Title

NoisyQuant: Noisy Bias-Enhanced Post-Training Activation Quantization for Vision Transformers

Authors

Liu, Yijiang
Yang, Huanrui
Dong, Zhen
et al.

Publication Date

2023-06-24

DOI

10.1109/cvpr52729.2023.01946

NoisyQuant: Noisy Bias-Enhanced Post-Training Activation Quantization for Vision Transformers

Yijiang Liu*[1], Huanrui Yang*[2], Zhen Dong[2], Kurt Keutzer[2], Li Du[1]✉, Shanghang Zhang[3]✉
[1]Nanjing University, [2]University of California, Berkeley
[3]National Key Laboratory for Multimedia Information Processing, School of Computer Science, Peking University
liuyijiang@smail.nju.edu.cn, {huanrui, zhendong, keutzer}@berkeley.edu
ldu@nju.edu.cn, shanghang@pku.edu.cn

Abstract

The complicated architecture and high training cost of vision transformers urge the exploration of post-training quantization. However, the heavy-tailed distribution of vision transformer activations hinders the effectiveness of previous post-training quantization methods, even with advanced quantizer designs. Instead of tuning the quantizer to better fit the complicated activation distribution, this paper proposes NoisyQuant, a quantizer-agnostic enhancement for the post-training activation quantization performance of vision transformers. We make a surprising theoretical discovery that for a given quantizer, adding a fixed Uniform noisy bias to the values being quantized can significantly reduce the quantization error under provable conditions. Building on the theoretical insight, NoisyQuant achieves the first success on actively altering the heavy-tailed activation distribution with additive noisy bias to fit a given quantizer. Extensive experiments show NoisyQuant largely improves the post-training quantization performance of vision transformer with minimal computation overhead. For instance, on linear uniform 6-bit activation quantization, NoisyQuant improves SOTA top-1 accuracy on ImageNet by up to 1.7%, 1.1% and 0.5% for ViT, DeiT, and Swin Transformer respectively, achieving on-par or even higher performance than previous nonlinear, mixed-precision quantization.

1. Introduction

Inspired by the success of Self Attention (SA)-based transformer models in Natural Language Processing (NLP) tasks [31], recent researches make significant progress in applying transformer models to the field of computer vision [3, 10, 22, 30]. In the meantime, the typical design of transformer models induces large model sizes, high computational consumption, and long training time. For instance, the widely used DeiT-Base model [30] contains 86M parameters, logs 18G floating-point operations for a single input, and requires 300 epochs of training on the ImageNet dataset. This leads to significant difficulties in hardware deployment. In facing such difficulty, a number of compression and acceleration methods are applied to vision transformer models, including pruning [6, 39], quantization [24, 42], and neural architecture search [5], *etc*.

Among these methods, quantization appears as one of the most effective and widely applied ways [11]. Quantization process uses a predefined "quantizer" function to convert the continuous representation of weights and activations into a small number of discrete symbols, therefore enabling low-precision representations for straightforward memory savings. For DNN models, the approximation error made by the quantizer inevitably leads to performance drop. A series of work focuses on Quantization-Aware Training (QAT) that finetunes the quantized model at low precision [8, 9, 25, 37, 47]. However, given the high training cost and the complicated computation graph of vision transformer models, retraining the model at low precision could be costly and unstable [42]. Alternatively, Post-Training Quantization (PTQ) is preferable for vision transformers as it eliminates the need for re-training or finetuning the quantized model, instead only adjusts the design of the quantizer based on the full-precision pretrained model and a small set of sampled calibration data [1, 2, 23, 32, 36].

For example, linear quantizers [7, 29, 36, 38] reduce quantization error by shifting, clipping, and scaling the values to be quantized. Nonlinear quantizers [13, 42, 44] further adjust the width and location of each quantization bin to better fit the distribution.

Unfortunately, though progresses are made in designing better PTQ quantizers, it still appears to be significantly challenging to quantize vision transformer models, especially

* Equal contribution.
✉ Corresponding Author.

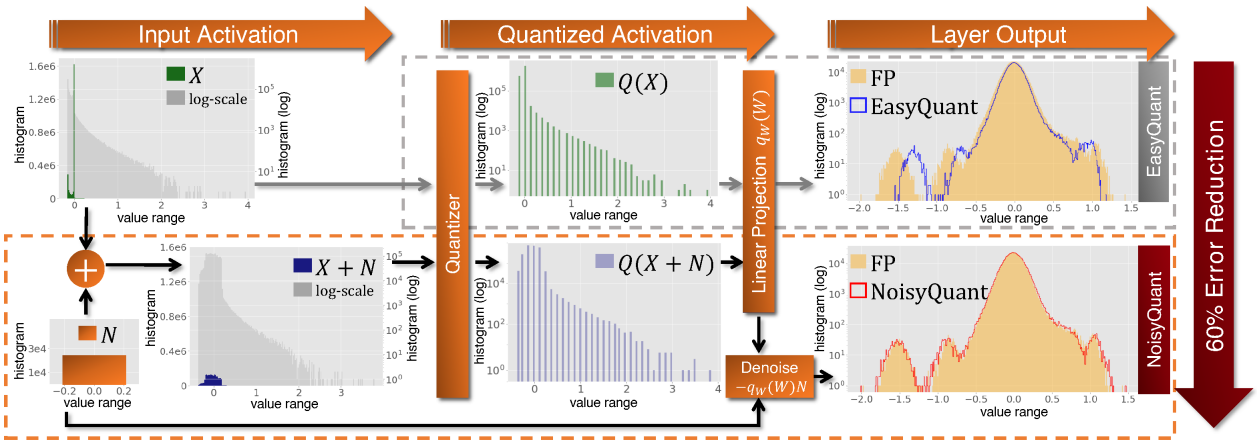


Figure 1. **Overview of the NoisyQuant pipeline (orange box) with comparison to EasyQuant [36] (gray box).** Histograms of input activations, quantized activations, and layer outputs are illustrated in the pipeline. Starting from the input activation X following a GELU function, where the green histogram is plotted in linear-scale and the gray in log-scale, NoisyQuant adds a fixed NoisyBias N sampled from a Uniform distribution onto X. Adding the noise before quantization flattens the peaks in the activation distribution to make it more friendly to quantization, as visualized in the linear/log-scaled histogram of $X + N$, and theoretically proved in Sec. 3.2. The noise is removed from the result of the fixed-point activation-weight multiplication with a denoising bias term, which we formulate in Sec. 3.3. The rightmost sub-figures illustrate the layer output distribution of EasyQuant and NoisyQuant. Compared to the output of the floating-point model (yellow), NoisyQuant output follows the distribution closely and achieves up to 60% output error reduction on some transformer layers, as shown in Sec. 4. The resulting PTQ performance improvement of the entire model is provided in Sec. 5.

for activation quantization. Transformer layers produce up to millions of activation values with sophisticated distributions. For instance, outputs of the GELU function [15] are asymmetrically distributed, with spikes in the histogram at some values and a long tail in a large range (see top-left of Fig. 1). Some linear projection layers also lead to significantly large activation values that are sparsely distributed in a very long tail [23]. Consequently, low-precision PTQ on vision transformer suffers from performance degradation, even utilizing non-linear or mixed-precision quantizers at the cost of additional data communication and computation overheads [23, 43]. *No* linear uniform PTQ method achieves good performance on vision transformer models.

