arXiv:2207.01405v4 [cs.CV] 7 Aug 2023

I-ViT: Integer-only Quantization for Efficient Vision Transformer Inference

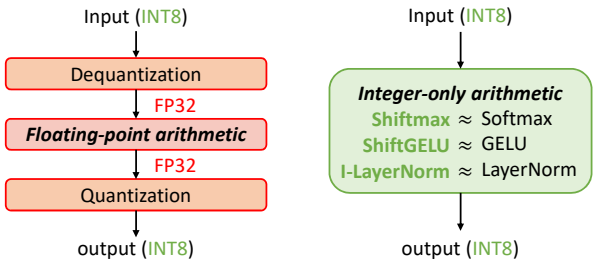
Zhikai Li[1,2], Qingyi Gu[1,*]
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
{lizhikai2020, qingyi.gu}@ia.ac.cn

Abstract

Vision Transformers (ViTs) have achieved state-of-the-art performance on various computer vision applications. However, these models have considerable storage and computational overheads, making their deployment and efficient inference on edge devices challenging. Quantization is a promising approach to reducing model complexity, and the dyadic arithmetic pipeline can allow the quantized models to perform efficient integer-only inference. Unfortunately, dyadic arithmetic is based on the homogeneity condition in convolutional neural networks, which is not applicable to the non-linear components in ViTs, making integer-only inference of ViTs an open issue. In this paper, we propose I-ViT, an integer-only quantization scheme for ViTs, to enable ViTs to perform the entire computational graph of inference with integer arithmetic and bit-shifting, and without any floating-point arithmetic. In I-ViT, linear operations (e.g., MatMul and Dense) follow the integer-only pipeline with dyadic arithmetic, and non-linear operations (e.g., Softmax, GELU, and LayerNorm) are approximated by the proposed light-weight integer-only arithmetic methods. More specifically, I-ViT applies the proposed Shiftmax and ShiftGELU, which are designed to use integer bit-shifting to approximate the corresponding floating-point operations. We evaluate I-ViT on various benchmark models and the results show that integer-only INT8 quantization achieves comparable (or even slightly higher) accuracy to the full-precision (FP) baseline. Furthermore, we utilize TVM for practical hardware deployment on the GPU's integer arithmetic units, achieving 3.72∼4.11× inference speedup compared to the FP model. Code of both Pytorch and TVM is released at https://github.com/zkkli/I-ViT.

1. Introduction

Vision Transformers (ViTs) have recently achieved great success on a variety of computer vision tasks [13, 10, 4]. Nevertheless, as compared to convolutional neural networks (CNNs), ViTs suffer from higher memory footprints, computational overheads, and power consumption, hindering their deployment and real-time inference on resource-constrained edge devices [25, 17, 15, 37]. Thus, compression approaches for ViTs are being widely researched.



(a) FasterTransformer [34] (b) I-ViT (ours)

Figure 1. Computation flows of Softmax, GELU, and LayerNorm in FasterTransformer [34] and our proposed I-ViT. I-ViT realizes the entire computational graph with integer-only arithmetic, which is more promising and practical for low-cost model deployment and efficient inference.

Model quantization, which reduces the representation precision of weight/activation parameters, is an effective and hardware-friendly way to improve model efficiency [12, 20, 7, 35]. With the quantized low-precision parameters, previous work [18] presents the dyadic arithmetic pipeline to realize integer-only inference, where the quantization scaling factors are collapsed into the integer multiplication and bit-shifting in the requantization process. This can enable the quantized models to fully benefit from the fast and efficient low-precision integer arithmetic units and thus provides promising speedup effects [41, 44]. For instance, the edge processor core in ARM Cortex-M family only support the deployment of the integer-only kernels; the recent Turing Tensor Cores in GPU server class also add support for integer logical units, and their high throughput capability enables notably lower latency compared to floating-point arithmetic.

However, the above integer-only pipeline is designed for

*Corresponding author.

CNNs and works under the homogeneity condition, making it only applicable to linear (*e.g.*, Dense) or piecewise linear (*e.g.*, ReLU) operations [18, 44]. Therefore, the non-linear operations (*e.g.*, Softmax, GELU, and LayerNorm) in ViTs cannot naively follow it. To cope with this problem, a brute-force scheme is to simply leave the non-linear operations as dequantized floating-point arithmetic, such as FasterTransformer [34] shown in Figure 1(a). Unfortunately, this scheme makes them tolerate the inefficiency of floating-point arithmetic units, and the cut of the computational graph also introduces communication costs between integer and floating-point units, which severely limits the speedup of inference. In addition, low-cost integer-only hardware cannot meet mixed-precision computing requirements, hence one has to design heterogeneous chips by adding floating-point arithmetic units, which definitely increases the budget for model deployment.

Consequently, integer-only arithmetic for non-linear operations is significant for low-cost deployment and efficient inference. To this end, several works have attempted on language Transformer models. Fully-8bit [29] employs L1 LayerNorm to replace the non-linear arithmetic of standard deviation, and I-BERT [19] proposes integer polynomial approximations for the non-linear operations. However, such approaches are inefficient and fail to fully exploit the benefits of hardware logic. Moreover, they are developed for language models, making it infeasible to properly transfer to ViTs due to differences in data distribution. For ViTs, FQ-ViT [30] preliminarily explores the feasibility of integer arithmetic for part of the operations (*e.g.*, Softmax), but it is simply built on I-BERT [19] and ignores the notable GELU operation, leaving a huge gap between it and integer-only inference. As a result, ***how to accurately perform the non-linear operations of ViTs with efficient integer-only arithmetic*** remains an open issue.

In this paper, we propose I-ViT, which quantizes the entire computational graph to fill the research gap of integer-only quantization for ViTs. Specifically, linear operations follow the dyadic arithmetic pipeline; and non-linear operations are approximated without accuracy drop by novel light-weight integer-only arithmetic methods, where Shiftmax and ShiftGELU perform most arithmetic with bit-shifting that can be efficiently executed with simple shifters in hardware logic [39], and I-LayerNorm calculates the square root with integer iterations instead.

