantization while maintaining the task performance of models[1–3], but the full-parameter training on the complete dataset is becoming impractical, especially for large pre-trained models. PTQ exhibits great compression efficiency compared to QAT since it is usually applied to quantize the model without retraining and requires just a small amount of calibration data for adjusting the quantizer and model parameters[4–6]. The development of PTQ can be roughly divided into three stages, as shown in Fig. 1: local quantizer optimization, layer-wise quantizer optimization, and layer-wise parameter optimization.

Local quantizer optimization reduces the output error by optimizing the local quantization loss offline and finding the local optimal quantizer for each quantized layer in advance[7–9]. Layer-wise quantizer optimization adopts a few calibration data to reconstruct the outputs of each layer in models to find a relatively optimal quantizer that can align the outputs of quantized models and float models[4, 10, 11]. Layer-wise parameter optimization is similar to layer-wise quantizer optimization, but it updates the quantizer parameters and layer parameters simultaneously to reconstruct the outputs, thus achieving lower precision quantization[5, 12, 13]. However, these optimizations are usually non-convex, which makes it difficult to recover the task performance of DNNs from ultra-low precision quantization since large output error.

In this paper, we minimize the output error caused by quantization from a new perspective of error compensation. We call it Bias Compensation (BC), which adds a bias vector to the outputs of each quantized layer in models and finds the optimal bias vector for minimizing output error. We demonstrate that identifying the optimal bias vector for each quantized layer is a convex problem, and solving the optimal solutions for minimal output error is attainable (See Section 3). It can be easily combined with the existing quantizers and enables ultra-low-precision quantization. Since the bias vector is added to the outputs after expensive operations, i.e., matrix multiplications or convolutions, it does not affect the computing efficiency of these operations and introduces almost no additional computation loads. We conduct extensive experiments to verify the validation of BC, and the results show that BC can significantly reduce the output error and improve the task performance of quantized models.

Our contributions are summarized as follows:

• We propose BC for minimizing the quantization-induced output error from a novel perspective. Instead of working on optimizing the non-convex quantization process as in previous

1 College of Software, Nankai University, Tianjin 300350, China
2 College of Computer Science, Nankai University, Tianjin 300350, China
3 Haihe Lab of ITAI, Tianjin 300450, China
4 Nankai International Advanced Research Institute (Shenzhen Futian), Nankai University, Shenzhen 518045, China
5 BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
Address correspondence to Tao Li, litao@nankai.edu.cn

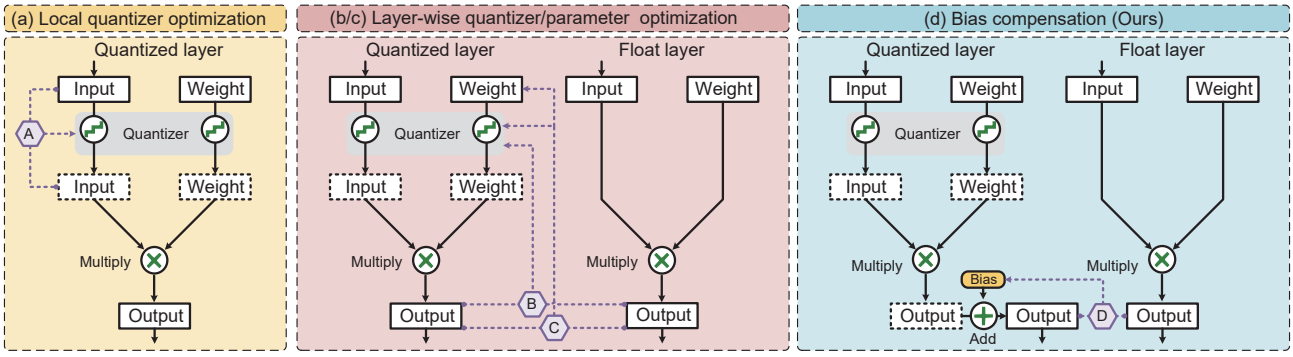

Fig. 1 Comparison between the previous PTQ methods and our proposed bias compensation. The hexagons represent different layer-wise optimization methods. The lines without arrows are the inputs of the optimization algorithms, and the arrows out of the hexagons indicate the output of algorithms. Specifically, (a) indicates local quantizer optimization methods. (b) and (c) are layer-wise quantizer and parameter optimization methods, respectively. Previous methods optimize the quantizer parameter or layer-wise weights to minimize the quantization loss or output error, which is non-convex and difficult to solve. Our method shown in (d) directly minimizes the output error by solving the best bias vector, which is convex and guarantees minimal output error.

methods, BC directly optimizes the output error by adding an optimizable bias vector to the outputs of quantized layers.

• We prove that the optimization of BC is convex, and the optimal bias vector can be obtained without fine-tuning. Additionally, we theoretically demonstrate that BC can always guarantee lower output error.

• Experiments on Vision Transformer Models (ViTs) and Large Language Models (LLMs) demonstrate that BC can significantly improve the task performance of quantized models. In particular, BC improves the accuracy of ViT-B* with 4-bit PTQ4ViT by 36.89% on the ImageNet-1K task, and decreases the perplexity of OPT-350M with 3-bit GPTQ by 5.97 on WikiText-2.

1 Related Work

This section presents a brief overview of the achievements in minimizing output error of DNN quantization.

1.1 Quantization aware training

QAT fine-tunes the model parameters on the whole dataset to recover model accuracy from quantization. LLM-QAT[3] proposes data-free distillation for training quantized models. QLoRA[14] quantizes weights into 4-bit NormalFloat format and employs Low-Rank Adaptation (LoRA)[15] for fine-tuning. Block-wise quantization[16] quantizes the states of optimizers for less memory usage in model training. PEQA[17] quantizes model weights into 4 bits and fine-tunes only the quantization scales of quantized LLMs. Despite tremendous achievements in QAT, fine-tuning the full parameters or even part parameters[14, 15] on the complete dataset is a heavy workload, and PTQ is studied to address this issue.

1.2 Post-training quantization

PTQ focuses on quantizing the pre-trained models with no data[7] or only a small amount of calibration data[5]. According to the type of optimization, the development of PTQ can be roughly divided into three stages as follows:

(1) Local quantizer optimization is a greedy strategy that simplifies the minimization of output error to the minimization of quantization loss in Fig. 1. Reference [7] employs the analytical clipping range and per-channel mixed-precision quantizer for low quantization loss. OCS[8] reduces the quantization loss by splitting channels into more channels. Reference [9] employs the fine-grainy kernel-wise and multi-tensor quantizer to reduce the quantization loss. LLM.int8()[18] uses a mixed-precision quantizer to isolate the outliers into 16 bits and quantize other values into 8 bits. SmoothQuant[19] smooths the activation before quantization to eliminate the outliers in different input channels, thus reducing quantization loss.

(2) Layer-wise quantizer optimization adopts a calibration dataset to align the outputs of the quantized model and float one, as shown in Fig. 1. AdaRound[4] learns a differentiable function that takes values between 0 and 1 to realize the adaptive rounding by minimizing layer-wise output error. BitSplit[11] constructs quantized values bit-by-bit using a squared output error objective on the residual error. BRECQ[10] reconstructs the outputs of quantized layers and residual blocks to learn the best step size and rounding variable of a quantizer. EasyQuant[20] jointly optimizes both scales of activation and weights for each layer, targeting the loss of the cosine similarity between original and quantized convolutional outputs. PTQ4ViT[6] proposes the twin uniform quantization for activation with asymmetric distributions in ViTs and introduces a Hessian-guided metric to determine the scaling factors of quantizers layer-by-layer. APQ-ViT[21] works on maintaining the power-law character of the softmax activations in ViTs. RepQ-ViT[22] studies a scale-reparametrized quantizer to minimize output error while maintaining efficient inference. Outlier Suppression+ (OS+)[23] measures the output change after quantization to pursue effective quantizing factors. AWQ[24] finds the proper scaling factors to preserve the outliers in activation, thus minimizing output error.

