

BLOB-Q: BOOSTING LOW BIT ViT QUANTIZATION VIA GLOBAL OPTIMIZATION ON MODEL DISTORTION

Anonymous authors

Paper under double-blind review

ABSTRACT

In this paper, we present a novel Mixed-Precision Post Training Quantization (PTQ) approach for Vision Transformers (ViTs). Different with prior works which typically optimize the output error of current layer (layer distortion), when performing quantization, our approach directly minimizes the output error of the last layer of the model (model distortion). As model distortion is highly related to accuracy, our approach can maximally maintain the accuracy even when quantized to low bit widths. We formulate the quantization of ViTs as a model distortion optimization problem, given the constraint of size. By solving the optimization problem, the optimal bit allocation across layers, i.e., the optimal bit width of each layer, can be obtained, with minimized model distortion. Directly solving the optimization problem is an NP-hard problem. We propose to adopt the second-order term of the Taylor series expansion to approximate model distortion, where an important additivity property can be derived under the approximation. Utilizing the second-order additivity property, the optimization problem can be decomposed into sub-problems and solved efficiently in an iterative manner. Specifically, we propose a dynamic programming algorithm to solve the optimization problem and efficiently find the globally optimal solution with only linear time complexity. Extensive experiments on six ViT models demonstrate the effectiveness of our approach. Results show that our approach significantly improves state-of-the-art and can further reduce the size of ViT models to 4 bits to 6 bits without hurting accuracy.

1 INTRODUCTION

Inspired by the success of the Transformer architecture in Natural Language Processing (NLP) Vaswani et al. (2017); Conneau et al. (2020); Devlin et al. (2019); Liu et al. (2019); Radford et al. (2018), Vision Transformer (ViT) Dosovitskiy et al. (2021); Touvron et al. (2021); Liu et al. (2021a) has become the mainstream architecture in Computer Vision and achieved state-of-the-art results in various vision tasks, from classification Krizhevsky et al. (2012), segmentation He et al. (2017) and style transfer Gatys et al. (2016); Johnson et al. (2016) of images to the synthesis of images Karras et al. (2018). However, the high accuracy of ViT models comes at the cost of high computational complexity Xu et al. (2024); Geng et al. (2024). ViT models typically require tens of millions or even more parameters to obtain a powerful performance. For example, DeiT Touvron et al. (2021) contains about 340 millions of parameters with 12 Transformer blocks. Such a large volume of parameters and its associated memory cost make it very difficult to deploy ViT models on resource-limited mobile devices (e.g., phones, drones, watches, automatic cars, etc).

Main Challenges. Quantization Han et al. (2015); Zhu et al. (2017); Zhang et al. (2018); Hubara et al. (2016); Zhou et al. (2017; 2016); Rastegari et al. (2016) is an effective approach to compress deep neural networks. In this paper, we focus on Post-Training Quantization (PTQ) for ViT models. PTQ does not require retraining or fine-tuning the model after quantization and is more efficient and feasible, since retraining itself is time-consuming, and even impossible in some cases when the training dataset is not available. What’s more, quantization with mixed-precision can benefit the accuracy. Because parameters in different layers react differently to quantization, instead of using equal bit width to quantize all layers, allocating different bit widths across the layers based on their sensitivity to quantization is more reasonable and can lead to higher precision. To find the optimal bit allocation of layers, one challenge is that search space of the hyperparameters increases exponentially with the number of layers. Given L layers and B bit width options, the search space is $O(B^L)$, where

in a deep neural network, L can be tens or hundreds of layers. Such a huge search space could make it extremely time-consuming for a heuristic search method, such as reinforcement learning Sutton & Barto (2018), to find a solution.

Existing Problems. Prior PTQ works for ViT models have several drawbacks. First, most existing works only consider layer distortion, i.e., the output error of current layer, when deciding the bit width and other quantization configurations of a layer. Because output error accumulates across layers, layer distortion can not reflect the real impact on the model. Small layer distortion may lead to a large output error in the last layer of the model. Using layer distortion as an indicator to make the decision of quantization configurations is not reasonable. Second, prior works typically use heuristic search methods (e.g., reinforcement learning) to find the bit allocation of layers. The heuristic search methods have high time complexity, require considerable computing resources, and may fall into a local solution in most cases. There are some other works using gradient descent methods or analytical solutions under the assumptions of the distributions of parameters. Again, these gradient descent methods can not guarantee a global-optimal solution and usually fall into a local solution. The analytical solutions require that parameters obey specific distributions (e.g., Laplacian distribution), which may not hold in practice.

Our Contributions. We propose a novel Mixed-Precision PTQ approach for ViTs. Our approach directly minimizes model distortion, i.e., the output error of the last layer, to maximally maintain the accuracy of the model. By solving the minimization problem, our approach can obtain optimal bit allocation of the layers with minimal model distortion. As the model distortion minimization problem is an NP-hard problem, we thus adopt the second-order term of the Taylor series expansion to approximate model distortion. We derive an important additivity property of model distortion under second-order approximation, which is that the model distortion caused by the quantization of all layers equals to the sum of all model distortion due to the quantization of individual layers. Utilizing this additivity property, the whole problem can be decomposed into sub-problems. We then develop an efficient dynamic programming algorithm to find the global-optimal solution of the problem with only linear time complexity. To the best of our knowledge, this is the first work that proposes a method to find the global-optimal solution for mixed-precision quantization. We summarize our contributions as follows:

- We propose a new Mixed-Precision PTQ approach for ViTs, which directly minimizes model distortion caused by quantization. Our approach finds the optimal bit allocation of layers with minimized model distortion and can well maintain the accuracy even when quantized to very low bit widths.
- We propose an efficient method to solve the model distortion minimization problem by adopting second-order approximation and utilizing the additivity property. An ultra-fast dynamic programming algorithm is developed to find the global-optimal solution with only linear time complexity.
- Our approach significantly outperforms state-of-the-art. For the first time, at 6 bits, we report the PTQ results on ViTs without hurting accuracy (loss < 1%). At 4 bits, our approach noticeably improves the PTQ results up to 11.49% compared with prior arts.

2 RELATED WORKS

Post-Training Quantization (PTQ). Model quantization can be generally categorized into Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ), where the latter PTQ performs quantization for a trained model. Recently, PTQ is more preferred as it is more lightweight without requiring to intensive training-from-scratch. Retrain-free Nagel et al. (2020a); Li et al. (2020); Zhao et al. (2019); Frantar & Alistarh (2022) approaches have become ones of desired traits of scalable PTQ for modern models. Plenty of PTQ techniques have been explored for ViTs Liu et al. (2021b); Li et al. (2023); Liu et al. (2023a); Lin et al. (2022); Di Wu (2020); Ding et al. (2022); Li & Gu (2023), mostly at 8 bits. Few others Yuan et al. (2021) explored low bit PTQ down to 4 bits with noticeable accuracy drops. We achieve 4-bits PTQ with vastly shrinked accuracy gaps, and 6-bits PTQ with full-precision level accuracies.

Mixed-Precision Quantization (MPQ). Adapting to different sensitivity to quantization on different layers of the model, MPQ aims to allocate varying bit widths within the model (commonly in layerwise granularity). Reinforcement-Learning (RL) based approaches like HAQ Wang et al. (2019)

and AutoQ Lou et al. (2020) pioneer MPQ towards searching optimal bit allocation, but can be costly for PTQ. Recent reconstruction based method OBQ Frantar & Alistarh (2022) are compatible with mixed-precision quantization, but are limited to layerwise optimization with greedy solver.

Model Distortion Minimization. To make the quantization strategy aware of task loss, model-level (Loss-aware) optimization approaches like AdaRound Nagel et al. (2020a) and BRECQ Li et al. (2020) are formulated on model-level loss to layerwisely (or blockwisely) reconstruct the quantization rounding. LAPQ Nahshan et al. (2021) and PTQ4ViT Yuan et al. (2021) also construct model-level objectives to search for optimal quantization parameters, but PTQ4ViT ends up local solutions, while LAPQ relies on a complicated multi-stage optimization procedure. Fig. 1 illustrates a comparison with our most related works.

Feature	HAQ	LAPQ	OBQ	BRECQ	HAWQ	PTQ4ViT	BLOB-Q
Model-level Optimization	✓	✓	✗	✗	✗	✗	✓
Global-Optimal Solver	✗	✗	✗	✗	✗	✗	✓
Mixed-Precision	✓	✗	✗	✗	✓	✗	✓
Second-order Method	✗	✓	✓	✓	✓	✓	✓

Figure 1: Comparisons of different PTQ techniques.

3 PRELIMINARIES AND PROBLEM STATEMENTS

Following prior works on Mixed-Precision Quantization (MPQ), *e.g.*, Wang et al. (2019); Dong et al. (2019); Liu et al. (2021b); Chen et al. (2023), we explore the MPQ problem for ViTs. Compared to Uniform-Precision Quantization (UPQ) which uses the same bit width throughout the model, MPQ works have shown better accuracy via unevenly allocating bit widths in the model. The general goal of MPQ is to solve the best quantization configuration, *a.k.a.* searching a set of bit widths $B_l^W \in \mathbb{R}^L$ and $B_l^A \in \mathbb{R}^L$ for all L layer weights and activations respectively in a model.

Layer Distortion Minimization. Previously, most PTQ approaches Nagel et al. (2020b); Li et al. (2020); Dong et al. (2019); Liu et al. (2021b); Chen et al. (2023) avoid optimizing on a model-level objective as it is NP-hard for non-linear DNNs. They generally can be summarized into the following framework, that individually solves a series of layer-level problems capturing the local impact of quantization on current layer output, under a generic compression constraint $\mathcal{R}(\widehat{\mathbf{W}}) \leq R$ that the total size $\mathcal{R}$ of the quantized model parametrized by all layer weights $\widehat{\mathbf{W}} = \{\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \dots, \widehat{\mathbf{W}}_L\}$ is under a budget size R :

$$\arg \min_{\widehat{\mathbf{W}}_l} \sum_l \Gamma(\mathbf{O}_l, \widehat{\mathbf{O}}_l), \quad \text{s.t.} \quad \mathcal{R}(\widehat{\mathbf{W}}) \leq R. \quad (1)$$

Here $\mathbf{O}_l$ denotes the output of the layer and $\widehat{\mathbf{O}}_l$ denotes the output of the quantized layer. Such layer distortion minimization framework works well for previous Uniform-Precision Quantization (UPQ) works Nagel et al. (2020b); Li et al. (2020) that reconstruct the quantized weight locally (layerwisely or blockwisely) via adjusting rounding parameters in the specific form of $\arg \min_{\widehat{\mathbf{W}}_l} \mathbb{E}_X(\|\mathbf{O}_l, \widehat{\mathbf{O}}_l\|_F^2)$. Most recent Mixed-Precision Quantization (MPQ) attempts Dong et al. (2019); Liu et al. (2021b); Chen et al. (2023) also deduct from this layer distortion minimization to search for optimal bit allocation resulting to ranking sensitivity by *e.g.*, nuclear forms Banner et al. (2018) and Hessian terms Dong et al. (2019).

Model Distortion Minimization. Despite most previous PTQ works circumvent the intimidating global optimization and resort to exploring local effect of quantization for efficient solution, in this paper, we aim to tackle the more difficult model-level optimization. Since the effect of quantization on model distortion directly links to the eventual model accuracy, theoretically it leads to a more accurate Mixed-Precision Quantization (MPQ) solution. Formally, model distortion minimization refers to the following framework:

$$\arg \min_{\widehat{\mathbf{W}}_l, \widehat{\mathbf{A}}_l} \Gamma(\mathbf{O}, \widehat{\mathbf{O}}), \quad \text{s.t.} \quad \mathcal{R}(\mathbf{W}, \mathbf{A}) \leq R, \quad (2)$$

where it directly optimizes the best quantization that leads to minimal model distortion in the output of the last layer $\mathbf{O} = \mathbf{O}_L$. Unlike many MPQ works which only search for weight quantization configurations, we include both weights and activations $\widehat{\mathbf{W}}_l, \widehat{\mathbf{A}}_l$ in the MPQ optimization. Similar with most prior works, we define distortion Γ as squared L2-distance: $\Gamma(\mathbf{O}, \widehat{\mathbf{O}}) = \mathbb{E}_X(\|\mathbf{O} - \widehat{\mathbf{O}}\|_F^2)$.

