Instance-Aware Group Quantization for Vision Transformers

Supplementary Material

A. More implementation details

A.1. Weight quantization

For weight quantization, we exploit a distinct quantizer for each output channel, following [7]. We designate the upper and lower bounds of weight quantizers with (100-ϵ)-th and ϵ-th percentiles of weight values in each output channel, where ϵ is a hyperparameter.

A.2. Hyperparameter settings

We set ϵ as 5e-2 and 1e-3 for 4/4-bit and 6/6-bit settings, respectively. The values of ϵ are found using a grid search, such that KL divergence between predictions of full-precision and quantized ViT-T from calibration data is minimized. In addition, we set N_{iter} and T in Algorithm 1 as 300 and 75, respectively, which are large enough for our algorithm to be converged.

A.3. Perturbation metric for Mask R-CNN models

We use the Mask R-CNN [5] and Cascade Mask R-CNN [2] with Swin transformer [9] for the tasks of object detection and instance segmentation. The perturbation metric in Eq. (9) cannot be applied directly to these models, typically having three outputs for each region of interest (RoI): (1) a class probability, (2) a bounding-box regression offset, and (3) a binary mask. The work of [5] proposes a multi-task loss on each sampled RoI as $L = L_{cls} + L_{box} + L_{mask}$, where the loss terms L_{cls}, L_{box}, and L_{mask} are defined as the discrepancy between the outputs (*i.e.*, class probability, bounding-box offset, and binary mask) and corresponding ground-truth labels. Note that L_{box} and L_{mask} are defined only on the bounding-box and mask corresponding to the ground-truth class for each RoI, respectively. Refer to [5] for more details.

For Mask R-CNN and Cascade Mask R-CNN models, we modify the perturbation metric for each RoI in Eq. (9) as follows:

$$\psi(g,l) = D_{\text{KL}}(y_l || y_l^g) + |b_l - b_l^g| + |m_l - m_l^g|, \quad \text{(A)}$$

where y_l^g, b_l^g, and m_l^g are predictions of class probability, bounding-box offset, and binary mask, respectively, for a model in which the l-th layer is quantized with a group size of g. We denote by y_l, b_l, and m_l the predictions of the model, where the l-th layer is left to be full-precision. Note that we consider only the distances for the ground-truth class during the computation of $|b_l - b_l^g|$ and $|m_l - m_l^g|$, following [5]. We use Eq. (A) to solve for Eq. (10) in the main paper.

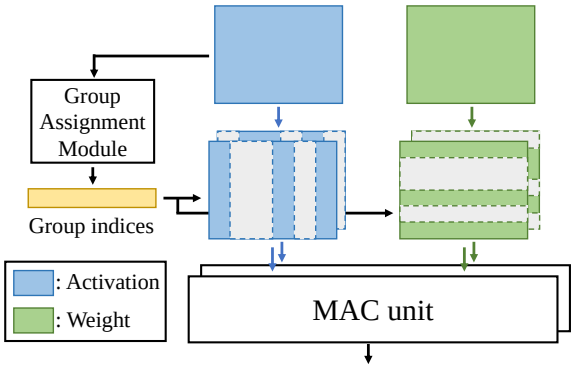

Figure A. Our instance-aware group quantization framework could be implemented using existing DNN accelerators, with a slight modification from the implementation of [3]. One might leverage a group assignment module that computes the group indices for each channel using Eq. (5), and the resulting indices are used to select channels that belong to each group during matrix multiplication. Corresponding rows of weight buffers are also selected.

B. Compatibility with existing hardwares

We believe that IGQ-ViT could be efficiently implemented using existing neural network accelerators, with a slight modification from the implementation of VS-quant [3]. Specifically, [3] divides channels of activations into a number of groups, and activation values assigned to each group are processed with separate multiply-accumulate (MAC) units. The outputs from each MAC unit are then scaled with different quantization parameters. In contrast, IGQ-ViT dynamically splits channels according to their statistical properties for each input instance. Compared to [3], IGQ-ViT requires additional computations, which includes computing the min/max values of each channel, and assigning channels to quantizers with the minimum distance. To address this, one might leverage a group assignment module that computes the min/max values of each channel, followed by obtaining group indices using Eq. (5) (Fig. A). The resulting indices are then used to select channels that belong to each group. Finally, a separate MAC unit is applied for activation values within each group, which contains a single quantization parameter. Note that computing the group indices for each channel is computationally cheap in terms of BOPs (See Table 1 in the main paper), and using an indexing scheme for efficient computation is a common practice in real devices. For example, the work of [12] implements a module providing indices to dynamically detect sparsity patterns of weight and activation values in each group. It then uses the indices to skip groups of zero-valued weights and activations for effi-

