FQ-ViT: Post-Training Quantization for Fully Quantized Vision Transformer

Yang Lin∗†, Tianyu Zhang∗, Peiqin Sun‡, Zheng Li and Shuchang Zhou
MEGVII Technology
linyang.zhh@gmail.com, {zhangtianyu, sunpeiqin, lizheng02, zsc}@megvii.com

Abstract

Network quantization significantly reduces model inference complexity and has been widely used in real-world deployments. However, most existing quantization methods have been developed mainly on Convolutional Neural Networks (CNNs), and suffer severe degradation when applied to fully quantized vision transformers. In this work, we demonstrate that many of these difficulties arise because of serious inter-channel variation in LayerNorm inputs, and present, Power-of-Two Factor (PTF), a systematic method to reduce the performance degradation and inference complexity of fully quantized vision transformers. In addition, observing an extreme non-uniform distribution in attention maps, we propose Log-Int-Softmax (LIS) to sustain that and simplify inference by using 4-bit quantization and the BitShift operator. Comprehensive experiments on various transformer-based architectures and benchmarks show that our Fully Quantized Vision Transformer (FQ-ViT) outperforms previous works while even using lower bit-width on attention maps. For instance, we reach 84.89% top-1 accuracy with ViT-L on ImageNet and 50.8 mAP with Cascade Mask R-CNN (Swin-S) on COCO. To our knowledge, we are the first to achieve lossless accuracy degradation (∼1%) on fully quantized vision transformers. The code is available at https://github.com/megvii-research/FQ-ViT.

1 Introduction

Transformer-based architectures have achieved competitive performance in various computer vision (CV) tasks, including image classification [Dosovitskiy et al., 2021; Touvron et al., 2021], object detection [Carion et al., 2020; Liu et al., 2021a], semantic segmentation [Zheng et al., 2021] and so on. Compared to the CNN counterparts, transformers usually have more parameters and higher computational costs. For example, ViT-L has 307M parameters and 190.7G FLOPs, reaching the accuracy of 87.76% in ImageNet with large-scale pre-training. However, the large number of parameters and computational overhead of transformer-based architectures present a challenge when deployed to resource-constrained hardware devices.

To facilitate deployment, several techniques have been proposed, including quantization [Zhou et al., 2016; Nagel et al., 2020; Shen et al., 2020; Liu et al., 2021b], pruning [Han et al., 2016], distillation [Jiao et al., 2020] and adaptation of architecture design [Graham et al., 2021]. We focus on the quantization technique in this paper and note that pruning, distillation, and architecture adaptation are orthogonal to our work and can be combined.

Most existing quantization approaches have been designed and tested on CNNs and lack proper handling of transformer-specific constructs. Previous work [Liu et al., 2021b] finds there have been a significant accuracy degradation when quantizing LayerNorm and Softmax of vision transformers. In this case, the models are not fully quantized, resulting in the need to retain floating-point units in the hardware, which will bring large consumption and significantly reduce the in-
ference speed [Lian et al., 2019]. So we revisit these two exclusive modules of the vision transformers and discover the reasons of degradation. Firstly, we find a serious inter-channel variation of LayerNorm inputs, which some channel ranges even exceed 40 × of the median. Traditional methods cannot handle such large fluctuations of activations, which will lead to large quantization error. Secondly, we find that the values of the attention map have an extreme non-uniform distribution, with most values clustered in 0 ∼ 0.01, and a few high attention values close to 1.

Based on the analysis above, we propose Power-of-Two Factor (PTF) to quantize the inputs of the LayerNorm. In this way, the quantization error is greatly reduced, and the overall computational efficiency is the same as that of layer-wise quantization thanks to the BitShift operator. In addition, we propose Log-Int-Softmax (LIS), which provides higher quantization resolution for small values and presents a more efficient integer inference for Softmax. Combining these methods, we are the first work to achieve post-training quantization for fully quantized vision transformers. As shown in Figure 1, our method significantly improves the performance of fully quantized vision transformers and obtains comparable accuracy with full precision counterparts.

Our contributions are four-fold:

- We revisit the fully quantized vision transformers and attribute the accuracy degradation to the serious inter-channel variation in LayerNorm inputs. Meanwhile, we observe an extreme non-uniform distribution of attention maps, resulting in another part of the quantization error.
- We propose Power-of-Two Factor (PTF), a simple yet efficient post-training method that can achieve accurate quantization on LayerNorm inputs with only one layer-wise quantization scale.
- We propose Log-Int-Softmax (LIS), a novel method that can perform 4-bit quantization on attention maps. With LIS, we can store attention maps on an aggressively low-bit and replace multiplication with BitShift operator. We achieve integer-only inference on Softmax modules, significantly reducing the inference consumption.
- We conduct extensive experiments on image classification and object detection with various transformer-based architectures. The results show that our fully quantized vision transformers with 8-bit weights/activations and 4-bit attention maps, can achieve comparable performance to floating-point versions.

