PRACTICAL EDGE KERNELS FOR INTEGER-ONLY VISION TRANSFORMERS UNDER POST-TRAINING QUANTIZATION

Zining Zhang, Bingsheng He, Zhenjie Zhang

MLSys 2023

Integer Only Kernels on Transformers

- Transformer models like ViT[1] and DeiT[2] have gained attention in the field of computer vision tasks.
  - Deploying them to edge devices requires low energy-consumption and latency integer kernels
- For Convolutional Neural Networks (CNNs), integer kernels are easy to implement for ReLU and BatchNorm
- For Transformers,
  - Softmax requires integer exp() function
  - GELU requires integer erf() function
  - LayerNorm requires taking mean and standard deviation (sqrt() function) during inference.
- We use polynomials to approximate the calculations (e.g. exp, erf, sqrt) and perform them on the quantized integer domain [3].

$$
\begin{aligned}
P(x) &= b_0 + b_1 x + \cdots + b_n x^n \\
&\approx b_0 + b_1 qS + \cdots + b_n (qS)^n \\
&\approx (\frac{b_0}{b_n S^n} + \frac{b_1}{b_n S^{n-1}} q + \cdots + q^n) b_n S^n \\
&\approx (\lfloor \frac{b_0}{b_n S^n} \rceil + \lfloor \frac{b_1}{b_n S^{n-1}} \rceil q + \cdots + q^n) b_n S^n \\
&= Q(q) \times b_n S^n
\end{aligned}
$$

[1] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[2]Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and J ´egou, H. Training data-efficient image transformers & distillation through attention. In International Con- ference on Machine Learning, pp. 10347–10357. PMLR, 2021

[3] I-bert Kim, S., Gholami, A., Yao, Z., Mahoney, M. W., and Keutzer, K. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pp. 5506– 5518. PMLR, 2021

Challenges on edge devices

- The polynomial approximations normally requires 64 bits integers for calculation[1, 2], while 64 bit operations like vmul, vmulh are seldomly supported on edge devices instruction sets (Neon [3], SVE/SVE2 [4]).

Table 1. Accuracy and Truncation Ratios in I-BERT (Kim et al., 2021) and FQ-ViT (Lin et al., 2022) on DeiT-S (Touvron et al., 2021) when deploying on edge devices. Each row means replacing the corresponding floating point kernels to integer kernels in the quantized model. Trunc. Ratio (%): the number of truncations in the result elements over the total number of elements. Truncations include left bit-shift and multiplications that exceed the range of int32, as well as smaller number dividing by larger number resulting in 0. Acc. (%): The image classification accuracy on ImageNet (Deng et al., 2009) evaluation dataset. In partial quantizations, it is **79.51** originally.

OPERATORS	TRUNC. RATIO % (I-BERT)	ACC. % (I-BERT)	TRUNC. RATIO % (FQ-VIT)	ACC. % (FQ-VIT)
INT. SOFTMAX	47.75	0.100	22.34	1.832
INT. GELU	92.41	0.140	N.A.	N.A.
INT. LAYERNORM	68.70	0.100	68.74	0.100

[1] I-bert Kim, S., Gholami, A., Yao, Z., Mahoney, M. W., and Keutzer, K. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pp. 5506– 5518. PMLR, 2021

[2] fqvit Lin, Y., Zhang, T., Sun, P., Li, Z., and Zhou, S. Fq-vit: Post-training quantization for fully quantized vision trans- former. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, pp. 1173–1179, 2022.

[3] Neon Introducing neon development article. https: //developer.arm.com/documentation/ dht0002/a/Introducing-NEON/ NEON-architecture-overview, 2022

[4] SVE Learn the architecture - introducing sve2. https: //developer.arm.com/documentation/ 102340/0001/Introducing-SVE2, 2022.

Truncation problems when performing int32 polynomials

- Softmax:
  - left shift (exp map decimal to int), and
  - final normalization
- LayerNorm: division in normalization
- GELU: overflow in 3rd order polynomials

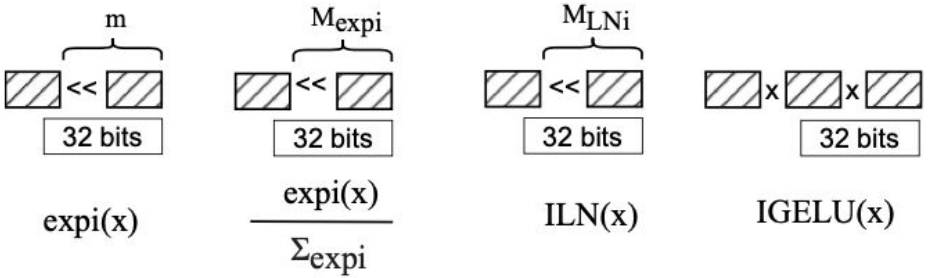


Figure 1. The truncation problems in integer version of Softmax, LayerNorm and GELU kernels.