The main contributions are summarized as follows:

- We propose I-ViT, which fully quantizes the computational graph of ViTs and allows performing the entire inference with integer arithmetic and bit-shifting, without any floating-point operations. To the best of our knowledge, this is the first work on integer-only quantization for ViTs.

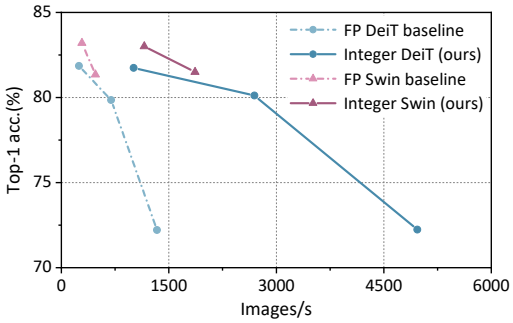


Figure 2. Accuracy-speed curves of I-ViT and the FP baseline on DeiT [38] and Swin [31]. Accuracy is evaluated on ImageNet dataset, and speed is obtained from the latency on an RTX 2080Ti GPU (batch=8). As we can see, I-ViT provides significant accelerations (3.72∼4.11×) while achieving similar (or even slightly higher) accuracy.

- We propose novel light-weight integer approximations for non-linear operations (as shown in Figure 1(b)), in particular, Shiftmax and ShiftGELU use integer bit-shifting to accomplish most arithmetic, which fully benefit from the efficient hardware logic.

- I-ViT is evaluated on various models for the large-scale classification task, achieving compression with similar (or even slightly higher) accuracy. Moreover, we deploy I-ViT on an RTX 2080Ti GPU using TVM[1] [6], which accelerates the integer-only inference of ViTs with Turing Tensor Cores, achieving a 3.72∼4.11× speedup over the FP model (as shown in Figure 2).

2. Related Works

2.1. Vision Transformers

Thanks to the global receptive fields captured by the attention mechanism, ViTs have shown superior performance on various computer vision tasks [13, 42, 14]. ViT [10] is the first effort to apply transformer-based models to vision applications and achieves high accuracy than CNNs on the classification task. DeiT [38] introduces an efficient teacher-student strategy via adding a distillation token, reducing the time and data cost in the training phase. Swin [31] presents shifted window attentions at various scales, which boosts the performance of ViTs. Furthermore, ViTs have also been applied to more complexed vision applications, such as object detection [4, 51], semantic segmentation [5], and video recognition [1].

Despite the promising performance, ViTs' complicated architectures with large memory footprints and computational overheads is intolerable in real-world applications [17, 43, 47, 27], especially in time/resource-constrained

[1] https://github.com/apache/tvm

scenarios. Thus, the compression approaches for ViTs are necessary for practical deployments.

2.2. Model Quantization

Model quantization, which converts the floating-point parameters to low-precision values, is a prevalent solution to compressing models in a hardware-friendly manner [20, 12, 26, 48]. Most previous works are designed to quantize CNNs. DoReFa [49] and LQ-Net [46] approximate the gradient propagation in quantization-aware training by straight-through estimator (STE) [2]. PACT [7] and LSQ [11, 3] treat the activation clipping value/step size as trainable parameters and achieve promising results on low-bit quantization. In addition, several notable works adopt more advanced quantization strategies, including non-uniform quantization [23], channel-wise quantization [22], and mixed-precision quantization [40, 9] etc.

Recently, several quantization methods oriented to ViTs' unique structures are proposed. Ranking loss [32] is presented to maintain the correct relative order of the quantized attention map. Q-ViT [28] proposes differentiable quantization for ViTs, taking the quantization bit-widths and scales as learnable parameters. PTQ4ViT [45] proposes twin uniform quantization and uses a Hessian guided metric to evaluate different scaling factors. FQ-ViT [30] introduces powers-of-two scale quantization and log-int quantization for LayerNorm and Softmax, respectively. RepQ-ViT [27] decouples the quantization and inference processes to address the extreme distributions of LayerNorm and Softmax activations. PSAQ-ViT [25, 24] pushes the quantization of ViTs to data-free scenarios based on patch similarity.

However, in the above approaches, all or part of the operations are performed with dequantized floating-point parameters during inference, which fails to fully use efficient low-precision arithmetic units and thus provides unsatisfactory model acceleration.

2.3. Integer-only quantization

Integer-only quantization, which eliminates dequantization and enables the entire inference to be performed with integer-only arithmetic, can potentially address the above challenges. Dyadic arithmetic is proposed to perform the integer-only pipeline for CNNs [18, 44], however, it is designed for linear and piecewise linear operations based on the homogeneity condition, and thus is not applicable to non-linear operations in ViTs.

Therefore, several studies are interested in how to achieve integer arithmetic for non-linear operations in language Transformer models. Fully-8bit [29] introduces L1 LayerNorm, which avoids the non-linearity of solving for the square root when calculating the standard deviation. I-BERT [19] focuses on integer polynomial approximations for the non-linear operations, including Softmax, GELU, and LayerNorm. It is worth noting that although these three component approximations can potentially enable integer-only inference of ViTs (partly verified by FQ-ViT [30]), the computation of high-order polynomials is inefficient in inference, and they are developed for language models that do not fit the data distribution of ViTs, leading to mismatched approximations. In addition, various approximation methods that hold floating-point arithmetic are presented [36, 50]; while they lower certain computational costs, they cannot meet the demands of integer arithmetic. As a result, integer-only quantization for ViTs remains a research gap.