The obvious challenge to tackle the above model-level optimization framework practically is the non-trivial task to find out close form relationship between perturbation on layer weights and activations of non-linear models. Reinforcement Learning (RL) based method like HAQ Wang et al. (2019) can get close-to-optimal solution for smaller CNNs, but can be extremely time-consuming to search a solution especially when model sizes scale up in modern models like ViTs. Layerwise pruning methods with a model-level objective like LAMP Lee et al. (2020) relies on greedy solver with no guarantee on global-optimum.

3.1 SECOND-ORDER APPROXIMATION AND ADDITIVITY PROPERTY

Recent works on model pruning Kurtic et al. (2022); Yang et al. (2023) indicate that second-order methods are essential for accurate layerwise pruning. This encourages us to explore the possibility to adopt a second-order approach for a quantization problem instead and potentially enhance the performance of Mixed-Precision Quantization. Recalling that the formulation of second-order Taylor expansion w.r.t. model output with small perturbation $\Delta \mathbf{W}_l$ on layer weight $\mathbf{W}_l$ is $\hat{\mathbf{O}} = \mathbf{O} + \mathbf{J}^\top(\mathbf{W}_l)\Delta \mathbf{W}_l + \frac{1}{2}\Delta \mathbf{W}_l^\top \mathbf{H}_l^w \Delta \mathbf{W}_l$, we start from applying second-order Taylor expansion at the model convergence point for both weight and activation quantization¹:

$$\Gamma_{\hat{\mathbf{W}}_l}(\mathbf{O}, \hat{\mathbf{O}}) = \mathbb{E}_X \left(\left\| \frac{1}{2} \Delta \mathbf{W}_l^\top \mathbf{H}_l^w \Delta \mathbf{W}_l \right\|^2 \right), \Gamma_{\hat{\mathbf{A}}_l}(\mathbf{O}, \hat{\mathbf{O}}) = \mathbb{E}_X \left(\left\| \frac{1}{2} \Delta \mathbf{A}_l^\top \mathbf{H}_l^a \Delta \mathbf{A}_l \right\|^2 \right), \quad (3)$$

where $\Gamma_{\hat{\mathbf{W}}_l}(\mathbf{O}, \hat{\mathbf{O}}), \Gamma_{\hat{\mathbf{A}}_l}(\mathbf{O}, \hat{\mathbf{O}})$ are the L2-distortion $\Gamma(\mathbf{O}, \hat{\mathbf{O}})$ with the quantized $\hat{\mathbf{W}}_l, \hat{\mathbf{A}}_l$ respectively, and $\Delta \mathbf{W}_l$ and $\Delta \mathbf{A}_l$ are the perturbation on layer weights and activations respectively by quantization (e.g. $\Delta \mathbf{W}_l = \mathbf{W}_l - \hat{\mathbf{W}}_l$), $\mathbf{H}_l^w, \mathbf{H}_l^a$ are the hessian matrices on $\mathbf{W}_l, \mathbf{A}_l$ respectively w.r.t. model output. Based on second-order Taylor approximation, we discover the following property when doing arbitrary quantization:

Property 1 (Second-Order Additivity). *The expected model distortion $\Gamma(\mathbf{O}, \hat{\mathbf{O}})$ between the output of a pretrained model $\mathbf{O}$ and a quantized one $\hat{\mathbf{O}}$ on dataset X can be approximated into the sum of model distortion due to the quantization of individual layers:*

$$\Gamma(\mathbf{O}, \hat{\mathbf{O}}) = \mathbb{E}_X(\|\mathbf{O} - \hat{\mathbf{O}}\|_F^2) \approx \sum_l \Gamma_{\hat{\mathbf{W}}_l}(\mathbf{O}, \hat{\mathbf{O}}) + \sum_l \Gamma_{\hat{\mathbf{A}}_l}(\mathbf{O}, \hat{\mathbf{O}}), \quad (4)$$

We provide a mathematical derivation for the additivity property in Appendix A and empirical evidences in Appendix A.2. This property also tells us that instead of optimizing the original model distortion which is NP-hard, we can instead solve an integer programming problem on layerwise contributions to the model distortion as in below Corollary.

Corollary 1 (Second-Order Model Distortion Optimization). *Given Eq. 4, the model distortion minimization problem in Eq. 2 can be turned into additive global objective:*

$$\arg \min_{\hat{\mathbf{W}}_l, \hat{\mathbf{A}}_l} \sum_l \Gamma_{\hat{\mathbf{W}}_l}(\mathbf{O}, \hat{\mathbf{O}}) + \sum_l \Gamma_{\hat{\mathbf{A}}_l}(\mathbf{O}, \hat{\mathbf{O}}), \quad \text{s.t.} \quad \mathcal{R}(\hat{\mathbf{W}}, \mathbf{A}) \leq R. \quad (5)$$

Since bit-widths B_l^W, B_l^A of single layer only control the quantization of single layer weights $\hat{\mathbf{W}}_l$ and activations $\hat{\mathbf{A}}_l$, they also only affect $\Gamma_{\hat{\mathbf{W}}_l}$ and $\Gamma_{\hat{\mathbf{A}}_l}$ of current layer. Furthermore, as the size constraint $\mathcal{R}$ can be instantiated as the linear function of bit-widths i.e., $\mathcal{R}(\mathbf{W}, \mathbf{A}) = \sum_l B_l^W \|\mathbf{W}_l\|_0 + B_l^A \|\mathbf{A}_l\|_0$. The model distortion minimization problem in Eq. 5 is equivalent to a integer programming problem:

$$\arg \min_{B_l^W, B_l^A} \sum_l \Gamma_{\hat{\mathbf{W}}_l}(\mathbf{O}, \hat{\mathbf{O}}) + \sum_l \Gamma_{\hat{\mathbf{A}}_l}(\mathbf{O}, \hat{\mathbf{O}}), \quad \text{s.t.} \quad \sum_l B_l^W \|\mathbf{W}_l\|_0 + B_l^A \|\mathbf{A}_l\|_0 \leq R, \quad (6)$$

with only the concrete values of $\Gamma_{\hat{\mathbf{W}}_l}$ and $\Gamma_{\hat{\mathbf{A}}_l}$ under different bit widths needed to be obtained via calibration, detailed in Section. 4. Supported by the additivity property in Eq. 4, we can guarantee to attain the global-optimal of the Model-level objective in Eq. 2 by solving the integer programming problem. We eventually devise a non-greedy solver for problem Eq. 6 detailed in Section. 5.

¹Prior works indicate first-order term can be regarded vanished on converged model

4 CALIBRATION FOR SECOND-ORDER MODEL DISTORTION

There are still remaining major challenges to leverage Property 1 for quantization practically for empirical ViTs. For $\mathbf{H}^w$, some existing second-order *pruning* methods Kurtic et al. (2022); Yang et al. (2023) pointed out some on possible ways for us. For $\mathbf{H}^a$, some second-order PTQ methods like BRECQ Li et al. (2020) involve Hessian on activations $\mathbf{H}^a$ but are designed for guiding weight quantization. Moreover, there was no attempt to seamlessly weld different Hessian approximation schemes (e.g. gradient Yang et al. (2023), empirical Fisher Kurtic et al. (2022), and diagonal gradient product Li et al. (2020)) for unified weight and activation MPQ search.

In this work, we unify the second-order approximation for Mixed-Precision Quantization (MPQ) search of both weights and activations. We start by selecting empirical Fisher Kurtic et al. (2022) scheme for approximating Hessian for both weight $\mathbf{H}^w$ and activation $\mathbf{H}^a$, to avoid potential mismatch of using different schemes in the unified allocation problem in Eq. 4:

Definition 1 (Empirical Fisher Kurtic et al. (2022)). *The Hessian $\mathbf{H}^w \in \mathbb{R}^{d \times d}$ of weight $\mathbf{W} \in \mathbb{R}^{d_{row} \times d_{col}}$ ($d = d_{row}d_{col}$) can be estimated on calibration dataset X_{cal} of N samples given the Jacobian matrix $\mathbf{J}_n(\mathbf{W})$ of the weight matrix on n -th sample as $\tilde{\mathbf{H}}^w = \kappa \mathbf{I}_d + \frac{1}{N} \sum_n \mathbf{J}_n(\mathbf{W}) \mathbf{J}_n^\top(\mathbf{W})$. κ is predefined hyperparameter.*

Empirical Fisher was originally used to approximate Hessian of layer weights in pruning. We attempt to similarly approximate Hessian of layer activations in this work as $\tilde{\mathbf{H}}^a = \kappa \mathbf{I}_d + \frac{1}{N} \sum_n \mathbf{J}_n(\mathbf{A}) \mathbf{J}_n^\top(\mathbf{A})$. However, even with such Hessian estimation, it is obviously still too expensive for processing real models to store large chunks of Hessian matrices for all layers. To solve global objective in Eq. 5, we need to sample all values of $\Gamma_{\hat{\mathbf{W}}_l}$ and $\Gamma_{\hat{\mathbf{A}}_l}$ of L layers under all possible quantized $\hat{\mathbf{W}}_l$ and $\hat{\mathbf{A}}_l$ during calibration. Therefore, we need $\Theta(\sum_l d_l^2)$ memory space to accommodate all layers Hessian matrices alone, which is near quadratic to the model size. In fact, there are still plenty of rooms for overhead and memory reductions, as we show that the expensive time and spatial costs can be significantly reduced in the followings.

Second-Order Approximation for Weights. For approximating distortion caused by weight quantization $\Gamma_{\hat{\mathbf{W}}_l}$, to obtain Hessian matrix based on empirical Fisher, we first consider using the approximated Hessian (Definition 1) to convert the second-order term inside L2 norm of Eq. 4 into

$$\frac{1}{2} \Delta \mathbf{W}^\top \left(\kappa \mathbf{I}_d + \frac{1}{N} \sum_n \mathbf{J}_n(\mathbf{W}) \mathbf{J}_n^\top(\mathbf{W}) \right) \Delta \mathbf{W} = \frac{\kappa}{2} \|\Delta \mathbf{W}\|_2^2 + \frac{1}{2N} \sum_n \|\mathbf{J}_n^\top(\mathbf{W}) \Delta \mathbf{W}\|_2^2, \quad (7)$$

where we notice it no longer needs to store the full Hessian and instead mainly the inner product between $\mathbf{J}_n(\mathbf{W})$, $\Delta \mathbf{W} \in \mathbb{R}^d$ which is a scalar value. The asymptotic time and spatial complexity of this form is then $\Theta((2N+2)d+N)$ and $\Theta(2d+N)$ respectively, which only has linear complexity w.r.t. the parameter size d . This form is still space for further improvement to parallelize the $\sum_n \|\mathbf{J}_n^\top(\mathbf{W}) \Delta \mathbf{W}\|_2^2$ for all N samples to trade for more time efficiency with parallel computing on GPUs. This can be done by concatenating N samples of Jacobians into $\mathbf{J}_N(\mathbf{W}) \in \mathbb{R}^{N \times d}$ and obtain a N -dimensional variable $\mathbf{P}^w = \mathbf{J}_N^\top(\mathbf{W}) \Delta \mathbf{W} \in \mathbb{R}^N$. Then the time cost can be slightly reduced to $\Theta((N+2)d+2N)$ by calculating $\frac{1}{2N} \text{trace}(\mathbf{P}^w \mathbf{P}^{w\top})$ instead, with slightly more space $\Theta(2d+N^2)$ to store the intermediate vector $\mathbf{P}^w$. Finally, since the above empirical Fisher estimation already encapsulates an averaging process on calibration data, everything inside expectation of $\Gamma_{\hat{\mathbf{W}}_l}$ (see Eq. 4) is already invariant to the data samples. Therefore the final expectation calculation step is unnecessary for weight quantization. Hence, the final second-order model distortion for weight quantization can be summarized as:

$$\Gamma_{\hat{\mathbf{W}}_l}(\mathbf{O}, \hat{\mathbf{O}}) \approx \frac{\kappa}{2} \|\Delta \mathbf{W}_l\|_2^2 + \frac{1}{2N} \text{trace}(\mathbf{P}_l^w \mathbf{P}_l^{w\top}). \quad (8)$$

Empirically, we observe the overhead of second-order calibration ranges from merely around 3 to 18 minutes between the smallest and the largest ViTs (ViT-S/224 and ViT-B/384) on a 4-L40 GPU server. We will give detailed efficiency analysis in the experimental section.