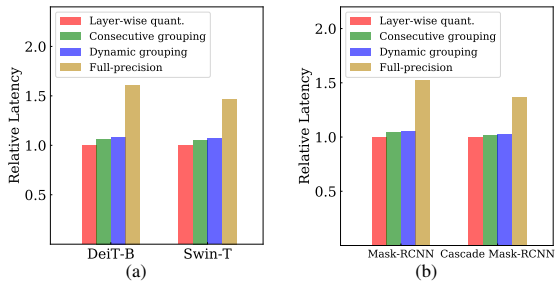

Figure B. Run-time latency for the tasks of (a) image classification and (b) object detection/instance segmentation using NVIDIA RTX 3090. For group quantization, we use a group size of 8 for all layers. For Mask-RCNN models, we use Swin-T as the backbone.

ciency (See Fig. 15.2.3 in [12]).

C. Latency on practical devices

To further validate the efficiency of IGQ-ViT, we conduct a simulation in PyTorch to compare the latencies between prior group quantization techniques [1, 3, 11] and ours. A key challenge is that most quantization methods exploit a fake quantization approach, following [6], which mimics the quantization process by discretizing the network's weights and activations into a finite set of floating-point values, thereby serving a surrogate for the true quantization process. This approach is inappropriate for estimating the latency on real hardwares as it does not change the actual precision of the data, but merely introduces the concept of lower precision during calibration. Accordingly, we directly convert the data formats of weights and activations into 8-bit representations to measure the latency more accurately. Specifically, we simulate IGQ-ViT for linear operations using Eq. (7), which requires low-bit matrix multiplication between weights and activations within each group, along with the summation of outputs for each group in full-precision. Since PyTorch does not support convolutional or linear layers that takes low-bit matrices as input, we have implemented their 8-bit counterparts. Note that we have implemented the group assignment algorithm (*i.e.*, Eq. (5)) in full-precision.

We compare in Fig. B the run-time latency of IGQ-ViT with its variants. We can see that IGQ-ViT introduces marginal overhead compared to layer-wise quantization, and consecutive grouping strategy, while achieving high quantization performances (See Table 5 in the main paper). This suggests that dynamic grouping of channels have a limited impact on actual latency.

D. Application to DETR

We show in Table A the results of quantizing a DETR model with a ResNet-50 [4] backbone on COCO [8]. To the best of our knowledge, PTQ for ViT [10] is the only PTQ method that provides quantization results for a DETR model under 6/6-bit setting. We can see that IGQ-ViT outperforms it by 0.8% for a group size of 12.

Table A. Results of quantizing a DETR model with a ResNet-50 [4] backbone on COCO [8].

Method	**#bits (W/A)**	**Box AP (Latency)**
Full-precision	32/32	42.0
PTQ for ViT [10]	6/6	40.5
Ours (#groups=8)	6/6	41.1
Ours (#groups=12)	6/6	**41.3**

References

[1] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Understanding and overcoming the challenges of efficient transformer quantization. *EMNLP*, 2021.

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

[3] Steve Dai, Rangha Venkatesan, Mark Ren, Brian Zimmer, William Dally, and Brucek Khailany. Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. *Proceedings of Machine Learning and Systems*, 2021.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[6] Yuhang Li, Mingzhu Shen, Jian Ma, Yan Ren, Mingxin Zhao, Qi Zhang, Ruihao Gong, Fengwei Yu, and Junjie Yan. Mqbench: Towards reproducible and deployable model quantization benchmark. *NeurIPS*, 2021.

[7] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. *ICCV*, 2023.

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

[9] 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*, 2021.

[10] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. In *NeurIPS*, 2021.

[11] 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 *AAAI*, 2020.

[12] Jinshan Yue, Xiaoyu Feng, Yifan He, Yuxuan Huang, Yipeng Wang, Zhe Yuan, Mingtao Zhan, Jiaxin Liu, Jian-Wei Su, Yen-Lin Chung, et al. 15.2 a 2.75-to-75.9 tops/w computing-in-memory nn processor supporting set-associate block-wise zero skipping and ping-pong cim with simultaneous computation and weight updating. In *ISSCC*, 2021.