2 Related Work

2.1 Vision Transformer

Recently, transformer-based architecture shows great power in CV tasks. Emerging works based on ViT [Dosovitskiy et al., 2021] demonstrated the effectiveness across all vision tasks such as classification [Touvron et al., 2021], detection [Carion et al., 2020] and segmentation [Zheng et al., 2021]. The newly proposed Swin Transformer [Liu et al., 2021a] even surpasses the state-of-the-art CNNs on almost traditional CV tasks, presenting strong expressive and generalization capability of transformer. However, these high-performing vision transformers are attributed to the large number of parameters and high computational overhead, limiting their adoption. Therefore, innovating a smaller and faster vision transformer becomes a new trend. Le-ViT [Graham et al., 2021] makes progress in faster inference with down-sampling, patch descriptors, and a redesign of Attention-MLP block. DynamicViT [Rao et al., 2021] presents a dynamic token sparsification framework to prune redundant tokens progressively and dynamically, achieving competitive complexity and accuracy trade-off. Evo-ViT [Xu et al., 2021] proposes a slow-fast updating mechanism that guarantees information flow and spatial structure, trimming down both the training and inference complexity. While the above works focus on efficient model designing, this paper boosts the compression and acceleration in the track of quantization.

2.2 Network Quantization

Current quantization methods can be divided into two categories: Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ). QAT [Zhou et al., 2016; Jacob et al., 2018] depends on training to achieve aggressively low-bit (e.g. 2-bit) quantization and promising performance, while it often requires a high-level expert knowledge and huge GPU resources for training or fine-tuning. To reduce above costs of quantization, PTQ, which is training-free, has received more widespread attention and lots of excellent works arise. OMSE [Choukroun et al., 2019] proposes to determine the value range of activation by minimizing the quantization error. AdaRound [Nagel et al., 2020] presents a novel rounding mechanism to adapt the data and the task loss. Besides works above specific to CNNs, Liu et al. proposes a post-training quantization method for vision transformers with similarity-aware and rank-aware strategies. However, this work does not quantize Softmax and LayerNorm modules, resulting in an incomplete quantization. In our FQ-ViT, we aim to implement an accurate, fully quantized vision transformer under the PTQ paradigm.

3 Proposed Method

In this section, we will introduce our proposed approach in detail. First in Section 3.1, we present the preliminary of network quantization. Then in Section 3.2 and 3.3, we analyze the reasons of degradation in fully quantized vision transformers and propose two novel quantization methods, Power-of-Two Factor (PTF) and Log-Int-Softmax (LIS), for LayerNorm and Softmax.

3.1 Preliminary

In this section, we explain the notations of network quantization. Assuming the quantization bit-width is b, the quantizer Q(X|b) can be formulated as a function that maps a floating-point number X ∈ R to the nearest quantization bin:

\[ Q(X|b) : R \rightarrow q, \quad q = \begin{cases} \{ -2^{b-1}, \ldots, 2^{b-1} - 1 \} & \text{Signed}, \\ \{ 0, 1, \ldots, 2^b - 1 \} & \text{Unsigned}. \end{cases} \]
There are various quantizer $Q(X|b)$, where uniform [Jacob et al., 2018] and log2 [Cai et al., 2018] are typically used.

**Uniform Quantization** is well supported on most hardware platforms. Its quantizer $Q(X|b)$ can be defined as:

$$Q(X|b) = \text{clip}(\frac{X}{s} + zp, 0, 2^b - 1),$$

where $s$ (scale) and $zp$ (zero-point) are quantization parameters determined by the lower bound $l$ and the upper bound $u$ of $X$, which are usually minimum and maximum values:

$$l = \min(X), \quad u = \max(X),$$

$$s = \frac{u - l}{2^b - 1}, \quad zp = \text{clip}(\frac{1}{s}l, 0, 2^b - 1).$$

**Log2 Quantization** converts the quantization process from linear to exponential variation. Its quantizer $Q(X|b)$ can be defined as:

$$Q(X|b) = \text{sign}(X) \cdot \text{clip}(\left\lfloor -\log_2 \frac{|X|}{\max(|X|)} \right\rfloor, 0, 2^b - 1).$$

In this paper, to achieve a fully quantized vision transformer, we quantize all modules, including Conv, Linear, MatMul, LayerNorm, Softmax, etc. Especially, uniform Min-Max quantization is used for Conv, Linear and MatMul modules and our following methods are used for LayerNorm and Softmax.

