2023 Summer Seminar

The Quantization of Vision Transformer

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
Jincheol Yang

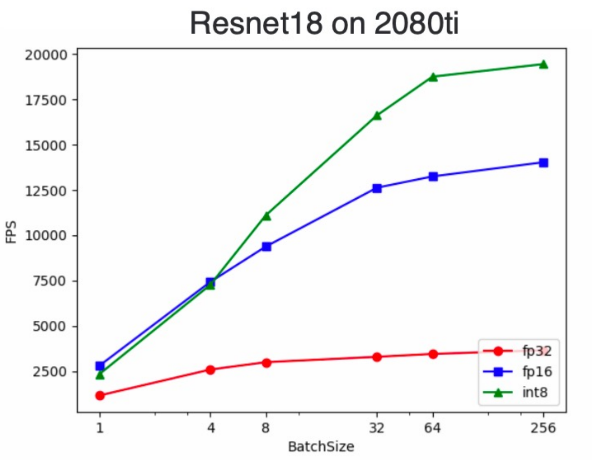
Outline

- Intro
  - What is quantization?
  - Post-training quantization and Quantization-aware training
  - CNNs vs ViTs
- Papers
  - PTQ4ViT: Post-Training Quantization for Vision Transformers with Twin Uniform Quantization (ECCV 2022)
  - I-ViT: Integer-only Quantization for Efficient Vision Transformer Inference (ICCV 2023)
- Conclusion

Intro

- What is quantization?
  - Motivation for optimizing models
    - Model size reduction
      - Computer vision models have huge model size
        - ✓ Improvements in the accuracy have highly over-parameterized
    - Performance benefits
      - Edge devices don’t have enough memory
        - ✓ Hardware efficiency on several metrics (latency, energy and power)
    - Applications such as real-time intelligent(health care monitoring, autonomous driving, …)
  - Method for optimizing models
    - Quantization
    - Pruning
    - Knowledge Distillation
    - Efficient Network Design

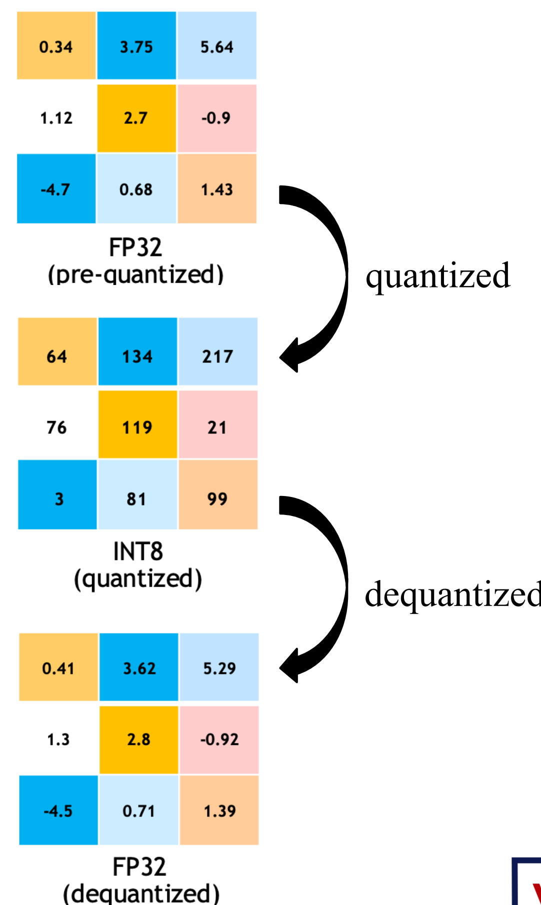
Comparison on latency



Intro

- What is quantization?
  - Process of reducing the precision of the model parameters(weights and activations)
    - Floating point(FP) value => INT value
  - Basic concepts
    - Quantization : $Q(r) = \left\lfloor \frac{r}{S} \right\rceil - Z; S = \frac{\beta-\alpha}{2^b-1}$
    - Dequantization : $\tilde{r} = S(Q(r) + Z)$

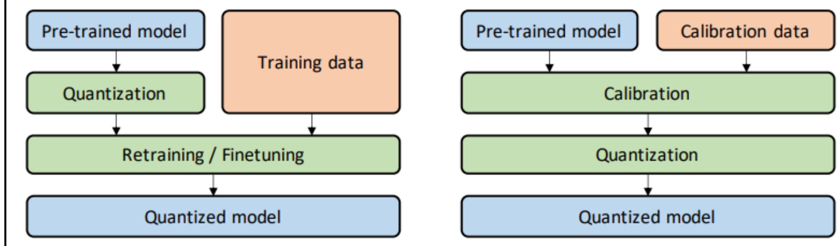
    Notations
      - $Q(r)$ = quantized representation of r
      - r = real value (FP)
      - S = scale factor
      - Z = zero-point
      - α, β =bounded range(clipping range)
      - b = bit width
      - $\lfloor \cdot \rceil$ = rounding function



Intro

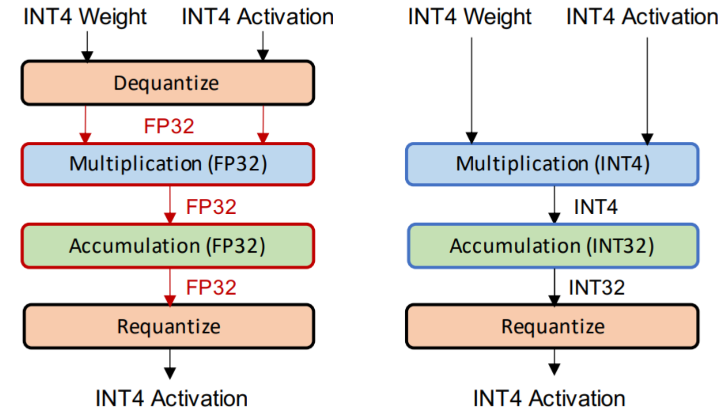
- What is quantization?
  - Basic concepts
    - Quantization : $Q(r) = \left\lfloor \frac{r}{S} \right\rceil - Z; S = \frac{\beta - \alpha}{2^b - 1}$
    - Dequantization : $\tilde{r} = S(Q(r) - Z)$
  - Considerations
    - Fine-tuning methods (QAT vs PTQ)
      - PTQ - Static vs Dynamic
    - Additional elements
      - Batch normalization folding
      - Symmetric vs Asymmetric
      - Uniform vs Non-uniform
      - Quantization granularity
    - Advanced concepts
      - Simulated vs Integer-only
      - Mixed-Precision
      - Combined with various method(Pruning, KD)