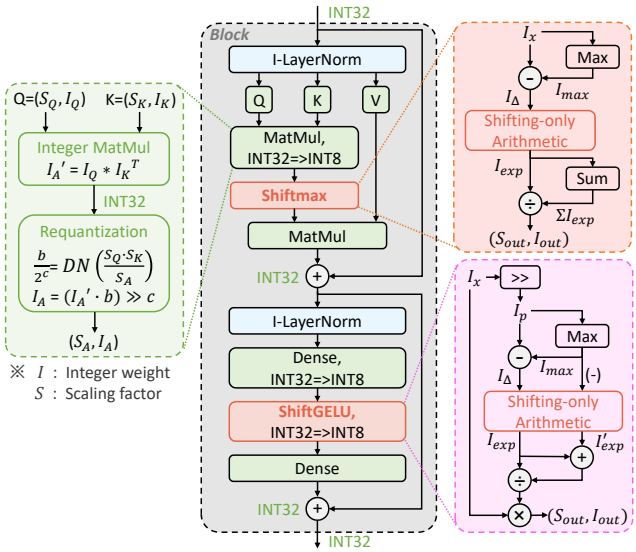


Figure 3. Overview of the proposed I-ViT. The entire computational graph is performed with integer-only arithmetic, where linear MatMul and Dense operations follow the dyadic arithmetic pipeline and the proposed Shiftmax, ShiftGELU, and I-LayerNorm accomplish the non-linear operations. Except for the labeled INT32, the remaining data streams are all INT8 precision.

3. Methodology

3.1. Overview

The overview of the proposed integer-only quantization scheme for ViTs is illustrated as Figure 3. The main body of ViTs is a stack of blocks, and each block is divided into a multi-head self-attention (MSA) module and a multi-layer perceptron (MLP) module, which can be formulated as follows:

$$\hat{X} = \text{MSA}(\text{LayerNorm}(X)) + X \tag{1}$$

$$Y = \text{MLP}(\text{LayerNorm}(\hat{X})) + \hat{X} \tag{2}$$

The MSA module learns inter-patch representations by

calculating the global attention as follows:

$$\text{MSA}(X) = \text{Concat}(\text{Attn}_1, \text{Attn}_2, \dots, \text{Attn}_h)W^O \quad (3)$$

$$\text{where} \;\; \text{Attn}_i = \text{Softmax}\left(\frac{Q_i \cdot K_i^T}{\sqrt{d}}\right) V_i \quad (4)$$

where h is the number of the attention heads, d is the size of hidden features, and $i = 1, 2, \cdots, h$. Here, Q_i, K_i, and V_i are query, key, and value, respectively, and they are obtained by linear projections, *i.e.*, $Q_i = XW_i^Q$, $K_i = XW_i^K$, $V_i = XW_i^V$. Then the MLP module employs two dense layers and a GELU activation function to learn high-dimensional representations as follows:

$$\text{MLP}(\hat{X}) = \text{GELU}(\hat{X}W_1 + b_1)W_2 + b_2. \quad (5)$$

In this work, we are interested in quantizing the entire computational graph of ViTs. To facilitate TVM implementation, we apply the simplest *symmetric uniform quantization* as follows:

$$I = \left\lfloor \frac{\text{clip}(R, -m, m)}{S} \right\rceil, \text{ where } S = \frac{2m}{2^k - 1} \quad (6)$$

where R and I denote the floating-point values and the quantized integer values, respectively, S is the scaling factor of quantization, m is the clipping value determined by the naive min-max method, k is the quantization bit-precision, and $\lfloor \cdot \rceil$ is the round operator.

With the quantized integer values, to avoid dequantization and achieve integer-only inference, we apply the dyadic arithmetic pipeline for linear operations, as detailed in Section 3.2. Since the above pipeline is based on the homogeneity condition (*e.g.*, MatMul($S_Q \cdot I_Q, S_K \cdot I_K$)$\equiv$ $S_Q \cdot S_K \cdot$MatMul(I_Q, I_K)), it is not applicable to the case of non-linearity (*e.g.*, Softmax($S_A \cdot I_A$)$\neq S_A \cdot$Softmax(I_A)). Thus, non-linear operations require accurate and efficient approximations by integer-only arithmetic. To this end, Shiftmax and ShiftGELU are proposed in this paper, which utilize efficient shifters in hardware logic to accomplish most arithmetic, and I-LayerNorm calculates the square root of the variance in an integer iterative manner. The above schemes are described in detail in Sections 3.3-3.5, respectively.

3.2. Dyadic Arithmetic for Linear Operations

The dyadic arithmetic pipeline, which uses integer bit-shifting to efficiently realize floating-point operations of scaling factors, allows linear operations to be performed with integer-only arithmetic. Although it is designed for CNNs [18, 44], it can also be followed for linear operations in ViTs, including Conv in the embedding layer, and MatMul and Dense in the transformer layer.

Taking MatMul as an instance, when the inputs are $Q = (S_Q, I_Q)$ and $K = (S_K, I_K)$, the output is calculated as follows:

$$A' = S_A{}' \cdot I_A{}' = S_Q \cdot S_K \cdot \left(I_Q * {I_K}^T\right) \quad (7)$$

where $I_A{}' = I_Q * {I_K}^T$ performs integer-only arithmetic. Following the principle of practical hardware implementation (*e.g.*, DP4A), when the inputs I_Q and I_K are INT8 types, the output $I_A{}'$ is INT32 type. Thus, we need to re-quantize $I_A{}'$ to INT8 type as the input for the next layer, which is calculated as follows:

$$I_A = \left\lfloor \frac{S_A{}' \cdot I_A{}'}{S_A} \right\rceil = \left\lfloor \frac{S_Q \cdot S_K}{S_A} \cdot \left(I_Q * {I_K}^T\right) \right\rceil \quad (8)$$

where S_A is the pre-calculated scaling factor of the output activation. Although the scaling factors remain floating-point values, their multiplication and division operations in Eq. 8 can be avoided by converting the rescaling to a dyadic number (DN) as follows:

$$DN\left(\frac{S_Q \cdot S_K}{S_A}\right) = \frac{b}{2^c} \quad (9)$$

where b and c are both positive integer values. In this case, the rescaling can be efficiently accomplished by integer multiplication and bit-shifting. To summarize, the integer-only arithmetic pipeline of MatMul can be denoted as follows:

$$I_A = \left(b \cdot \left(I_Q * {I_K}^T\right)\right) \gg c \quad (10)$$

where $\gg$ indicates right bit-shifting.