This paper provides a brand new perspective in dealing with the complicated vision transformer activation distribution. Instead of adding more tricks in the quantizer design to fit the activation distribution, this work explores the potential of actively and cost-effectively altering the distribution being quantized, making it more friendly to a given quantizer. The pipeline of our proposed method is illustrated in Fig. 1. Specifically, we make a surprising discovery that for any quantizer, the quantization error can be significantly reduced by adding a fixed noisy bias sampled from a Uniform distribution to the activation before quantization. We theoretically prove the condition when the quantization error reduction can be achieved. On this basis, we propose **NoisyQuant**, a *plug-and-play*, quantizer-agnostic enhancement on the post-training activation quantization of vision transformer models. For each layer, we sample a Noisy Bias based on the input activation distribution following our theoretical analysis, and compute the corresponding denoising bias to retain the correct output of the layer. At inference time, the Noisy Bias is added to the input activation before quantization, and the denoising bias is removed from the layer output. This process significantly reduces the quantization error with minimal computation overhead. NoisyQuant leads to significant improvement in the PTQ performance of state-of-the-art vision transformers. Applying NoisyQuant on top of a uniform linear quantization achieves on-par performance to SOTA mixed-precision, nonlinear PTQ methods [23, 42]. Adding NoisyQuant on top of these nonlinear quantizers achieves further performance gain.

To the best of our knowledge, this paper makes the following novel contributions:

- Theoretically shows the possibility and proves feasible conditions for reducing the quantization error of heavy-tailed distributions with a fixed additive noisy bias;
- Proposes NoisyQuant, a quantizer-agnostic enhancement for PTQ performance on activation quantization. NoisyQuant achieves the first success in actively refining the distribution being quantized to reduce quantization error following the theoretical results on additive noisy bias, with minimal computation overhead;
- Demonstrates consistent performance improvement by applying NoisyQuant on top of existing PTQ quantizers. For 6-bit PTQ, NoisyQuant improves the ImageNet

top-1 accuracy of uniform linear quantized vision transformer models by up to 1.7%, and improves SOTA nonlinear quantizer PTQ4ViT [43] by up to 0.7%.

2. Related work

2.1. Transformer models in computer vision

Since Self-Attention (SA) [31]-based transformer models have shown impressive performance in Natural Language Processing tasks, researchers transfer the idea into computer vision. Pioneering work on Vision Transformer [10] for the first time builds a totally SA-based model in the field of computer vision and shows its effectiveness. Since then, a variety of transformer-based vision models have emerged with boosted performance, claiming state-of-the-art in multiple computer vision tasks. Notable advances include additional data augmentation and distillation [30], incorporating multi-stage architecture [22, 33], and explore novel attention design [14,22,41]. Transformer models also advance to tasks beyond classification, such as object detection [3], semantic segmentation [46], and image generation [17]. Furthermore, the development of vision transformer brings unification of model architecture and representation across image and text, enabling stronger vision-language connection via image-text-pair contrastive learning [27].

The rapid development of vision transformer models motivates the exploration of quantization, as quantization brings straightforward memory and computation savings agnostic to model architectures. Compared to CNN models, the use of advanced activation functions like GELU [15] and more complicated computation graphs like the attention mechanism makes vision transformers more sensitive to quantization [43]. This work aims to resolve the difficulty of post-training vision transformer activation quantization caused by the heavy-tailed activation distribution, which we believe will facilitate the deployment of state-of-the-art transformer models into real-world computer vision applications.

2.2. Post-training quantization methods

Post-training quantization (PTQ) is a preferable quantization method in scenarios with limited training data or limited computational resources for the expensive quantization-aware fine-tuning process. Previous methods have dedicatedly studied PTQ on CNNs. For instance, EasyQuant [36] presents a quick searching mechanism to determine the proper clipping range for quantization. ZeroQ [2] proposes a distillation mechanism to generate proxy input images, which can leverage inherent statistics of batch normalization layers to conduct PTQ. SQuant [12] calibrates the model for lower quantization error on the basis of the sensitivity obtained by the Hessian spectrum. On vision transformers, Liu *et al*. [23] presents a linear PTQ method that uses Pearson correlation coefficients and ranking loss to determine scaling factors. PTQ4ViT [43] introduces a nonlinear Twin Uniform Quantization based on vision transformers' activation distributions, which set different scaling factors for 1) positive and negative activations of GeLU, and 2) small and large values of Softmax, which can reduce the activation quantization error to some extent with the cost of additional computation overhead. [19, 34] reparameterize the LN layer to suppress outliers by scaling down activation values.

Unlike previous PTQ methods that analyze the activation distribution and fit the quantizer accordingly, our method takes a novel perspective of actively modifying the distribution being quantized with a sampled noisy bias. We show our proposed method can bring quantization error reduction and performance enhancement for all the PTQ quantizers for vision transformer activation quantization.

3. Method

In this section, we provide a preliminary introduction to the notation and typical formulation of DNN quantization in Sec. 3.1, theoretically analyze the potential of reducing quantization error with pre-sampled additive noisy bias in Sec. 3.2, and formulate NoisyQuant, a noisy bias-enhanced post-training activation quantization method with reduced quantization error in Sec. 3.3.

3.1. Preliminary

To start with, we recap the post-training quantization process on DNN models. Given the dominance of fully-connected (FC) linear projection layers in transformer models, here we focus our discussion on the FC layer.

The computation performed in a FC layer with weight $W \in \mathbb{R}^{k\times m}$, input activation $X \in \mathbb{R}^{m\times n}$, and bias $B \in \mathbb{R}^{k\times 1}$ can be formulated as

$$f(X) = WX + B. \tag{1}$$

The main computation burden in Eq. (1) lies in the matrix multiplication of WX, which requires a total of $k \times m \times n$ multiply-accumulate computations (MACs). Post-training quantization aims to reduce the computation cost of matrix multiplication by converting X and W to fixed-point quantized values, therefore replacing floating-point MACs with much cheaper fixed-point operations [16, 40]. The quantized FC layer can therefore be represented as

$$f_q(X) = q_W(W)q_A(X) + B, \tag{2}$$

where $q_W(\cdot)$ and $q_A(\cdot)$ denotes the quantizer function of weight and activation respectively. Previous research has observed that the heave-tailed activation distribution of X in transformer is causing significant quantization error between X and $q_A(X)$ at low precision, leading to significant performance degradation. Previous PTQ methods modify the design of quantizer $q_A(\cdot)$ to mitigate the quantization

error. In this paper, we propose an alternative approach, where we modify the distribution of activation X with a pre-sampled noisy bias before quantization, and prove it to be a plug-and-play enhancement on any quantizer design.

3.2. Theoretical analysis on quantization error

As previous research mainly blames the heavy-tailed distribution of transformer activation as the main cause for large quantization errors [23, 42], in this section we analyze how the quantization error change as we actively alter the activation distribution being quantized. A straightforward way to alter the input activation distribution is to add a fixed "Noisy Bias" sampled from a Uniform random distribution. Denoting the input activation as X and the Noisy Bias as N, where N and X have the same dimensions, here we formulate the quantization error difference between X and $X+N$ for a quantizer Q in Eq. (3)

$$D(X,N) = QE(X+N) - QE(X) = \\ ||(Q(X+N) - X - N)||_2^2 - ||(Q(X) - X)||_2^2. \tag{3}$$

Theorem 1 states the condition where $D(x,N) \leq 0$ holds for each histogram snapshot x of activation X.