Second-Order Approximation for Activations. To scalably approximate the second-order model distortion for activation $\Gamma_{\hat{\mathbf{A}}_l}$, most ideas from above $\Gamma_{\hat{\mathbf{W}}_l}$ work with an exception. Since the activation map changes for different data samples, we need to keep N copies of activations $\mathbf{A}^N =$

$\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \dots, \mathbf{A}^{(N)}\}$, which raises another question how to similarly get $\mathbf{P}^a$ from $\mathbf{A}^N \in \mathbb{R}^{N \times dT}$ and $\mathbf{J}_N \in \mathbb{R}^{N \times dT}$ (T is the number of visual tokens in ViTs) recalling from $\mathbf{P}^w$ above. Here we calculate it with a slight variation as $\mathbf{P}^a = \text{diag}(\mathbf{J}_N(\mathbf{A}^N) \Delta \mathbf{A}^{N\top}) \in \mathbb{R}^N$. With other parts similar to the case of weight quantization above, this gives the following formulation:

$$\Gamma_{\hat{\mathbf{A}}_l}(\mathbf{O}, \hat{\mathbf{O}}) \approx \frac{\kappa}{2} \|\Delta \mathbf{A}_l\|_2^2 + \frac{1}{2N} \text{trace}(\mathbf{P}_l^a \mathbf{P}_l^{a\top}). \quad (9)$$

The time and space complexity would be $\Theta((N^2 + N)\|s_i\|_0 + N)$ and $\Theta(2N\|s_i\|_0 + N^2)$ respectively, which are still only linear complexity. We have a more detailed analysis in supplemental materials.

Efficiency of Second-Order Calibration.

After the above optimization, the calibration process is now streamlined to a one-time forward+backward pass on calibration set to obtain necessary gradient information $\mathbf{J}_N(\mathbf{W})$, $\mathbf{J}_N(\mathbf{A})$, followed by applying Eq. 8 and Eq. 9 individually to rapidly approximate global output distortions $\Gamma_{\mathbf{W}_l}$, $\Gamma_{\mathbf{A}_l}$ under arbitrary quantizing weights and activations. In practice, one may iterate through a finite set of different quantization configurations to sample $\hat{\mathbf{W}}$, $\hat{\mathbf{A}}$, e.g. a set of possible bit-widths and quantization step sizes, etc. We summarize the estimated end-to-end time and space complexity analysis in Tab. 1.

Table 1: Time and spatial complexity of the second-order calibration for whole model. D_W and D_A denote the total parameter size and total feature map size for a model respectively.

	Weight	Activation
Time	$\Theta(Nl + (N + 2)D_W)$	$\Theta((N^2 + N)D_A)$
Space	$\Theta(2 \max_l \ \mathbf{W}_l\ _0 + N^2 l)$	$\Theta(2N \max_l \ \mathbf{A}_l\ _0 + N^2 l)$

The total calibration time costs scale linearly with the sizes of parameters and activation states, and the maximum memory required is capped at the layer with largest weight or activation footprints ($\|\mathbf{W}_l\|_0$ and $\|\mathbf{A}_l\|_0$). In Tab. 1 we omit the cost for quantization itself. This calibration scheme is also compatible with blockwise MPQ (assign bit-width on a block of weight/activation within layer). Compared to vanilla layerwise MPQ, blockwise MPQ does not increase much time overhead and has less memory requirement as one can deduct from Tab. 1, as it can be seen as changing the problem space with now total number of weight/activation blocks becoming the number of layers and each with less parameter sizes. We explain more blockwise MPQ settings in experiment section.

5 GLOBAL OPTIMAL SOLVER VIA DYNAMIC PROGRAMMING

We adopt Dynamic Programming (DP) algorithm to recursively solve the integer problem by decomposing the whole problem into multiple sub-problems and then processing each sub-problem separately. Defining the state function required by DP as $S(l, r) \in \mathbb{R}$ is a function of discrete values layer $l \in [1, L]$ and remaining model total bits $r \in [1, R]$, this indicates the the minimal achievable model distortion at current intermediate state. Then the DP algorithm can be completed by the following state transferring function to recursively solve the original problem at state $S(L, R)$:

$$S(l, r) = \min_{1 \leq b_1 \leq B} \{\Gamma_{q(\mathbf{Z}_l, b_1)}(\mathbf{O}, \hat{\mathbf{O}}) + S(l - 1, r - b_1 d_l)\}, \quad (10)$$

where $\mathbf{Z}_l$ represents the either one between $\mathbf{W}_l$ and $\mathbf{A}_l$ and we treat states for weights and activates the same as they both contribute to the total bit budget R . $q(\mathbf{Z}_l, b)$ is the optimal b -bit quantizer for input $\mathbf{Z}_l$, B is the maximum allowed bit-width for each layer (we set to $B = 10$). We set the boundary condition of $S(0, \cdot) = 0$. As we populate through the above states $S(l, r)$, we use another function $G(l, r) \in \mathbb{Z}_+$ to record all intermediate selections of b_1 made:

$$G(l, r) = \arg \min_{1 \leq b_1 \leq B} \{\Gamma_{q(\mathbf{Z}_l, b_1)}(\mathbf{O}, \hat{\mathbf{O}}) + S(l - 1, r - b_1 d_l)\}. \quad (11)$$

Using $G(l, r)$, we can iterate through layers to find the optimal bit-width B_l at layer l .

The theoretical time complexity of the DP-based solver is $O(2LRB^2)$, as we have $2L \times R$ different states since we consider two variable bit-widths i.e. weight and activation for each layer. It shows the algorithm scales linearly with the model size. Moreover, the solution found by dynamic programming is the globally optimal solution. Taking advantages of modern parallel compute like GPU, the DP-solver can be further accelerated towards practical usages. We provide more details of our

Algorithm 1 BLOB-Q: Model-level Global-optimal Mixed-precision Quantization Algorithm.

Require: Calibration dataset X_{cal} with N samples, pretrained model with L layers, maximum allowed bit-width B , target total model size R .

Ensure: Optimal bit-widths for layer Weights and activations $B_l^W, B_l^A \in \mathbb{R}, 1 \leq l \leq L$.

- 1: A forward+backward pass on X_{cal} to get $\mathbf{A}_N^{(l)}, J_N(\mathbf{W}_l), J_N(\mathbf{A}_l)$ ▷ Start calibration.
- 2: **for each** layer $l \in [1, L]$ **do**
- 3: **for each** bit $b \in [1, B]$ **do**
- 4: $\hat{\mathbf{W}}_l = q(\mathbf{W}_l, b), \hat{\mathbf{A}}_l = q(\mathbf{A}_l, b)$ ▷ Quantize weights and activations to b -bit.
- 5: $\Delta \mathbf{W}_l = \mathbf{W}_l - \hat{\mathbf{W}}_l, \Delta \mathbf{A}_l = \mathbf{A}_l - \hat{\mathbf{A}}_l$ ▷ Quantization errors.
- 6: $\mathbf{P}_l^w = \mathbf{J}_N^T(\mathbf{W}_l) \Delta \mathbf{W}_l, \mathbf{P}_l^a = \text{diag}(\mathbf{J}_N(\mathbf{A}_l^N) \Delta \mathbf{A}_l^{N^T})$
- 7: $\Gamma_{q(\mathbf{W}_l, b)}(\mathbf{O}, \hat{\mathbf{O}}) = \frac{\kappa}{2} \|\Delta \mathbf{W}_l\|_2^2 + \frac{1}{2N} \text{trace}(\mathbf{P}_l^w \mathbf{P}_l^{w^T})$. ▷ Model Distortion w.r.t. W-quant. (Eq. 8)
- 8: $\Gamma_{q(\mathbf{A}_l, b)}(\mathbf{O}, \hat{\mathbf{O}}) = \frac{\kappa}{2} \|\Delta \mathbf{A}_l\|_2^2 + \frac{1}{2N} \text{trace}(\mathbf{P}_l^a \mathbf{P}_l^{a^T})$. ▷ Model Distortion w.r.t. A-quant. (Eq. 9)
- 9: **end for each**
- 10: **end for each**
- 11: Create matrices $\mathbf{S}, \mathbf{G} \in \mathbb{R}^{2L+1, R}$ ▷ Start DP-solver (Sec.5).
- 12: Initialize $\mathbf{S}[1, r] \leftarrow 0, \forall r \in [1, R]$ if $l = 0$ else ∞ .
- 13: **for each** layer $l \in [1, 2L]$ **do** ▷ (have $2L$ for both weights and activations)
- 14: **for each** size $r \in [1, R]$ **do**
- 15: Update $\mathbf{S}[1, r]$ using Eq. 10, Update $\mathbf{G}[1, r]$ using Eq. 11 ▷ Recursively populate states.
- 16: **end for each**
- 17: **end for each**
- 18: $B_L^W = \mathbf{G}[2L, R]$ ▷ Last layer weight bit-width.
- 19: **for each** layer l reversely from L to 1 **do** ▷ Recursively obtain layerwise bit-widths.
- 20: $B_l^A = \mathbf{G}[2l, R - B_l^W \|\mathbf{W}_l\|_0]$ ▷ optimal activation bit-width.
- 21: $R \leftarrow R - B_l^W \|\mathbf{W}_l\|_0$
- 22: $B_l^W = \mathbf{G}[2l-1, R - B_l^A \|\mathbf{A}_l\|_0]$ ▷ layerwise weight bit-width.
- 23: $R \leftarrow R - B_l^A \|\mathbf{A}_l\|_0$
- 24: **end for each**

implemented DP-solver in the supplementary materials, incorporating strategies like vectorized updating and Run-length coding, with source code attached. Empirically, we find that DP-solver only requires to perform in few seconds as we show in Tab. 5. We provide the end-to-end pseudocode of our proposed BLOB-Q algorithm in Algo. 1.

To this end, we proposed a unified second-order approximation scheme for model output distortion caused by the quantization of both weights and activations. We started by discovering an important property on ViT models under second-order taylor approximation, which allows us to transform the NP-hard Global distortion optimization problem into subproblems. This allows us to solve the model distortion minimization problem in a global-optimal and non-greedy manner via Dynamic Programming. Finally, utilizing Hessian matrix information, we developed an ultra-fast algorithm to calculate the second-order distortion during post-training calibration efficiently considering the practical settings.