3.3. Integer-only Softmax: Shiftmax

Softmax in ViTs translates the attention scores into probabilities, which acts on the hidden features and is calculated as follows:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j^d e^{x_j}} = \frac{e^{S_{x_i} \cdot I_{x_i}}}{\sum_j^d e^{S_{x_j} \cdot I_{x_j}}} \quad (11)$$

where $i = 1, 2, \cdots, d$. Due to the non-linearity, Softmax cannot follow the dyadic arithmetic pipeline discussed above, and the exponential arithmetic in Eq. 11 is typically unsupported by integer-only logic units [36]. To address the above issues, we propose the approximation method Shiftmax, which can utilize simple hardware logic to achieve accurate and efficient integer-only arithmetic of Softmax. First, to smooth the data distribution and prevent overflow, we restrict the range of the exponential arithmetic as follows:

$$\text{Softmax}(x_i) = \frac{e^{S_{\Delta_i} \cdot I_{\Delta_i}}}{\sum_j^d e^{S_{\Delta_j} \cdot I_{\Delta_j}}} = \frac{e^{S_{x_i} \cdot (I_{x_i} - I_{max})}}{\sum_j^d e^{S_{x_j} \cdot (I_{x_j} - I_{max})}} \quad (12)$$

where $I_{max} = \max\{I_{x_1}, I_{x_2}, \cdots, I_{x_d}\}$. Here, $I_{\Delta_i} = I_{x_i} - I_{max}$ is a non-positive value and $S_{\Delta_i} = S_{x_i}$, and we simplify them as I_Δ and S_Δ in the following part for easier expression.

Then, we are motivated to convert the base from e to 2 to fully utilize the efficient shifters. Instead of a brute-force conversion, we perform an equivalent transformation using the base changing formula of the exponential function. Importantly, since $\log_2 e$ can be approximated by binary as $(1.0111)_b$, the floating-point multiplication with it can be achieved by integer shifting as follows:

$$\begin{aligned} e^{S_\Delta \cdot I_\Delta} &= 2^{S_\Delta \cdot (I_\Delta \cdot \log_2 e)} \\ &\approx 2^{S_\Delta \cdot (I_\Delta + (I_\Delta \gg 1) - (I_\Delta \gg 4))} \end{aligned} \tag{13}$$

The power term is denoted as $S_\Delta \cdot I_p$, which is not ensured as an integer and cannot be directly used for shifting. Thus, we decompose it into an integer part and a decimal part as follows:

$$2^{S_\Delta \cdot I_p} = 2^{(-q) + S_\Delta \cdot (-r)} = 2^{S_\Delta \cdot (-r)} \gg q \tag{14}$$

where $S_\Delta \cdot (-r) \in (-1, 0]$ is the decimal part, and q and r are both positive integer values. For low-cost computation, we approximate $2^{S_\Delta \cdot (-r)}$ in range $(-1, 0]$ by the linear function as follows:

$$\begin{aligned} 2^{S_\Delta \cdot (-r)} &\approx [S_\Delta \cdot (-r)]/2 + 1 \\ &= S_\Delta \cdot [((-r) \gg 1) + I_0] \end{aligned} \tag{15}$$

where $I_0 = \lfloor 1/S_\Delta \rceil$. The above completes the approximation of the numerator in Eq. 12, *i.e.*, $S_\Delta \cdot I_{exp} \approx e^{S_\Delta \cdot I_\Delta}$, where S_Δ can be removed via fraction reduction since the scaling factor of the denominator obtained by summing is also S_Δ. This turns Eq. 12 into an integer division, which is calculated with the specified output bit-precision k_{out} as follows:

$$\begin{aligned} I_{out_i} &= \frac{S_\Delta \cdot I_{exp_i}}{S_\Delta \cdot \sum_j^d I_{exp_j}} \\ &= \text{IntDiv}(I_{exp_i}, \sum_j^d I_{exp_j}, k_{out}) \\ &= \left(\left\lfloor \frac{2^M}{\sum_j^d I_{exp_j}} \right\rfloor \cdot I_{exp_i} \right) \gg (M - (k_{out} - 1)) \\ S_{out_i} &= 1/2^{k_{out}-1} \end{aligned} \tag{16}$$

where $\text{IntDiv}(I_1, I_2, k)$ implements the integer division function, and I_1, I_2, and k are integer dividend, integer divisor and output bit width, respectively. Here, M is a sufficiently large integer, and $S_{out_i} \cdot I_{out_i}$ [2] can approximate the result of $\text{Softmax}(x_i)$.

[2] S_{out} is the scaling factor for the k_{out}-bit symmetric quantization with $m \approx 1$.