Softmax

- Dillema : Left shift too little vs. left shift too much
- Solution: Requantize the range of operators to 32-M*, where M* is the desired left-shift bits from an additional calibration or optimization (next slide).

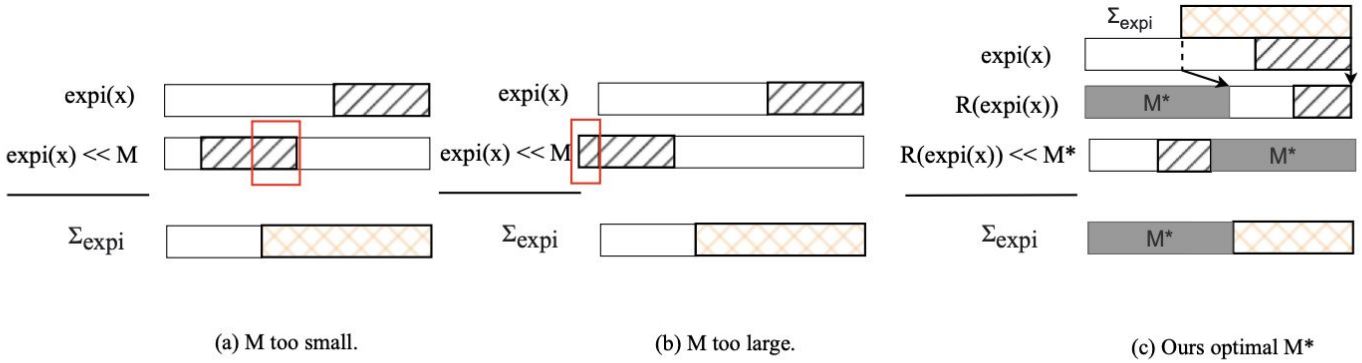
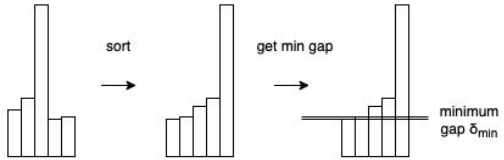


Figure 2. Comparisons among different left shift bit M selections. (a) and (b) show the hard decisions of M. Sketched black boxes represent bits occupied by $\exp i(x)$ and orange ones represent those of $\sum_{\exp i}$. (c) represents the proposed requantization method followed by M^* bit shift, where determination of M^* is discussed in Section 3.1.2.

Choosing the number of bits for left shifts M*

- Output-oriented: use a calibration to collect the floating point gaps
  - The target is to make the unit of the last place (ulp) less than the minimum gap δ_{min}



$$\frac{\exp(x)}{\sum_{\exp}} \leq 1$$
$$\frac{R(\exp i(x))}{\sum_{\exp i}} \leq 1 \qquad (14)$$
$$\frac{R(\exp i(x)) << M}{\sum_{\exp i}} \leq 2^M \qquad M^* \geq \lceil -\log \delta_{min} \rceil$$

- Loss-based: Using the M with lowest mean squared error between floating point softamx and integer softmax.

$$M^* = \text{argmin}_M \left\| \frac{\exp(x)}{\sum_{\exp}} - \lfloor \frac{R(\exp i(x)) << M}{\sum_{\exp i}} \rceil \times \frac{1}{2^M} \right\|_2$$

LayerNorm

- Similar to Softmax, the division may cause truncation problems

$$
\begin{aligned}
\mathrm{LNi}(x) &= \frac{\gamma}{S_o}\frac{(q-zp)S-\mathbb{E}[q-zp]S}{\sqrt{Var[q-zp]S^2+1}} + \lfloor\frac{\beta}{S_o}\rceil + zp_o \\
&\approx \frac{\gamma}{S_o}\bigotimes\lfloor\frac{q-\mathbb{E}[q]}{\sqrt{\mathbb{E}[(q-\mathbb{E}[q])^2]+1}}\rceil + \lfloor\frac{\beta}{S_o}\rceil + zp_o \\
&= \frac{\gamma}{S_o}\bigotimes\lfloor\frac{q-\mathbb{E}[q]}{\sqrt{\mathbb{E}[q^2]-\mathbb{E}[q]^2+1}}\rceil + \lfloor\frac{\beta}{S_o}\rceil + zp_o
\end{aligned}
$$

- Because LayerNorm numerator not involving polynomials, thus it requires less bits. Simply setting M= 32-17 works well in LayerNorm

$$
\mathrm{LNi}(x) = \frac{\gamma/2^M}{S_o}\bigotimes\lfloor\frac{(q-\mathbb{E}[q])<<M}{\sqrt{\mathbb{E}[q^2]-\mathbb{E}[q]^2+1}}\rceil + \lfloor\frac{\beta}{S_o}\rceil + zp_o
$$

Channel-wise [1] Quantized LayerNorm Better Than Tensor-wise?