(3) Layer-wise parameter optimization also utilizes a calibration dataset to minimize output error directly. It updates both layer parameters and quantizer parameters to reduce output error further, as shown in Fig. 1. AdaQuant[12] proposes layer-by-layer optimization to learn the optimal quantization step size and update layer weights to minimize the layer output error. ZeroQuant[25] uses layer-by-layer knowledge distillation to fine-tune the layer parameters. FlexRound[26] learns a different scaling factor for each pre-trained weight to reconstruct each layer or block output. OBC[13] expands the Optimizing Brain Surgery (OBS)[27, 28] algorithm to general quantization scenes. It iteratively quantizes a portion of the weights into discrete ones and updates the remaining float weights to eliminate output error during the quantization process. OBC is time-consuming and infeasible for large models. The following GPTQ[5] proposes a fast and robust implementation of OBC for large model quantization.

These methods above have achieved great success in aligning the outputs. However, minimizing quantization output error remains an open challenge in PTQ for recovering task performance from ultra-low-precision quantization[5, 25].

1.3 Bias-related studies

The idea of optimizing quantization with bias has a long history. HAWQ[29] applies bias with different variances to weights to simulate quantization and estimate the sensitivities of layers. Reference [9] adds a constrained bias tensor to the float weight tensor to implement quantization and measure the output error caused by the bias. LSQ+[30] solves the asymmetric distribution problem of activation quantization in QAT by adding an offset to shift the distribution before quantized module computation. AutoRound[4], AutoQuant[12], and BRECQ[10] add a continuous bias variable to weights for adaptive weight optimization while minimizing the output error. Recent methods[7, 31] find that applying bias correction to align the means of weights and activation before and after quantization can reduce accuracy degradation. NoisyQuant[32] adds a noise bias on activation before quantization to flatten the activation distribution for less quantization loss. The Pseudo Quantization Training (PQT) methods[33, 34] add learnable noise parameters to floating-point values to address the non-differentiable problems in QAT. It can only be deduced as adding an offline bias to the output in simple modules such as Linear, but cannot handle Attention output.

Although studies based on bias are prevalent in quantization, the proposed BC in this paper is a completely new method that is entirely different from the above methods in terms of 4 aspects: (1) BC aligns the outputs of the float and quantized layers and uses a bias vector to compensate for the quantized output. This is the first method in PTQ and QAT that applies a bias to compensate for output errors after quantized module computation directly. (2) Optimization in PTQ is challenging since PTQ cannot fine-tune quantized models on the entire dataset. Notably, we prove that the optimization of BC is convex and can guarantee the lowest output error for PTQ, making it a significant innovation of our paper. (3) BC is applied after quantized module computation directly and is decoupled with the quantizer. It can be easily applied to any quantizer and module, reducing output error. (4) Quantizers with offsets or biases cache a large bias tensor with a module output size, greatly increasing memory occupation. BC only needs to cache a bias vector to reduce output error, which occupies little additional memory and computation.

1.4 Motivations

We propose BC mainly driven by two motivations: (1) PTQ methods face challenges in achieving low-precision quantization due to significant output error. (2) Applying bias to output has the potential to reduce output error further, but to the best of our knowledge, relevant work has yet to explore it. Therefore, we propose BC to address these issues. We prove that the minimization of output error through BC is convex, and its optimal solution can be easily obtained.

2 Methodology

In this section, we first introduce the quantization and output error, which is the essential issue degrading model task performance. Then we propose bias compensation from a novel perspective to reduce the output error.

2.1 Quantization and output error

Let a neural network has L layers, and the i-th layer with float weight $W_i \in \mathbb{R}^{m\times n}$ processes the float input $X_i \in \mathbb{R}^{b\times m}$, where b is the min-batch size, m is the feature size of each row in X, and n is the output feature size of i-th layer. The typical computation of i-th layer is as follows:

$$X_{i+1} = X_i W_i \tag{1}$$

The matrix multiplication of $X_i W_i$ is the typical computational bottleneck in DNNs, and the quantization is proposed to reduce its computational load.

Let a quantizer $Q(\cdot)$ map any float value in weight and input matrices to a quantized one from $\mathbb{Q}$, and the float matrix multiplication can be replaced with following implementation:

$$\begin{aligned} &X^q_{i+1} = Q(X_i)\cdot Q(W_i) \\ &\text{s.t.}\quad Q(X_i)\in \mathbb{Q}^{b\times m}, Q(W_i)\in \mathbb{Q}^{m\times n} \end{aligned} \tag{2}$$

Given that the space of $\mathbb{Q}$ is significantly smaller than that of $\mathbb{R}$, it becomes feasible to represent elements in $\mathbb{Q}$ using fewer bits. Consequently, operations on these elements can be executed using low-bit integer operations, as shown in Fig. 2, leading to reduced computational load.

However, the smaller space implies inevitable numerical errors, and the error matrix can be defined as follows:

$$\begin{cases} \mathcal{N}_{X_i} &= X_i - Q(X_i), \\ \mathcal{N}_{W_i} &= W_i - Q(W_i) \end{cases} \tag{3}$$

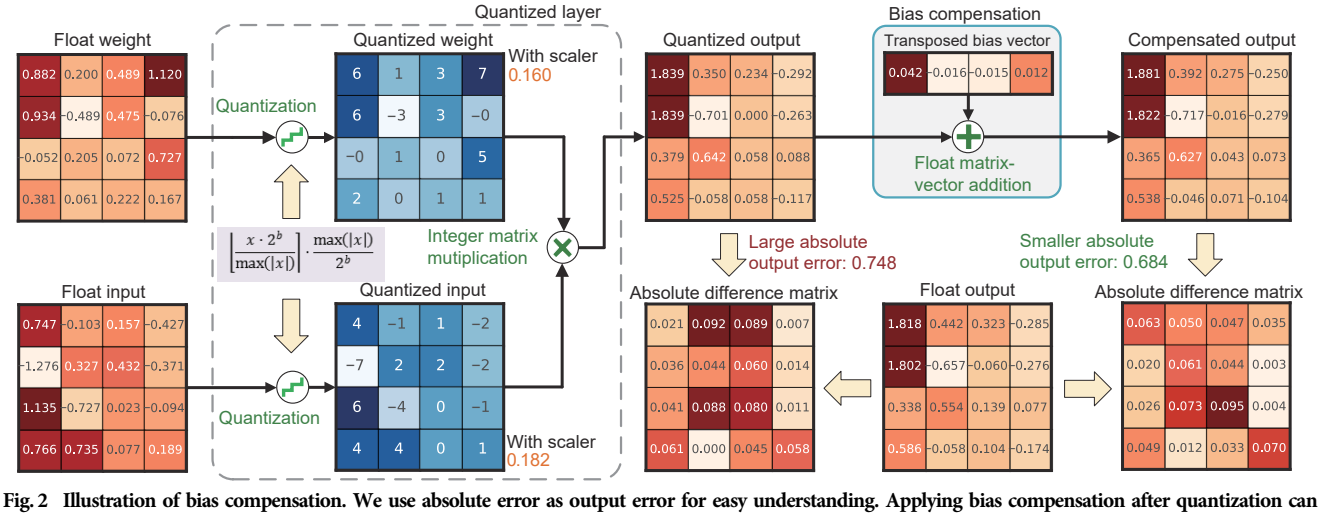

Fig. 2 Illustration of bias compensation. We use absolute error as output error for easy understanding. Applying bias compensation after quantization can significantly reduce the output error without increasing additional computational complexity.

where $\mathcal{N}_{X_i}$ and $\mathcal{N}_{W_i}$ are usually called noise matrixes since the numerical error of quantization can be regarded as applying a noise for each element in X_i and W_i[4, 12, 29]. In many previous studies[7, 8, 9], the L2 norms of $\mathcal{N}_{X_i}$ and $\mathcal{N}_{W_i}$, i.e., $||\mathcal{N}_{X_i}||_2^2$ and $||\mathcal{N}_{W_i}||_2^2$, are optimized for improving the task performance of quantized models.