Algorithm 1: Integer-only Softmax: **Shiftmax**

Input: I_{in} : Integer input
S_{in} : Input scaling factor
k_{out} : Output bit-precision
Output: I_{out} : Integer output
S_{out} : Output scaling factor

Function `ShiftExp` (I, S) **:**
  $I_p \leftarrow I + (I \gg 1) - (I \gg 4)$; ▷ $I \cdot \log_2 e$
  $I_0 \leftarrow \lfloor 1/S \rceil$;
  $q \leftarrow \lfloor I_p / (-I_0) \rfloor$; ▷ Integer part
  $r \leftarrow -(I_p - q \cdot (-I_0))$; ▷ Decimal part
  $I_b \leftarrow ((-r) \gg 1) + I_0$; ▷ Eq. 15
  $I_{exp} \leftarrow I_b \ll (N - q)$; [3] ▷ Eq. 14
  $S_{exp} \leftarrow S/(2^N)$;
  return (I_{exp}, S_{exp}); ▷ $S_{exp} \cdot I_{exp} \approx e^{S \cdot I}$
End Function

Function `Shiftmax` $(I_{in}, S_{in}, k_{out})$ **:**
  $I_\Delta \leftarrow I_{in} - \max(I_{in})$; ▷ Eq. 12
  $(I_{exp}, S_{exp}) \leftarrow$ `ShiftExp`(I_Δ, S_{in});
  $(I_{out}, S_{out}) \leftarrow$ `IntDiv`$(I_{exp}, \sum I_{exp}, k_{out})$; ▷ Eq. 16
  return (I_{out}, S_{out}); ▷ $I_{out} \cdot S_{out} \approx \text{Softmax}(I_{in} \cdot S_{in})$
End Function

The integer-only flow of Shiftmax is summarized in Algorithm 1. Instead of complex second-order polynomial approximations [19], Shiftmax performs all arithmetic with bit-shifting, except for one integer subtraction, summation, and division, which significantly improves computational efficiency. In addition, only Eqs. 13 and 15 are mathematically approximated, while all others are equivalent transformations, which ensures the accuracy of Shiftmax.

3.4. Integer-only GELU: ShiftGELU

GELU is the non-linear activation function in ViTs, which, from the study [16], can be approximated by a sigmoid function as follows:

$$\begin{aligned} \text{GELU}(x) &= x \cdot \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt \\ &\approx x \cdot \sigma(1.702x) \\ &= S_x \cdot I_x \cdot \sigma(S_x \cdot 1.702 I_x) \end{aligned} \tag{17}$$

Thus, the challenge becomes the realization of the sigmoid function's integer-only arithmetic. First, 1.702 can be approximated by binary as $(1.1011)_b$, thus $1.702 I_x$ can be achieved by integer shifting, *i.e.*, $I_p = I_x + (I_x \gg 1) + (I_x \gg 3) + (I_x \gg 4)$. Then, we equivalently transform

[3] To avoid too small values after right shifting, we first have a N-bit left shifting.

Algorithm 2: Integer-only GELU: **ShiftGELU**

Input: I_{in} : Integer input
S_{in} : Input scaling factor
k_{out} : Output bit-precision
Output: I_{out} : Integer output
S_{out} : Output scaling factor

Function $\texttt{ShiftGELU}(I_{in}, S_{in}, k_{out})$ **:**

$I_p \leftarrow I_{in} + (I_{in} \gg 1) + (I_{in} \gg 3) + (I_{in} \gg 4)$; ▷ $1.702I$

$I_\Delta \leftarrow I_p - \max(I_p)$;

$(I_{exp}, S_{exp}) \leftarrow \texttt{ShiftExp}(I_\Delta, S_{in})$;

$(I'_{exp}, S'_{exp}) \leftarrow \texttt{ShiftExp}(-\max(I_p), S_{in})$;

$(I_{div}, S_{div}) \leftarrow \texttt{IntDiv}(I_{exp}, I_{exp}+I'_{exp}, k_{out})$; ▷ Eq. 18

$(I_{out}, S_{out}) \leftarrow (I_{in} \cdot I_{div}, S_{in} \cdot S_{div})$;

return (I_{out}, S_{out}); ▷ $I_{out} \cdot S_{out} \approx \text{GELU}(I_{in} \cdot S_{in})$

End Function

the sigmoid function as follows:

$$\begin{aligned} \sigma(S_x \cdot I_p) &= \frac{1}{1 + e^{-S_x \cdot I_p}} \\ &= \frac{e^{S_x \cdot I_p}}{e^{S_x \cdot I_p} + 1} \\ &= \frac{e^{S_x \cdot (I_p - I_{max})}}{e^{e^{S_x \cdot (I_p - I_{max})}} + e^{S_x \cdot (-I_{max})}} \end{aligned} \tag{18}$$

where, interestingly, the numerator is in exact correspondence with the numerator of Eq. 12, thus the two implementations are identical. After that, the integer approximation of GELU is done by following the integer division in Eq. 16 and then multiplying it with $S_x \cdot I_x$.

Algorithm 2 shows the integer-only flow of ShiftGELU. Except for a few fundamental arithmetic operations, ShiftGELU utilizes shifters in hardware logic to perform all other arithmetic and thus enables the efficient inference of ViTs. Furthermore, compared to the second-order polynomial method that only approximates for a specific interval [19], the approximation of ShiftGELU works on the entire domain of definition, which can potentially provide higher accuracy and robustness.

3.5. Integer-only LayerNorm: I-LayerNorm

LayerNorm in ViTs normalizes the input in the hidden feature dimension as follows:

$$\text{LayerNorm}(x) = \frac{x - \text{Mean}(x)}{\sqrt{\text{Var}(x)}} \cdot \gamma + \beta \tag{19}$$

where γ and β are linear affine factors. In contrast to BatchNorm that holds fixed parameters from training and can be folded during inference, LayerNorm needs to dynamically compute statistics ($i.e.$, mean and standard deviation) in the inference phase. The integer arithmetic units allow straightforward calculation of the mean and variance of the data, yet they fail to support the square root arithmetic for obtaining the standard deviation [29]. Thus, we improve the light-weight integer iterative approach [8] via bit-shifting as follows:

$$\begin{aligned} I_{i+1} &= (I_i + \lfloor \text{Var}(I_x)/I_i \rfloor)/2 \\ &= (I_i + \lfloor \text{Var}(x)/I_i \rfloor) \gg 1 \end{aligned} \tag{20}$$

where I_i is the result of the i-th iteration, and I_0 is initialized as $2^{\lfloor \text{bit}(\text{Var}(I_x))/2 \rfloor}$. The naive stopping criterion for the iterations is $I_{i+1} \geq I_i$, which unfortunately cannot guarantee a constant latency. We experimentally find that 10 iterations can achieve most convergence, thus we modify the stopping criterion to the iteration counts to facilitate hardware implementation.