Overview of QAT and PTQ



< Left : QAT, Right : PTQ >

Overview of Simulated and Integer-only quantization



< Left : Simulated, Right : Integer-only >

서강대학교 SOGANG UNIVERSITY

Intro

- Post-training quantization and Quantization-aware training
  - Post-training quantization(PTQ)
    - A method of quantizing the resulting parameter values at pre-trained model
      - Advantages : No fine-tuning required
      - Disadvantages : For small models with large parameter size, accuracy drop is large
    - Static vs Dynamic method
      - Static : The quant parameters of weight and activation values are kept unchanged in inference
      - Dynamic : Weights are statically quantized, but the quant parameters of activations changed per-sample
  - Quantization-aware training(QAT)
    - A method of quantization finds optimal parameter values during training
      - Advantages : Accuracy drop is very small
      - Disadvantages : Fine-tuning required

Intro

- CNNs vs ViTs
  - The trend of Vision Transformer on paperswithcode.com

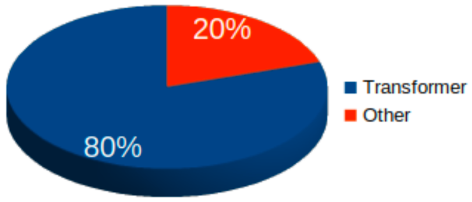
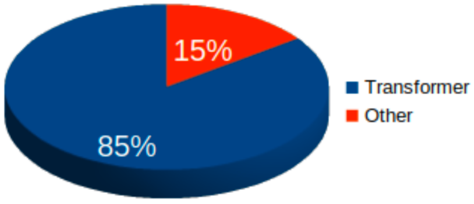


Image classification
on ImageNet-1k



Object Detection
on COCO

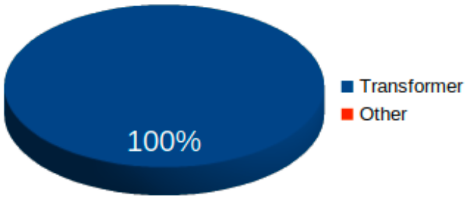
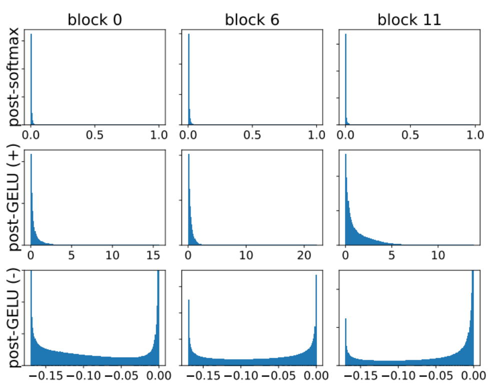


Image Segmentation
on ADE-20k

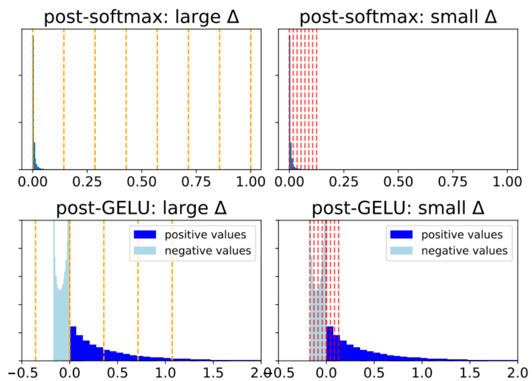
  - Motivation of Vision Transformer
    - Transformer has achieved remarkable performance on a variety of computer vision application
    - Vision Transformers are often of sophisticated architectures, which are more difficult to be developed on mobile devices compared with CNN

PTQ4ViT

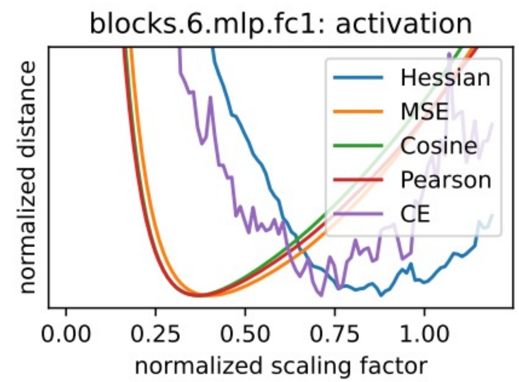
- Keyword
  - Weight, Activation map / Uniform, Static(calibration, clipping)
  - Simulation(fake quant) / Post-training
- Abstract
  - Post-training quantization method
  - Using twin uniform quantization method and Hessian guided metric
    - Why do we use twin uniform quantization and Hessian metric?