6 EXPERIMENTS

Implementation Details. We randomly select 64 samples from the ImageNet training set as the calibration dataset without any augmentations, and evaluate the quantized ViTs on full validation set. We conduct all experiments on Nvida 4-L40 with 48GB of VRAM on each GPU. For bias correction, we adopt the strategies same as Liu et al. (2021b) to rectify the layerwise output distribution. For finetuning, we use the same optimizer and criterion as original ViT strategy, with learning rate $1e-6$ with batchsize of 128 for larger ViT-B/384 and DeiT-B/384 and 256 for the rest. We quantize all the Conv2D and Linear layers in ViTs, including the first and last layer and excluding LayerNorm and Embedding layers. For quantizer, we choose vanilla symmetric uniform quantizer for both weights and activations. We perform blockwise MPQ in the experiments as earlier introduced in Section 4, *i.e.* allocating a group of channels in the layer weight/activation once a time, where we set the blocksize to 64-channels for larger ViT-B/384 and DeiT-B/384 and 32-channel for the rest. We provide more details in supplementary materials.

Main Results. We compare our approach with prior Post-Training Quantization (PTQ) methods for ViTs at six ViT models (ViT-S, ViT-B, ViT-B/384, DeiT-S, DeiT-B, and DeiT-B/384) on the ImageNet validation dataset, including PTQ Liu et al. (2021b), PTQ4ViT Yuan et al. (2022), Percentile Li et al. (2019), EasyQuant Wu et al. (2020), APQ-ViT Ding et al. (2022), NoisyQuant Liu et al. (2023b), I-ViT Li & Gu (2023), and FQ-ViT Lin et al. (2021). Tab. 2 illustrates the results. Our approach consistently outperforms the baseline methods in most of the cases, especially at low bit widths. Specifically, at 4 bits, our approach outperforms other methods by 1.72% and 11.49% on ViT-S and ViT-B, respectively. On DeiT-S and DeiT-B, at 4 bits, our approach outperforms others by 6.15% and 3.54%, respectively. It is worth mentioning that our approach can quantize the ViT models to 6 bits without hurting accuracy (loss < 1%). We notice that most method can obtain very well accuracy at 8 bits. This is because that 8 bits are long enough to maintain the precision.

Table 2: Results on the ImageNet validation dataset when ViT models are quantized to 4 bits, 6 bits, and 8 bits. Top-1 classification accuracy is evaluated. MP denotes quantization with Mixed-Precision. For a fair comparison, we report the results with the same average bit widths. For example, 4 MP means that we quantize the ViT Models into 4 bits on average across the layers. The best result of each group is highlighted with bold.

Method	Size (Bit)	ViT-S	ViT-B	ViT-B/384	DeiT-S	DeiT-B	DeiT-B/384
Full Precision	32	81.39%	84.54%	86.05%	79.87%	81.85%	83.12%
PTQ4ViT Yuan et al. (2022)	4	42.57%	30.69%	-	34.08%	64.39%	-
APQ-ViT Ding et al. (2022)	4	47.95%	41.41%	-	43.55%	67.48%	-
RepQ-ViT Li et al. (2023)	4	65.60%	68.48%	-	69.03%	75.61%	-
BLOB-Q	4 MP	67.32%	79.97%	71.20%	75.18%	79.48%	80.17%
PTQ Liu et al. (2021b)	6 MP	70.24%	75.26%	46.88%	75.10%	77.47%	68.44%
PTQ4ViT Yuan et al. (2022)	6	78.63%	81.65%	83.34%	76.28%	80.25%	81.55%
Percentile Li et al. (2019)	6 MP	67.74%	77.63%	77.60%	70.49%	73.99%	78.24%
EasyQuant Wu et al. (2020)	6	75.13%	81.42%	82.02%	73.26%	75.86%	81.26%
APQ-ViT Ding et al. (2022)	6	79.10%	82.21%	-	77.76%	80.42%	-
RepQ-ViT Li et al. (2023)	6	80.43%	83.62%	-	77.76%	80.42%	-
NoisyQuant Liu et al. (2023b)	6	78.65%	82.32%	83.22%	77.43%	80.70%	81.65%
BLOB-Q	6 MP	80.53%	83.66%	85.20%	79.22%	81.58%	82.72%
PTQ Liu et al. (2021b)	8 MP	80.46%	76.98%	85.35%	-	75.94%	-
PTQ4ViT Yuan et al. (2022)	8	81.00%	84.25%	85.82%	79.47%	81.48%	82.97%
Percentile Li et al. (2019)	8 MP	78.77%	80.12%	82.53%	73.98%	75.21%	80.02%
EasyQuant Wu et al. (2020)	8	80.75%	83.89%	85.53%	76.59%	79.36%	82.10%
APQ-ViT Ding et al. (2022)	8	81.25%	84.26%	-	79.78%	81.72%	-
NoisyQuant Liu et al. (2023b)	8	81.15%	84.22%	85.86%	79.51%	81.45%	82.49%
I-ViT Li & Gu (2023)	8	81.27%	84.76%	-	80.12%	81.74%	-
FQ-ViT Lin et al. (2021)	8	-	83.31%	-	79.17%	81.20%	-
BLOB-Q	8 MP	81.54%	84.84%	85.93%	79.59%	81.75%	82.85%

Ablation Studies. Both model distortion minimization and global-optimal solution contribute to the results. We conduct an ablation study to evaluate the effectiveness of each of them, which is shown in Tab. 3. In the ablation study, we evaluate the results of three methods which are (a) minimize layer distortion with a local solution, (b) minimize model distortion with a local solution, and (c) minimize model distortion with a global-optimal solution. As we can see in Tab. 3, both of model distortion minimization and global-optimal solution can make a noticeable impact on the accuracy. For example, at 4 bits, with model distortion minimization, the accuracy is improved from 0.11 to 20.52 on ViT-S. With a global-optimal solution, the accuracy is improved from 20.25 to 61.45. Similarly noticeable results can be also observed on other models and bit widths.

Impact of the Calibration Set. We evaluate the impact of the calibration set on our approach. Tab. 4 illustrates the results of our approach with 32 and 64 calibration images. First, our approach only requires a very small calibration set to obtain the statistical data of model distortion. We notice that increasing the calibration size from 32 images to 64 images, our approach can obtain slightly higher accuracy. With more calibration images (e.g., 128), the accuracy can not be improved any more but the computation complexity can be significantly increased. We thus use 64 calibration images in all of our experiments.

Executive Time. We also evaluate the execution time of the optimization method and compare it with other baselines, including PTQ4ViT Yuan et al. (2022), REPQ-ViT Li et al. (2023), EasyQuant Wu et al. (2020), and NoisyQuant Liu et al. (2023b). Our method is much faster than others by orders

Table 3: Ablation studies of our approach. The effectiveness of model distortion minimization and global-optimal solution are evaluated. We use a greedy algorithm to find a local solution in this table. For layer distortion, we use the output error of current layer in the learning objective. The supplementary materials provide more details about the local solution.

Comparison	Size	ViT-S	ViT-B	ViT-B/384	DeiT-S	DeiT-B	DeiT-B/384
Layer Distortion + Local	4 Bits	0.11%	0.10%	0.09%	0.10%	0.12%	0.12%
Model Distortion + Local	4 Bits	20.52%	1.25%	0.60%	0.16%	16.51%	49.92%
Model Distortion + Global	4 Bits	61.45%	77.31%	51.69%	75.14%	79.21%	79.46%
Layer Distortion + Local	6 Bits	0.10%	0.12%	0.11%	0.17%	14.01%	50.80%
Model Distortion + Local	6 Bits	62.87%	15.27%	11.04%	40.25%	44.08%	74.70%
Model Distortion + Global	6 Bits	78.88%	83.47%	84.71%	79.21%	81.58%	82.72%
Layer Distortion + Local	8 Bits	39.81%	1.25%	14.43%	0.12%	45.81%	76.08%
Model Distortion + Local	8 Bits	67.78%	52.34%	61.23%	74.88%	80.81%	81.12%
Model Distortion + Global	8 Bits	81.37%	84.78%	85.69%	79.64%	81.76%	82.99%

Table 4: Impact of calibration size on the results. We randomly select 32 or 64 images from the ImageNet training dataset as the calibration set.

Bit Width Calibration Size	4 Bits		6 Bits		8 Bits	
	32 Images	64 Images	32 Images	64 Images	32 Images	64 Images
ViT-S	51.04%	59.71%	81.90%	79.98%	83.26%	81.30%
ViT-B	78.82%	74.51%	84.10%	83.58%	84.46%	83.95%
ViT-B/384	66.01%	52.09%	84.42%	85.18%	85.35%	86.01%
DeiT-S	68.75%	72.80%	78.12%	78.97%	81.25%	79.51%
DeiT-B	78.12%	78.96%	81.25%	81.81%	81.25%	81.78%
DeiT-B/384	78.59%	79.06%	82.25%	82.72%	82.59%	82.56%

of magnitude. As shown in Tab. 5, our method is $72\times$ faster than PTQ4ViT Yuan et al. (2022), $85\times$ faster than REPQ-ViT Li et al. (2023), $146\times$ faster than EasyQuant Wu et al. (2020), and $543\times$ faster than NoisyQuant Liu et al. (2023b). Different with prior works which need to search the solution from a huge space with exponential complexity, our method utilizes the additivity property and adopts dynamic programming find the global-optimal solution. As a result, our optimization algorithm has only linear time complexity and is much faster than others. We analyzed the time and memory efficiency in more detail in Appendix D.4.

Table 5: Evaluation of the executive time of optimization. We test the time of our dynamic programming algorithm to find the global-optimal bit allocation. For other baseline methods, we run their code on the same hardware (Nvidia L40) to evaluate the time.

Method	Executive Time of Optimization Method (s)											
	ViT-S			ViT-B			DeiT-S			DeiT-B		
	4 Bits	6 Bits	8 Bits	4 Bits	6 Bits	8 Bits	4 Bits	6 Bits	8 Bits	4 Bits	6 Bits	8 Bits
PTQ4ViT Yuan et al. (2022)	103.8	101.4	101.4	193.2	182.4	196.2	100.6	100.2	100.8	196.8	198.0	207.6
REPQ-ViT Li et al. (2023)	123.4	126.1	118.8	247.0	247.9	252.8	114.2	119.0	119.1	244.6	251.9	251.6
EasyQuant Wu et al. (2020)	211.1	211.5	211.1	463.0	463.6	464.0	214.3	212.4	211.2	463.0	463.6	463.7
NoisyQuant Liu et al. (2023b)	781.0	780.8	784.5	1844.5	1850.4	1851.5	787.6	777.7	778.3	1845.2	1848.6	1848.4
BLOB-Q	1.438	1.209	1.298	1.749	2.291	2.236	1.408	1.329	1.212	1.814	2.121	2.276

7 CONCLUSION

In conclusion, we novelly introduced BLOB-Q, a global-optimal model distortion minimized Mixed-precision Quantization method. Our approach builds on a key property observed in ViT models under second-order Taylor approximation, enabling the transformation of the NP-hard global distortion optimization problem into manageable subproblems. We further proposed a non-greedy and lightweight DP-solver to solve MPQ bit-allocations that finds globally optimal solutions. Furthermore, we proposed a unified hessian approximation scheme, accompanied by an ultra-fast algorithm with minimal memory footprint, ensuring scalability of the method. Experiments show that BLOB-Q significantly boost PTQ performance on ViT models, and successfully bridge the performance gap on as low as 4-bit even without retraining.

REPRODUCIBILITY STATEMENT

We ensure to provide all necessary details to reproduce the experimental results shown. In the Supplementary Materials, we provide the source code, containing:

- Experimentation scripts for second-order calibration and DP-solver to 4/6/8-bits.
- Code for bias-correction and retraining for auxiliary results.
- A README file providing sample commands and information on how to run all scripts.