Recent studies show that aligning the outputs of quantized layers and their corresponding float layers is the pivot factor for high model task performance[5, 13, 25]. The difference between the outputs of X_{i+1} and X_{i+1}^q is defined as follows:

$$\mathcal{N}_{i+1} = X_{i+1} - X_{i+1}^q \tag{4}$$

where $\mathcal{N}_{i+1} \in \mathbb{R}^{b\times n}$ is a difference matrix, and each element in it shows the error caused by the quantization $Q(X_i)$ and $Q(W_i)$ as shown in Fig. 2. Our target is to minimize the L2 norm of the difference matrix $\mathcal{N}_{i+1}$ as follows:

$$l_{i+1} = ||\mathcal{N}_{i+1}||_2^2 = ||\mathcal{N}_{X_i}\mathcal{N}_{W_i} - \mathcal{N}_{X_i}W_i - X_i\mathcal{N}_{W_i}||_2^2 \tag{5}$$

l_{i+1} is the output error of i-th layer, which is only decided by noise matrixes $\mathcal{N}_{X_i}$ and $\mathcal{N}_{W_i}$. To our best knowledge, almost all previous studies focus on finding the relatively optimal noise matrixes $\mathcal{N}_{X_i}$ and $\mathcal{N}_{W_i}$ that meet quantization constraints while minimizing the output error l_{i+1}:

$$\begin{aligned} &\arg\min_{\mathcal{N}_{X_i},\mathcal{N}_{W_i}} ||\mathcal{N}_{i+1}||_2^2 \\ &\text{s.t.}\quad \forall s \in (X_i - \mathcal{N}_{X_i}) \cup (W_i - \mathcal{N}_{W_i}), s \in \mathbb{Q} \end{aligned} \tag{6}$$

However, this optimization is non-convex since the quantization is discrete. It is difficult to solve the optimal noise matrixes for minimal output error.

2.2 Bias compensation

Instead of optimizing the noise matrixes $\mathcal{N}_{X_i}$ and $\mathcal{N}_{W_i}$, we innovatively introduce the bias compensation to minimize output error. Our idea is to find an extra bias vector to compensate for the quantized output matrix X_{i+1}^q as follows:

$$\begin{aligned} &\hat{X}_{i+1}^q(j) = X_{i+1}^q(j) + \mathcal{B}_{i+1}, \\ &\text{s.t. } j = 1, 2, \ldots, b \end{aligned} \tag{7}$$

where $\mathcal{B}_{i+1} \in \mathbb{R}^n$ is a bias vector, and $X_{i+1}^q(j)$ is the j-th row of matrix X_{i+1}^q. Attaching the bias vector $\mathcal{B}_{i+1} \in \mathbb{R}^n$ to X_{i+1}^q follows the quantized matrix multiplication as shown in Fig. 2. It does not alter the quantization and computation procedures of quantized layers. Besides, the addition of bias vector and output matrix is an efficient operation that does not substantially amplify computational load while offering the potential for reduced output error. With bias compensation for the quantized output, the output error can be reformulated:

$$l_{i+1} = ||X_{i+1} - \hat{X}_{i+1}^q||_2^2 = \sum_{j=1}^{b} ||\mathcal{N}_{i+1}(j) - \mathcal{B}_{i+1}||_2^2 \tag{8}$$

where $\mathcal{N}_{i+1}(j)$ is the j-th row of matrix $\mathcal{N}_{i+1}$. Differing from the previous optimization target in Formula (6), our target is to find an optimal bias vector $\mathcal{B}_{i+1}$ for the quantized output of i-th layer, and minimizes the output error l_{i+1}:

$$\arg\min_{\mathcal{B}_{i+1}} \sum_{j=1}^{b} ||\mathcal{N}_{i+1}(j) - \mathcal{B}_{i+1}||_2^2 \tag{9}$$

Notably, any value of $\mathcal{B}_{i+1}$ will not affect $\mathcal{N}_{i+1}$, thus the optimization in Formula (9) is irrelated to the optimization in Formula (6). It implies that bias compensation can be easily combined with various quantizers for lower output error.

3 Optimization

In this section, we present a proof demonstrating that the optimal bias vector of BC can be solved and it always guarantees a lower output error. To do this, we first prove that the minimization of output error with respect to the bias vector is convex, and then solve the optimal bias vector based on a given quantizer and calibration dataset. Finally, we present the inference of a Transformer layer with BC.

3.1 Convex optimization proof

Let H be the Hessian matrix of l_{i+1} with respect to $\mathcal{B}_{i+1}$ in Eq. (8), the elements in H are computed as follows.

$$H(p,q) = \frac{\partial^2 l_{i+1}}{\partial \mathcal{B}_{i+1}(p)\partial \mathcal{B}_{i+1}(q)} = \begin{cases} b > 0, & p = q; \\ 0, & p \neq q \end{cases} \tag{10}$$

where $H(p,q)$ refers to the element in the p-th row and q-th column of H. Clearly, each element in H is greater than or equal to 0, which indicates that H is positive-semidefinite and the optimization in Formula (9) is convex.

3.2 Obtain optimal bias vector

For a convex optimization, we can solve for its optimal analytical solution directly. Let the optimal bias vector be $\mathcal{B}_{i+1}^*$ and the optimal solution is obtained as follows:

$$\mathcal{B}_{i+1}^* = \frac{1}{b}\sum_{j=1}^{b}\mathcal{N}_{i+1}(j) = \frac{1}{b}\sum_{j=1}^{b}(X_{i+1}(j) - X_{i+1}^q(j)) \tag{11}$$

where b refers to the number of samples of a calibration dataset, and X_{i+1} and X_{i+1}^q are certain for a given quantizer and input. The optimal bias vector $\mathcal{B}_{i+1}$ for i-th quantized layer can be obtained with two feedforward inferences (computing X_{i+1} and X_{i+1}^q) without fine-tuning.

Employing the optimal bias vector $\mathcal{B}_{i+1}^*$, the lowest output error can be obtained as follows.

$$\begin{aligned} &l_{i+1} = \sum_{k=1}^{b} ||\mathcal{N}_{i+1}(k) - \mathcal{B}_{i+1}^*||_2^2 = \sum_{k=1}^{b} ||\mathcal{N}_{i+1}(k) - \frac{1}{b}\sum_{j=1}^{b}\mathcal{N}_{i+1}(j)||_2^2 = \\ &||\mathcal{N}_{i+1}||_2^2 - b||\frac{1}{b}\sum_{j=1}^{b}\mathcal{N}_{i+1}(j)||_2^2 \leqslant ||\mathcal{N}_{i+1}||_2^2 \end{aligned} \tag{12}$$

It demonstrates that BC can always guarantee the lower output error with a given quantizer and calibration dataset.