< Distribution of post-softmax, post-GELU >



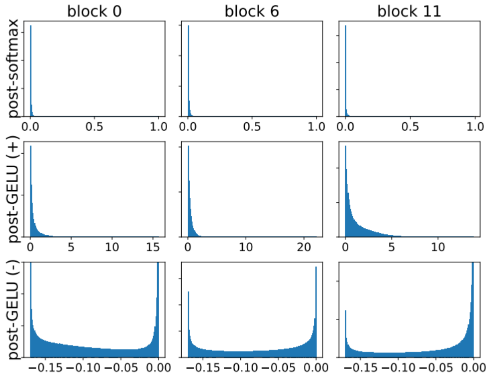
< Different scaling factor >



< Distance between CE and various metric >

PTQ4ViT

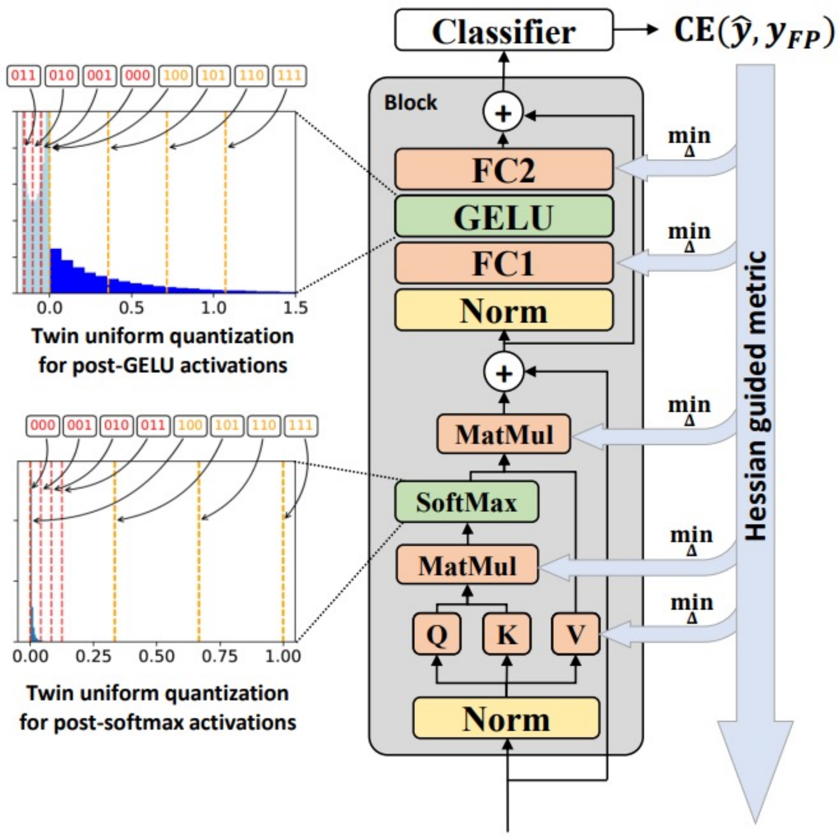
- Challenges
  - PTQ has achieved great success on CNN
    - But directly bringing it to vision transformer results in more than 1% accuracy drop
  - Why?
    - Softmax → unbalanced distribution → most of values are very close to zero
      - Large scaling factor to make small values to zero -> it least to a large error
    - GELU → highly asymmetrical distribution → difficult to quantify both the positive and negative values



< Distribution of post-softmax, post-GELU >

PTQ4ViT

- Overview of the proposed framework



① Twin uniform quantization(adjusting scale)
- It can be efficiently processed on existing hardware devices(CPU, GPU)
  - Post GELU activations
  - Post softmax activations

② Hessian guided metric
- The metric to determine the optimal scaling factor is not accurate on vision transformers
  - MSE, Cosine distance [EasyQuant[2)]] - CNN
  - Pearson correlation coefficient [PTQ for ViT[3)]]
- Hessian guided metric to determine the quantization parameters

PTQ4ViT

- Base PTQ Method
  - Basic concepts
    - The main body of ViTs is a stack of blocks, each block is divided into a multi-head self-attention(MSA) module and a multi-layer perceptron(MLP)
    - The simplest symmetric uniform quantization

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

  - Find optimal scales with

$$\min_{\Delta_A \Delta_B} distance(O, \hat{O})$$

$$\hat{O} = \Delta_A \Delta_B A_q B_q$$

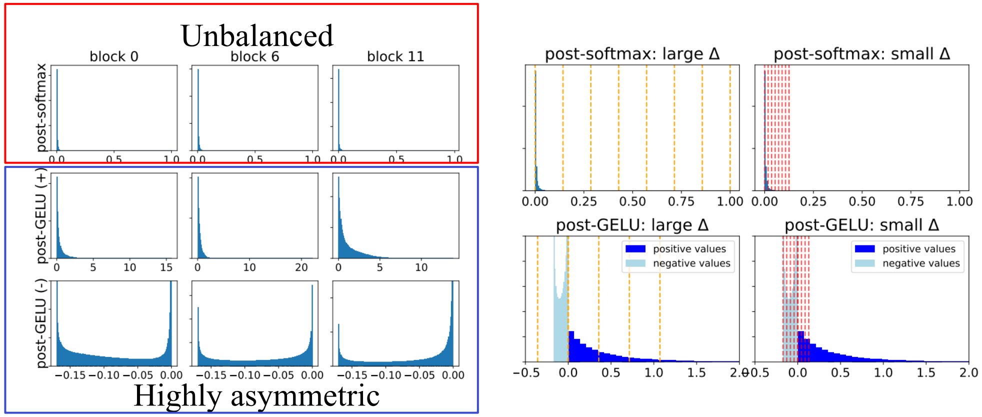
    - EasyQuant[2)] uses cosine distance as the metric to calculate the distance
    - Search $\Delta_A \Delta_B$ from (In EasyQuant, $\alpha, \beta = 0.5, 1.2$)

$$\left[\alpha \frac{A_{max}}{2^{k-1}}, \beta \frac{A_{max}}{2^{k-1}}\right], \left[\alpha \frac{B_{max}}{2^{k-1}}, \beta \frac{B_{max}}{2^{k-1}}\right]$$

    - Base PTQ results in more than 1% accuracy drop

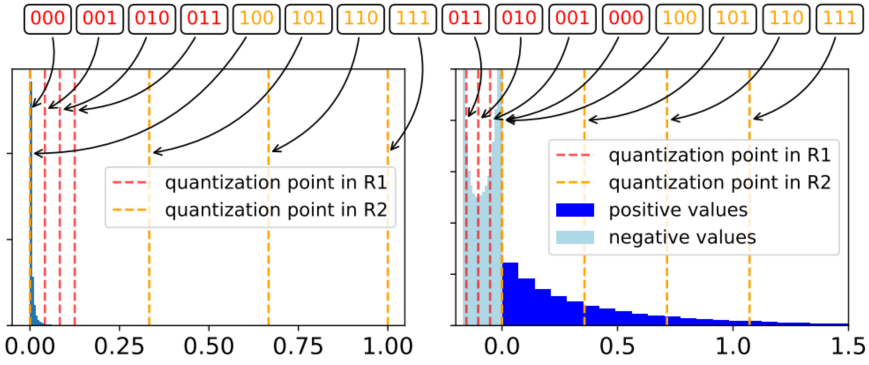
PTQ4ViT

- Method
  - Twin Uniform Quantization
    - Large values after softmax → high correlation (two patches) → large scaling factors



Simply,
- Large values → large scaling factors
- Small values → small scaling factors