REFERENCES

- Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment. *arXiv preprint arXiv:1810.05723*, 2018.
- Jun Chen, Shipeng Bai, Tianxin Huang, Mengmeng Wang, Guanzhong Tian, and Yong Liu. Data-free quantization via mixed-precision compensation without fine-tuning. *Pattern Recognition*, 143: 109780, 2023.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *ACL*, 2020.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv*, 2019.
- Yongle Zhao Ming Zhang Debing Zhang Ying Fu Di Wu, Qi Tang. Easyquant: Post-training quantization via scale optimization. 2020.
- Yifu Ding, Haotong Qin, Qinghua Yan, Zhenhua Chai, Junjie Liu, Xiaolin Wei, and Xianglong Liu. Towards accurate post-training quantization for vision transformer. In *Proceedings of the 30th ACM international conference on multimedia*, pp. 5380–5388, 2022.
- Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *arXiv*, 2019.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=YicbFdNTTy>.
- Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.
- L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.
- Xue Geng, Zhe Wang, Chunyun Chen, Qing Xu, Kaixin Xu, Chao Jin, Manas Gupta, Xulei Yang, Zhenghua Chen, Mohamed M Sabry Aly, et al. From algorithm to hardware: A survey on efficient and safe deployment of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.
- Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.
- K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In *CVPR*, 2017.
- I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. In *arXiv:1609.07061*, 2016.
- J. Johnson, A. Alahi, and F.-F. Li. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

- T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.
- Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 4163–4181, 2022.
- Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. Layer-adaptive sparsity for the magnitude-based pruning. In *International Conference on Learning Representations*, 2020.
- Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2810–2819, 2019.
- Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2020.
- Zhikai Li and Qingyi Gu. I-vit: Integer-only quantization for efficient vision transformer inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17065–17075, 2023.
- Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17227–17236, 2023.
- Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Post-training quantization for fully quantized vision transformer. *arXiv preprint arXiv:2111.13824*, 2021.
- Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Post-training quantization for fully quantized vision transformer. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 1173–1179, 2022.
- Yijiang Liu, Huanrui Yang, Zhen Dong, Kurt Keutzer, Li Du, and Shanghang Zhang. Noisyquant: Noisy bias-enhanced post-training activation quantization for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20321–20330, June 2023a.
- Yijiang Liu, Huanrui Yang, Zhen Dong, Kurt Keutzer, Li Du, and Shanghang Zhang. Noisyquant: Noisy bias-enhanced post-training activation quantization for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20321–20330, 2023b.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. In *arXiv*, 2019.
- 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. In *ICCV*, 2021a.
- 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, 2021b.
- Qian Lou, Feng Guo, Minje Kim, Lantao Liu, and Lei Jiang. Autoq: Automated kernel-wise neural network quantizations. In *ICLR*, 2020.
- Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. *PMLR*, 2020a.

- Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020b.
- Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson. Loss aware post-training quantization. *Machine Learning*, 110(11-12): 3245–3262, 2021.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. In *arXiv*, 2018.
- M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-NET: Imagenet classification using binary convolutional neural networks. In *arXiv preprint arXiv:1603.05279*, 2016.
- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, 2017.
- Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8612–8620, 2019.
- 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.
- Kaixin Xu, Zhe Wang, Chunyun Chen, Xue Geng, Jie Lin, Xulei Yang, Min Wu, Xiaoli Li, and Weisi Lin. Lpvit: Low-power semi-structured pruning for vision transformers. In *European Conference on Computer Vision*, pp. 269–287. Springer, 2024.
- Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18547–18557, 2023.
- Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization framework for vision transformers with twin uniform quantization. *arXiv preprint arXiv:2111.12293*, 2021.
- Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In *European conference on computer vision*, pp. 191–207. Springer, 2022.
- D. Zhang, J. Yang, D. Ye, and G. Hua. Lq-nets: learned quantization for highly accurate and compact deep neural networks. In *ECCV*, 2018.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.
- Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. *PMLR*, 2019.
- A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *arXiv preprint arXiv:1702.03044*, 2017.
- S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. In *arXiv preprint arXiv:1606.06160*, 2016.
- Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- C. Zhu, S. Han, H. Mao, and W.J. Dally. Ternary weight networks. In *ICLR*, 2017.

A PROOF OF PROPERTY 1 (SECOND-ORDER ADDITIVITY)

Recall that $\Delta \mathbf{W}_l$ and $\Delta \mathbf{A}_l$ are the perturbation on layer weights and activations respectively by quantization, *i.e.*, $\Delta \mathbf{W}_l = \mathbf{W}_l - \widehat{\mathbf{W}}_l$, $\Delta \mathbf{A}_l = \mathbf{A}_l - \widehat{\mathbf{A}}_l$. The derivation of Property 1 utilizes a basic assumption as follows:

Assumption 1. Quantization perturbation Layer-independency Zhou et al. (2018): The perturbation on weight/activation by individual quantizing on each layer is (1) zero-meaned: $\forall 1 \leq i \leq L, \mathbb{E}(\Delta \mathbf{W}_i) = \mathbb{E}(\Delta \mathbf{A}_i) = 0$, and (2) independent across layer: $\forall 1 \leq i \neq j \leq L, \mathbb{E}(\Delta \mathbf{W}_i) \mathbb{E}(\Delta \mathbf{W}_j) = \mathbb{E}(\Delta \mathbf{A}_i) \mathbb{E}(\Delta \mathbf{A}_j) = \mathbb{E}(\Delta \mathbf{W}_i) \mathbb{E}(\Delta \mathbf{A}_j) = 0$.

We provide more discussion and evidences in Appendix A.1.

Proof. According to second-order taylor series expansion when under perturbations on l -th layer's weight (first order term can be neglected on pretrained models):

$$\widehat{\mathbf{O}} \approx \mathbf{O} + \frac{1}{2} \Delta \mathbf{W}_l^\top \mathbf{H}_l^w \Delta \mathbf{W}_l, \quad (12)$$

The L2 distortion when all the layer weights are perturbed can be written as

$$\begin{aligned} \Gamma_{\widehat{\mathbf{W}}}(\mathbf{O}, \widehat{\mathbf{O}}) &= \mathbb{E}_X \left[\|\mathbf{O} - \widehat{\mathbf{O}}\|_F^2 \right] \approx \mathbb{E}_X \left[\left(\sum_{i=1}^l \frac{1}{2} \Delta \mathbf{W}_i^\top \mathbf{H}_i^w \Delta \mathbf{W}_i \right)^\top \left(\sum_{j=1}^l \frac{1}{2} \Delta \mathbf{W}_j^\top \mathbf{H}_j^w \Delta \mathbf{W}_j \right) \right] \\ &= \mathbb{E}_X \left[\sum_{i,j=1}^l \left(\frac{1}{2} \Delta \mathbf{W}_i^\top \mathbf{H}_i^w \Delta \mathbf{W}_i \right)^\top \left(\frac{1}{2} \Delta \mathbf{W}_j^\top \mathbf{H}_j^w \Delta \mathbf{W}_j \right) \right]. \end{aligned} \quad (13)$$

The summation inside expectation can be further swap out with the expectation:

$$\Gamma_{\widehat{\mathbf{W}}}(\mathbf{O}, \widehat{\mathbf{O}}) \approx \sum_{i,j=1}^l \mathbb{E}_X \left[\frac{1}{4} \Delta \mathbf{W}_i \mathbf{H}_i^{w\top} \Delta \mathbf{W}_i \Delta \mathbf{W}_j^\top \mathbf{H}_j^w \Delta \mathbf{W}_j \right]. \quad (14)$$

Further utilize independency assumption in Assumption 1 on the expectation on matrices: (replace $\mathbb{E}_X$ with $\mathbf{E}$ for visual simplicity)

$$\Gamma_{\widehat{\mathbf{W}}}(\mathbf{O}, \widehat{\mathbf{O}}) \approx \frac{1}{4} \mathbf{E}[\Delta \mathbf{W}_i] \mathbf{H}_i^{w\top} \mathbf{E}[\Delta \mathbf{W}_i \Delta \mathbf{W}_j^\top] \mathbf{H}_j^w \mathbf{E}[\Delta \mathbf{W}_j]. \quad (15)$$

Notice that (1) $\mathbf{E}[\Delta \mathbf{W}_i]$ and $\mathbf{E}[\Delta \mathbf{W}_j]$ in the above complies with the zero-mean assumption when $i \neq j$ in Assumption 1, (2) $\mathbf{E}[\Delta \mathbf{W}_i \Delta \mathbf{W}_j^\top] = 0$ since assumption 1 indicates $\Delta \mathbf{W}_i$ and $\Delta \mathbf{W}_j$ are independent, and (3) the fact that all-zero matrix multiplied with any matrix results in zero, we can eliminate such cross terms $i \neq j$ in the summands, resulting in:

$$\Gamma_{\widehat{\mathbf{W}}}(\mathbf{O}, \widehat{\mathbf{O}}) \approx \sum_{i=1}^l \mathbf{E} \left(\left\| \frac{1}{2} \Delta \mathbf{W}_i^\top \mathbf{H}_i^w \Delta \mathbf{W}_i \right\|^2 \right). \quad (16)$$

The above derivation also applies to the case of activation quantization, resulting:

$$\Gamma_{\widehat{\mathbf{A}}}(\mathbf{O}, \widehat{\mathbf{O}}) \approx \sum_{i=1}^l \mathbf{E} \left(\left\| \frac{1}{2} \Delta \mathbf{A}_i^\top \mathbf{H}_i^a \Delta \mathbf{A}_i \right\|^2 \right). \quad (17)$$

Therefore, the original expected L2 distortion under weights and activation perturbation is:

$$\begin{aligned} \Gamma(\mathbf{O}, \widehat{\mathbf{O}}) &\approx \sum_{i=1}^l \Gamma_{\widehat{\mathbf{W}}_i}(\mathbf{O}, \widehat{\mathbf{O}}) + \sum_{i=1}^l \Gamma_{\widehat{\mathbf{A}}_i}(\mathbf{O}, \widehat{\mathbf{O}}) \\ &= \sum_{i=1}^l \mathbf{E} \left(\left\| \frac{1}{2} \Delta \mathbf{W}_i^\top \mathbf{H}_i^w \Delta \mathbf{W}_i \right\|^2 \right) + \sum_{i=1}^l \mathbf{E} \left(\left\| \frac{1}{2} \Delta \mathbf{A}_i^\top \mathbf{H}_i^a \Delta \mathbf{A}_i \right\|^2 \right). \end{aligned} \quad (18)$$

□

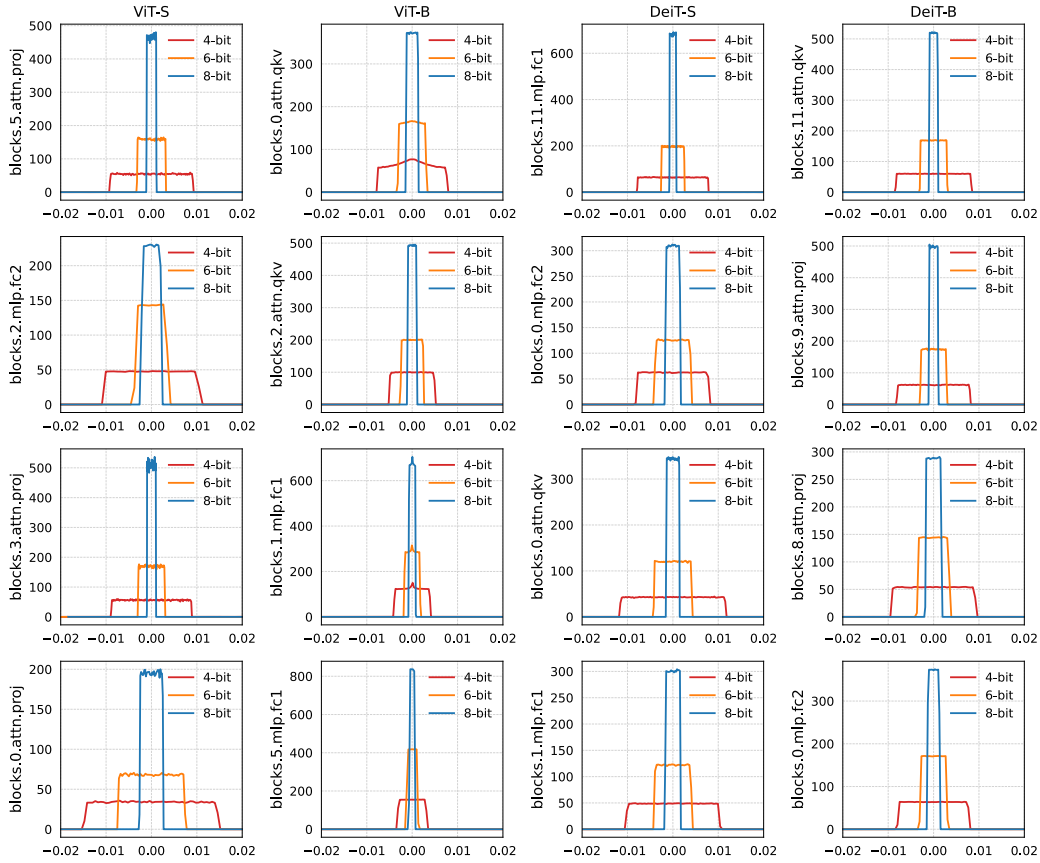


Figure 2: Empirical statistics of histogram of weight perturbation ΔW_i when quantizing individual layers. We randomly select 4 layers from each models to display the distribution quantization error matrices under various bit-widths. We use the same symmetric uniform quantizer with least MSE scaling factors.