4. Experiments

We evaluate I-ViT in both accuracy on the large-scale classification task and latency on the practical hardware to fully demonstrate the superiority, and I-ViT can accelerate 3.72∼4.11× over the FP model while achieving similar (or even slightly higher) accuracy. Besides the *FP baseline*, I-ViT is compared end-to-end with the following methods:

- *FasterTransformer* [34]: leaving non-linear operations as floating-point arithmetic.
- *I-BERT* [19]: approximating non-linear operations with integer second-order polynomials.

Notably, several methods achieve integer inference for only part of the operations, making it impractical to perform end-to-end comparisons, hence we evaluate the following individual components in ablation studies.

- *L1 LayerNorm* in Fully-8bit [29]: using L1 norm to replace the calculation of standard deviation.
- *LIS* in FQ-ViT [30]: using Log-Int-Softmax, which builds opon I-BERT and adds the logarithmic function.

4.1. Accuracy Evaluation

Implementation Details: I-ViT is evaluated on various popular models, including ViT [10], DeiT [38], and Swin [31] on ImageNet (ILSVRC-2012) [21] dataset for the large-scale image classification task. The pre-trained models are all obtained from timm[4] library. First, we use Eq. 6 to quantize the weights of the pre-trained FP model for the initialization of I-ViT. Then, we perform quantization-aware fine-tuning using naive STE [2] to recover the accuracy. The optimizer we adopt is AdamW [33], and the search space of the learning rate is [2e-7, 5e-7, 1e-6, 2e-

[4] https://github.com/rwightman/pytorch-image-models

Table 1. Accuracy and latency results on various model benchmarks. Here, accuracy is evaluated on ImageNet dataset, and latency is evaluated on an RTX 2080Ti GPU (batch=8). Compared to the FP baseline, I-ViT, which quantizes the entire computational graph and enables integer-only inference on Turing Tensor Cores, can achieve similar or even slightly higher accuracy and provides a significant 3.72∼4.11× speedup. In addition, I-ViT consistently outperforms existing works FasterTransformer [34] and I-BERT [19] in terms of both accuracy and latency.

Model	Method	Bit-prec.	Size (MB)	Int.-only	Top-1 Acc. (%)	Diff. (%)	Latency (ms)	Speedup
ViT-S	Baseline	FP32	88	×	81.39	-	11.5	×1.00
	FasterTransformer [34]	INT8	22	×	81.07	-0.32	3.26	×3.53
	I-BERT [19]	INT8	22	✓	80.47	-0.92	3.05	×3.77
	I-ViT (ours)	INT8	22	✓	**81.27**	**-0.12**	**2.97**	**×3.87**
ViT-B	Baseline	FP32	344	×	84.53	-	32.6	×1.00
	FasterTransformer [34]	INT8	86	×	84.29	-0.24	8.51	×3.83
	I-BERT [19]	INT8	86	✓	83.70	-0.83	8.19	×3.98
	I-ViT (ours)	INT8	86	✓	**84.76**	**+0.23**	**7.93**	**×4.11**
DeiT-T	Baseline	FP32	20	×	72.21	-	5.99	×1.00
	FasterTransformer [34]	INT8	5	×	72.06	-0.15	1.74	×3.45
	I-BERT [19]	INT8	5	✓	71.33	-0.88	1.66	×3.61
	I-ViT (ours)	INT8	5	✓	**72.24**	**+0.03**	**1.61**	**×3.72**
DeiT-S	Baseline	FP32	88	×	79.85	-	11.5	×1.00
	FasterTransformer [34]	INT8	22	×	79.66	-0.19	3.26	×3.53
	I-BERT [19]	INT8	22	✓	79.11	-0.74	3.05	×3.77
	I-ViT (ours)	INT8	22	✓	**80.12**	**+0.27**	**2.97**	**×3.87**
DeiT-B	Baseline	FP32	344	×	81.85	-	32.6	×1.00
	FasterTransformer [34]	INT8	86	×	81.63	-0.22	8.51	×3.72
	I-BERT [19]	INT8	86	✓	80.79	-1.06	8.19	×3.88
	I-ViT (ours)	INT8	86	✓	**81.74**	**-0.11**	**7.93**	**×4.11**
Swin-T	Baseline	FP32	116	×	81.35	-	16.8	×1.00
	FasterTransformer [34]	INT8	29	×	81.06	-0.29	4.55	×3.69
	I-BERT [19]	INT8	29	✓	80.15	-1.20	4.40	×3.82
	I-ViT (ours)	INT8	29	✓	**81.50**	**+0.15**	**4.29**	**×3.92**
Swin-S	Baseline	FP32	200	×	83.20	-	27.8	×1.00
	FasterTransformer [34]	INT8	50	×	83.04	-0.34	7.35	×3.78
	I-BERT [19]	INT8	50	✓	81.86	-1.34	7.13	×3.90
	I-ViT (ours)	INT8	50	✓	**83.01**	**-0.19**	**6.92**	**×4.02**

6]. The above implementations are done on PyTorch[5], and the model inference details (*e.g.*, bit-shifting) follow the TVM implementation to ensure consistent accuracy with the TVM deployment.

Table 1 reports the accuracy results of I-ViT and various baselines on multiple benchmark models on ImageNet dataset. Although I-ViT reduces the bit-precision of the parameters and enables integer-only inference, it maintains comparable accuracy, even slightly more than the FP baseline, which adequately demonstrates the effectiveness and robustness of the proposed approximation schemes. For instance, DeiT-S obtained by I-ViT achieves 80.12% Top-1 accuracy with 8-bit integer-only inference, which is even 0.27% higher than the FP baseline. In addition, I-ViT is consistently superior to FasterTransformer [34] and I-BERT [19], and in particular, the naive application of I-BERT to ViTs suffers from mismatched approximations, making the results far from satisfactory. For Swin-S, I-BERT results in a noticeable 1.34% accuracy drop, while I-ViT still offers high robustness.