### 3.2 Power-of-Two Factor for LayerNorm Quantization

During inference, LayerNorm [Ba et al., 2016] computes the statistics $\mu_X, \sigma_X$ in each forward step and normalizes input $X$. Then, affine parameters $\gamma, \beta$ rescale the normalized input to another learned distribution. The above process can be written as:

$$\text{LayerNorm}(X) = \frac{X - \mu_X}{\sqrt{\sigma_X^2 + \epsilon}} \cdot \gamma + \beta.$$  

Unlike BatchNorm [Ioffe and Szegedy, 2015], commonly used in CNNs, LayerNorm cannot be folded into the previous layer due to its dynamic computational property, so we have to quantize it separately. However, we observe a significant performance degradation while applying post-training quantization on it. Looking into the inputs of LayerNorm layers, we find there is a serious inter-channel variation. Figure 2 presents the channel-wise ranges of activation in the last LayerNorm layer. In addition, we also display the cases of ResNets [He et al., 2016] for comparison. Considering that there is no LayerNorm in ResNets, we choose the activations at the same position (fourth stage’s outputs) to exhibit.

It is observed that the channel-wise ranges fluctuate more wildly in vision transformers than those in ResNets. For instance, the maximum range/median range of ResNet152 is only 21.6/4.2, while it goes up to 622.5/15.5 in Swin-B. Based on such extreme inter-channel variation, layer-wise quantization, which applies the same quantization parameters to all channels, will lead to an intolerable quantization error. A possible solution is using group-wise quantization [Shen et al., 2020] or channel-wise quantization [Li et al., 2019], which assign different quantization parameter to different group or channel. However, these will still induce the calculation of mean and variance in the floating-point domain, resulting in a high hardware overhead.

In this paper, we propose a simple yet efficient method, Power-of-Two Factor (PTF), for LayerNorm quantization. The core idea of PTF is to equip different channels with different factors, rather than different quantization parameters. Given the quantization bit-width $b$, the input activation $X \in \mathbb{R}^{N \times L \times C}$, the layer-wise quantization parameters $s, zp \in \mathbb{R}^1$, and the PTF $\alpha \in \mathbb{N}^C$, then the quantized activation $X_Q$ can be formulated as:

$$X_Q = Q(X|b) = \text{clip}(\frac{X}{2^\alpha s} + zp, 0, 2^b - 1),$$

with

$$s = \frac{\max(X) - \min(X)}{2^b - 1},$$

$$zp = \text{clip}(\left\lfloor -\frac{\min(X)}{2^\alpha s} \right\rfloor, 0, 2^b - 1),$$

$$\alpha_c = \arg\min_{\alpha_c \in \{0, 1, \ldots, K\}} \left\| X_c - \left\lfloor \frac{X_c}{2^{\alpha_c s}} \right\| 2^\alpha s \right\|_2.$$
all models, we set $K = 3$ as default. Detailed experiments can be found in supplementary materials.

At this point, each channel has its own Power-of-Two Factor $\alpha$ and layer-wise parameters $s, zp$. During inference, layer-wise parameters $s$ and $zp$ can be extracted, so the computation of $\mu, \sigma$ could be done in the integer domain rather than floating-point, which reduces the energy and area costs [Lian et al., 2019]. Meanwhile, thanks to the nature of powers of two, PTF $\alpha$ can be efficiently combined with layer-wise quantization by BitShift operator, avoiding floating-point calculations of group-wise or channel-wise quantization. The whole process can be processed with two phases:

Phase 1: Shift the quantized activation with Power-of-Two Factor $\alpha$:
\[
\hat{X}_Q = (X_Q - zp) << \alpha.
\] (12)

Phase 2: Calculate the mean and variance based on the shifted activation $\hat{X}_Q$:
\[
\mu(X) \approx \mu(2^\alpha s \cdot (X_Q - zp)) = s \cdot \mu(\hat{X}_Q),
\] (13)
\[
\sigma(X) \approx \sigma(2^\alpha s \cdot (X_Q - zp)) = s \cdot \sigma(\hat{X}_Q).
\] (14)

### 3.3 Log-Int-Softmax for Softmax Quantization

Multi-head Self-Attention (MSA) is one of the most important components in transformer-based architectures, while it is considered the most resource-intensive due to the quadratic complexity to the number of token, the division of the image resolution by the patch size. As model performance proved to benefit from higher resolution and smaller patch size [Dosovitskiy et al., 2021], when the increasing resolution and reducing patch size, the storage and computation of attention maps become the bottleneck which directly affect the throughput and latency of inference. Therefore, smaller attention maps and more efficient inference become an urgent need.

#### Log2 Quantization for Attention Map

In order to compress the attention maps to smaller size and speed up the inference, we quantize attention maps to lower bit-width. As experimenting on quantization of attention maps from 8-bit to 4-bit with uniform quantization, all vision transformers show severe performance drop. For example, DeiT-T only results in 8.69% top-1 accuracy on ImageNet with 4-bit uniform quantized attention maps, decreasing 63.05% from 8-bit case.

Inspired by the idea of sparse attention in DynamicViT [Rao et al., 2021], we probe into the distribution of attention maps, as Figure 3 shows. We observe a distribution centering at a fairly small value, while only a few outliers have larger values close to 1. Averaging on all attention maps in ViT-L, about 98.8% of values are smaller than 1/16. Compared with 4-bit uniform quantization which only assigns 1 bin for such many values, log2 method can allocate 12 bins for them. Moreover, following the purpose of ranking-aware loss [Liu et al., 2021b], log2 quantization can retain much order consistency between full precision and quantized attention maps. Consequently, we save the extreme degradation in 4-bit quantization of attention maps and achieve equal performance as 8-bit uniform quantization with 50% less memory footprint.