    - Twin uniform quantization → efficiently processing on CPU and GPUs



Flag expression

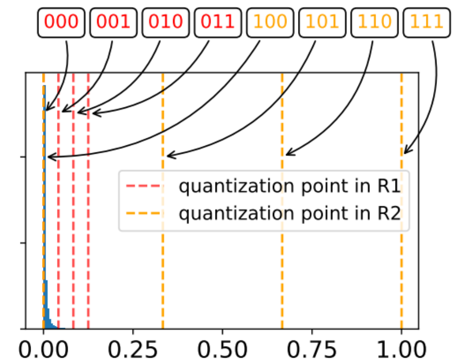
000

- For sign bit
  - 0 → large scaling factors
  - 1 → small scaling factors

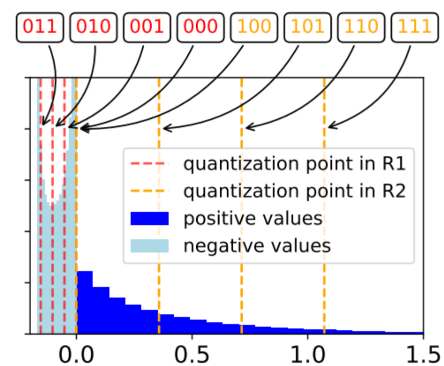
PTQ4ViT

- Method
  - Twin Uniform Quantization
    - Two quantization ranges (R1, R2) are controlled by two scaling factors Δ_{R1}, Δ_{R2}

$$T_K(x, \Delta_{R1}, \Delta_{R2}) = \begin{cases} \varphi_{k-1}(x, \Delta_{R1}), x \in R1 \\ \varphi_{k-1}(x, \Delta_{R2}), otherwise \end{cases}$$



    - Post Softmax case
      - values $\in R1 = [0, 2^{k-1}\Delta^{s}_{R1})$ → well quantized by a small Δ^{s}_{R1}
      - Use fixed range $\Delta^{s}_{R2} = \frac{1}{2^{k-1}}$, $R2 = [0,1]$ → large values in R2
    - Post GELU case
      - $R1 = [-2^{k-1}\Delta^{g}_{R1}, 0]$
        - ✓Use fixed range = Δ^{g}_{R1}
        - ✓R1 covers the entire range of negative numbers
      - $R2 = [0, 2^{k-1}\Delta^{g}_{R2}]$
    - When calibrating the network, search for the optimal Δ^{s}_{R1} , Δ^{g}_{R2}

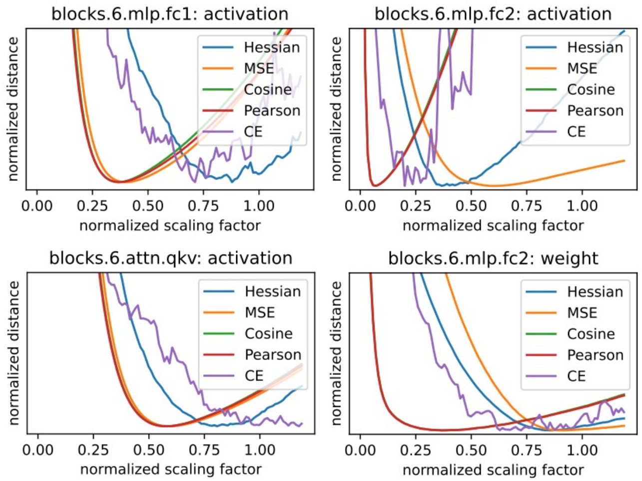


서강대학교 Sogang University

VDS LAB

PTQ4ViT

- Method
  - Hessian Guided Metric
    - Prior papers greedily determine the scaling factors of inputs and weights layer by layer
      - MSE, cosine distance, Pearson correlation → inaccurate
      - Blocks.6.mlp.fc1:activation → $\frac{0.4A_{max}}{2^{k-1}}$ → optimal = 0.75
    - The distance between the last layer's output before and after quantization can be more accurate in PTO
      - Executing the network many times to calculate the last layer's output, which consumes too much time



Hessian metric is most similar to CE

< Distance between the layer outputs before and after quantization and CE >

PTQ4ViT

- Method
  - Hessian Guided Metric
    - Hessian guided metric to determine the scaling factors → high accuracy and quick quantization
      - $L = CE(\hat{y}, y)$, *where y is FP32 result* ; *CE*:cross-entropy
    - Quantization brings a small perturbation ϵ on weight
      - $\widehat{W} = W + \epsilon$
    - Analyze the influence of quantization on task loss by Taylor series expansion
      - $\mathbb{E}[L(\widehat{W})] - \mathbb{E}[L(W)] \approx \epsilon^T \bar{g}^{(W)} + \frac{1}{2}\epsilon^T \bar{H}^{(W)} \epsilon$
        - ✓ $\bar{g}^{(W)}$ is gradients and $\bar{H}^{(W)}$ is the Hessian matrix, $\mathbb{E}[L(W)]$ is the expectation of loss
    - The target is to find the scaling factors to minimize the influence
      - $\min_{\Delta}(\mathbb{E}[L(\widehat{W})] - \mathbb{E}[L(W)])$
    - The optimization can be approximated
      - $\min_{\Delta}(\mathbb{E}\left[(\hat{O}^l - O^l)^T diag\left(\left(\frac{\partial L}{\partial O_1^l}\right)^2, \dots, \left(\frac{\partial L}{\partial O_{|O^l|}^l}\right)^2\right)(\hat{O}^l - O^l)\right]$
        - ✓ $\hat{O}^l, O^l$ are the outputs of the l_{th} layer before and after quantization, respectively

Approximation because of the difficulty of direct calculation

Firstly, Use Taylor series expansion in AdaRound[2)]

Use term in BRECQ proposed

Minimum $\hat{O}^l, -O^l$

서강대학교 SOGANG UNIVERSITY

VDS LAB

1) Zhihang, Yuan, et al. "PTQ4ViT: Post-Training Quantization for Vision Transformers with Twin Uniform Quantization",(ECCV, 2022)