[5] https://github.com/pytorch/pytorch

4.2. Latency Evaluation

Implementation Details: We deploy I-ViT on an RTX 2080Ti GPU using TVM to measure the real hardware latency. First, we use TVM to build and compile the same model as PyTorch, followed by the auto-tuning to optimize the computational schedule, and then we perform the end-to-end latency tests. Note that although the GPU is not an integer-only hardware, depending on the DP4A instructions, I-ViT can perform efficient integer-only inference on its Turing Tensor Cores. Since ViT [10] and DeiT [38] have the same model structure in the inference process, ViT enjoys the same acceleration as DeiT.

The latency results of I-ViT on an RTX 2080Ti GPU (batch=8) are also shown in Table 1. FasterTransformer [34], which leaves non-linear operations as floating-point arithmetic and cannot be deployed on integer-only hardware, produces disappointing acceleration effects. In the case of DeiT-T and DeiT-S quantization, it only accelerates the model by 3.45× and 3.53×, respectively. Note that the disappointing acceleration stems not only from the inefficiency of the floating-point arithmetic units, but also

Table 2. Ablation studies of accuracy and latency of Shiftmax, ShiftGELU, and I-LayerNorm. Latency is evaluated on an RTX 2080Ti GPU (batch=8). Replacing (→) Shiftmax and ShiftGELU with second-order polynomial approximations [19] leads to lower accuracy and higher latency, and I-LayerNorm suffers from non-trivial accuracy loss due to the mismatch in the data distribution.

Model	Method	Shifting-oriented	Top-1 Acc. (%)	Diff. (%)	Latency (ms)	Diff. (ms)
DeiT-B	I-ViT(ours)	✓	**81.74**	-	7.93	-
	Shiftmax → Poly. [19]	×	81.62	-0.12	8.04	+0.11
	ShiftGELU → Poly. [19]	×	80.88	-0.86	8.10	+0.17
	Shiftmax → LIS [30]	×	81.66	-0.08	8.05	+0.12
	I-LayerNorm → L1 LayerNorm [29]	-	79.25	-2.49	**7.91**	-0.02
Swin-S	I-ViT(ours)	✓	**83.01**	-	6.92	-
	Shiftmax → Poly. [19]	×	82.79	-0.22	7.02	+0.10
	ShiftGELU → Poly. [19]	×	82.10	-0.91	7.08	+0.16
	Shiftmax → LIS [30]	×	82.89	-0.12	7.03	+0.11
	I-LayerNorm → L1 LayerNorm [29]	-	79.69	-3.32	**6.90**	-0.02

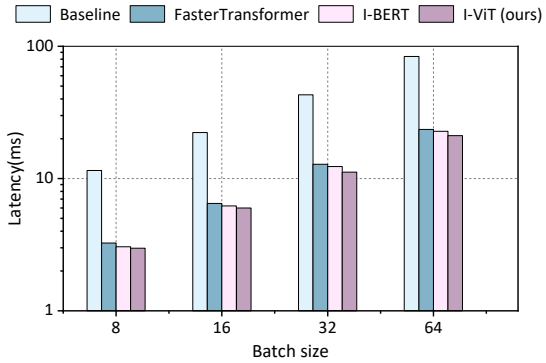


Figure 4. Latency results of DeiT-S [38] evaluated on an RTX 2080Ti GPU with various batch sizes. I-ViT maintains a constant acceleration effect for the same model architecture at various batch sizes.

from the data interaction overheads between the integer and floating-point arithmetic units, since the integer results from the previous layers need to be passed to the floating-point units and returned later. In contrast, I-BERT [19] and I-ViT can achieve integer-only inference by utilizing the integer arithmetic units of Turing Tensor Cores. Importantly, compared to I-BERT, our proposed I-ViT makes fuller use of the efficient shifters in hardware logic and thus has a more advantageous 3.72∼4.11× speedup. For instance, for DeiT-B with 32.6ms latency at FP baseline, the integer inference latencies of the quantized models obtained by I-BERT and I-ViT are 8.19ms and 7.93ms, respectively, with the latter being 0.26ms faster. Moreover, from the results, I-ViT is more effective in accelerating more computationally-intensive models.

4.3. Ablation Studies

Here, we perform ablation studies for comparison with the second-order polynomial approximations in I-BERT [19], LIS in FQ-ViT [30], and L1 LayerNorm in Fully-8bit [29], and the results are shown in Table 2. Due to the differences in data distribution of ViTs and language models, replacing Shiftmax and ShiftGELU with the polynomial approximations results in severe accuracy degradation, with performance losses of 0.95% and 1.15% in the quantization of DeiT-B and Swin-S, respectively. In particular, polynomial GELU that only approximates for the specific interval is not applicable to ViTs and thus has most contribution in the accuracy degradation. For instance, polynomial GELU reduces the Top-1 accuracy by 0.86% and 0.91% compared to ShiftGELU in the quantization of DeiT-B and Swin-S, respectively. It is also worth mentioning that the proposed schemes are shifting-oriented arithmetic and can thus benefit more from the efficient hardware logic, while the second-order polynomial approximations lack this advantage. LIS also encounters the above problems, since it is simply built on top of I-BERT. For L1 LayerNorm, although it simplifies the computation to achieve faster speed, its low approximation capability leads to non-trivial accuracy loss.