3.3 Inference with bias compensation

In the implementation of model inference using BC, the bias vector is attached after each quantized layer in order to reduce output error. In a typical Transformer architecture, the fully connected layers and Batch Matrix Multiplication (BMM) layers are quantized to minimize computational costs. The BC modules are then embedded after each quantized module, as illustrated in Fig. 3. Compared to the basic quantized model inference process, the quantized layers with BC modules only require an additional matrix-vector addition. For the Linear layer, the bias vector of BC can be fused into the bias term. These attached modules do not impact the computational efficiency of the quantized layers since the bias vectors are added to the outputs after the expensive

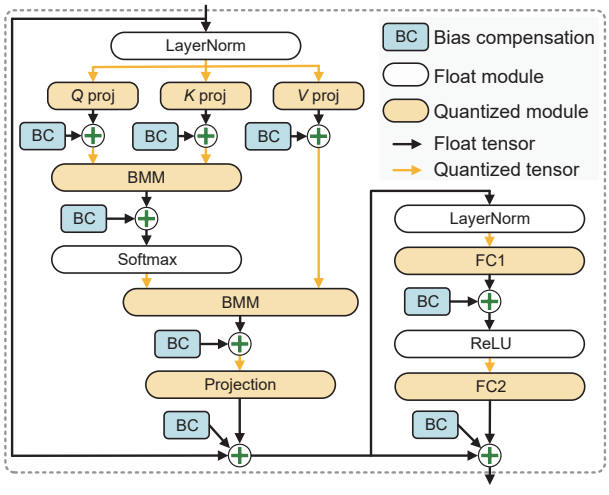

Fig. 3 Bias compensation positions in Transformer.

operations. BC has minimal impact on the computational and spatial complexity of quantized models, while ensuring lower output error with the given quantizer and calibration dataset.

4 Experiment

In this section, we perform experiments on ViTs and LLMs to validate the effectiveness of the proposed method. At each part, we take the state-of-the-art quantizers as baselines and verify the validation of BC.

4.1 Experiments on ViTs

ViTs have demonstrated that pre-training Transformer architectures with a large number of parameters on a massive visual dataset can achieve high accuracy[35]. However, achieving low-precision ViTs remains a challenge. In this experiment, we aim to demonstrate that BC can significantly enhance the accuracy of ViTs under low-bit quantization.

4.1.1 Baselines

We take the PTQ4ViT[6] as the base quantizer for BC and the state-of-the-art quantizers including Base PTQ in Ref. [6], EasyQuant[20], APQ-ViT[21], NoisyQuant[32], and RepQ-ViT[22] as baselines for comparison. The experimental results of the baselines are cited from their original paper, and the results of PTQ4ViT are obtained based on the open implementation[1]

4.1.2 Models & datasets

We evaluate BC on ImageNet-1K[36] dataset with ViT[37], DeiT[38], and Swin[39] models. All the pre-trained full-precision models and the default data augmentation strategies are cited from Timm[40]. We randomly select 32 images from the ImageNet-1K training dataset as calibration images in the model quantization as did in Ref. [6]. The default data augmentation in Ref. [40] for ImageNet-1K is adopted.

4.1.3 Experimental setting

We quantize all the weights and inputs for all the fully-connect layers and the matrix multiplications in self-attention modules in ViTs. Then, we attach BC modules following these quantized layers, as shown in Fig. 3. The layer outputs in ViTs are typically three-dimensional tensors, encompassing batch, sequence, and token dimensions. The size of the bias vector of BC module is identified by the product of the token dimension size and the sequence dimension size. During model inference, the bias vector is added to each batch slice of the output tensor.

4.1.4 Results & analysis

The accuracy results are shown in Table 1. BC consistently outperforms the base quantizer, i.e., PTQ4ViT[6], across all bit quantization and models, and significantly improves the model accuracy. The average accuracy improvements are 0.12%, 0.85%, and 13.06% for W8A8, W6A6, and W4A4 quantizing settings, respectively, across all models. It is attributed to the fact that PTQ4ViT often results in a large output error when using low-precision quantization. BC minimizes the error and aligns the float outputs and quantized outputs, thus improving model accuracy. Remarkably, BC enables low-precision quantization of PTQ4ViT without quantizer parameter or layer weight optimization. For instance, ViT-B* with 4-bit PTQ4ViT provides an accuracy of 31.40%. BC improves it by 36.89%, achieving an accuracy of 68.29%.

BC outperforms state-of-the-art baselines and achieves competitive accuracy results. Compared to Base PTQ[6], EasyQuant[20], and APQ-ViT[21], BC consistently achieves higher accuracy across various bits and models. It can be attributed to the high base accuracy of PTQ4ViT, coupled with BC to reduce layer-wise output errors further. When compared to NoisyQuant[32], BC outperforms it in terms of accuracy across all models except for some cases of Swin models, where the results are slightly lower by a maximum margin of 0.43%. BC demonstrates comparable or even higher accuracy than RepQ-ViT[22] on certain models and bits, showcasing competitive performance. However, it yields lower results in certain cases. This discrepancy primarily arises from the significant contrast in accuracy between the quantizers, PTQ4ViT and RepQ-ViT. The optimization upper bound of BC remains constrained by the base quantizer, i.e., PTQ4ViT, as illustrated in Eq. (12), and the large quantization error of the base quantizer can impact its optimization effectiveness and final achievable accuracy. Despite the limitations, BC greatly improves the task performance of the base quantizer and attains high accuracy. Remarkably, BC achieves this without requiring any updates of the quantizer parameters and layer weights, or the need for fine-tuning. This robustly showcases the effectiveness of BC in ViT quantization.

To explicitly demonstrate the effectiveness of BC in aligning outputs and reducing output error, we present the attention output distribution of the first layer of the ViT-S Transformer with 4-bit quantization on the calibration dataset, as shown in Fig. 4. It can be seen that the output of PTQ4ViT diverges significantly from the float output, resulting in a substantial output error. Remarkably, BC introduces a bias vector to compensate for the deviations and align the PTQ4ViT's output with float output, thereby considerably reducing the output error.

4.2 Experiments on LLMs

LLMs have achieved impressive performance in recent years, but their deployments are difficult due to the massive parameters and operations. Therefore, verifying the effectiveness of quantization on LLMs is crucial for promoting their practical application. In this subsection, we conduct experiments to verify the effectiveness of BC in LLM quantization.

4.2.1 Baselines

We take the naive Rounding-To-Nearest (RTN) (implemented in

1 https://github.com/hahnyuan/PTQ4ViT

Table 1 Top-1 accuracy results of ViTs, Deits, and Swins. The numbers in the brackets following method names are the number of parameters with the unit of million. The default input resolution is 224 × 224 (* means 384 × 384). BC enables PTQ4ViT to achieve competitive accuracy in ultra-low-precision quantization without quantizer parameter or layer weight optimization. (%)

Model	W/A	ViT-S (22.1)	ViT-B (86.6)	ViT-B* (86.9)	Deit-S (22.4)	Deit-B (86.6)	Deit-B* (86.9)	Swin-T (28.3)	Swin-S (49.9)	Swin-B (88.1)	Swin-B* (90.8)
FP32	32/32	81.39	84.54	86.00	79.85	81.80	83.11	81.39	83.23	85.27	86.44
Base PTQ	8/8	80.46	83.89	85.35	77.65	80.94	82.33	80.96	82.75	84.79	86.16
EasyQuant	8/8	80.75	83.89	85.53	78.98	81.19	82.10	80.95	83.00	85.10	86.39
PTQ4ViT	8/8	81.00	84.25	85.82	79.47	81.48	82.97	81.24	83.10	85.14	86.39
APQ-ViT	8/8	81.25	84.26	-	79.78	81.72	-	-	83.16	85.16	86.40
NoisyQuant	8/8	81.15	84.22	85.86	79.51	81.45	82.49	81.25	83.13	85.20	86.44
PTQ4ViT+BC	8/8	81.15	84.33	85.96	79.58	81.76	83.12	81.34	83.18	85.21	86.44
Base PTQ	6/6	70.24	75.66	46.88	72.26	78.78	68.44	78.45	81.74	83.35	85.22
EasyQuant	6/6	75.13	81.42	82.02	75.27	79.47	81.26	79.51	82.45	84.30	85.89
PTQ4ViT	6/6	78.63	81.65	83.34	76.28	80.25	81.55	80.47	82.38	84.01	85.38
APQ-ViT	6/6	79.10	82.21	-	77.76	80.42	-	-	82.67	84.18	85.60
NoisyQuant	6/6	78.65	82.32	83.22	77.43	80.70	81.65	80.51	82.86	84.68	86.03
RepQ-ViT	6/6	80.43	83.62	-	78.90	81.27	-	-	82.79	84.57	-
PTQ4ViT+BC	6/6	79.22	83.00	85.00	78.60	81.29	82.44	80.56	82.46	84.28	85.60
PTQ4ViT	4/4	34.02	35.30	31.40	24.05	60.72	75.93	74.46	76.46	74.49	77.26
APQ-ViT	4/4	47.95	41.41	-	43.55	67.48	-	-	77.15	76.48	80.84
RepQ-ViT	4/4	65.05	68.48	-	69.03	75.61	-	-	79.45	78.32	-
PTQ4ViT+BC	4/4	56.11	64.60	68.29	65.90	74.97	80.15	74.74	76.79	77.78	79.97