Theorem 1. *Consider a given quantizer Q with quantization bin width $2b$. For each histogram snapshot of X where all elements X_i have the same distance x away from the center of the quantization bin, and for a Noisy Bias N sampled from $N \sim \mathcal{U}(-n,n)$ where $x \leq n \leq 2b - x$, we have*

$$D(x,N) \leq 0 \ \textit{iff} \ 0 \leq x \leq n\left(1 - \sqrt{\frac{n}{3b}}\right). \tag{4}$$

Proof. Without loss of generality, we consider the quantization bin of quantizer Q which x falls into to be centered at the 0 point, therefore $0 \leq x \leq b$.

In this way, $Q(X_i) = b$ for all elements in the histogram snapshot, leading to a quantization error

$$QE(X) = (b-x)^2. \tag{5}$$

Consider adding $N \sim \mathcal{U}(-n,n)$. $X+N$ will follow $\mathcal{U}(x-n, x+n)$. In the case of $x \leq n \leq 2b - x$, $Q(X_i + N_i) = b$ if $N_i \in [-x, n]$, and $Q(X_i + N_i) = -b$ if $N_i \in [-n, -x)$ for all elements in $X+N$. Though we only sample a single instance of N from the Uniform distribution, given the large number of elements in the histogram snapshot, the empirical quantization error $QE(X+N)$ can be estimated with an expectation over all $N \sim \mathcal{U}(-n,n)$ following the Weak Law of Large Numbers, as

$$\begin{aligned} \mathbf{E}_N[QE(X+N)] &= \frac{1}{2n}\int_{x-n}^{x+n} QE(x+z)\,dz \\ &= \frac{1}{2n}\left[\int_{x-n}^{0}(z+b)^2\,dz + \int_{0}^{x+n}(z-b)^2\,dz\right] \\ &= x^2 - \frac{b}{n}x^2 + \frac{n}{3} - nb + b^2. \end{aligned} \tag{6}$$

Combining Eq. (5) and Eq. (6), we have

$$\begin{aligned} D(x,N) &= \mathbf{E}_N\left[QE(X+N)\right] - QE(X) \\ &= -\frac{b}{n}x^2 + 2bx + \frac{n^2}{3} - nb. \end{aligned} \tag{7}$$

We verify this derivation empirically in Sec. 4.

It can be observed that given b and n, $D(x,N)$ is a quadratic function with respect to the activation value x. The inequality $D(x,N) \leq 0, 0 \leq x \leq n$ can be easily solved with respect to x as

$$0 \leq x \leq n\left(1 - \sqrt{\frac{n}{3b}}\right), \tag{8}$$

which is always feasible given both b and n are positive. □

Theorem 1 indicates that adding Noisy Bias can always reduce the quantization error of elements close to the center of the quantization bin, which is the source of large quantization error in a heavy-tailed distribution.

In practice, to choose a proper n for a layer in a pretrained model, we can first acquire the empirical distribution of activation X by passing a small amount of calibration data through the pretrained model. With the distribution of X, we can estimate the expected quantization error reduction of the layer numerically as a function of n

$$\mathcal{L}(n) = \sum_{x \in X}[D(x,N)]. \tag{9}$$

Note that x here denotes the distance between each activation value and the corresponding quantization bin center. Though we cannot find a closed-form solution for the minima of $\mathcal{L}(n)$, we can easily find a good-enough choice with a linear search over a small range of n.

3.3. NoisyQuant formulation and pipeline

Given our theoretical analysis on reducing quantization error with Noisy Bias, we propose to apply the Noisy Bias as an enhancement to the PTQ methods of vision transformer models, which we name as "**NoisyQuant**". The pipeline of NoisyQuant is visualized in Fig. 1. Specifically, before performing PTQ and deploying the model, we sample a single Noisy Bias $N \in \mathbb{R}^{m \times 1}$ for each layer from a Uniform distribution, and fix it throughout the inference. The range of N is determined by the calibration objective defined in Eq. (9). At inference time, we add the Noisy Bias N to the input activation X under the broadcasting rule before going through the activation quantizer, therefore actively altering the distribution being quantized to reduce the input quantization error. After the activation-weight multiplication in the linear layer, we remove the impact of N by adjusting the bias term in the linear layer with a denoising bias computed from N, therefore retrieving the correct output. Formally,

NoisyQuant converts the quantized FC computation defined in Eq. (2) into

$$f_{Nq}(X) = q_W(W)q_A(X+N) + (B - q_W(W)N)\,. \quad (10)$$

Since N is sampled before the deployment of the model and fixed it throughout the inference, the denoising output bias $B' = B - q_W(W)N$ only needs to be computed once before the deployment, without any computation overhead in the inference process. The only computation overhead brought by NoisyQuant at inference time is the on-the-fly summation of $X + N$, which is negligible comparing to the cost of the matrix multiplications. Both B' and N shall be stored with higher precision. In practice, we store both variables as INT16 to enable integer-only inference.

Comparing to the output of the floating-point FC layer defined in Eq. (1), the output quantization error induced by the activation quantization defined in Eq. (2) is

$$QE_O(X) = ||f_q(X) - f(X)||_2^2 = ||W[Q(X) - X]||_2^2, \quad (11)$$

where $Q(\cdot)$ is short of $q_A(\cdot)$, and we omit the weight quantization $q_W(\cdot)$ as we are focusing on the activation quantization. Similarly, the output quantization error after the application of NoisyQuant can be computed as

$$\begin{aligned} QE'_O(X) &= ||f_{Nq}(X) - f(X)||_2^2 \\ &= ||W[Q(X+N) - X - N]||_2^2. \end{aligned} \quad (12)$$

As we prove in Theorem 1 that the Noisy Bias enables $X + N$ to have a lower quantization error than X, we can achieve lower output quantization error in Eq. (12) than in Eq. (11) when NoisyQuant is applied, therefore improving the PTQ performance of the model.

4. Theoretical insight verification

In this section, we provide empirical verification of our theoretical analysis on reducing the activation quantization error with NoisyQuant. We numerically verify Theorem 1 with simulated data in Sec. 4.1, and demonstrate the reduction in both input and output quantization errors under the true inference-time activation distribution in Sec. 4.2.

4.1. Empirical verification of Theorem 1

Here we follow the settings in Theorem 1 to empirically verify its theoretical derivation. Specifically, we set the quantization bin range $b = 1$, and explore how the quantization error difference induced by the Noisy Bias change with different choice of activation value x and noisy bias range n. For all empirical results we experiment with 10 instances of independently sampled Noisy Bias N, and report the mean and standard derivation of $D(X, N)$ defined in Eq. (3) across the 10 instances. We consider input activation X to be a tensor with 20 dimensions. Given the tens to hundreds

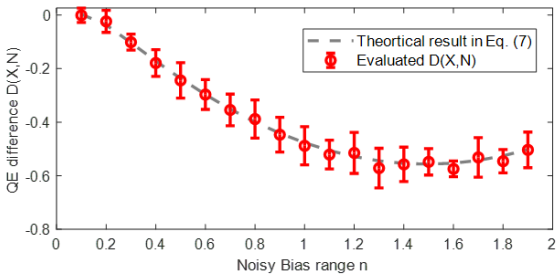


Figure 2. Verifying Eq. (7) with respect to Noisy Bias range n. Circle indicates the mean and error bar the std for evaluated results.