PTQ4ViT

- Method

Algorithm

Searches for the optimal scaling factors of each layer

1 **for** l *in* 1 *to* L **do**
2 | forward-propagation $O^l \leftarrow A^l B^l$;
3 **end**
4 **for** l *in* L *to* 1 **do**
5 | backward-propagation to get $\frac{\partial L}{\partial O^l}$;
6 **end**
7 **for** l *in* 1 *to* L **do** α , β = [0.5, 1.2] , n =100
8 | initialize $\Delta^*_{B^l} \leftarrow \frac{B^l_{max}}{2^{k-1}}$;
9 | generate search spaces of Δ_{A^l} and Δ_{B^l};
10 | **for** r = 1 *to* #*Round* **do**
11 | | search for $\Delta^*_{A^l}$ using Eq. (7);
12 | | search for $\Delta^*_{B^l}$ using Eq. (7);
13 | **end**
14 **end**

- Line1~6 : Phase1
  - Collect output and gradient of the output in each layer before quantization on the calibration dataset
    - ✓ l_{th} output → O^l → forward
    - ✓ Backward → gradients $\frac{\partial L}{\partial O^l_1}, \dots, \frac{\partial L}{\partial O^l_a}$

- Line7~14 : Phase2
  - Search for the optimal scaling factors layer by layer
    - ✓ Different scaling factors are used to quantize the activation and weight values
    - ✓ $\hat{O}^l$ is calculated → search for the optimal scaling factor Δ → minimize

서강대학교 SOGANG UNIVERSITY

VDS LAB

PTQ4ViT

- Experimental results

Results between base PTQ and PTQ4ViT

- **Base PTQ** : EasyQuant results more than 1% accuracy drop
- **PTQ4ViT** : low or slightly high accuracy

Model	FP32	Base PTQ W8A8	Base PTQ W6A6	PTQ4ViT W8A8	PTQ4ViT W6A6
ViT-S/224/32	75.99	73.61(2.38)	60.14(15.8)	75.58(0.41)	71.90(4.08)
ViT-S/224	81.39	80.46(0.91)	70.24(11.1)	81.00(0.38)	78.63(2.75)
ViT-B/224	84.54	83.89(0.64)	75.66(8.87)	84.25(0.29)	81.65(2.89)
ViT-B/384	86.00	85.35(0.64)	46.88(39.1)	85.82(0.17)	83.34(2.65)
DeiT-S/224	79.80	77.65(2.14)	72.26(7.53)	79.47(0.32)	76.28(3.51)
DeiT-B/224	81.80	80.94(0.85)	78.78(3.01)	81.48(0.31)	80.25(1.55)
DeiT-B/384	83.11	82.33(0.77)	68.44(14.6)	82.97(0.13)	81.55(1.55)
Swin-T/224	81.39	80.96(0.42)	78.45(2.92)	81.24(0.14)	80.47(0.91)
Swin-S/224	83.23	82.75(0.46)	81.74(1.48)	83.10(0.12)	82.38(0.84)
Swin-B/224	85.27	84.79(0.47)	83.35(1.91)	85.14(0.12)	84.01(1.25)
Swin-B/384	86.44	86.16(0.26)	85.22(1.21)	86.39(0.04)	85.38(1.04)

Results of the effect of the proposed method

- When Hessian Guided metric is used, accuracy is high
- Overall, when all methods are used, various model accuracy is high

Model	Hessian Guided	Softmax Twin	GELU Twin	Top-1 Accuracy W8A8	Top-1 Accuracy W6A6
ViT-S/224 81.39				80.47	70.24
	✓			80.93	77.20
	✓	✓		81.11	78.57
	✓		✓	80.84	76.93
		✓	✓	79.25	74.07
	✓	✓	✓	81.00	78.63
ViT-B/224 84.54				83.90	75.67
	✓			83.97	79.90
	✓	✓		84.07	80.76
	✓		✓	84.10	80.82
		✓	✓	83.40	78.86
	✓	✓	✓	84.25	81.65
ViT-B/384 86.00				85.35	46.89
	✓			85.42	79.99
	✓	✓		85.67	82.01
	✓		✓	85.60	82.21
		✓	✓	84.35	80.86
	✓	✓	✓	85.89	83.19

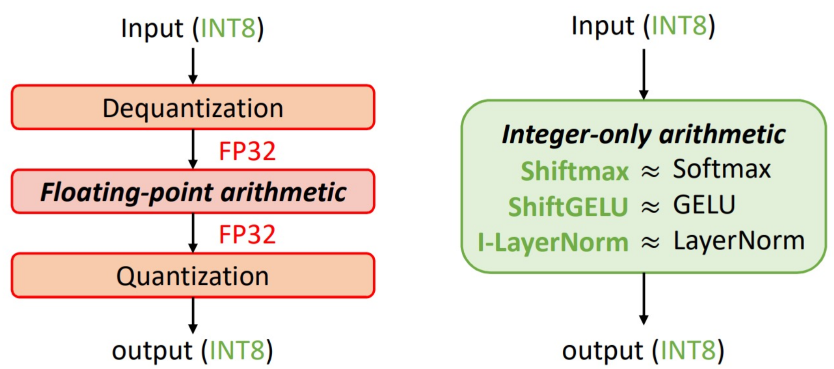
Conclusion

- Conclusion
  - Twin uniform quantization and a Hessian guided metric are proposed
  - They can decrease the quantization error and improve the prediction accuracy

- Limitations
  - Do not quantize Non-linear layer
    - Softmax, GELU, LayerNorm → Integer-only quantization?
  - Taylor series expansion is the approximation
    - CE and Hessian do not match completely