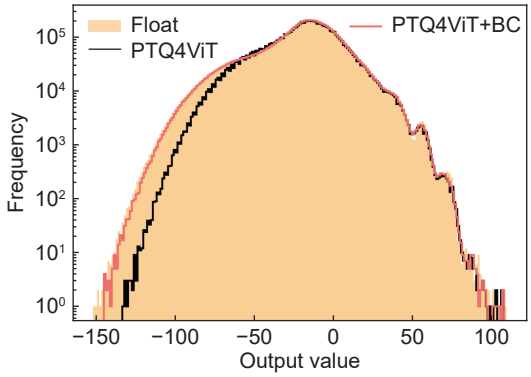

Fig. 4 Attention output distribution of the first layer of ViT-S Transformer using 4-bit quantization. The output of PTQ4ViT significantly deviates from the float output. BC compensates for the output of PTQ4ViT and aligns it with the float one.

GPTQ[5]), GPTQ[5], AWQ[24], and BiLLM[41] as baselines in this experiment. We use the open implementations of GPTQ[2], AWQ[3], BiLLM[4], and pre-trained LLM models in HuggingFace to obtain reproducible results.

4.2.2 Models & datasets

The OPT-125M[42], OPT-350M, and BLOOM-560M[43] are adopted in this experiment and their open implementations in HuggingFace are utilized for reproducibility. Similar to GPTQ, we randomly select 128 random 2048 token segments from the C4 dataset[44] as calibration data to obtain the best bias vectors for compensation. We profile the quantized models on perplexity evaluation on WikiText-2[45], PTB[46], and C4[44] since perplexity can stably reflect the LLM's performance[47].

4.2.3 Experimental setting

Only weight quantization is performed and the activation maintains FP16 as in Refs. [5, 24]. We apply BC for all the quantized modules in LLMs. The size of the bias vector is set to the product of the token dimension size and the sequence dimension size of the module's output.

4.2.4 Results & analysis

Perplexity results are shown in Table 2. The results of LLMs adopting BC consistently outperform that of baselines across all conditions. The average perplexity improvements across models and datasets are 2.61 for 4-bit quantization, 68.55 for 3-bit quantizing setting, and 1033.64 for 2-bit quantization, respectively. These improvements can be attributed to the fact that BC can significantly reduce quantization output error and recover the perplexity of LLMs under low-precision quantization. Besides, BC achieves significantly better perplexity results in the lower quantizing bits. It indicates that BC is beneficial for promoting ultra-low precision quantization in LLMs. Especially, by applying BC, the perplexity of OPT-125M with 3-bit GPTQ has been reduced by 13.59 on the PTB dataset, and that of OPT-350M with 4-bit GPTQ on the WikiText-2 achieves 22.85, which is approaching the result of the float model (22.00). BC significantly improves the results of LLMs in 2-bit and binary quantization, with an order of magnitude improvement in many results.

2 https://github.com/IST-DASLab/gptq
3 https://github.com/mit-han-lab/llm-awq
4 https://github.com/Aaronhuang-778/BiLLM

Table 2 Perplexity results of LLMs. BC substantially reduces the perplexity of LLMs with GPTQ on various bits and datasets. 4g256 indicates 4-bit quantization with a group size of 256, and other symbols are similar. The best results are underlined.

Method	Bit	OPT-125M			OPT-350M			BLOOM-560M		
		WikiText-2	PTB	C4	WikiText-2	PTB	C4	WikiText-2	PTB	C4
FP16	16	27.66	32.55	24.61	22.00	26.08	20.71	22.42	41.26	24.38
RTN	4	37.28	45.11	31.64	25.94	31.12	23.94	25.89	48.57	27.42
GPTQ	4	31.22	36.93	26.94	24.20	28.89	22.60	23.98	44.53	25.60
GPTQ+BC	4	29.90	36.11	26.03	22.85	27.23	21.34	23.77	44.44	25.46
RTN	4g256	31.66	37.37	27.54	24.88	29.61	23.10	24.17	45.17	26.01
GPTQ	4g256	30.16	35.78	26.38	23.67	28.14	22.20	23.39	42.99	25.08
AWQ	4g256	30.38	35.23	26.35	-	-	-	28.45	52.93	34.96
GPTQ+BC	4g256	28.99	33.96	25.40	22.48	26.85	21.14	23.24	42.68	24.97
RTN	4g128	30.48	36.47	26.80	24.51	29.20	22.79	23.89	44.75	25.76
GPTQ	4g128	29.79	35.25	25.95	23.17	27.93	21.94	23.23	42.72	24.93
AWQ	4g128	29.14	34.96	25.90	-	-	-	28.22	55.37	34.93
GPTQ+BC	4g128	28.50	33.77	25.21	22.48	26.73	21.08	23.10	42.68	24.83
RTN	3	1276.92	1209.34	731.60	64.56	81.85	50.14	56.98	117.15	58.96
GPTQ	3	54.68	65.74	38.27	33.75	38.66	28.53	32.45	62.66	32.25
GPTQ+BC	3	45.19	52.15	33.78	27.78	33.79	24.79	31.49	60.36	31.35
RTN	3g256	79.27	98.82	64.14	40.81	48.85	33.65	35.32	68.46	36.20
GPTQ	3g256	41.64	52.96	32.46	30.01	34.86	26.05	26.76	51.06	28.11
AWQ	3g256	41.87	50.92	35.55	-	-	-	33.21	62.04	41.32
GPTQ+BC	3g256	35.65	44.33	29.04	25.79	31.11	23.23	26.35	50.11	27.69
RTN	3g128	51.21	64.98	40.09	36.33	42.53	30.60	31.06	60.79	32.46
GPTQ	3g128	37.78	43.20	30.00	27.62	32.96	24.91	25.52	48.44	27.15
AWQ	3g128	36.52	44.02	32.02	-	-	-	31.77	62.97	39.66
GPTQ+BC	3g128	33.63	39.54	27.79	24.61	29.77	22.68	25.15	47.51	26.84
RTN	2g64	7042.44	5057.60	3869.38	4354.61	3560.32	2346.04	502.39	627.17	326.20
GPTQ	2g64	192.96	200.53	114.07	263.21	205.00	124.54	74.06	182.64	59.23
AWQ	2g64	133.27	147.97	97.18	-	-	-	111.49	316.53	145.12
GPTQ+BC	2g64	108.98	137.12	70.11	80.33	89.48	55.11	69.12	178.73	56.43
BiLLM	Binary	2409.61	2581.31	1632.83	1949.35	2567.82	822.90	-	-	-
BiLLM+BC	Binary	348.64	362.20	199.67	116.31	138.91	100.99	-	-	-

Compared to GPTQ and BiLLM, its average improvement reaches 63.43 and 1782.85, respectively. Compared to AWQ, experimental results indicate that AWQ achieves lower perplexity than GPTQ on OPT-125M over some datasets across all bits. However, GBTQ+BC consistently outperforms AWQ across all the models and quantizing configurations. For example, 2-bit AWQ with a group size of 64 achieves 133.27 point of perplexity on OPT-125M over WikiText-2, and GPTQ+BC reduces it by up to 24.27 point on the same model and dataset with the same quantizing configuration. These results demonstrate the significant potential of BC in suppressing output error and enhancing the task performance of quantized LLMs.