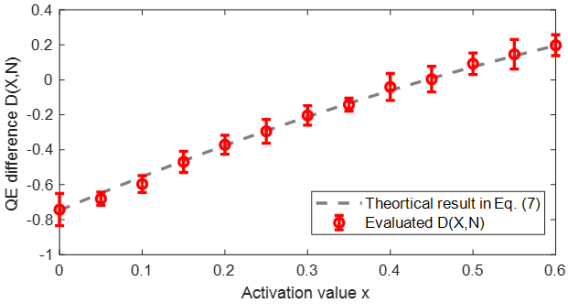


Figure 3. Verifying Eq. (7) with respect to activation value x. Circle indicates the mean and error bar the std for evaluated results.

of thousands of activation values in each transformer layer, it is likely to see more than 20 activation elements taking the same value. As we base our theoretical derivation on the Weak Law of Large Numbers, having more elements taking the same value will lead to less variance than the simulation results provided in this section.

For the first experiment, we fix all elements in X to take value $x = 0.1$, and alter n in the range of $[0.1, 1.9]$. We compare the empirical $D(X, N)$ and the theoretical result derived in Eq. (7) in Fig. 2. The evaluated results closely follow the theoretical line. Even with only 20 activation elements the standard deviation across independently sampled Noisy Bias is much smaller than the quantization error benefit brought by adding the Noisy Bias.

Similarly, we fix $n = 1.4$ and alter activation value x in the range of $[0, 0.6]$. The results are shown in Fig. 3, where we see the same trend of close mean and small variance as in the previous experiment. Quantization error can be reduced by Noisy Bias when $x < 0.44$, which follows the numerical computation result of Eq. (4). Both results show the correctness and stability of our derivation of Eq. (7) and the proof of Theorem 1.

4.2. Quantization errors in transformer layers

As mentioned in Sec. 3.2, NoisyQuant enables the reduction of both input activation quantization error (QE) and output error (QE_O) of a linear layer given a quantizer. In this section, we verify the error reduction on the real acti-

Table 1. **Averaged quantization error on different types of layers.** The calculation is performed with 5,120 images randomly selected from the ImageNet validation set on the Swin-T model. Layer names are defined as same as in Timm [35]. We compute the input quantization error difference D and compare the output errors between NoisyQuant and EasyQuant.

Layer Name	Eq. (3) $D(X)$	Eq. (11) $QE_O(X)$	Eq. (12) $QE'_O(X)$	Output QE drop
qkv	-0.029	0.107	0.096	10%
proj	-0.106	0.093	0.081	13%
fc1	-0.527	0.126	0.123	2%
fc2	-5.130	1.049	0.852	19%

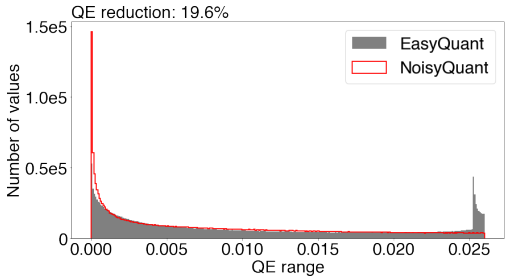
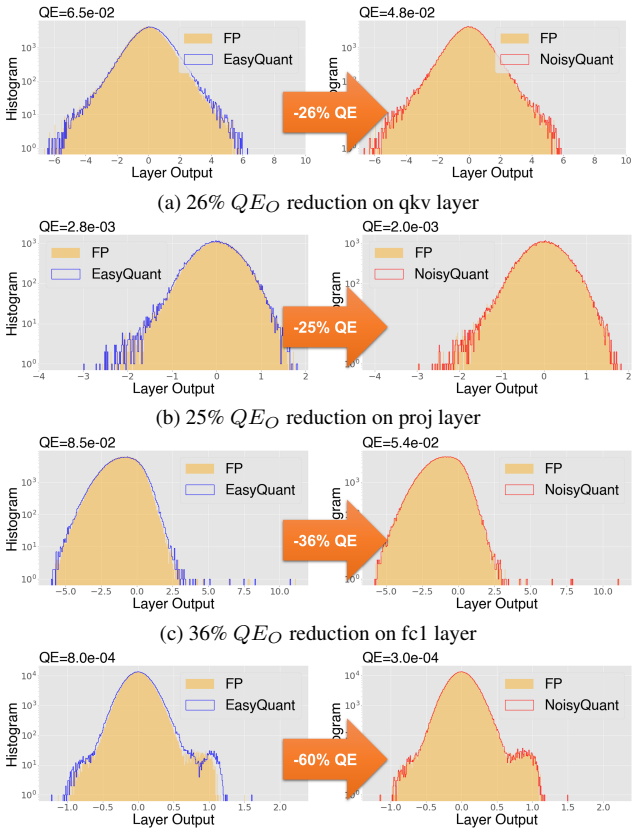


Figure 4. Quantization error histogram of fc2 input activation. We apply NoisyQuant on top of the EasyQuant quantizer, and compare the quantization error distribution (red) with EasyQuant (grey).

vation distribution of vision transformer model. Practically, we run evaluation on randomly selected 5120 images of validation set with linear PTQ quantizer EasyQuant [36] as baseline, and calculate QE and QE_O with or without adding Noisy Bias. Experiments are ran on different types of layers, namely qkv, proj, fc1, and fc2, respectively. The results are averaged across all layers of each type. As shown in Tab. 1, input quantization error difference $D(X)$ defined in Eq. (3) is consistently lower than 0, which meets our expectation that Noisy Bias reduces input quantization error. This further leads to significant drop in output error between the quantized and floating-point layers for all layer types. Up to 19% averaged output error drop can be achieved in fc2 layers.

To further understand the impact NoisyQuant makes, we visualize both the input and output distribution of each layer. We start with highlighting fc2 as it achieves the greatest QE reduction. The fc2 layer takes in the output of GELU activations, which has significantly complicated distribution as introduced in Sec. 1. Fig. 4 visualizes the distribution of input quantization error with or without NoisyQuant. With EasyQuant only, significant amount of activation elements are showing large quantization error, which is most probably owing to bad quantization on densely distributed areas. Instead, NoisyQuant generally makes QE distributed plainly and for most elements near zero, therefore leading to significant QE reduction overall. Subsequently, we plot the



(a) 26% QE_O reduction on qkv layer

(b) 25% QE_O reduction on proj layer

(c) 36% QE_O reduction on fc1 layer

(d) 60% QE_O reduction on fc2 layer

Figure 5. Output histogram in selected transformer layers before and after activation quantization. NoisyQuant significantly reduces the output quantization error over EasyQuant baselines.

histogram of selected layers' outputs in Fig. 5. NoisyQuant output consistently follows the floating-point layer output distribution closer than that of EasyQuant, especially at values with large magnitudes. This leads to up to 60% error reduction in the output of some vision transformer layers.