Log2 quantization proved to be suitable combining with MSA from two aspects. Firstly, comparing to Equation (6), the fixed output range $(0, 1)$ of Softmax makes the log2 function calibration-free:
\[
\text{Attn}_Q = Q(\text{Attn}_b) = \text{clip}([- \log_2(\text{Attn})], 0, 2^b - 1).
\] (15)

This ensures that the quantization of attention maps will not be affected by the fluctuation of calibration data.

Secondly, it introduces the merits of converting the MatMul to BitShift between the quantized attention map $(\text{Attn}_Q)$ and values $(V_Q)$ as:
\[
\text{Attn} \cdot V_Q = 2^{-\text{Attn}_Q} \cdot V_Q = V_Q \gg \text{Attn}_Q
\] (16)
\[
= \frac{1}{2^N} \cdot (V_Q << (N - \text{Attn}_Q)),
\] (17)

with $N = 2^b - 1$. Noticing that directly right shift $V_Q$ with the results of $\text{Attn}_Q$ may lead to severe truncation error. We...
use \((N - \text{Attn}_Q)\) as the quantized output with a scale equaling \(1/2^N\), which ensures a left-shift operation to prevent from truncation error.

### Integer-only Inference

Previous works [Liu et al., 2021b] chose to not quantize Softmax because the negligibility of calculation amount in Softmax, and quantization may lead to significant accuracy degradation. However, data moving between CPU and GPU/NPU, doing dequantization and requantization, will induce great difficulties in hardware design, which is not a negligible consumption.

Combining log2 quantization with \(i\cdot\exp\) [Kim et al., 2021], which is a polynomial approximation of exponential function, we propose Log-Int-Softmax, an integer-only, faster, low consuming Softmax:

\[
\exp(s \cdot X_Q) \approx s' \cdot i\cdot\exp(X_Q), \quad (18)
\]

\[
\text{LIS}(s \cdot X_Q) = N - \log_2\left[\frac{\sum i\cdot\exp(X_Q)}{i\cdot\exp(X_Q)}\right], \quad (19)
\]

with \(N = 2^b - 1\). An integer log2 function can be easily implemented by using BitShift to find the first bit index where the value is 1 (we call it \text{FindFirstOne} function), and adding the value of bit right behind that. Detailed derivation can be found in the supplementary materials.

The difference between normal MSA and our method are shown in Figure 4, with the data type of every stage labeled. In the multi-head self-attention with unquantized Softmax shown on the left, the matrix multiplication of queries (Q) and keys (K) needs to be dequantized to full precision before Softmax, and requantized after it. When our Log-Int-Softmax adopted, shown on the right, the entire data type can be in pure integer, with quantization scale individually and paralleling calculated. It is worth noting that LIS uses an aggressively 4-bit representation on attention maps, which significantly reduces memory footprint.

### 4 Experiments

In this section, we present experimental results on vision transformers for image classification and object detection. We state the detailed experimental configuration firstly and then exhibit the comparison of our method with existing post-training quantization methods in ImageNet [Krizhevsky et al., 2012] and COCO [Lin et al., 2014] benchmarks. In the end, ablation studies are conducted to evaluate the effectiveness of Power-of-Two Factor (PTF) and Log-Int-Softmax (LIS).

#### 4.1 Implementation Details

We randomly sample 1000 training images from ImageNet or COCO as the calibration data, and use the validation set to evaluate performance. Apart from special notes, we perform symmetric channel-wise quantization for weights and asymmetric layer-wise quantization for activations. For a fair comparison, the quantization for weights is fixed as MinMax. The hyperparameter K in Power-of-Two Factor is set to 3.

#### 4.2 Comparison with State-of-the-art Methods

This paper employs several current post-training quantization methods, including MinMax, EMA [Jacob et al., 2018], Percentile [Li et al., 2019], OMSE [Choukroun et al., 2019], Bit-Split [Wang et al., 2020] and PTQ for ViT [Liu et al., 2021b]. Note that PTQ for ViT [Liu et al., 2021b] is closest to our work, however it does not quantize the LayerNorm and Softmax, while we quantize all modules.

#### Image Classification on ImageNet

To demonstrate the effectiveness of proposed methods, we conduct extensive experiments in ImageNet [Krizhevsky et al., 2012] with various vision transformers, i.e., ViT [Dosovitskiy et al., 2021], DeiT [Touvron et al., 2021], Swin Transformer [Liu et al., 2021a]. The overall top-1 accuracy results are reported in Table 1. It is obvious that all current methods can’t capture fully quantized vision transformers, while our FQ-ViT does it and achieves a nearly lossless quantization even with an aggressively low-bit on attention maps. Meanwhile, our FQ-ViT significantly exceeds PTQ for ViT [Liu et al., 2021b], whose LayerNorm and Softmax are not quantized. For instance, our FQ-ViT achieves 81.20% accuracy on DeiT-B in the case of all modules quantized to 8-bit, and it can still achieve 80.85% accuracy when the attention maps are compressed to 4-bit.