To explicitly demonstrate the reduction of layer-wise output error, we compare the output errors of 12 Transformer layers of the OPT-125M model using 3-bit RTN, GPTQ, and GPTQ+BC, respectively, on the calibration dataset. As shown in Fig. 5, BC greatly decreases the output errors across all layers in comparison to RTN and GPTQ. It is consistent with the statement in Section 3. Besides, we observe that the deeper layers exhibit greater output errors, which is consistent with the previous observation[9]. BC

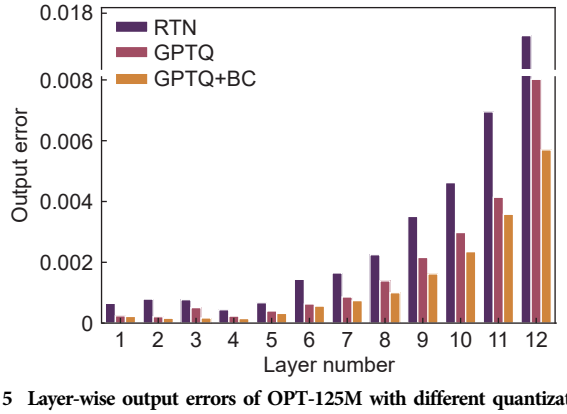

Fig. 5 Layer-wise output errors of OPT-125M with different quantization methods on calibration dataset. BC greatly decreases the output errors across all layers.

effectively compensates for these large output errors, thus improving model performance.

We present the distribution of bias vectors of BC modules for the multi-head attention layers in the OPT-125M model with 3-bit quantization, as shown in Fig. 6. Noticeably, different bias vectors are obtained for different layers for low output errors, and the bias vectors for deeper layers have larger absolute means and variances, reflecting that the deeper layers have larger output errors, as described above.

4.3 Ablation studies

In this subsection, we take ablation studies to verify the impact of the calibration dataset and the compensation position of applying BC on the model accuracy, as well as the impact of BC on model parameters and inference.

4.3.1 Impact of calibration dataset

In Section 4.1, we initially use the same calibration dataset for BC as PTQ4ViT to ensure a fair comparison with the baselines. In this study, we aim to assess the generalization of BC with varying numbers of samples and different random sampling seeds. To do this, we redesign our experiments to separate the calibration dataset of BC from PTQ4ViT. We take ViTs with 4-bit PTQ4ViT (W4/A4) on ImageNet-1K for this experiment.

Specifically, the seeds for random sampling are selected from a set of $\{3, 0, 1993\}$, where 3 is the default seed in PTQ4ViT, 0 is the default setting in PyTorch, and 1993 is a randomly chosen seed for this experiment. The numbers of samples include $\{32, 64, 128, 1024\}$, where $\{32, 64, 128\}$ are the default settings in PTQ4ViT project, and 1024 is used to evaluate the generalization of BC on a large calibration dataset. It is worth noting that we keep a consistent experimental setup (32 samples with random seed 3) for the base method PTQ4ViT in all PTQ4ViT+BC experiments, and only vary the random sampling seeds and dataset sizes for BC calibration.

The results in Table 3 demonstrate that, in most cases, incorporating more calibration datasets can effectively enhance the performance of BC. This observation supports that BC optimized on a limited amount of calibration data may exhibit generality issues, and increasing the calibration dataset can partially mitigate this problem.

Furthermore, we have observed that the selection of random seeds also impacts the results, particularly when the number of calibration samples is small. Increasing the number of samples helps stabilize the results. It can be attributed to the fact that a smaller calibration dataset is more prone to deviating from the true distribution of the entire training dataset, resulting in more significant fluctuations. Despite these challenges, BC consistently outperforms the base method PTQ4ViT across all configurations, demonstrating a significant performance improvement.

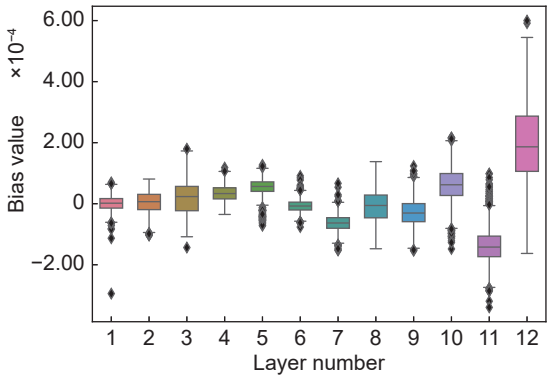

Fig. 6 Distribution of bias values in different layers of OPT-125M using 3-bit GPTQ quantization with BC. Bias vectors for deeper layers have larger absolute means and variances.

4.3.2 Impact of compensation position

This study evaluates the impact of different compensation positions with BC. We still take ViTs with 4-bit PTQ4ViT (W4A4) on ImageNet-1K for this study. The calibration dataset size is 32 and the random sampling seed is 3, as in Section 4.1. The compensation positions consist of specific layers, including fine-grained layers, and coarse-grained layers. The fine-grained layers encompass Linear (L) and BMM (B) layers, while the coarse-grained layers include Self-attention (S) and MLP (M) layers. When optimizing either the fine-grained or coarse-grained layers separately, BC optimization processes can be executed simultaneously. However, when applying BC to both fine-grained and coarse-grained layers together, the optimization of coarse-grained layers must follow that of fine-grained layers, and they cannot be parallelized.

The results are shown in Table 4. In regards to fine-grained layers, we observe that applying BC to Linear or BMM layers

Table 3 Accuracy results of ViT and DeiT models with BC calibrated with different calibration datasets.

Method	Seed	# Samples	Accuracy (%)			
			ViT-S	ViT-B	DeiT-S	DeiT-B
PTQ-ViT	3	32	34.02	35.30	24.05	60.72
PTQ-ViT+BC	3	32	56.11	64.60	65.90	74.97
	0	32	55.78	63.21	65.24	74.52
	1993	32	54.71	63.90	65.09	74.78
	3	64	57.49	65.30	65.73	74.77
	0	64	57.32	63.69	65.76	74.70
	1993	64	56.94	65.51	65.22	74.79
	3	128	57.90	64.94	66.07	74.91
	0	128	57.35	64.39	65.95	74.94
	1993	128	57.76	65.20	65.92	75.04
	3	1024	58.52	64.61	66.13	74.99
	0	1024	57.94	64.54	66.25	75.08
	1993	1024	58.22	65.03	66.22	75.03

Table 4 Accuracy results of ViT and DeiT models with different compensation positions. L is the Linear layer, B is the BMM layer, S is the Self-attention layer, and M is the MLP layer.

Method	Config	Accuracy (%)			
		ViT-S	ViT-B	DeiT-S	DeiT-B
PTQ4ViT	-	34.02	35.30	24.05	60.72
Fine-grained	L	53.17	64.44	65.85	74.32
	B	40.04	39.57	27.30	59.93
	L+B	56.11	64.60	65.90	74.97
Coarse-grained	S	42.94	50.11	37.80	58.82
	M	55.34	63.63	60.14	73.69
	S+M	54.06	60.48	54.78	71.76
Fine-grained + Coarse-grained	L+B+S	55.07	64.67	65.59	74.79
	L+B+M	55.30	59.57	65.68	74.47
	L+B+S+M	54.23	61.59	65.51	74.46

individually can enhance the accuracy of PTQ4ViT for ViTs. The best accuracy of quantized models can be achieved by applying BC to all fine-grained layers. When it comes to coarse-grained layers, applying BC to Self-Attention or MLP layers separately can both improve model accuracy. However, compensating for all coarse-grained layers with BC can reduce model accuracy compared to only compensating for MLP layers. Compensating for all fine-grained and coarse-grained layers with BC can significantly enhance model accuracy compared to PTQ4ViT results, but there is no advantage compared to compensating for only fine-grained layers.