5. Experiments

This section evaluates the impact of NoisyQuant on the PTQ performance of some commonly used vision transformer architectures. First, we evaluate the performance on image classification tasks with the ViT [10], DeiT [30] and Swin [22] transformer models. We also evaluate the performance of object detection with the DETR model [4]. Finally, we provide ablation study on how NoisyQuant improves the PTQ performance when added to each type of layers in the vision transformer model.

5.1. Experimental settings

Dataset. The classification tasks are implemented on the ImageNet-2012 classification dataset [28], with 1.2 million training images and 50,000 validation images in 1000 classes. For object detection task, the MSCOCO 2017 dataset [20]

Table 2. **PTQ accuracy of ViT models.** Top-1 accuracy on ImageNet is reported in the table. W/A represents weight and activation bit-width, respectively. The default patch size is 16×16 and the default input resolution is 224×224. "*" after model name indicates 384×384 input. "MP" represents mixed-precision. For our methods, "NoisyQuant-Linear" is implemented on top of the EasyQuant quantizer, whereas "NoisyQuant-PTQ4ViT" is implemented on the nonlinear PTQ4ViT quantizer. All NoisyQuant results are shown with mean±STD of 20 repeated experiments.

Method	W/A	ViT-S	ViT-B	ViT-B*
Pretrained	32/32	81.39	84.54	86.00
Percentile [18]	6/6	67.74	77.63	77.60
Liu *et al*. [23]	6 MP	-	75.26	-
PTQ4ViT-Linear [43]	6/6	70.24	75.66	46.88
EasyQuant [36]	6/6	75.13	81.42	82.02
NoisyQuant-Linear	6/6	**76.86**±0.06	**81.90**±0.11	**83.00**±0.09
PTQ4ViT [43]	6/6	78.63	81.65	**83.34**
NoisyQuant-PTQ4ViT	6/6	**78.65**±0.07	**82.32**±0.09	83.22±0.10
Percentile [18]	8/8	78.77	80.12	82.53
Liu *et al*. [23]	8 MP	-	76.98	-
FQ-ViT [21]	8/8	-	83.31	-
PTQ4ViT-Linear [43]	8/8	80.46	83.89	85.35
EasyQuant [36]	8/8	80.75	83.89	85.53
NoisyQuant-Linear	8/8	**80.81**±0.01	**84.10**±0.03	**85.56**±0.01
PTQ4ViT [43]	8/8	81.00	**84.25**	85.82
NoisyQuant-PTQ4ViT	8/8	**81.15**±0.02	84.22±0.01	**85.86**±0.01

Table 3. **PTQ accuracy of DeiT models.** Abbreviations are the same as Tab. 2.

Pretrained	W/A	DeiT-S	DeiT-B	DeiT-B*
Pretrained	32/32	79.85	81.80	83.11
Percentile [18]	6/6	70.49	73.99	78.24
Bit-Split [32]	6/6	74.58	76.39	-
Liu *et al*. [23]	6/6	74.58	77.02	-
Liu *et al*. [23]	6 MP	75.10	77.47	-
PTQ4VIT-Linear [43]	6/6	72.26	78.78	68.44
EasyQuant [36]	6/6	75.27	79.47	81.26
NoisyQuant-Linear	6/6	**76.37**±0.13	**79.77**±0.08	**81.40**±0.06
PTQ4VIT [43]	6/6	76.28	80.25	81.55
NoisyQuant-PTQ4ViT	6/6	**77.43**±0.08	**80.70**±0.11	**81.65**±0.04
Percentile [18]	8/8	73.98	75.21	80.02
Bit-Split [32]	8/8	76.39	79.42	-
Liu *et al*. [23]	8/8	77.47	80.48	-
Liu *et al*. [23]	8 MP	78.09	81.29	-
FQ-ViT [21]	8/8	79.17	81.20	-
PTQ4VIT-Linear [43]	8/8	77.65	80.94	82.33
EasyQuant [36]	8/8	78.98	81.19	82.10
NoisyQuant-Linear	8/8	**79.11**±0.02	**81.30**±0.03	**82.23**±0.02
PTQ4VIT [43]	8/8	79.47	**81.48**	**82.97**
NoisyQuant-PTQ4ViT	8/8	**79.51**±0.01	81.45±0.02	82.49±0.02

is utilized to evaluate the PTQ performance, which contains 118,000 training images and 5,000 validation images. For both tasks we randomly sample 1024 images from the training set as the calibration data for all PTQ methods implemented in our experiments.

Pretrained model architecture. We perform uniform 6-bit and 8-bit PTQ on different variants of pretrained vision transformer models. For classification tasks we quantize the model family of ViT [10], DeiT [30], and Swin [22] provided by the Timm library [35], including large-scale models with 384×384 input resolution (marked with "*"). For detection, we perform PTQ on the official implementation of DETR [4] with ResNet-50 backbone. We use the pretrained model checkpoint provided by the official source of each model, whose floating-point performances are reported in the results tables and match their original papers.

Table 4. **PTQ accuracy of Swin models.** Abbreviations are the same as Tab. 2.

Method	W/A	Swin-T	Swin-S	Swin-B	Swin-B*
Pretrained	32/32	81.39	83.23	85.27	86.44
Percentile [18]	6/6	77.75	80.41	81.90	82.94
PTQ4VIT-Linear [43]	6/6	78.45	81.74	83.35	85.22
EasyQuant [36]	6/6	79.51	82.45	84.30	85.89
NoisyQuant-Linear	6/6	**80.01**±0.06	**82.78**±0.04	**84.57**±0.04	**85.96**±0.04
PTQ4VIT [43]	6/6	80.47	82.38	84.01	85.38
NoisyQuant-PTQ4ViT	6/6	**80.51**±0.03	**82.86**±0.05	**84.68**±0.06	**86.03**±0.07
Percentile [18]	8/8	79.88	80.93	83.08	84.31
FQ-ViT [21]	8/8	80.51	82.71	-	-
PTQ4VIT [43]	8/8	80.96	82.75	84.79	86.16
EasyQuant [36]	8/8	80.95	83.00	85.10	86.39
NoisyQuant-Linear	8/8	**81.05**±0.03	**83.07**±0.03	**85.11**±0.04	**86.42**±0.02
PTQ4VIT [43]	8/8	81.24	83.10	85.14	86.39
NoisyQuant-PTQ4ViT	8/8	**81.25**±0.02	**83.13**±0.01	**85.20**±0.03	**86.44**±0.01

Implementation Details. Following the mainstream quantization methods [23, 43], we quantize all the weights and inputs involved in matrix multiplication. More specifically, we quantize weights of qkv-projectors, attention output projectors, MLPs, linear embeddings, and model heads. Besides, we also quantize input activations of linear layers and matrix multiplication operators. Following [23, 26, 43, 45], we keep layer normalization and softmax operations as full precision. Apart from special notes, we perform symmetric layer-wise quantization for activations and symmetric channel-wise quantization for weights. In particular, weights are quantized by absolute MinMax values without clamping. We implement NoisyQuant on top of linear quantizer EasyQuant [36] and nonlinear quantizer PTQ4ViT [43]. After we apply the selected quantizer to determine the quantization bin width and location on calibration data, we decide the parameters of the Noisy Bias distribution through a linear search. We use the empirical activation quantization error as defined in Eq. (9) as the search objective, where we find the Noisy Bias parameter that can minimize the quantization error.