#### Object Detection on COCO

We also conduct experiments on the object detection benchmark COCO [Lin et al., 2014]. We choose Swin series [Liu et al., 2021b] for Table 1: Comparison of the top-1 accuracy with state-of-the-art methods on ImageNet dataset. * indicates that all LayerNorm and Softmax modules are not quantized.

| Method       | W/A/Attn | DeiT-T | DeiT-S | DeiT-B | ViT-B | ViT-L | Swin-T | Swin-S | Swin-B |
|--------------|----------|--------|--------|--------|-------|-------|--------|--------|--------|
| Full Precision | 32/32/32 | 72.21  | 79.85  | 81.85  | 84.53 | 85.81 | 81.35  | 83.20  | 83.60  |
| MinMax       | 8/8/8    | 70.94  | 75.05  | 78.02  | 23.64 | 3.37  | 64.38  | 74.37  | 25.58  |
| EMA [Jacob et al., 2018] | 8/8/8 | 71.17  | 75.71  | 78.82  | 30.30 | 3.53  | 70.81  | 75.05  | 28.00  |
| Percentile [Li et al., 2019] | 8/8/8 | 71.47  | 76.57  | 78.37  | 46.69 | 5.85  | 78.78  | 78.12  | 40.93  |
| OMSE [Choukroun et al., 2019] | 8/8/8 | 71.30  | 75.03  | 79.57  | 73.39 | 11.32 | 79.30  | 78.96  | 48.55  |
| Bit-Split* [Wang et al., 2020] | 8/8/8 | -      | 77.06  | 79.42  | -    | -     | -      | -      | -      |
| PTQ for ViT* [Liu et al., 2021b] | 8/8/8 | -      | 77.47  | 80.48  | -    | -     | -      | -      | -      |
| FQ-ViT       | 8/8/8    | 71.61  | 79.17  | 81.20  | 83.31 | 85.03 | 80.51  | 82.71  | 82.97  |
| FQ-ViT       | 8/8/4    | 71.07  | 78.40  | 80.85  | 82.68 | 84.89 | 80.04  | 82.47  | 82.38  |
| Method          | W/A/Attn | Mask R-CNN w/ Swin-S | Cascade Mask R-CNN w/ Swin-S |
|-----------------|----------|----------------------|-----------------------------|
| Full Precision  | 32/32/32 | 48.5                 | 52.0                        |
| MinMax          | 8/8/8    | 32.8                 | 35.2                        |
| EMA [Jacob et al., 2018] | 8/8/8    | 37.9                 | 40.4                        |
| Percentile [Li et al., 2019] | 8/8/8    | 41.6                 | 44.7                        |
| OMSE [Choukroun et al., 2019] | 8/8/8    | 42.6                 | 44.9                        |
| FQ-ViT          | 8/8/8    | 47.8                 | 51.4                        |
|                 | 8/8/4    | 47.2                 | 50.8                        |

Table 2: Comparison of the bbox mAP with state-of-the-art methods on COCO dataset.

| Method | PTF | LIS | BitOPs (G) | Acc. (%) |
|--------|-----|-----|------------|----------|
| Full Precision | -   | -   | -          | 84.53    |
| Baseline #8 | ×   | ×   | 1118.51    | 23.64    |
|            | ✓   | ×   | 1118.52    | 83.31    |
| Baseline #4 | ✓   | ✓   | 1118.34    | 7.96     |
|            | ✓   | ✓   | 1117.76    | 82.68    |

Table 3: Effect of the Power-of-Two Factor (PTF) and Log-Int-Softmax (LIS). We evaluate the performance of full precision and quantized ViT-B on ImageNet validation set. We choose MinMax with 8-bit weights and activations as "Baseline" and # indices the bit-width of attention maps.

4.3 Ablation Studies
To study the effect of our methods, Power-of-Two Factor (PTF) and Log-Int-Softmax (LIS), we fully quantize ViT-B under a variety of strategies and report the results in Table 3. We design two baselines. "Baseline #8" indices the model is fully quantized by MinMax with 8-bit weights, activations and attention maps, while "Baseline #4" has a lower bit-width (4-bit) on attention maps. From the results, we have several observations. Firstly, the model with PTF and LIS outperforms the baseline and achieves almost lossless accuracy. Secondly, thanks to the BitShift operator, PTF only introduces little amount of BitOPs, while LIS reduces that.

4.4 Visualization of Quantized Attention Map
We visualize the attention maps to see the difference between uniform quantization and our LIS as Figure 5 shows. When both using 8-bit, uniform quantization focuses on the high activation area, while LIS keeps more texture in the low activation area, which retains more relative rank of the attention map. This divergence does not make a big difference in the case of 8-bit. However, when quantized to lower bit-width, as the 6-bit and 4-bit cases show, uniform quantization sharply degrades and even deactivates all the attention area. On the contrary, LIS still presents acceptable performance similar to 8-bit.