This study reveals that the accuracy of ViTs can be significantly enhanced by applying BC to the Linear or MLP layers. While the BMM and Self-attention layers can also improve model accuracy, the improvement is relatively small.

4.3.3 Impact on model parameters and model inference

The proposed BC requires an additional matrix-vector addition for compensating output, which could potentially impact both the model parameters and inference speed. In this study, we employ the OPT-125M and BLOOM-560M models to evaluate the impact of BC on model parameters and inference, as the LLMs are more sensitive to model parameters than ViTs.

The hidden states of Transformers are represented as three-dimensional tensors, consisting of batch, sequence, and token dimensions. When it comes to the size of the bias tensor of BC, we have the ability to set it as either the token size or the product of the sequence length and token size of the output hidden state. Setting the vector size equal to the token size will not increase the model parameters or computational load significantly, as it can be fused with the bias term of the Linear layer. Increasing the vector size by the times of the sequence length of hidden state, allows BC to further enhance the task performance of LLMs. In specific implementations, we set the bias vector size as the token size (denoted as GPTQ+BC-1) or the product of the sequence length and token size (GPTQ+BC-2) for a Linear output hidden state. We use fake quantization for easy evaluation in PyTorch, which may not improve the inference speed compared to the FP16 model. All results are obtained using a single NVIDIA 3 090 GPU device.

The results are shown in Table 5. Notably, GPTQ-BC-1 does not increase model parameters, while GPTQ-BC-2 doubles the parameters, resulting in the lowest perplexity. Neither approach significantly increases the inference latency. In practice, the choice between GPTQ-BC-1 and GPTQ-BC-2 can be made based on the desired task performance and memory requirements of quantized models.

4.3.4 Results on CNNs

We conduct an experiment to assess the impact of BC on CNNs. The AdaQuant[12] is selected as the base method due to its widespread use as a layer-wise parameter optimization quantizer for CNNs. We evaluated the effectiveness of the quantizer by measuring the top 1 accuracy of quantized ResNet models, including ResNet18, ResNet34, and ResNet50, on ImageNet dataset. The results, presented in Table 6, show that our method significantly improves the accuracy of AdaQuant across all bitwidth configurations and models. Specifically, the average

Table 5 Perplexity results and parameters of LLMs with BC. GPTQ+BC-1 indicates the BC with a vector size equal to the token size of the output tensor, and GPTQ+BC-2 is the BC with a vector size equal to the product of the sequence length and token size of the output tensor. Fake quantization is adopted for the sake of comparison.

Model	Method	Bit	Speed (10^3 token/s)	# Param (10^6)	Perplexity		
					WikiText-2	PTB	C4
OPT-125M	FP16	16	38.05	125.24	27.66	32.55	24.61
	GPTQ	4	35.61	125.24	31.22	36.93	26.94
	GPTQ+BC-1	4	36.93	125.33	30.97	36.10	26.60
	GPTQ+BC-2	4	30.31	313.98	29.90	36.11	26.03
BLOOM-560M	FP16	16	10.02	559.21	22.42	41.26	24.38
	GPTQ	4	11.27	559.21	23.98	44.53	25.60
	GPTQ+BC-1	4	10.80	559.46	23.83	44.22	25.49
	GPTQ+BC-2	4	10.77	1062.53	23.78	44.17	25.45

Table 6 Accuracy results of ResNet models with BC over ImageNet.

Method	W/A	Accuracy (%)		
		ResNet18	ResNet34	ResNet50
FP32	32/32	69.80	73.30	76.10
AdaQuant	8/8	69.70	73.30	76.00
AdaQuant+BC	8/8	69.70	73.30	76.10
AdaQuant	4/4	62.10	69.20	68.60
AdaQuant+BC	4/4	65.50	70.80	70.20

accuracy improvements are 0.03% and 2.20% for W8A8 and W4A4, respectively. These findings demonstrate that BC can be effectively applied to layer-wise parameter quantizers for CNNs, resulting in enhanced model accuracy.

5 Conclusion

In this paper, we introduce a novel method called BC aimed at minimizing the output error induced by quantization in neural networks. We theoretically prove that BC optimization is convex and that the optimal solution for minimal output error can be solved without the need for fine-tuning. Additionally, we theoretically demonstrate that applying BC can always guarantee a lower output error with a given quantizer and calibration dataset. Through extensive experiments conducted on ViTs and LLMs, the effectiveness of BC is highlighted in significantly enhancing the task performance of quantized models, particularly for ultra-low-precision quantizing settings of 4 bits, 3 bits, and even 2 bits.

Acknowledgment

This work was partially supported by the China Postdoctoral Science Foundation (No. 2022M721707), the Fundamental Research Funds for the Central Universities (No. Nankai University, 070-63243150), the National Natural Science Foundation of China (Nos. 62272248 and 62476143), the Natural Science Foundation of Tianjin (Nos. 23JCZDJC01010 and 23JCQNJC00010), the Youth Program of National Science Fund of Tianjin, China (No. 22JCQNJC01340).

Article History

Received: 12 May 2024; Revised: 17 June 2024; Accepted: 21 June 2024

References

[1] C. Gong, Y. Chen, Y. Lu, T. Li, C. Hao, and D. Chen, VecQ: Minimal loss DNN model compression with vectorized weight quantization, *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 696–710, 2021.

[2] C. Gong, Y. Lu, K. Xie, Z. Jin, T. Li, and Y. Wang, Elastic significant bit quantization and acceleration for deep neural networks, *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 3178–3193, 2022.

[3] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra, LLM-QAT: Data-free quantization aware training for large language models, arXiv preprint arXiv: 2305.17888, 2023.

[4] M. Nagel, R. A. Amjad, M. V. Baalen, C. Louizos, and T. Blankevoort, Up or down? Adaptive rounding for post-training quantization, in *Proc. 37th Int. Conf. Machine Learning*, virtual, 2020, pp. 7197–7206.

[5] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, OPTQ: Accurate quantization for generative pre-trained transformers, in *Proc. 11th Int. Conf. Learning Representations* (*ICLR*), Kigali, Rwanda, 2023.

[6] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, PTQ4ViT: Post-training quantization for vision transformers with twin uniform quantization, in *Proc. 17th European Conf. Computer Vision* (*ECCV*), Tel Aviv, Israel, 2022, pp. 191–207.

[7] R. Banner, Y. Nahshan, and D. Soudry, Post training 4-bit quantization of convolutional networks for rapid-deployment, in *Proc. 33rd Conf. Neural Information Processing Systems* (*NeurIPS 2019*), Vancouver, Canada, 2019, pp. 7950–7958.

[8] R. Zhao, Y. Hu, J. Dotzel, C. De Sa, and Z. Zhang, Improving neural network quantization without retraining using outlier channel splitting, in *Proc. 36th Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 7543–7552.

[9] Y. Choukroun, E. Kravchik, and P. Kisilev, Low-bit quantization of neural networks for efficient inference, in *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision Workshop* (*ICCVW*), Seoul, Republic of Korea, 2019, pp. 3009–3018.

[10] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu, BRECQ: Pushing the limit of post-training quantization by block reconstruction, in *Proc. 8th Int. Conf. Learning Representations* (*ICLR*), virtual, 2020.

[11] P. Wang, Q. Chen, X. He, and J. Cheng, Towards accurate post-training network quantization via bit-split and stitching, in *Proc. 37th Int. Conf. Machine Learning*, virtual, 2020, pp. 9847–9856.