5.2. NoisyQuant performance

Here we compare NoisyQuant with state-of-the-art PTQ methods, including percentile [18], bit-split [32], Liu *et al*. [23], FQ-ViT [21], EasyQuant [36], and PTQ4ViT [43], on ImageNet. PTQ performance of different variants of ViT, DeiT, and Swin transformer models are provided in Tab. 2, Tab. 3, and Tab. 4, respectively. Experiments are conducted for both linear and nonlinear quantizers.

Table 5. **PTQ accuracy of DETR models.** Mean average precision (mAP) evaluated on the MSCOCO 2017 dataset is reported.

Model	Method	W/A	mAP
DETR [4] (COCO2017)	Pretrained	32/32	42.0
	Percentile [18]	8/8	38.6
	Bit-Split [32]	8/8	40.6
	Liu *et al*. [23]	8/8	41.2
	EasyQuant [36]	8/8	41.1
	NoisyQuant-Linear	8/8	**41.4**$\pm$0.05

Linear quantizer. Previous linear PTQ methods suffer from severe performance degradation on vision transformer activation quantization. EasyQuant [36] achieves the best performance among linear quantizers, yet still induce 2-6% of accuracy drop under 6-bit quantization compared to the floating-point baseline. By applying the proposed NoisyQuant method on top of the EasyQuant quantizer, the resulted NoisyQuant-Linear quantizer consistently and significantly improves the PTQ accuracy on all variants of vision transformer models. Under the W6A6 setting, NoisyQuant-Linear achieves performance improvement of 1.73% on ViT-S, 0.98% on ViT-B*, 1.1% on DeiT-S, and 0.5% on SWIN-T over EasyQuant. Under the W8A8 setting, as the performance gap between EasyQuant and floating-point gets smaller, NoisyQuant still appears beneficial in further improving the PTQ performance. It is worth noting that although nonlinear quantizer like PTQ4ViT [43] consistently achieves higher PTQ performance than the linear EasyQuant quantizer, the enhancement brought by NoisyQuant enables the NoisyQuant-Linear quantizer to achieve on-par or even better performance than the nonlinear PTQ4ViT, with much lower overhead in hardware deployment.

Nonlinear quantizer. As we claim NoisyQuant as a quantizer-agnostic enhancement of PTQ, we further implement NoisyQuant on top of the nonlinear quantizer PTQ4ViT [43], namely NoisyQuant-PTQ4ViT. NoisyQuant-PTQ4ViT outperforms most results of PTQ4ViT, notably achieving for a 1.25% improvement on DeiT-S, 0.67% on ViT-S, and 0.67% on Swin-B in the W6A6 setting. The W6A6 PTQ performance of Swin-S and Swin-B* for the first time improved to within 0.5% of the floating-point baseline by the NoisyQuant-PTQ4ViT quantizer, which can never be achieved without the help of NoisyQuant.

Besides classification models, we report the PTQ performance of the DETR object detection model in Tab. 5. NoisyQuant implemented on top of the EasyQuant quantizer also outperforms the EasyQuant baseline, and all previous linear PTQ quantizers including percentile [18], bit-split [32], and Liu *et al*. [23].

Table 6. **Applying NoisyQuant on different layer types.** Starting from W6A6 PTQ with the EasyQuant quantizer, we add NoisyQuant to each type of layers in the transformer model.

Model	qkv noise	proj noise	fc1 noise	fc2 noise	Top-1 W6A6
DeiT-S [30]	✗	✗	✗	✗	75.27
	✓	✗	✗	✗	75.38
	✗	✓	✗	✗	75.45
	✗	✗	✓	✗	75.33
	✗	✗	✗	✓	76.21
	✓	✓	✓	✓	**76.37**
Swin-T [22]	✗	✗	✗	✗	79.51
	✓	✗	✗	✗	79.53
	✗	✓	✗	✗	79.56
	✗	✗	✓	✗	79.52
	✗	✗	✗	✓	79.80
	✓	✓	✓	✓	**80.01**

5.3. NoisyQuant's impact on different layer types

As we show the effectiveness of NoisyQuant in improving the PTQ performance of the entire model, here we explore if NoisyQuant is helpful for all types of layers in the vision transformer model. Tab. 6 exhibits the effect of applying NoisyQuant on different types of layers. The checkmark in the table means that we apply NoisyQuant to the input activation of all the layers of that particular type, otherwise the layer is quantized with EasyQuant only. For all layer types, applying NoisyQuant consistently improves the PTQ performance of the entire model. Specifically, 'fc2' layers derive the greatest benefit from NoisyQuant, which corresponds to our previous analysis that the activation distribution after GELU function brings the major challenge to PTQ, which can be resolved with the proposed NoisyQuant method. After adding NoisyQuant to the input activation of all linear layers, the model achieves the maximum performance boost.

6. Conclusions

This work proposes NoisyQuant, a noisy bias-enhanced post-training activation quantization method for the complicated activation distribution of vision transformer models. We theoretically prove the quantization error reduction brought by adding noisy bias to the input activation before quantization, and empirically show the effectiveness of NoisyQuant on both linear and nonlinear quantizers on vision transformer models. We hope this work opens up a new direction to reduce PTQ quantization error through actively changing the distribution being quantized, and inspires better Noisy Bias distribution design or generation method based on quantizer and activation characteristics.

Acknowledgement This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4400900. The Berkeley team acknowledges support from Berkeley Deep Drive, Intel Corporation, and Panasonic.

References

[1] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2978–2985, 2020. 1

[2] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13166–13175, 2020. 1, 3

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 1, 3

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *ArXiv*, abs/2005.12872, 2020. 6, 7, 8

[5] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12270–12280, 2021. 1

[6] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021. 1

[7] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 1

[8] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020. 1

[9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 293–302, 2019. 1

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 3, 6, 7, 11

[11] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021. 1

[12] Cong Guo, Yuxian Qiu, Jingwen Leng, Xiaotian Gao, Chen Zhang, Yunxin Liu, Fan Yang, Yuhao Zhu, and Minyi Guo. Squant: On-the-fly data-free quantization via diagonal hessian approximation. *arXiv preprint arXiv:2202.07471*, 2022. 3

[13] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1

[14] Ali Hatamizadeh, Hongxu Yin, Jan Kautz, and Pavlo Molchanov. Global context vision transformers. *arXiv preprint arXiv:2206.09959*, 2022. 3

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 2, 3, 11

[16] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *ISSCC*, 2014. 3, 11

[17] Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, and Ce Liu. Vitgan: Training gans with vision transformers. *arXiv preprint arXiv:2107.04589*, 2021. 3

[18] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2805–2814, 2019. 7, 8

[19] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. *arXiv preprint arXiv:2212.08254*, 2022. 3, 11

[20] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6

[21] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Post-training quantization for fully quantized vision transformer. In *IJCAI*, 2022. 7

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021. 1, 3, 6, 7, 8, 11

[23] Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. In *NeurIPS*, 2021. 1, 2, 3, 4, 7, 8, 11