In addition, we also evaluate the latency of DeiT-S with various batch sizes, as shown in Figure 4. It can be seen that I-ViT is robust to the batch size and can maintain a constant acceleration effect. Also, it should be highlighted that despite the significant speedup on the RTX 2080Ti GPU that provides an evident strength of I-ViT, both the software support of TVM and the hardware support of Turing Tensor Cores are not optimal. For instance, there is no full parallelism after increasing the batch size in both FP and quantized cases, $i.e.$, increasing the batch size results in a corresponding increase in latency. Therefore, it is believed that deploying I-ViT on dedicated hardware ($e.g.$, FPGAs) will further enhance the acceleration potential.

5. Conclusions

In this paper, we propose I-ViT, which is the first integer-only quantization scheme for ViTs to the best of our knowledge. I-ViT quantizes the entire computational graph to enable the integer-only inference, where linear operations follow the dyadic arithmetic pipeline; and non-linear operations are performed by the proposed novel light-weight integer-only approximation methods. In particular, Shiftmax and ShiftGELU perform most arithmetic with bit-

shifting, which can fully benefit from the efficient hardware logic. Compared to the FP baseline, I-ViT achieves similar (or even slightly higher) accuracy on various benchmarks. In addition, we utilize TVM to deploy I-ViT on an RTX 2080Ti GPU, whose Turing Tensor Cores can accelerates the integer-only inference of ViTs, achieving a 3.72$\sim$4.11$\times$ speedup over the FP model.

In the future, we will consider deploying I-ViT on dedicated integer-only hardware ($e.g.$, FPGAs) to obtain better acceleration performance. Furthermore, we also plan to extend I-ViT to more complex vision tasks ($e.g.$, object detection and semantic segmentation).

Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant 2022ZD0119402; in part by the National Natural Science Foundation of China under Grant 62276255.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021. 2

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3, 6

[3] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 696–697, 2020. 3

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, Cham, 2020. 1, 2

[5] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021. 2

[6] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, 2018. 2

[7] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 1, 3

[8] Richard Crandall and Carl Pomerance. *Prime numbers*. Springer, 2001. 6

[9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019. 3

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 1, 2, 6, 7

[11] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 3

[12] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021. 1, 3

[13] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 2022. 1, 2

[14] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[15] Zhiwei Hao, Jianyuan Guo, Ding Jia, Kai Han, Yehui Tang, Chao Zhang, Han Hu, and Yunhe Wang. Learning efficient vision transformers via fine-grained manifold distillation. In *Advances in Neural Information Processing Systems*, 2022. 1

[16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 5

[17] Zejiang Hou and Sun-Yuan Kung. Multi-dimensional vision transformer compression via dependency guided gaussian process search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3669–3678, 2022. 1, 2

[18] 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*, pages 2704–2713, 2018. 1, 2, 3, 4

[19] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR, 2021. 2, 3, 5, 6, 7, 8

[20] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 1, 3

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 6

[22] 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*, pages 2810–2819, 2019. 3

[23] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2020. 3

[24] Zhikai Li, Mengjuan Chen, Junrui Xiao, and Qingyi Gu. Psaq-vit v2: Towards accurate and general data-free quantization for vision transformers. *arXiv preprint arXiv:2209.05687*, 2022. 3

[25] Zhikai Li, Liping Ma, Mengjuan Chen, Junrui Xiao, and Qingyi Gu. Patch similarity aware data-free quantization for vision transformers. In *European conference on computer vision*, pages 154–170, 2022. 1, 3

[26] Zhikai Li, Liping Ma, Xianlei Long, Junrui Xiao, and Qingyi Gu. Dual-discriminator adversarial framework for data-free quantization. *Neurocomputing*, 2022. 3

[27] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. *arXiv preprint arXiv:2212.08254*, 2022. 2, 3

[28] Zhexin Li, Tong Yang, Peisong Wang, and Jian Cheng. Q-vit: Fully differentiable quantization for vision transformer. *arXiv preprint arXiv:2201.07703*, 2022. 3

[29] Ye Lin, Yanyang Li, Tengbo Liu, Tong Xiao, Tongran Liu, and Jingbo Zhu. Towards fully 8-bit integer inference for the transformer model. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3759–3765, 2021. 2, 3, 6, 8

[30] 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*, pages 1173–1179, 2022. 2, 3, 6, 8

[31] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2, 6

[32] 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, 2021. 3

[33] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 6

[34] NVIDIA. FasterTransformer, https://github.com/nvidia/fastertransformer.git, 2022. 1, 2, 6, 7

[35] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020. 1

[36] Jacob R Stevens, Rangharajan Venkatesan, Steve Dai, Brucek Khailany, and Anand Raghunathan. Softermax: Hardware/software co-design of an efficient softmax for transformers. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 469–474, 2021. 3, 4

[37] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12165–12174, 2022. 1

[38] 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*, pages 10347–10357. PMLR, 2021. 2, 6, 7, 8

[39] Hanrui Wang, Zirui Li, Jiaqi Gu, Yongshan Ding, David Z Pan, and Song Han. On-chip qnn: Towards efficient on-chip training of quantum neural networks. *arXiv preprint arXiv:2202.13239*, 2022. 2

[40] 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*, pages 8612–8620, 2019. 3

[41] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020. 1

[42] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10033–10041, 2021. 2

[43] Huanrui Yang, Hongxu Yin, Pavlo Molchanov, Hai Li, and Jan Kautz. Nvit: Vision transformer compression and parameter redistribution. *arXiv preprint arXiv:2110.04869*, 2021. 2

[44] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021. 1, 2, 3, 4

[45] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization framework for vision transformers. *arXiv preprint arXiv:2111.12293*, 2021. 3

[46] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018. 3

[47] Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Minivit: Compressing vision transformers with weight multiplexing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12145–12154, 2022. 2

[48] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017. 3

[49] 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. 3

[50] Danyang Zhu, Siyuan Lu, Meiqi Wang, Jun Lin, and Zhongfeng Wang. Efficient precision-adjustable architecture for softmax function in deep learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(12):3382–3386, 2020. 3

[51] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2020. 2