5 Conclusions
In this paper, we propose a method to fully quantize vision transformers. Specifically, we propose Power-of-Two Factor (PTF) to deal with the serious inter-channel variations in the inputs of LayerNorm. In addition, we propose an integrated quantization solution Log-Int-Softmax (LIS) to implement 4-bit quantization of attention maps and utilize the BitShift operator instead of MatMul during inference, which effectively reduces hardware resource requirement. Experimental results show that our fully quantized vision transformers achieve comparable performance with the full precision models. Altogether, we provide a higher baseline for future works, hoping that FQ-ViT’s strong performance will encourage research into even lower bit-width quantization of vision transformers, which will boost their real-world adoptions.
References

[Ba et al., 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

[Cai et al., 2018] Jingyong Cai, Masashi Takemoto, and Hironori Nakajo. A deep look into logarithmic quantization of model parameters in neural networks. In Proceedings of the 10th International Conference on Advances in Information Technology, 2018.

[Carion et al., 2020] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In European Conference on Computer Vision. Springer, 2020.

[Choukroun et al., 2019] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In IEEE/CVF International Conference on Computer Vision Workshops. IEEE, 2019.

[Dosovitskiy et al., 2021] 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.

[Graham et al., 2021] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet’s clothing for faster inference. arXiv preprint arXiv:2104.01136, 2021.

[Han et al., 2016] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In International Conference on Learning Representations, 2016.

[He et al., 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning. PMLR, 2015.

[Jacob et al., 2018] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[Jiao et al., 2020] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling BERT for natural language understanding. In Findings of the Association for Computational Linguistics: EMNLP, 2020.

[Kim et al., 2021] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In International Conference on Machine Learning. PMLR, 2021.

[Krizhevsky et al., 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, 2012.

[Li et al., 2019] 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, 2019.

[Lian et al., 2019] Xiaocang Lian, Zhenyu Liu, Zhourui Song, Jiwu Dai, Wei Zhou, and Xiangyang Ji. High-performance fpga-based cnn accelerator with block-floating-point arithmetic. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019.

[Lin et al., 2014] 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 European Conference on Computer Vision. Springer, 2014.

[Liu et al., 2021a] 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. arXiv preprint arXiv:2103.14030, 2021.

[Liu et al., 2021b] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. In Thirty-Fifth Conference on Neural Information Processing Systems, 2021.

[Nagel et al., 2020] 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. PMLR, 2020.

[Rao et al., 2021] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. arXiv preprint arXiv:2106.02034, 2021.

[Shen et al., 2020] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In Proceedings of the AAAI Conference on Artificial Intelligence, 2020.

[Touvron et al., 2021] 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. PMLR, 2021.

[Wang et al., 2020] Peisong Wang, Qiang Chen, Xiangyu He, and Jian Cheng. Towards accurate post-training network quantization via bit-split and stitching. In International Conference on Machine Learning. PMLR, 2020.

[Xu et al., 2021] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. arXiv preprint arXiv:2108.01390, 2021.

[Zheng et al., 2021] Suxiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiutian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[Zhou et al., 2016] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160, 2016.
A Quantized Inference of LayerNorm

A.1 Quantized Inference of LayerNorm

LayerNorm has been widely used in neural networks to accelerate the convergence during the training process, which can be formulated as:

\[ \text{LayerNorm}(X) = \frac{X - \mu_X}{\sigma_X + \epsilon} \cdot \gamma + \beta \]  
(20)

where \( \mu \) and \( \sigma \) are the statistics which need to be calculated based on the input of LayerNorm. In this paper, proposed Power-of-Two Factor is used for LayerNorm input \( X \in \mathbb{R}^{B \times L \times C} \) and its quantized value \( X_0 \) can be written as:

\[ X_0 = Q(X|b) = \text{clip}(\frac{X}{2^s s} + zp, 0, 2^b - 1), \]  
(22)

where \( s, zp \in \mathbb{R}^1 \) are quantization parameters and \( \alpha \in \mathbb{N}^C \) is Power-of-Two Factor.