[24] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103, 2021. 1

[25] A. Polino, R. Pascanu, and D. Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018. 1

[26] Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. Fully quantized transformer for improved translation. *ArXiv*, abs/1910.10485, 2019. 7

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 3

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 6, 11

[29] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821, 2020. 1

[30] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herv'e J'egou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 3, 6, 7, 8, 11

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 3

[32] Peisong Wang, Qiang Chen, Xiangyu He, and Jian Cheng. Towards accurate post-training network quantization via bit-split and stitching. In *ICML*, 2020. 1, 7, 8

[33] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 3

[34] Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. *arXiv preprint arXiv:2209.13325*, 2022. 3, 11

[35] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6, 7

[36] Di Wu, Qi Tang, Yongle Zhao, Ming Zhang, Ying Fu, and Debing Zhang. Easyquant: Post-training quantization via scale optimization. *arXiv preprint arXiv:2006.16669*, 2020. 1, 2, 3, 6, 7, 8, 11

[37] Lirui Xiao, Huanrui Yang, Zhen Dong, Kurt Keutzer, Li Du, and Shanghang Zhang. Csq: Growing mixed-precision quantization scheme with bi-level continuous sparsification. *arXiv preprint arXiv:2212.02770*, 2022. 1

[38] Huanrui Yang, Lin Duan, Yiran Chen, and Hai Li. Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization. *arXiv preprint arXiv:2102.10462*, 2021. 1

[39] Huanrui Yang, Hongxu Yin, Pavlo Molchanov, Hai Li, and Jan Kautz. Nvit: Vision transformer compression and parameter redistribution. *arXiv preprint arXiv:2110.04869*, 2021. 1

[40] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021. 3

[41] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 3

[42] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization framework for vision transformers. *arXiv preprint arXiv:2111.12293*, 2021. 1, 2, 4

[43] Zhihang Yuan, Chenhao Xue, Yi-Qiang Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In *ECCV*, 2022. 2, 3, 7, 8

[44] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018. 1

[45] Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. Ternarybert: Distillation-aware ultra-low bit bert. In *EMNLP*, 2020. 7

[46] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 3

[47] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 1

This document provides additional visualizations and experimental results to support the main paper. We demonstrate results of quantization error on model output in Appendix A, visualize prediction scores for ImageNet classes in Appendix B, illustrate more histogram examples of the input and output activation distributions on different transformer layers in Appendix C, discuss memory and computation overhead in Appendix D, and show additional experimental results in Appendix E.

A. Quantization error of model output

In this section, we show the comparison of the output logits between EasyQuant [36] and NoisyQuant with 6-bit ViT [10], DeiT [30] and Swin [22] models. Here NoisyQuant is implemented on top of EasyQuant with the proposed noisy bias enhancement. We go through the whole ImageNet validation set and calculate the mean-square error of model output logits on each quantized model compared to the pretrained floating-point counterpart. As shown in Tab. 7, NoisyQuant achieves quantization error reduction on all model outputs, especially for ViT-S (17%) and Swin-T (16%) models.

Table 7. **Quantization error of model output.** Models are quantized by EasyQuant and NoisyQuant with the W6A6 setting.

Model	EasyQuant [36]	NoisyQuant	Reduction
ViT-S	1.0400	0.8583	0.1818 **(17%)**
ViT-B	0.6365	0.5982	0.0383 **(6%)**
ViT-B*	0.6956	0.6360	0.0596 **(9%)**
DeiT-S	0.3270	0.2934	0.0335 **(10%)**
DeitT-B	0.2869	0.2584	0.0284 **(10%)**
DeiT-B*	0.1984	0.1760	0.0224 **(11%)**
Swin-T	0.0913	0.0765	0.0148 **(16%)**
Swin-S	0.0296	0.0289	0.0007 **(2%)**
Swin-B	0.0505	0.0457	0.0047 **(9%)**
Swin-B*	0.0412	0.0399	0.0013 **(3%)**

B. Visualization of model output

Following Appendix A, we visualize model output in Fig. 6 to give further perspectives on how the reduced quantization error achieved by NoisyQuant improved final accuracy. Specifically, we plot prediction logits produced by the floating-point (red), EasyQuant (gray), and NoisyQuant (green) models on the 1000 ImageNet [28] classes, respectively. The highest logits are marked with stars, where the location of the red star corresponds to the ground truth class. With less quantization error, NoisyQuant logits closely match that of the floating-point model, thus achieving better performance than EasyQuant.

Table 8. Comparing to reparameterization.

Model	W/A	Reparam.	**NoisyQuant**
ViT-S	6/6	76.66	**78.90** $\pm$ 0.06
DeiT-B	6/6	81.03	**81.26** $\pm$ 0.04
Swin-S	6/6	82.46	**82.83** $\pm$ 0.04

C. Additional input and output activation histogram

In this section, we present more histogram examples of layer input and output as previously described in Sec. 4.2 of the main paper. We briefly illustrate the pipeline of EasyQuant and NoisyQuant in Fig. 7. The top-left sub-figure refers to input activation X, and EasyQuant follows the gray arrow while NoisyQuant follows the blue. NoisyQuant utilizes the proper-selected noisy bias N to refine the input before quantization (shown in the bottom-left sub-figure). The output histograms are shown in the right sub-figures, and we point out the mismatch caused by EasyQuant with the orange arrow.

As we have emphasized in the main paper, transformer layers produce sophisticated activation distributions. Fig. 7 gives more examples from different transformer layers. Fig. 7a and Fig. 7b show fc2 layers in ViT-S and DeiT-S which takes GELU [15] activations as input; the asymmetric and heavy-tailed input activation distribution makes a negative impact on the layer output produced by EasyQuant. Instead, NoisyQuant refines the distribution to achieve a better match in the quantized layer output. Fig. 7c gives an example of the downsample layer in Swin models which as well enjoys the noisy bias enhancement.

D. Memory and computation overhead

Memory overhead. In practice, for weights $W \in \mathbb{R}^{k\times m}$ and activations $X \in \mathbb{R}^{m\times n}$, we follow the standard implementation to set bias $B \in \mathbb{R}^{k\times 1}$ and sample noise $N \in \mathbb{R}^{m\times 1}$, so the denoising bias $B' = B - q_W(W)N$ is also $\mathbb{R}^{k\times 1}$, where $q_W(\cdot)$ is the quantizer. The sum follows the broadcasting rule. Storing N brings minimal overhead, for instance, DeiT-B* has 86.9M params, with only 0.06M (0.07%) for storing the noise.

Computation overhead. The matrix multiplication, i.e., WX, dominates the computation of ViT linear layers, requiring $10^3\times$ more MAC than the number of adds in $X + N$ and bias. So the cost of FP32 add is negligible (<0.4%) to that of INT8 layer. Further, N and B' can be INT16 rather than FP32, enabling integer-only inference and reducing the cost of add to <0.03%. We observe no accuracy differences in using INT16 or FP32 for N and B' in our experiments. We estimate the energy cost with 0.23pJ/Int8-MAC, 0.9pJ/FP32-Add, and 0.05pJ/Int16-Add following [16].

E. Additional experiments

Ablation study on calibration size. We follow [23]'s setting for calibration size 1024. Further experiments show that calibration size as low as 32 can still produce similar performance (see Tab. 9).