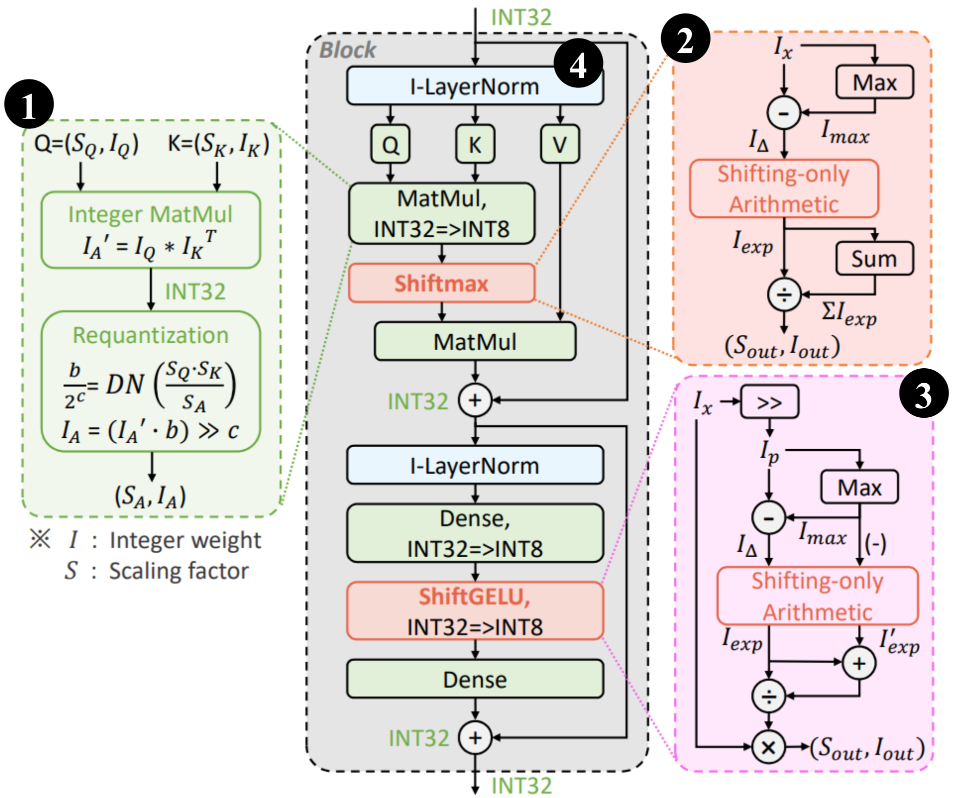
I-ViT

- Keyword
  - All layer(weight, activation map, softmax, GELU, LayerNorm) / Uniform
  - Integer-only / Simulation(fake quant) / QAT
- Abstract
  - Quantization-aware training method
  - First work on integer-only quantization for ViTs.
    - Apply to Quantization of Softmax, GELU, LayerNorm
  - What is integer-only quantization?
    - Eliminates dequantization and enables to be performed with integer-only arithmetic



I-ViT

- Overview of the proposed framework



① Dyadic Arithmetic for Linear Operations
- Use integer bit-shifting
  - Embedding, MatMul, Dense layer

② Integer-only Softmax: Shiftmax
- Due to the non-linearity, use the approximation and bit-shifting

③ Integer-only GELU: ShiftGELU
- Due to the non-linearity, use the approximation by sigmoid function and bit-shifting

④ Integer-only LayerNorm: I-LayerNorm
- Use integer iterative approach via bit-shifting

I-ViT

- Method
  - Basic concepts
    - The main body of ViTs is a stack of blocks, each block is divided into a multi-head self-attention(MSA) module and a multi-layer perceptron(MLP)
    - The simplest symmetric uniform quantization

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

- $\hat{X} = MSA(LayerNorm(X)) + X$

$$MSA(X) = Concat(Attn_1, Attn_2, \dots, Attn_h)W^O$$

$$;\ Attn_i = Softmax\left(\frac{Q_i \cdot K_i^T}{\sqrt{d}}\right)V_i$$

- $Y = MLP(LayerNorm(\hat{X})) + \hat{X}$

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

- $\hat{X} = MSA(I - LayerNorm(X)) + X$

$$MSA(X) = Concat(Attn_1, Attn_2, \dots, Attn_h)W^O$$

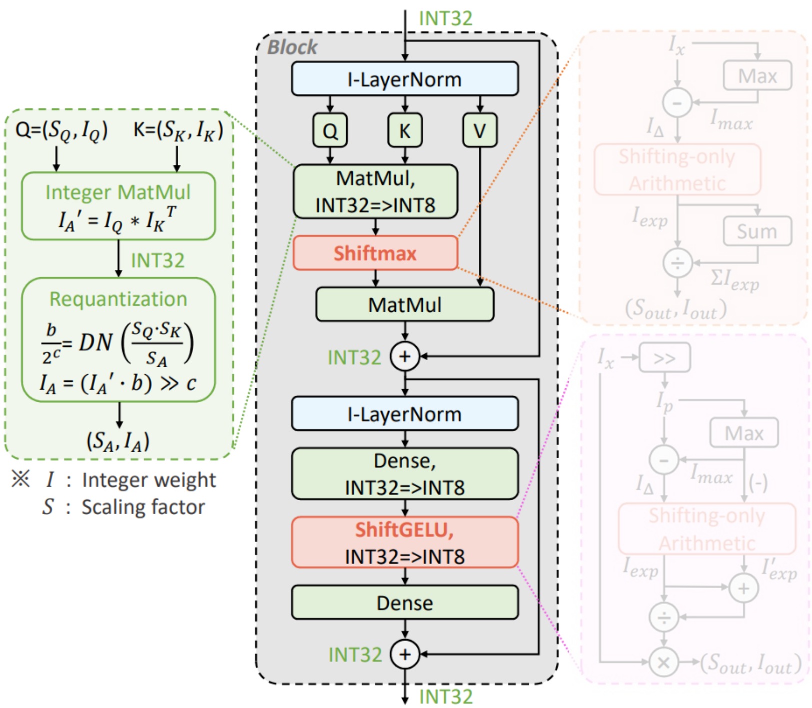
$$;\ Attn_i = Shiftmax\left(\frac{Q_i \cdot K_i^T}{\sqrt{d}}\right)V_i$$

- $Y = MLP(I - LayerNorm(\hat{X})) + \hat{X}$

$$MLP(\hat{X}) = ShiftGELU(\hat{X}W_1 + b_1)W_2 + b_2$$

I-ViT

- Method
  - Dyadic Arithmetic for Linear Operations
    - The dyadic arithmetic pipeline, which uses integer bit-shifting
    - MatMul, Dense layer (INT32=>INT8)