A.1 EVIDENCES OF ASSUMPTION USED

We now show the empirical observations on real ViT models to further support the assumptions used in the above theoretical proof.

Assumption 1 (1): Zero-mean: Fig. 2 illustrates the distributions of weight perturbation ΔW in different layers of different models with different bitwidths. Under majority circumstances, the distributions of the element-wise perturbations display a symmetric pattern around zero, supporting that zero-mean assumption holds universally for ViTs.

Assumption 1 (2): Interlayer independency: Interlayer independency can be evaluated by observing the covariance between ΔW_i and ΔW_j . Fig. 3 illustrates the full pair-wise combination of all quantizable layers in the model, where each entry is a sample of covariance of layerwise weight perturbation of two layers averaged on a random batch of images from ImageNet. It displays a clear diagonal pattern that the covariances vanish on off-diagonal locations where $i \neq j$, which is observed on all 4 ViTs. This supports the interlayer-independence assumption.

A.2 EVIDENCES OF ADDITIVITY

To further support the theoretically discovered Property 1 (Section A), we further provide direct empirical evidences on ViT models as in Fig. 4 and Fig. 5 at low bit width weight/activation quantization from 5 bits down to 2 bits. As one can see, at 4 to 5 bits, most of points lie in the diagonal meaning that the additivity property holds well. However, when the bit width is less than 4

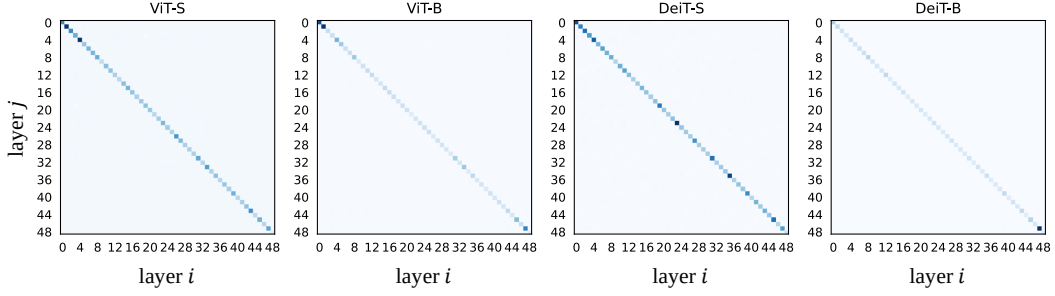


Figure 3: Empirical statistics of covariance matrix $E(\|\Delta \mathbf{W}_i \Delta \mathbf{W}_j\|^2)$ when quantizing individual layers in pretrained model. Deeper blue color indicates larger value, vice versa.

bits, outlier points start to increase. In practice, our approach can quantize the model into 6 bits to 8 bits without hurting accuracy. There is a noticeable accuracy drop when the model is quantized to lower bit width (e.g, 4 bits or less).

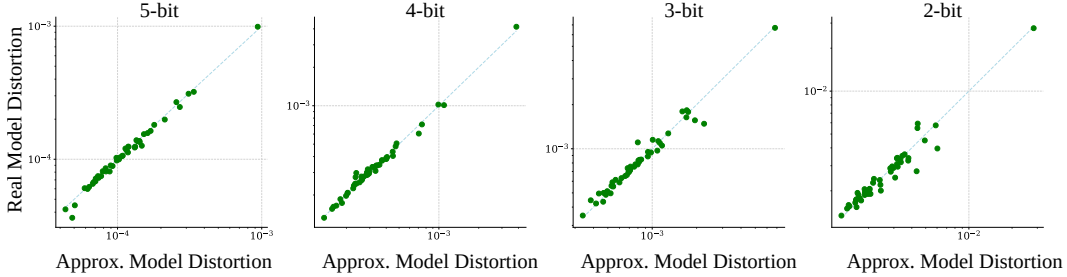


Figure 4: Empirical Evidences of Second-order additivity property on ViT-S. The X-axis represents the model distortion when approximating via second-order additivity property (Property 1) when quantizing all layer **weights**. The Y-axis represents the real model distortion by directly measuring the L2 distortion on model output. Data points inclines to the diagonal lines very well, showing that the additivity property holds.

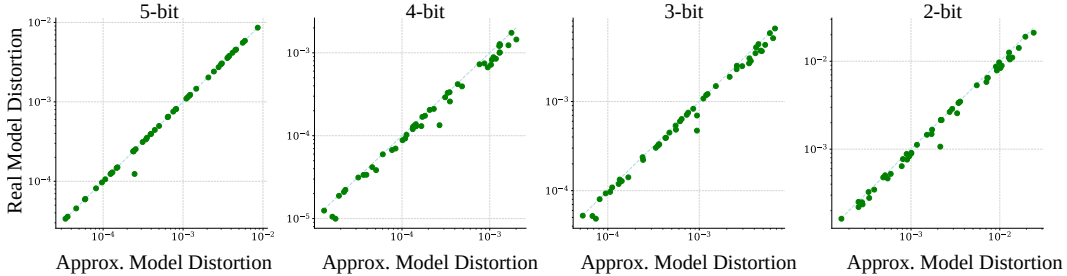


Figure 5: Empirical Evidences of Second-order additivity property on DeiT-B. The X-axis represents the model distortion when approximating via second-order additivity property (Property 1) when quantizing all layer **activations**. The Y-axis represents the real model distortion by directly measuring the L2 distortion on model output. Data points inclines to the diagonal lines very well, showing that the additivity property holds.

B RELATION BETWEEN MODEL DISTORTION AND MODEL ACCURACY

Our BLOB-Q method optimizes the mixed-precision bit-allocation directly on model distortion, i.e. the L2 distortion on model output between FP and quantized model. This is based on the hypothesis that model distortion directly reflects the task performance. To evaluate this hypothesis, we evaluate this correlation on empirical models in Fig. 6. As it shows, the model distortion is closely correlated to accuracy under various model bit-rates under 3 models, where the large distortion is associated

with poor model accuracy. This strongly supports the hypothesis that is also one of the motivations of BLOB-Q.

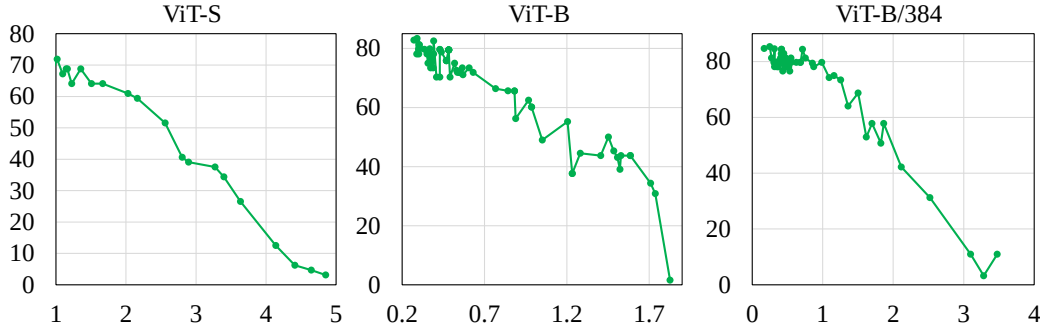


Figure 6: Model Distortion (X-axis) vs. Model Accuracy (Y-axis). We quantize each model to decreasing target mixed-precision bit-widths using BLOB-Q. We evaluate the average L2 distortion on 128 calibration images from ImageNet training set.

C PARALLEL DP-SOLVER DESIGN

A naive DP-solver can be implemented based on Section 4 which by default performs sequentially on CPU, but we can harness the parallel-computing power of GPU to further accelerate the theoretical time cost to enable DP-Solver for larger models at scale. DP-solver to solve the integer problem which can be regarded as finding the minimally achievable bitwidth (“weight”) for each group of to-be-quantized layer weight/activation (“item”). In the first step, since from Eq. 10 we can see that the states $S[1, :]$ only depends on previous layer $S[1-1, :]$, and the fact that the final solutions are only related to G , thus we can reduce the memory cost of the outer-most loop (Algorithm 1 L13) by allocate a smaller 2-by- R matrix for S to reduce the memory allocation by $L/2$ times. This avoids OOM issue on large models like DeiT-B/384 with higher target rates *e.g.* 8-bit. Secondly, in the states updating in DP-solver (Algorithm 1 L13-17), the number of states needed to iterate the inner-loop can be actually reduced to $\sum_i d_i B \leq R$ as this is the largest reachable accumulated sizes when all visited layers (from 1-st to l -th) quantized to the maximum bitwidth B . This can reduces the complexity for the nested for-loop by virtually half. Also, to avoid OOM for large layer sizes, we perform parallel state updating piecewisely for the inner-loop, allowing us to adjust the piece length to fully utilize VRAM. Finally, for the matrix G , we can offload it to storage layerwisely for $G[1, :]$ since as Eq. 11 indicates it does not depend on G of other layers, and we can load back to perform Algorithm 1 L19-23. However, for large models *e.g.* DeiT-B with target bit-rate of 8-bit, the storage needed for offloaded G can be as large as a long matrix of $50 \times 8.6 \times 10^7 \times 8 \times 8 \text{bytes} \approx 256 \text{GB}$, which is unacceptable. We notice that due to the discrete behavior of the state transfer, at each layer, a lot of entries in G repeat themselves for as long as d_l . Therefore we perform run-length-coding to encode G before offloading to storage and efficiently decode it back for the final steps of the algorithm. Empirically this significantly reduces the encoded size of G to only around 20MiB for DeiT-B model. The attached source code contains the complete details of the proof-of-concept implementation of the above described parallel DP-solver.

D ADDITIONAL EXPERIMENT RESULTS

D.1 RESULTS WITH FINETUNING

Although BLOB-Q does not require finetuning to compete with plenty of SOTA methods as in Tab. 1 in main text, for completeness we investigate the effect of finetuning on top of raw BLOB-Q results. As illustrated in Tab. 6, we can see that finetuning burings noticeable gain from the retraining-free results on all 6 models in most cases, especially the case under low bit rate like 4-bit MP. On higher bit rates (6/8-bit), given the already small performance gaps from dense model, finetuning brings small improvements for most cases, and maintains similar performances of retraining-free results for

Table 6: Additional Finetuning results after Post-training Quantization (marked as “+ FT”). Top-1 classification accuracy is evaluated. MP denotes quantization with Mixed-Precision. For a fair comparison, we report the results with the same average bit widths. For example, 4 MP means that we quantize the ViT Models into 4 bits on average across the layers. The best result of each group is highlighted with bold.