Additional baselines. Concurrent works [19, 34] introduce the reparameterization approach which reparameterizes LN layer to suppress outliers by scaling down activation values. NoisyQuant is orthogonal as we actively change the activation distribution being quantized without scaling. So NoisyQuant can be plugged in after reparameterization. We reproduce the reparameterization used in the two works and subsequently add NoisyQuant to show consistent improvement in Tab. 8.

Additional models. Beyond ViT, on ResMLP-24 with W6A6, NoisyQuant (76.71%) beat EasyQuant (76.48%) by 0.23%.

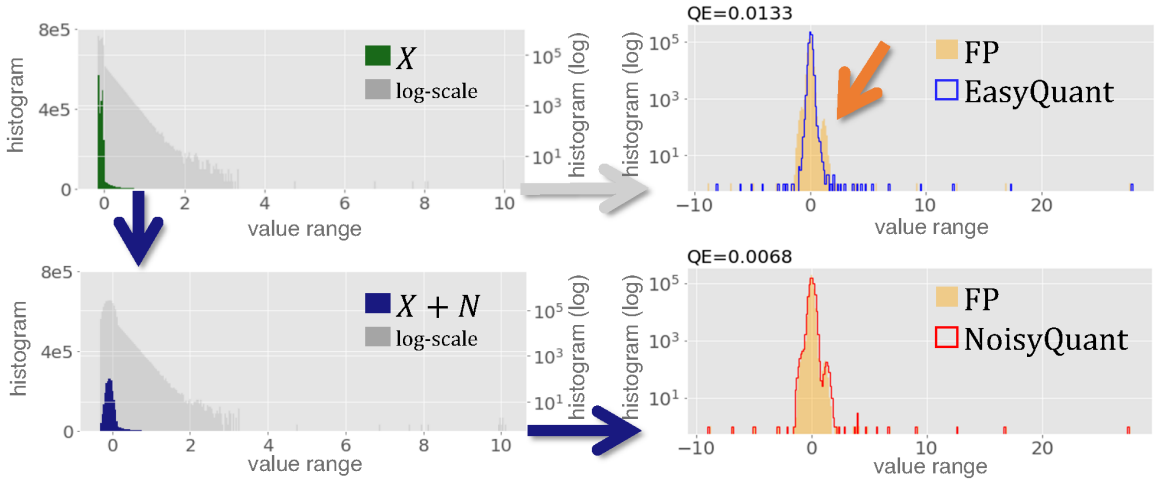
(a) Ground truth = 597

(b) Ground truth = 38

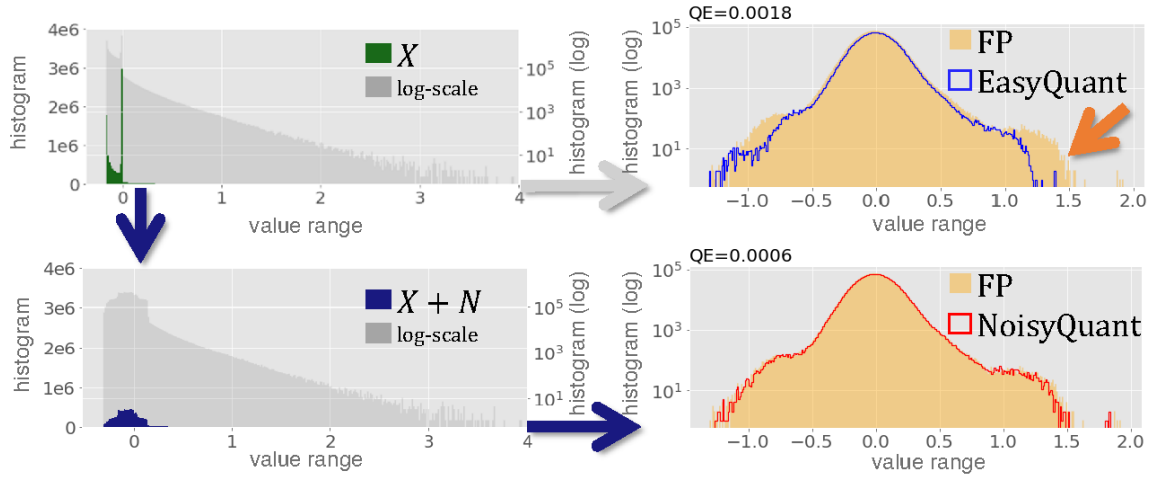
Figure 6. Model output of the floating-point (red), EasyQuant (green), and NoisyQuant (gray) model. The floating-point and NoisyQuant models give correct predictions (red/green star) while EasyQuant gives wrong prediction (gray star).

Table 9. Performance with smaller calibration set.

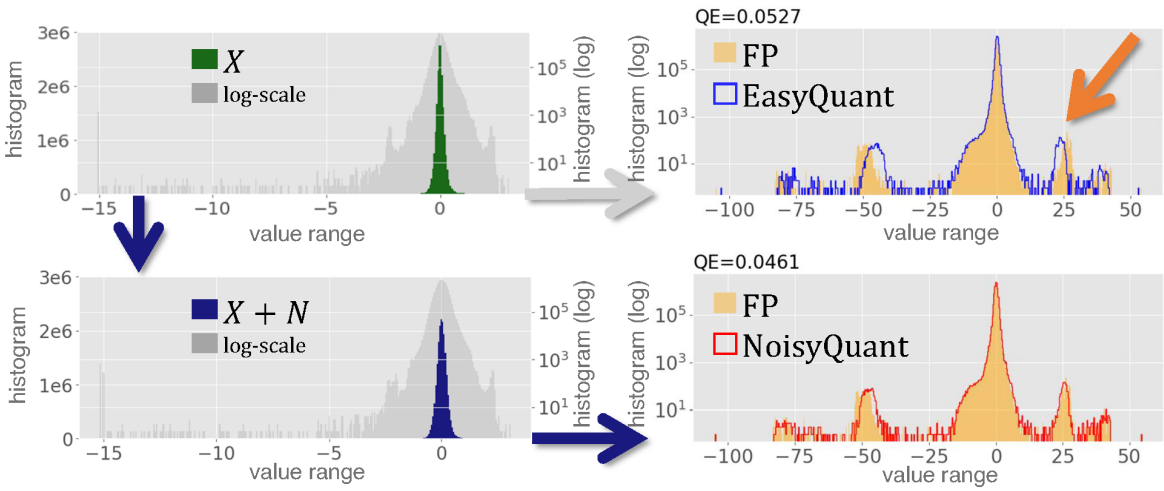
Size	W/A	ViT-S	ViT-B	ViT-B*	DeiT-S	DeiT-B	DeiT-B*	Swin-T	Swin-S	Swin-B	Swin-B*
32	6/6	76.81	81.89	82.81	76.30	79.71	81.19	79.97	82.74	84.55	85.90
128	6/6	76.87	81.97	82.86	76.47	80.20	81.24	80.13	82.68	84.44	86.00
1024	6/6	76.86	81.90	83.00	76.37	79.77	81.40	80.01	82.78	84.57	85.90



(a) fc2 layer in ViT-S



(b) fc2 layer in DeiT-S



(c) downsample layer in Swin-B

Figure 7. Input (left) and output (right) histogram on different layers.