[12] I. Hubara, Y. Nahshan, Y. Hanani, and R. Banner, D. Soudry, Accurate post training quantization with small calibration sets, in *Proc. 38th Int. Conf. Machine Learning*, virtual, 2021, pp. 4466–4475.

[13] E. Frantar and D. Alistarh, Optimal brain compression: A framework for accurate post-training quantization and pruning, in *Proc. 36th Conf. Neural Information Processing Systems* (*NeurIPS 2022*), New Orleans, LA, USA, 2022, pp. 4475–4488.

[14] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, QLoRA: Efficient finetuning of quantized LLMs, in *Proc. 37th Conf. Neural Information Processing Systems* (*NeurIPS 2023*), New Orleans, LA, USA, 2023, pp. 10088–10115.

[15] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, LoRA: Low-rank adaptation of large language models, in *Proc. 10th Int. Conf. Learning Representations* (*ICLR*), virtual, 2022.

[16] T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, 8-bit optimizers via block-wise quantization, in *Proc. 10th Int. Conf. Learning Representations* (*ICLR*), virtual, 2022.

[17] J. Kim, J. H. Lee, S. Kim, J. Park, K. M. Yoo, S. J. Kwon, and D. Lee, Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization, in *Proc. 37th Conf. Neural Information Processing Systems* (*NeurIPS 2023*), New Orleans, LA, USA, 2023, pp. 36187–36207.

[18] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, LLM.int8(): 8-bit matrix multiplication for transformers at scale, in *Proc. 36th Conf. Neural Information Processing Systems* (*NeurIPS 2022*), New Orleans, LA, USA, 2022, pp. 30318–30332.

[19] G. Xiao, J. Lin, M. Seznec, J. Demouth, and S. Han, SmoothQuant: Accurate and efficient post-training quantization for large language

models, in *Proc. 40th Int. Conf. Machine Learning*, Honolulu, HI, USA, 2023, pp. 38087–38099.

[20] D. Wu, Q. Tang, Y. Zhao, M. Zhang, Y. Fu, and D. Zhang, EasyQuant: Post-training quantization via scale optimization, arXiv preprint arXiv: 2006.16669, 2020.

[21] Y. Ding, H. Qin, Q. Yan, Z. Chai, J. Liu, X. Wei, and X. Liu, Towards accurate post-training quantization for vision transformer, in *Proc. 30th ACM Int. Conf. Multimedia*, Lisboa, Portugal, 2022, pp. 5380–5388.

[22] Z. Li, J. Xiao, L. Yang, and Q. Gu, RepQ-ViT: Scale reparameterization for post-training quantization of vision transformers, in *Proc. 2023 IEEE/CVF Int. Conf. Computer Vision* (*ICCV*), Paris, France, 2023, pp. 17181–17190.

[23] X. Wei, Y. Zhang, Y. Li, X. Zhang, R. Gong, J. Guo, and X. Liu, Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling, in *Proc. 2023 Conf. Empirical Methods in Natural Language Processing*, Singapore, 2023, pp. 1648–1665.

[24] J. Lin, J. Tang, H. Tang, S. Yang, W. M. Chen, W. C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, AWQ: Activation-aware weight quantization for LLM compression and acceleration, in *Proc. 7th Annu. Conf. Machine Learning and Systems* (*MLSys 2024*), Santa Clara, CA, USA, pp. 87–100.

[25] Z. Yao, R. Y. Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers, in *Proc. 36th Conf. Neural Information Processing Systems* (*NeurIPS 2022*), New Orleans, LA, USA, 2022, pp. 27168–27183.

[26] J. H. Lee, J. Kim, S. Kwon, and D. Lee, FlexRound: Learnable rounding based on element-wise division for post-training quantization, in *Proc. 40th Int. Conf. Machine Learning*, Honolulu, HI, USA, 2023, pp. 18913–18939.

[27] B. Hassibi, D. G. Stork, and G. J. Wolff, Optimal Brain Surgeon and general network pruning, in *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, CA, USA, 1993, pp. 293–299.

[28] E. Frantar, E. Kurtic, and D. Alistarh, M-FAC: Efficient matrix-free approximations of second-order information, in *Proc. 35th Conf. Neural Information Processing Systems* (*NeurIPS 2021*), virtual, 2021, pp. 14873–14886.

[29] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, HAWQ: Hessian aware quantization of neural networks with mixed-precision, in *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision* (*ICCV*), Seoul, Republic of Korea, 2019, pp. 293–302.

[30] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, and N. Kwak, LSQ+: Improving low-bit quantization through learnable offsets and better initialization, in *Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshop* (*CVPRW*), Seattle, WA, USA, 2020, pp. 2978–2985.

[31] M. Nagel, M. Van Baalen, T. Blankevoort, and M. Welling, Data-free quantization through weight equalization and bias correction, in *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision* (*ICCV*), Seoul, Republic of Korea, 2019, pp. 1325–1334.

[32] Y. Liu, H. Yang, Z. Dong, K. Keutzer, L. Du, and S. Zhang, NoisyQuant: Noisy bias-enhanced post-training activation quantization for vision transformers, in *Proc. 2023 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Vancouver, Canada, 2023, pp. 20321–20330.

[33] J. Shin, J. So, S. Park, S. Kang, S. Yoo, and E. Park, NIPQ: Noise proxy-based Integrated Pseudo-Quantization, in *Proc. 2023 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Vancouver, Canada, 2023, pp. 3852–3861.

[34] P. H. P. Savarese, X. Yuan, Y. Li, and M. Maire, Not all bits have equal value: Heterogeneous precisions via trainable noise, in *Proc. 36th Conf. Neural Information Processing Systems* (*NeurIPS 2022*), New Orleans, LA, USA, 2022, pp. 35769–35782.

[35] B. Dong, W. Wang, D. P. Fan, J. Li, H. Fu, and L. Shao, Polyp-PVT: Polyp segmentation with pyramid vision transformers, *CAAI Artif. Intell. Res.*, vol. 2, p. 9150015, 2023.

[36] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li, ImageNet: A large-scale hierarchical image database, in *Proc. 2009 IEEE Conf. Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009, pp. 248–255.

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, in *Proc. 9th Int. Conf. Learning Representations* (*ICLR*), virtual, 2021.

[38] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, Training data-efficient image transformers & distillation through attention, in *Proc. 38th Int. Conf. Machine Learning*, virtual, 2021, pp. 10347–10357.

[39] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, in *Proc. 2021 IEEE/CVF Int. Conf. Computer Vision* (*ICCV*), Montreal, Canada, 2021, pp. 9992–10002.

[40] R. Wightman, Pytorch image models, https://github.com/rwightman/pytorch-image-models, 2019.

[41] W. Huang, Y. Liu, H. Qin, Y. Li, S. Zhang, X. Liu, M. Magno, and X. Qi, Billm: Pushing the limit of post-training quantization for LLMs, arXiv preprint arXiv: 2402.04291, 2024.

[42] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al., OPT: Open pre-trained transformer language models, arXiv preprint arXiv: 2205.01068, 2022.

[43] BigScience Workshop, T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, et al., BLOOM: A 176B-Parameter open-access multilingual language model, arXiv preprint arXiv: 2211.05100, 2022.

[44] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 5485–5551, 2020.

[45] S. Merity, C. Xiong, J. Bradbury, and R. Socher, Pointer sentinel mixture models, in *Proc. 5th Int. Conf. Learning Representations* (*ICLR*), Toulon, France, 2017.

[46] M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger, The Penn Treebank: Annotating predicate argument structure, in *Proc. 2nd APRA Human Language Technology Workshop*, Plainsboro, NJ, USA, 1994, pp. 114–119.

[47] T. Dettmers and L. Zettlemoyer, The case for 4-bit precision: k-bit inference scaling laws, in *Proc. 40th Int. Conf. Machine Learning*, Honolulu, HI, USA, 2023, pp. 7750–7774.