For example ; about MatMul

- Input(query, key)
  - $Q = (S_Q, I_Q), K = (S_K, I_K)$
- Output
  - $A' = S_{A'} \cdot I_{A'} = S_Q \cdot S_K \cdot (I_Q * I_K^T)$
- Requantization (INT32 => INT8)
  - $I_A = \left\lfloor \frac{S_{A'} \cdot I_{A'}}{S_A} \right\rceil = \left\lfloor \frac{S_Q \cdot S_K}{S_A} \cdot (I_Q * I_K^T) \right\rceil$
- Convert the rescaling to a dyadic number(DN)
  - $DN(\frac{S_Q \cdot S_K}{S_A}) = \frac{b}{2^c}$
- The integer-only arithmetic pipeline of MatMul
  - $I_A = \left(b \cdot (I_Q * I_K^T)\right) \gg c$

Notation

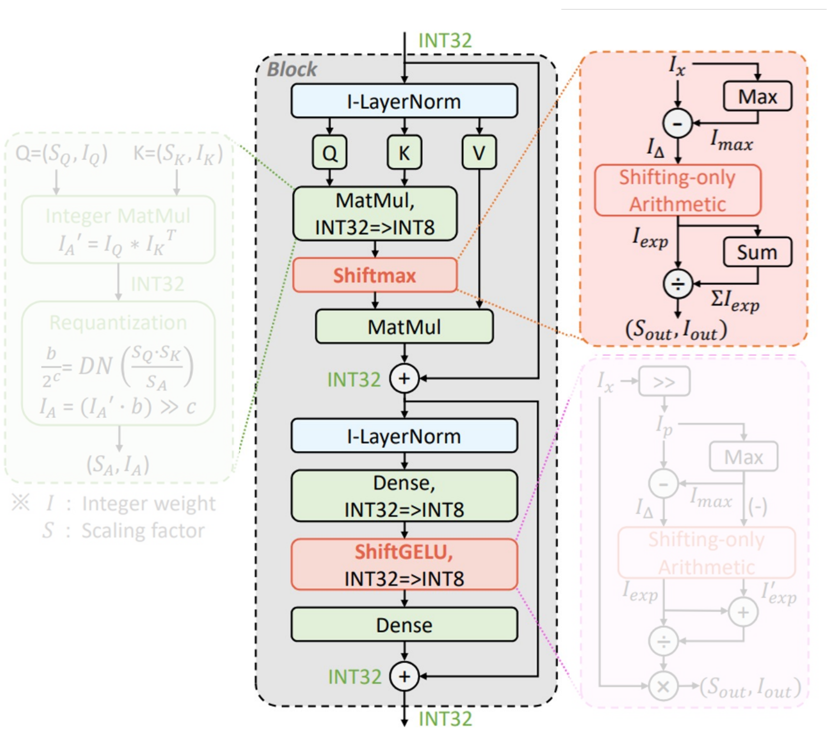
I : intput(INT8; quantized)
S : scaling factor
$I_{A'}$: $I_Q * I_K^T$ (INT32)
S_A : pre-calculated scaling factor of the output activation(FP)
DN : fraction whose denominator is a power of two

서강대학교 SOGANG UNIVERSITY

VDS LAB

I-ViT

- Method
  - Integer-only Softmax: Shiftmax
    - Due to the non-linearity, Softmax cannot follow the dyadic arithmetic
    - The approximation method Shiftmax



Pseudo code

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) **:**
- $I_p \leftarrow I + (I \gg 1) - (I \gg 4)$; ▷ $I \cdot \log_2 e$
- $I_0 \leftarrow \lceil 1/S \rfloor$;
- $q \leftarrow \lfloor I_p / (-I_0) \rfloor$; ▷ 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}); ▷ $I_{out} \cdot S_{out} \approx \mathrm{Softmax}(I_{in} \cdot S_{in})$

End Function

1) Zhikai, Li, et al. "I-ViT: Integer-only Quantization for Efficient Vision Transformer Inference", (ICCV 2023)

I-ViT

- Method
  - Integer-only Softmax: Shiftmax

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` $(I_\Delta, S_{in});$
  $(I_{out}, S_{out}) \leftarrow$ `IntDiv` $(I_{exp}, \sum I_{exp}, k_{out});$ ▷ Eq. 16
  return $(I_{out}, S_{out});$
  ▷ $I_{out} \cdot S_{out} \approx \mathrm{Softmax}(I_{in} \cdot S_{in})$
End Function

$$\mathrm{Softmax}(x) = \frac{e^{S_{\Delta_i} \cdot I_{\Delta_i}}}{\sum_j e^{S_{\Delta_j} \cdot I_{\Delta_j}}}$$

ShiftExp function

- Line1 : To use shifter, convert the base e to 2 (approximation)
  $\therefore \log_2 e = (1.0111)_b$
  $e^{S_\Delta \cdot I_\Delta} = 2^{S_\Delta \cdot (I_\Delta \cdot \log_2 e)} \approx 2^{S_\Delta \cdot (I_\Delta + (I_\Delta \gg 1) - (I_\Delta \gg 4))}$
- Line3, 4: integer and decimal part
  Due to not integer, calculating integer and decimal part respectively
- Line5: Approximate the linear function for low-cost computation
  $\therefore 2^{S_\Delta \cdot I_\Delta} \approx S_\Delta \cdot I_\Delta$
- Line6: To avoid too small values