Method	Size (Bit)	ViT-S	ViT-B	ViT-B/384	DeiT-S	DeiT-B	DeiT-B/384
Full Precision	32	81.39%	84.54%	86.05%	79.87%	81.85%	83.12%
PTQ4ViT Yuan et al. (2022)	4	42.57%	30.69%	-	34.08%	64.39%	-
APQ-ViT Ding et al. (2022)	4	47.95%	41.41%	-	43.55%	67.48%	-
RepQ-ViT Li et al. (2023)	4	65.60%	68.48%	-	69.03%	75.61%	-
BLOB-Q	4 MP	67.32%	79.97%	71.20%	75.18%	79.48%	80.17%
BLOB-Q + FT	4 MP	68.53%	80.66%	72.39%	77.42%	80.15%	81.0%
PTQ Liu et al. (2021b)	6 MP	70.24%	75.26%	46.88%	75.10%	77.47%	68.44%
PTQ4ViT Yuan et al. (2022)	6	78.63%	81.65%	83.34%	76.28%	80.25%	81.55%
Percentile Li et al. (2019)	6 MP	67.74%	77.63%	77.60%	70.49%	73.99%	78.24%
EasyQuant Wu et al. (2020)	6	75.13%	81.42%	82.02%	73.26%	75.86%	81.26%
APQ-ViT Ding et al. (2022)	6	79.10%	82.21%	-	77.76%	80.42%	-
RepQ-ViT Li et al. (2023)	6	80.43%	83.62%	-	77.76%	80.42%	-
NoisyQuant Liu et al. (2023b)	6	78.65%	82.32%	83.22%	77.43%	80.70%	81.65%
BLOB-Q	6 MP	80.53%	83.66%	85.20%	79.22%	81.58%	82.72%
BLOB-Q + FT	6 MP	80.31%	84.23%	85.15%	79.53%	81.85%	82.98%
PTQ Liu et al. (2021b)	8 MP	80.46%	76.98%	85.35%	-	75.94%	-
PTQ4ViT Yuan et al. (2022)	8	81.00%	84.25%	85.82%	79.47%	81.48%	82.97%
Percentile Li et al. (2019)	8 MP	78.77%	80.12%	82.53%	73.98%	75.21%	80.02%
EasyQuant Wu et al. (2020)	8	80.75%	83.89%	85.53%	76.59%	79.36%	82.10%
APQ-ViT Ding et al. (2022)	8	81.25%	84.26%	-	79.78%	81.72%	-
NoisyQuant Liu et al. (2023b)	8	81.15%	84.22%	85.86%	79.51%	81.45%	82.49%
I-ViT Li & Gu (2023)	8	81.27%	84.76%	-	80.12%	81.74%	-
FQ-ViT Lin et al. (2021)	8	-	83.31%	-	79.17%	81.20%	-
BLOB-Q	8 MP	81.54%	84.84%	85.93%	79.59%	81.75%	82.85%
BLOB-Q + FT	8 MP	81.47%	84.89%	86.01%	79.79%	81.89%	83.04%

few others. We currently finetune for only 100 iterations on ImageNet training set. With dedicated hyperparameter tuning, e.g. with larger epochs and other optimizer choices, we may see even more performance gain.

D.2 SCALABILITY OF LOWER QUANTIZATION BIT-WIDTH

Beyond the 4-bit quantization that most existing methods, we further evaluated the quantization results on even lower bit-width (3-bit), and we found more significant improvement on 3-bit compared to others, up to 41.15% on ViT-L as shown in Tab. 7. As claimed in our paper, one of our main contributions of this paper is to boost the performance of ViT quantization on low bit-width. This is useful in practice when the model is required to be quantized to very small sizes for deployment.

Table 7: Extended experiments on under 3-bit. Both weights and activations are quantized.

Method	Bit-width	ViT-B	ViT-L
NoisyQuant Liu et al. (2023b)	3	0.138	1.72
RepQ-ViT Li et al. (2023)	3	0.27	0.85
Ours	3 MP	36.36	42.87

D.3 BIT ALLOCATION STATISTICS

We some example visualizations of the bit allocations resulted from BLOB-Q on ViTs. Fig. 7 and Fig. 8 show the distribution of the averaged bits allocated blockwisely inside each layer under two ViT models, ViT-S and ViT-B, and quantized to 4 and 6-bits respectively. We notice some patterns for the allocation results. Firstly, aligning with previous observations in many compression works that first and last layers are usually sensitive to quantization thus requires more bits, we also observe similar phenomenon on ViTs especially for activation quantization. Beside that, we also found an

interesting pattern that the `fc1` layers in latter transformer blocks requires slightly more bits to quantize their activations, which is constantly observed under multiple settings.

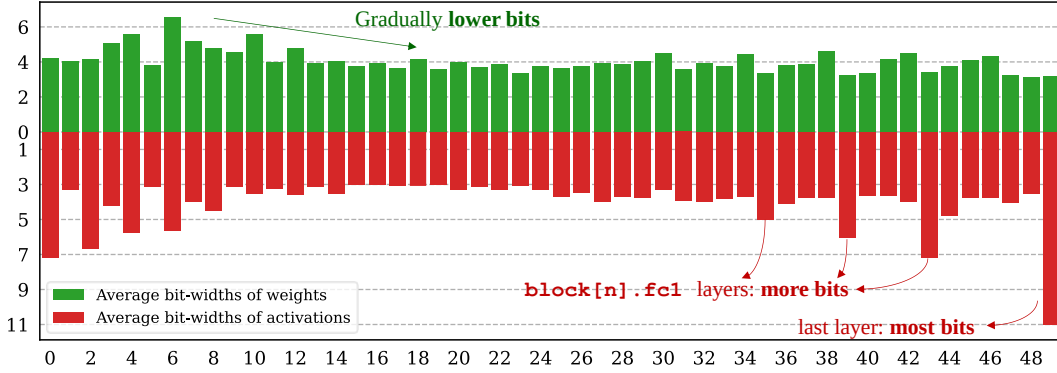


Figure 7: Bit allocation visualization of BLOB-Q on 4-bit ViT-B. We show the averages of bit rates allocated to each block of weight/activation tensor within each layer.

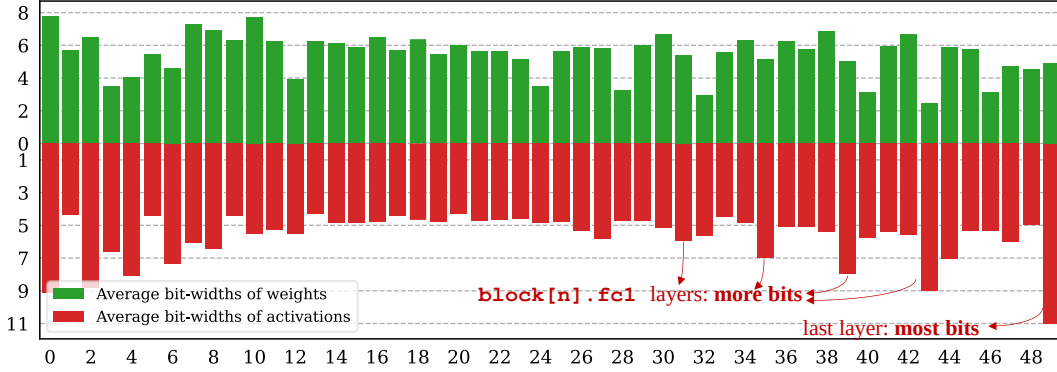


Figure 8: Bit allocation visualization of BLOB-Q on 6-bit ViT-B. We show the averages of bit rates allocated to each block of weight/activation tensor within each layer.

D.4 EMPIRICAL TIME AND MEMORY COSTS

Fig. 9 shows the time costs of our proposed BLOB-Q compared to the most competitive methods to us. BLOB-Q consists of (1) **BLOB-Q-Calib** a one-time calibration on weight/activation quantization which can be used to solve MPQ with arbitrary target rates, and (2) **BLOB-Q-DP** the DP-solver to get final solution for a specific bit-width target (*e.g.* 4/6/8-bit). As the figure shows, our method displays minimal optimization times (BLOB-Q-DP) and significantly faster than NoisyQuant Liu et al. (2023b). Moreover, the total times (BLOB-Q-Total) combining two steps together still remain competitive within manageable range. Note that in practice, the calibration is only required to perform once and versatile to arbitrary target bit rates, therefore the calibration time (BLOB-Q-Calib) does not significantly contribute to overall time consumption for *e.g.* multi-target scenario.

Fig. 10 shows the memory costs of calibrating ViT models on L40 machine. As it shows, the VRAM memory required to perform PTQ ranges reasonably between around 2GB and 6GB for the evaluated models, making our BLOB-Q accessible to more use cases.

D.5 MORE ViT VARIANTS AND NON-TRANSFORMER MODELS

We further evaluate the scalability of our method to other ViT variants, including ViT-L and Swin variants like Swin-T and Swin-S, as well as pre-ViT architectures like ResNet-18. As shown in Tab. 8, we observe consistent advantage compared to baseline method on various bit-widths, especially for lower bit-widths where we improve from baseline method by up to 11.24% under 4-bit. This shows that the effectiveness of the BLOB-Q framework is not constrained to ViT architecture.

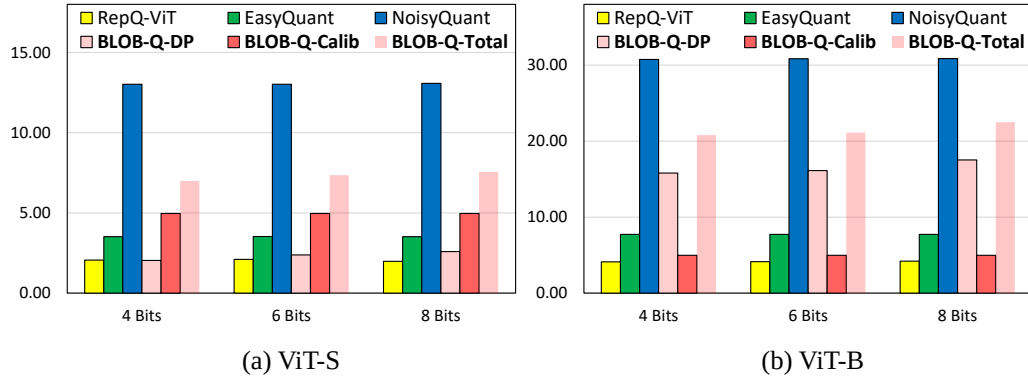


Figure 9: Time costs (minutes) of different PTQ methods.

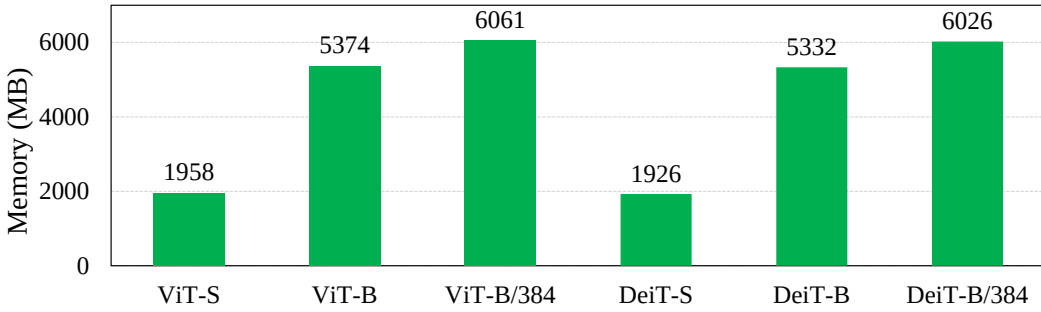


Figure 10: VRAM Memory consumptions (MB) of BLOB-Q calibration.

Table 8: More ViT variants and CNNs results on ImageNet-1k .