- Channel-wise quantization in Layernorm means that every channel **c** has different quantization parameters $\mathbf{s}_c$ and $\mathbf{z}_c$, i.e., different quantization domains,
- Tensor-wise quantization, uses the same **s** and **z** for all channels
- Some work [1] report a better accuracy whenusing 8 bit channel-wise quantization in LayerNorm
- However, LayerNorm calculates mean and standard deviation across all channels, this needs to bring all the channels into the same quantization domain. Thus channel-wise quantization in LayerNorm is not meaningful, and introduces additional domain alignment costs.

[1] qvit Lin, Y., Zhang, T., Sun, P., Li, Z., and Zhou, S. Fq-vit: Post-training quantization for fully quantized vision trans- former. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, pp. 1173–1179, 2022.

GELU

- Since GELU is more complicated than Softmax, it requires 3rd order polynomials to be accurate.
- This results in overflow during multiplications.
- By requantizing the #bits of inputs to lower than 10, GELU avoids overflow.

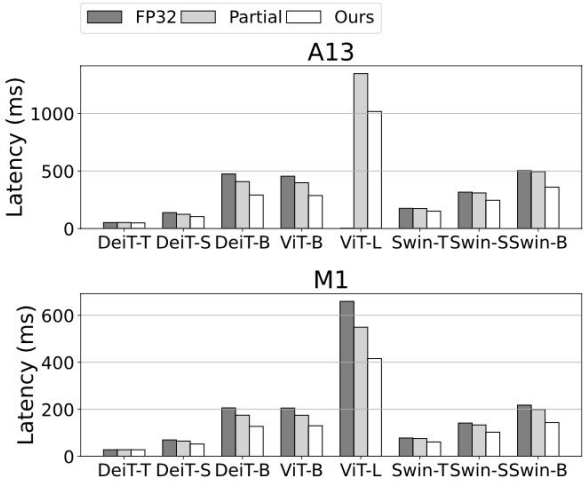
$$
\begin{aligned}
&\lfloor \frac{b_0}{b_2 S^2} \rceil q + \lfloor \frac{b_1}{b_2 S} \rceil q^2 + q^3 \\
\leq & 2^{2b} + 2^{3b} + 2^{3b} \\
\leq & 2^{3b+1} \times (1 + 2^{-b-1}) \\
\leq & 2^{3b+1+\log(1+2^{-b-1})}
\end{aligned}
$$

Experiment results

- When testing on ImageNet, our method consistently outperform other integer quantization methods.

METHOD	DEIT-T	DEIT-S	DEIT-B	VIT-B	VIT-L	SWIN-T	SWIN-S	SWIN-B
FULL FP32	72.13	79.83	81.80	84.54	85.83	81.38	83.23	83.60
PARTIAL FP32 (UINT16 LN)	71.87	79.51	81.49	83.67	85.46	81.04	83.14	83.50
PARTIAL FP32 (UINT8 LN)	70.93	74.88	77.52	28.51	3.60	64.38	74.37	25.58
PTQ FOR VIT	-	77.47	80.48	-	-	-	-	-
FQ-VIT+IGELU(INT64)	70.42	77.88	80.36	81.20	84.21	79.71	82.14	82.33
OURS (MSE)	**71.08**	**78.49**	**80.74**	81.69	84.47	80.03	**82.29**	**82.67**
OURS (OUTPUT)	71.06	78.47	80.73	**81.81**	**84.84**	**80.07**	82.27	82.65

- While partial integer quantizations bring minor inference latency improvements, our method significantly reduces the latency.



Ablation study

- Our integer-only kernels have 3~5 times speed up compared with full-precision kernels when tested on two ARM CPUs A13 and M1

Table 3. Latency comparison on **A13**. Tensor shapes: softmax $(12, 197, 197)$, GELU $(197, 3072)$, layer normalization $(197, 768)$. B means batch size.

OPERATORS	FP32 (MS)	OURS (MS)	SPEEDUP
ISOFTMAX (B=1)	1.90	0.54	3.52×
IGELU (B=1)	6.22	1.42	4.38×
ILN (B=1)	0.50	0.09	5.56×
ISOFTMAX (B=16)	34.77	11.04	3.15×
IGELU (B=16)	111.14	24.17	4.60×
ILN (B=16)	6.95	1.42	4.89×

Table 4. Latency comparison on **M1**. Tensor shapes: softmax (12, 197, 197), GELU (197, 3072), layer normalization (197, 768). B means batch size.

OPERATORS	FP32 (MS)	OURS (MS)	SPEEDUP
ISOFTMAX (B=1)	1.02	0.23	4.43×
IGELU (B=1)	2.71	0.87	3.11×
ILN (B=1)	0.33	0.07	4.71×
ISOFTMAX (B=16)	17.02	3.85	4.42×
IGELU (B=16)	49.18	15.50	3.17×
ILN (B=16)	4.01	0.82	4.89×

Thanks!