Following the LayerNorm's definition, we should calculate the statistics of input \( X \). As described in this paper, the whole process can be divided into two phases. In the first phase, we shift the quantized activation \( X_0 \) with PTF \( \alpha \):

\[ \hat{X}_0 = (X_0 - zp) << \alpha. \]  
(23)

Then, in the second phase, we need to calculate the statistics based on the shifted activation \( \hat{X}_0 \). Firstly, we gauge the mean of \( X \) and \( X^2 \) as follows:

\[ \mu_X \approx \frac{1}{C} \sum_{i=1}^{C} (\hat{X}_{0i} \cdot s) = \frac{s}{C} \sum_{i=1}^{C} (\hat{X}_{0i}) \rightarrow \frac{s}{C} M_1, \]  
(24)

\[ \mu_{X^2} \approx \frac{1}{C} \sum_{i=1}^{C} (\hat{X}_{0i} \cdot s)^2 = \frac{s^2}{C} \sum_{i=1}^{C} (\hat{X}_{0i})^2 \rightarrow \frac{s^2}{C} M_2, \]  
(25)

where \( C \) is the number of channels in \( X \). Secondly, we utilize \( \mu_X \) and \( \mu_{X^2} \) to calculate \( \sigma_X^2 \):

\[ \sigma_X^2 = \mu_{X^2} - \mu_X^2 \approx \frac{s^2}{C} (CM_2 - M_1^2), \]  
(26)

and approximate \( \sqrt{\sigma_X^2 + \epsilon} \) as:

\[ \sqrt{\sigma_X^2 + \epsilon} \approx \frac{s}{C} \sqrt{CM_2 - M_1^2}. \]  
(27)

Thus, we obtain the statistics of input \( X \) based on integer-only calculations.

After the calculations of statistics, we need to do the integrated inference of LayerNorm. Equation (20) can be written as follows by quantizing the input \( X \) and output \( Y_0 \):

\[ Y_0 = \left\lfloor \frac{s_{in} \gamma}{s_{out} \sqrt{\sigma_X^2 + \epsilon}} \cdot \hat{X}_0 + \frac{\beta \sqrt{\sigma_X^2 + \epsilon} - \gamma \mu_X}{s_{out} \sqrt{\sigma_X^2 + \epsilon}} \right\rfloor + zp_{out}, \]  
(28)

where \( s_{in} \in \mathbb{R}^1 \) is the scale of input \( X \), and \( s_{out}, zp_{out} \in \mathbb{R}^1 \) are the scale of output \( Y_0 \).

In order to simplify the equation, we fuse each term as follows:

\[ A = \frac{s_{in} \gamma}{s_{out} \sqrt{\sigma_X^2 + \epsilon}}, \]  
(29)

\[ B = \frac{\beta \sqrt{\sigma_X^2 + \epsilon} - \gamma \mu_X}{s_{out} \sqrt{\sigma_X^2 + \epsilon}}. \]  
(30)

Figure 6: Comparison between exponential (exp) and integer-only exponential (i-exp)

To obtain the integer-only inference, we approximate \( A \) as:

\[ N_1 = b - 1 - \lfloor \log_2 |A| \rfloor, N_2 = \lfloor |A|2^{N_1} \rfloor, \]  
(31)

\[ A = \text{sign}(A) \cdot \frac{N_2}{2^{N_1}}, \]  
(32)

where \( b \) is target bit-width. Finally, the quantized inference for LayerNorm can be formulated as:

\[ Y_0 = [A\hat{X}_0 + B] + zp_{out} \]  
(33)

\[ = \lfloor \text{sign}(A) \cdot N_2\hat{X}_0 + [B \cdot 2^{N_1}] \rfloor + zp_{out}. \]  
(34)

A.2 Quantized Inference of Softmax

Softmax is generally used in the Vision Transformers and it can be formulated as:

\[ \text{Softmax}(X)_i = \frac{\exp(X_i)}{\sum_{j=1}^{J} \exp(X_j)}. \]  
(35)

In practice, we subtract the maximum value of the input to avoid overflowing:

\[ \hat{X}_i = X_i - \max(X), \]  
(36)

and thus the Softmax can be written as:

\[ \text{Softmax}(X)_i = \frac{\exp(\hat{X}_i)}{\sum_{j=1}^{J} \exp(\hat{X}_j)}. \]  
(37)

Now, all inputs, \( \hat{X} \), become non-positive. We decompose it as \( (-\ln 2)z + p \), where \( z \) is a non-negative integer and \( p \) is a real number in \( (-\ln 2, 0] \). Then, the exponential of \( \hat{X} \) can be written as:

\[ \exp(\hat{X}) = 2^{-z} \exp(p) = \exp(p) >> z, \]  
(38)

where \( >> \) is the right shifting.

I-BERT use a second-order polynomial to approximate the exponential function in the interval of \( p \in (-\ln 2, 0] \):

\[ L(p) = 0.3585(p + 1.353)^2 + 0.344 \approx \exp(p), \]  
(39)

\[ \exp(\hat{x}) \approx i\text{-exp}(\hat{x}) := L(p) >> z, \]  
(40)

where \( z = \lfloor \hat{X} / \ln 2 \rfloor \) and \( p = \hat{X} + z \ln 2 \).
Algorithm 1 Integer-only Exponential

1: Input: \( q, s \): quantized input and scale
2: Output: \( q_{\text{out}}, s_{\text{out}} \): quantized output and scale