Method	W-bit	A-bit	ViT-L	Swin-T	Swin-S	ResNet-18
Full Precision	32	32	85.59	81.2	83.23	71.61
BLOB-Q	8 MP	8 MP	85.82	81.12	83.11	71.74
BLOB-Q	6 MP	6 MP	85.77	80.91	82.82	70.23
BLOB-Q	4 MP	4 MP	77.09	76.41	80.53	63.02

D.6 SCALABILITY TO LANGUAGE MODELS

Aside from vision tasks, we also explored the performance of BLOB-Q on language models. We evaluated on a decoder-only model OPT-350M Zhang et al. (2022). From the results on text generation task shown in Tab. 9, BLOB-Q also preserve generation quality on wikitext2 dataset with bearable drop in perplexity (PPL). The performance could be further boosted by incorporating more LLM-specific quantizer such as GPTQ [3] quantizer as well as additional finetuning.

Table 9: BLOB-Q on language model OPT-350M results on text generation.

Model	Method	Bit-width	Wikitext2 PPL ↓
OPT-350M	FP	32	23.5607
OPT-350M	BLOB-Q	8 MP	25.7557
OPT-350M	BLOB-Q	6 MP	25.9515
OPT-350M	BLOB-Q	4 MP	34.6864

D.7 GENERALIZED RESULTS ON DOWNSTREAM TASK

To evaluate generalizability of the proposed BLOB-Q beyond standard image classification task, we conduct PTQ experiments on three pretrained Mask-RCNN models with different ViT backbones: Swin-T and Swin-S, obtained by varying training settings such as with multi-scale training data.

Table 10: Detection and segmentation performances under different bit widths on COCO2017 *val* dataset. [†] means models with multi-scaled training. MaskRCNN-SWIN-T[†] is evaluated.

Method	Bit Width	Model Size (MB)	Bbox mAP			Mask mAP		
			mAP	mAP@50	mAP@75	mAP	mAP@50	mAP@75
Original	32	182.59	42.7	65.2	46.8	39.3	62.2	42.2
BLOB-Q	6 (MP)	34.24	41.8	64.4	46.5	38.9	61.5	41.9
BLOB-Q	5 (MP)	29.53	39.6	63.1	43.0	37.3	60.1	39.6
BLOB-Q	4 (MP)	22.82	32.9	56.0	35.3	33.2	53.8	35.4

Table 11: Detection and segmentation performances under different bit widths on COCO2017 *val* dataset. [†] means models with multi-scaled training. MaskRCNN-SWIN-T is evaluated.

Method	Bit Width	Model Size (MB)	Bbox mAP			Mask mAP		
			mAP	mAP@50	mAP@75	mAP	mAP@50	mAP@75
Original	32	182.59	46.0	68.1	50.4	41.6	65.2	44.7
BLOB-Q	6 (MP)	34.24	45.0	67.4	49.3	41.1	64.6	44.2
BLOB-Q	5 (MP)	29.53	42.5	64.9	47.2	39.9	62.3	43.4
BLOB-Q	4 (MP)	22.82	26.2	46.8	26.5	27.7	45.8	29.5

Table 12: Detection and segmentation performances under different bit widths on COCO2017 *val* dataset. [†] means models with multi-scaled training. MaskRCNN-SWIN-S is evaluated.

Method	Bit Width	Model Size (MB)	Bbox mAP			Mask mAP		
			mAP	mAP@50	mAP@75	mAP	mAP@50	mAP@75
Original	32	264.16	48.2	69.9	52.8	43.2	67.1	46.0
BLOB-Q	6 (MP)	49.53	47.2	69.4	51.8	42.7	66.6	46.0
BLOB-Q	5 (MP)	41.28	46.0	67.9	51.0	42.0	65.2	45.3
BLOB-Q	4 (MP)	33.02	27.9	45.4	31.0	28.4	44.4	31.3

We first demonstrate the effectiveness of the proposed BLOB-Q on extremely low bit rates for three ViT-based detection models. We observe a consistent advantage over the uniform quantization method on detection scores across different models and bit rate targets (4 to 6-bit). In Table 10, Table 11, and 12, we achieved near-FP (full-precision) model performance on 6-bit, where three models only drop 0.9–1% on bbox mAP and < 1% drop on mask mAP from their respective FP models. On 6-bit BLOB-Q induces less detection error compared to the uniform baseline, while on lower bit rates like 5-bit, the performance gap becomes even larger, where we are still able to boost the mAP to the level comparable to FP models. Particularly, we observe a significant boost on extremely low bit such as 4-bit. Compared to the three 4-bit models obtained by uniform quantization where all detection scores are extremely poor, we significantly improve the performances to acceptable ranges using BLOB-Q. Specifically, we improve the bbox mAP score by around 22 $\times$, from mere 1.4 to 33.0 by employing BLOB-Q. The other two models also show similar improvement.



Figure 11: Visualization of the detection results. First row shows uniform INT4 quantization results and second row shows F-LBQ INT4 results.

Qualitative Results. Fig. 11 shows the qualitative detection results on COCO 2017 validation images. At bit rates as low as 4-bit, we observe that although there are some cases of incomplete bounding boxes (e.g., the boat in the second image), the detection quality is generally decent and drastically better than the baseline results.

E QUANTIZATION WITH MIXED-PRECISION ON HARDWARE

In practice, there are three stages of data movement for Mixed-Precision Quantization on hardware: global memory stage, buffer stage, and processing stage (or compute unit stage). At global memory stage, weights are stored and packed based on bit-width to maximize storage efficiency. Then at buffer stage, weights will be unpacked and aligned before sending to compute unit. For example, if a) memory has a 16-bit word, b) weights are quantized to 4-bit, and c) 8-bit int is used in compute logic, then a 16-bit word will be unpacked to 4 weights and each weight will be padded to a 8-bit integer with zero-padding. Last, the unpacked and padded weights will be sent to compute units. The hardware dynamically processes weights with different bit widths based on the memory packing and alignment mechanisms.

Here is an example when weights are quantized to 4-bit, with bitwise operations involved in the process. Assume the length of a memory word is 16-bit. 4 weights are packed into a single word. Let $w_1 = 0xA$ (1010 in binary), $w_2 = 0x4$ (0100 in binary), $w_3 = 0x5$ (0101 in binary), and $w_4 = 0x7$ (0111 in binary) denote 4 weights. They will be packed into a word:

$$\text{packed_word} = [w_1 w_2 w_3 w_4] = 1010\ 0100\ 0101\ 0111 \text{ (packed into a 16 bit word)}.$$

When the packed data is fetched from global memory to buffer, bitwise operations are used to extract the individual values from the packed word. For example, to extract the 4-bit weight w_1 , perform a bitwise AND operation with a mask and shift the result: mask = 0xF000 (in binary: 1111 0000 0000 0000),

$$\text{extract } w_1 : \text{packed_word} \& \text{mask} >> 12.$$

If 8-bit int is used in compute logic, each weight will be padded to a 8-bit integer,

$$\text{padding} : w_1 = 0000\ 1010, w_2 = 0000\ 0100, w_3 = 0000\ 0101, w_4 = 0000\ 0111.$$

The padded weights will be then sent to compute unit.

We implemented 8-bit and 4-bit matrix multiplication and convolution for GPU acceleration based on Nvidia’s CUTLASS template library² as CUDA extensions to python-based main process, and replaced them with Pytorch’s default convolution and linear module and tested the speed up on GPUs. Since Nvidia’s TensorCore is supported on Ampere machines with the latest versions of the CUDA compiler, our implementation leverages the TensorOp GEMMs provided by CUTLASS. For INT8 and INT4 convolution, we adopt the implicit convolution for signed 8-bit (s8) and signed 4-bit (s4) respectively. For INT4 operations, since Pytorch does not support s4 data type, we first perform packing operations for both inputs and weights (quantized to effective 4-bit but contained in torch.int8 data type), which packs two adjacent s4 values into a byte, then transfer them from pytorch in s8 type to C++. The packed s8 data are further casted into s4 type using reinterpret_cast<cutlass::int4b_t *>. We implemented the packing for two s4 values f_1, f_2 as below:

$$\text{pack}(f_1, f_2) = f_1 \ll 4 + f_2 + 16 \cdot \mathbb{I}(f_2 < 0). \quad (19)$$

This packing process is implemented in parallel to reduce the overhead when applied to tensors.

To efficiently utilize CUDA kernels, CUTLASS adopts 128-bit vector memory access. As a result, INT8 convolution and matmul require the input channels to be a multiple of 16, and the INT4 ones require a multiple of 32 instead. Therefore, we pad the input channels to meet this alignment requirement for INT4 or INT8 inference. We then utilize the implemented INT4 and INT8 operations to perform mixed-precision quantization on Vision Transformers (ViTs). We constrain the quantization parameters and bit allocation options to only 4 and 8 bits. We evaluate such constrained mixed-precision on DeiT-B when quantized to 6 bits on average. Tab. 13 shows the performance of the inference time on DeiT-B and the speedup under different batch sizes. The Mixed-Precision Quantization at 6 bits on average with the above implementation can bring up to $1.6\times$ speedups.

²<https://github.com/NVIDIA/cutlass>

Table 13: Benchmark performances of Mixed-Precision Quantization at 6 bits on DeiT-B inference using INT8 and INT4 GEMM on Nvidia A100.

Batch Size	Time Cost (ms)			
	32	64	128	256
FP32	2.38	2.42	2.40	2.38
6 Bits	1.45	1.44	1.52	1.54
Speedup	1.64	1.68	1.58	1.55

E.1 DISCUSSION ON MEMORY BANDWIDTH

To explore the empirical benefit of adopting a mixed-precision quantization recipe compared to uniform quantization, we conducted further comparison of our mixed-precision approach with uniform quantization method, regarding the hardware performance. Firstly, we compare the inference latency with the SOTA uniform quantization method (RepQ-ViT) on an 80GB-A100 GPU platform, with a high memory bandwidth at 2.039 TB/s. Tab. 14 summarizes the inference speed resulting from our mixed-precision quantization approach. In Tab. 14, we choose the settings of two models for both approaches where accuracy loss is less than 0.5%. For all settings, we show the average latency on 8192 test images with batchsize of 64. The results show we gain latency speedups from uniform quantization up to 19%, without hurting accuracy.

Table 14: GPU latency (ms) of mixed-precision inference compared to uniform quantization.

Method	Latency on DeiT-S	Latency on DeiT-B
FP	65.4	179
RepQ-ViT (Uniform)	10.5	113
Ours (MP)	8.78	102

E.2 DISCUSSION ON THE EFFECT ON MEMORY BANDWIDTH

We also evaluated the hardware performance on two hardware platforms: TPU V3 and Eyeriss, which are the neural-network-target hardware accelerators targeted for edge devices. The memory bandwidths on these platforms are much less than A100 GPU, at 12.5GB/s and 1GB/s for TPU and Eyeriss respectively. Using the standard SCALE-sim simulator, we simulated the inference speed on DeiT-S and DeiT-B, when deployed onto the two hardware platforms. Tab. 15 summarizes the inference speed resulting from our mixed-precision quantization approach, compared with RepQ-ViT. In Tab. 15, we also choose the settings of both approaches where accuracy loss is less than 0.5%. As we can see, our mixed-precision quantization approach can further reduce the inference time noticeably by up to 25%, compared with the uniform quantization approach. The improvement comes from the low-bit quantization of our approach. As the proposed mixed-precision quantization can quantize models to lower bit widths, it thus leads to smaller model size and less memory read/write time.

Table 15: Hardware performance on Edge devices with different memory bandwidths.

Method	Latency on DeiT-S (ms)		Latency on DeiT-B (ms)	
	TPU V3	Eyeriss	TPU V3	Eyeriss
FP	10.8	646.4	18.6	1113.2
RepQ-ViT (Uniform)	9.1	449.7	15.7	774.4
Ours (MP)	6.8	337.3	12.9	580.8