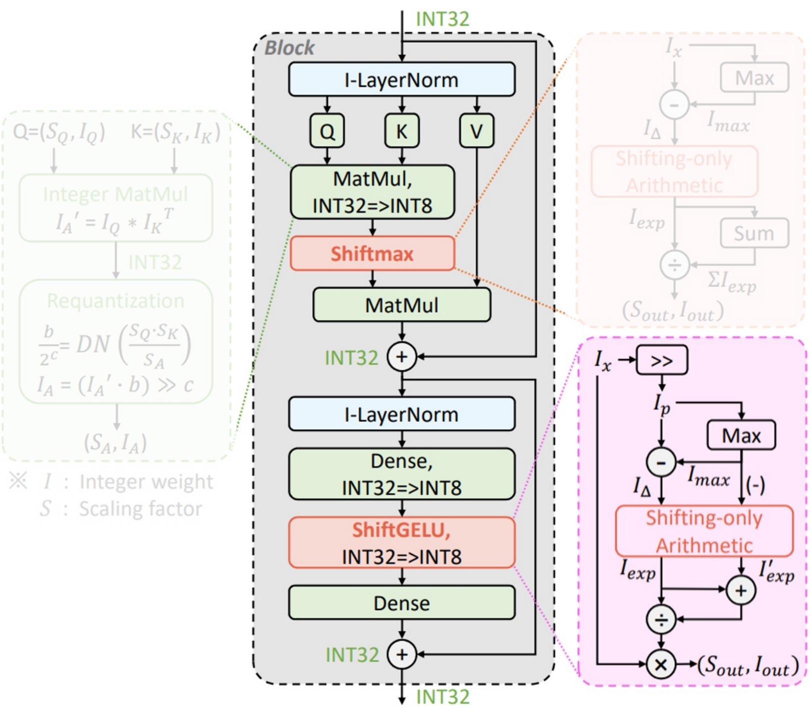
Shiftmax function

- Line1 : prevent overflow
- Line2 : output to use Shiftmax function
  Applying the shift to the e to output the transformed input and the scale
- Line3 : IntDiv function
  Output : $\frac{e^{S_{\Delta_i} \cdot I_{\Delta_i}}}{\sum_j e^{S_{\Delta_j} \cdot I_{\Delta_j}}} \Rightarrow \frac{S_{\Delta_j} \cdot I_{\Delta_j}}{S_{\Delta_j} \cdot \sum_j I_{\Delta_j}}$

I-ViT

- Method
  - Integer-only GELU: ShiftGELU
    - GELU is the non-linear activation function

※ $GELU(x) = x \cdot \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt \approx x \cdot \sigma(1.702x)$



Pseudo code

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 ShiftGELU(I_{in}, S_{in}, k_{out}):

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

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

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

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

$(I_{div}, S_{div}) \leftarrow$ 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}); ▷ $I_{out} \cdot S_{out} \approx \mathrm{GELU}(I_{in} \cdot S_{in})$

End Function

I-ViT

- Method
  - Integer-only GELU: ShiftGELU

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 `ShiftGELU`$(I_{in}, S_{in}, k_{out})$**:**

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

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

$(I_{exp}, S_{exp}) \leftarrow$ `ShiftExp`(I_Δ, S_{in});

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

$(I_{div}, S_{div}) \leftarrow$ `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}); ▷ $I_{out} \cdot S_{out} \approx \mathrm{GELU}(I_{in} \cdot S_{in})$

End Function

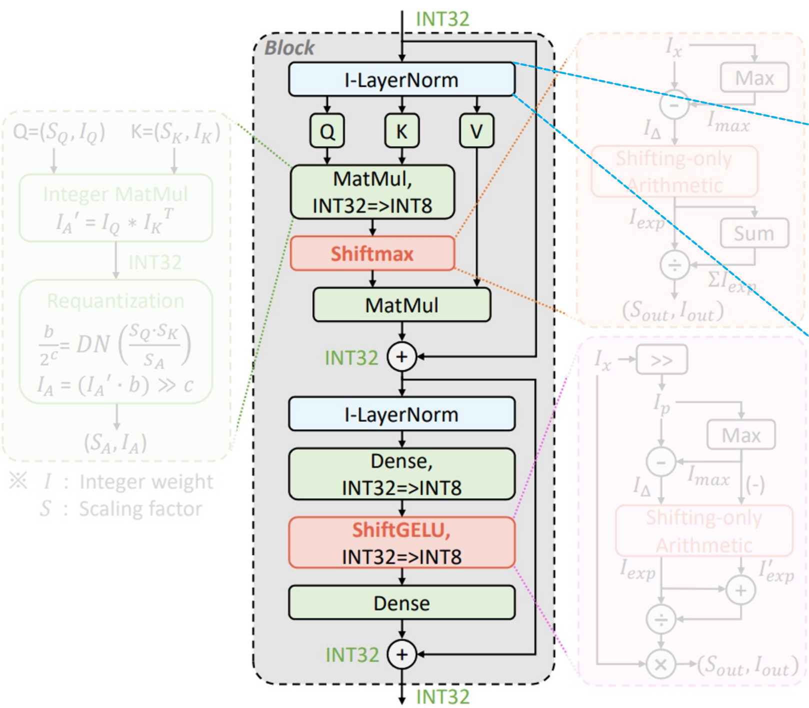
$$GELU(x) = x \cdot \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt \approx x \cdot \sigma(1.702x)$$

ShiftGELU function

- Line1 : To use shifter(approximation)
  $x \cdot \sigma(1.702x) = S_x \cdot I_x \cdot \sigma(S_x \cdot 1.702 I_x)$
  $\therefore 1.702 I_x = (1.1011)_b$
- Line3 : Calculate the numerator
- Line4 : integer approximation of GELU
- Line5 : IntDiv function
  $$\therefore \sigma(S_x \cdot I_P) = \frac{1}{1 + e^{-S_x \cdot I_P}} = \frac{e^{S_x \cdot I_P}}{e^{S_x \cdot I_P} + 1} = \frac{e^{S_x \cdot (I_P - I_{max})}}{e^{e^{S_x \cdot (I_P - I_{max})}} + e^{S_x \cdot (-I_{max})}}$$
- Line6 : Requantization

I-ViT

- Method
  - Integer-only LayerNorm: I-LayerNorm
    - LayerNorm needs to dynamically compute statistics(mean, std)
    - Due to the square root arithmetic, using bit-shifting



- $\text{LayerNorm(x)} = \frac{x - Mean(x)}{\sqrt{Var(x)}} \cdot \gamma + \beta$
- $I_{i+1} = (I_i + \lfloor \frac{Var(I_x)}{I_i} \rfloor)/2$
  $= \left(I_i + \lfloor \frac{Var(I_x)}{I_i} \rfloor\right) \gg 1$
- Experimentally 10 iterations can achieve convergence

I-ViT

- Experimental results
  - Accuracy and latency results on various model(ViT, DeiT, Swin) on ImageNet dataset

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**

- Top-1 accuracy : comparable or slightly higher
- Latency : 3.72~4.11 X inference speedup

Conclusion

- Conclusion
  - First integer-only quantization for Vision Transformer
  - I-ViT quantized the entire computational graph
    - Dyadic arithmetic pipeline
      - Linear operation (MatMul, Dense layer)
    - Integer-only approximation methods
      - Non-linear operation(Softmax, GELU, LayerNorm)
  - Compared to the FP model, similar or slightly higher accuracy
  - 3.72~4.11 X speedup

- Limitations
  - Factors in accuracy loss using approximate methods

Thank you