3: function I-\text{EXP}(q, s)
4: \( q_{\text{in2}} \leftarrow \lfloor -\ln 2/s \rfloor \)
5: \( q \leftarrow \max(q, n \cdot q_{\text{in2}}) \)
6: \( z \leftarrow \lfloor q/q_{\text{in2}} \rfloor \)
7: \( q_p \leftarrow q - z \cdot q_{\text{in2}} \)
8: \( q_0, q_c \leftarrow \lfloor b/s \rfloor, \lfloor c/a s^2 \rfloor \)
9: \( s_L \leftarrow \lfloor a s^2 \rfloor \)
10: \( q_s = (q + q_0)^2 + q_c \)
11: \( q_{\text{out}} = q_L << n - z, s_L/2^n \)
12: return \( q_{\text{out}}, s_{\text{out}} \)
13: end function

Algorithm 2 Log-Int-Softmax

Input: \( q, s \): quantized input and input scale
Output: \( q_{\text{out}}, s_{\text{out}} \): quantized output and output scale

function I-\text{LOG2}(q)
\( M \leftarrow \text{FindFirstOne}(q) \)
\( \chi \leftarrow (M - 1)^{\text{th}} \text{ Bit of } q \)
\( q_{\text{out}} \leftarrow M + \chi \)
return \( q_{\text{out}} \)
end function

function LOG-INT-\text{SOFTMAX}(q, s)
\( q' \leftarrow q - \max(q) \)
\( q_{\text{exp}}, s_{\text{exp}} \leftarrow \text{I-\text{EXP}}(q', s) \)
\( q_{\text{rev}} \leftarrow \sum(q_{\text{exp}})/q_{\text{exp}} \)
\( q_{\text{out}}, s_{\text{out}} \leftarrow N - \text{I-\text{LOG2}}(q_{\text{rev}}), 1/2^N \)
return \( q_{\text{out}}, s_{\text{out}} \)
end function

Figure 6 plots the result of \( i-\text{exp} \), which is nearly identical to the exponential function. The largest gap between these two functions is only \( 1.9 \times 10^{-3} \), which is relatively negligible comparing to 8-bit quantization error of \( 1/2^{255} = 3.9 \times 10^{-3} \).

Based on \( i-\text{exp} \) (Algorithm 1), we propose Log-Int-Softmax (LIS) as Algorithm 2 shows. Firstly, the maximum value of inputs is subtracted to ensure all the inputs are non-positive, and the results are sent to the \( i-\text{exp} \). After that, we replace the original Softmax by its reciprocal to ensure the results of integer division to be larger than 1. Last but not least, we perform a Log2 quantization on the output of reverse Softmax as:

\[
\text{LIS}(s \cdot X_Q) = N - \text{clip} \left( (\frac{\sum i-\text{exp}(X_Q)}{i-\text{exp}(X_Q)}), 0, 2^b - 1 \right) 
\]

\[
= N - \text{Attn}_Q, \quad (41)
\]

with \( N = 2^b - 1 \). \( X_Q, s \) are the quantized input and scale.

The Log2 function can be calculated with integer arithmetic as showed in Algorithm 2. To obtain the Log2 of the integer, we introduce the FindFirstOne function which returns the index of the most significant one bit of the input. The whole process can be done in integer domain. For example, if the input of I-\text{LOG2} is 0000 1101 1010 1100, the M, \( \chi \) will be calculated as 11, 1 and the rounding result will be 12.

The following step of Softmax is the matrix multiplication between attention map \( \text{Attn} \) and values \( V \). Considering \( V_Q \) is the quantized value of \( V \) and its scale and zero point are \( sv, zp \in \mathbb{R}^3 \), the matrix multiplication can be written as:

\[
\text{Attn} \cdot V = \frac{2^{N-\text{Attn}_Q}}{2^N} \cdot (V_Q - zp_V)sv \quad (43)
\]

\[
= \frac{sv}{2^N} \cdot (V_Q - zp_V) \ll (N - \text{Attn}_Q). \quad (44)
\]

As we can see, the multiplication is converted to BitShift operator with an output scale \( \frac{1}{2^N} \), where \( N \) equals \( 2^4 - 1 = 15 \) in 4-bit quantization.

Based on the above processes, we realize fully fixed-point inference for Softmax and accelerate the calculation of attention map and values by using BitShift.

B Hyperparameter K of Power-of-Two Factor

We conduct experiments on the setting of hyperparameter \( K \) in PTF. As shown in Figure 7, we evaluate the top-1 accuracy of vision transformers with different \( K \). It is observed that the accuracy is nearly saturated when \( K \) is taken as 3. As a result, to meet the different inter-channel variants between all models as much as possible, we choose 3 as the default value.

C Inter-channel Variation for Vision Transformers and ResNets

In Figure 8, we present the channel-wise minimum and maximum values of Vision Transformers and ResNets. For comparison, we choose the input of the last LayerNorm layer for Vision Transformers and the output of the fourth stage for ResNets to show. It is observed that a serious inter-channel variation are found in Vision Transformers.
Figure 8: Channel-wise minimum and maximum values of Vision Transformers and ResNets.