I-BERT: Integer-only BERT Quantization

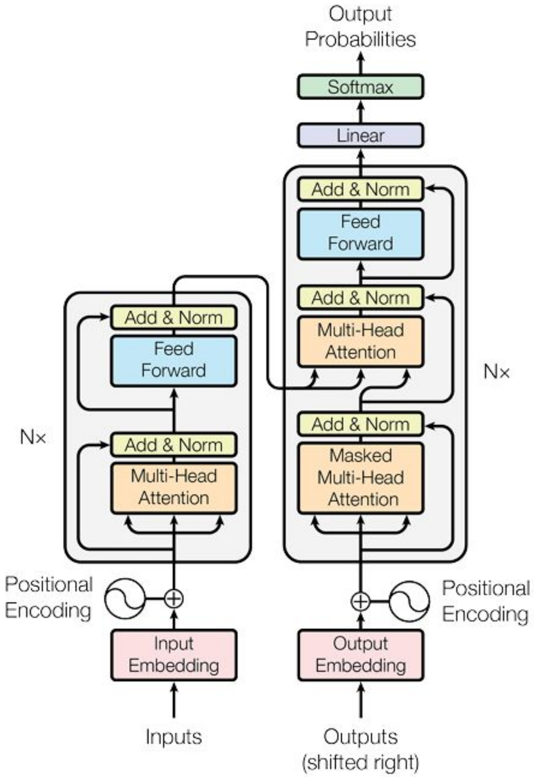
Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, Kurt Keutzer (UC Berkeley)
ICML 2021

DL Compiler Study (2021-11-11)

Summary

- I-BERT, a novel integer-only quantization scheme for Transformers, where the entire inference is performed with **pure integer arithmetic**.
- Key elements of I-BERT are **approximation methods for nonlinear operations** such as GELU, Softmax, and LayerNorm, which enable their approximation with integer computation.
- Empirically evaluated **I-BERT on RoBERTa-Base/Large models**, where our quantization method improves the average GLUE score by 0.3/0.5 points as compared to baseline.
- Furthermore, we directly deployed the quantized models and measured the end-to-end inference latency, showing that I-BERT can achieve up to 4.00× speedup on a Tesla T4 GPU as compared to floating point baseline.

Problems



- Pre-trained Transformer models are generally orders of magnitude larger than prior models
  - Huge number of parameters
  -
- Resources (energy, memory footprint, compute)
  - real-time inference
  - edge devices

Challenges

- Quantization
  - compresses NN models into smaller size by representing parameters and/or activations with low bit precision (reducing memory footprint, faster inference time)
  - Drawbacks: simulated quantization
    - (Bhandare et al., 2019; Shen et al., 2020; Zafrir et al., 2019)
    - GELU, Soft-max, Layer Normalization requires FP calcs.
    - Cannot be deployed to neural accelerators or edge processors
- Integer-only inference
  - Simple CNN layers, Batch-Norm, ReLU -> all linear, piece-wise linear operators
  - Transformer architecture: GELU, Softmax, LayerNorm -> non-linear

Solution

- New kernel for **GELU and Softmax** (second-order polynomials)
  - maximum error of 1.8×10−2 for GELU, and 1.9×10−3 for Softmax.
- For **LayerNorm**, integer-only computation by a known algorithm for integer calculation of square root (Crandall & Pomerance, 2006).
- New design for integer-only quantization for Transormer based models.
  - Processed Embedding and matrix multiplication (MatMul) with INT8 multiplication and INT32 accumulation.
  - The following non-linear operations (GELU, Softmax, and LayerNorm) are then calculated on the INT32 accumulated result and then requantized back to INT8.
  - Represent parameters and activations in the entire computational graph with integers, and never cast them into floating point.
- Evaluation
  - Apply to RoBERTa-Base/Large, and we evaluate their accuracy on the GLUE (Wang et al., 2018) downstream tasks.
  - 0.3 and 0.5 on the GLUE downstream tasks for RoBERTa-Base and RoBERTaLarge, respectively.
- INT8 inference achieves up to 4× speedup as compared to FP32 inference.

Integer-only GELU (1) - Previous approaches

- GELU (Hendrycks & Gimpel,2016)

$$\mathrm{GELU}(x) := x \cdot \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right],$$

$$\text{where } \mathrm{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)dt.$$

$$\mathrm{GELU}(x) \approx x\sigma(1.702x),$$

Sigmoid is another non-linear func.

- h-GELU (Howardetal.,2019)

$$\text{h-GELU}(x) := x\frac{\mathrm{ReLU6}(1.702x+3)}{6} \approx \mathrm{GELU}(x).$$

huge accuracy drop

Integer-only GELU (2) - Proposed solution

- Solution: use polynomials to approximate GELU by solving optimization problem
  - only optimize L(x) in a limited range since erf approaches to 1(−1) for large values of x.

$$\min_{a,b,c} \frac{1}{2}\left\| \mathrm{GELU}(x) - x \cdot \frac{1}{2}\left[1 + \mathrm{L}(\frac{x}{\sqrt{2}})\right]\right\|_2^2,$$

$$\text{s.t.} \quad L(x) = a(x+b)^2 + c,$$

- After finding the best interpolating points, i.e., $L(x) = \sum_{i=0}^{n} f_i l_i(x) \text{ where } l_i(x) = \prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j}.$

$$\text{i-GELU}(x) := x \cdot \frac{1}{2}\left[1 + \mathrm{L}(\frac{x}{\sqrt{2}})\right] \quad \mathrm{L}(x) = \mathrm{sgn}(x)\left[a(\mathrm{clip}(|x|, \max = -b) + b)^2 + 1\right],$$

Integer-only GELU (3)

Algorithm 2 Integer-only GELU

Input: q, S: quantized input and scaling factor
Output: q_{out}, S_{out}: quantized output and scaling factor

function I-ERF(q, S) $\triangleright$ $qS = x$
  $a, b, c \leftarrow -0.2888, -1.769, 1$
  $q_{\text{sgn}}, q \leftarrow \text{sgn}(q), \text{clip}(|q|, \max = -b/S)$
  $q_L, S_L \leftarrow$ I-POLY(q, S) with a, b, c $\triangleright$ Eq. 8
  $q_{out}, S_{out} \leftarrow q_{\text{sgn}} q_L, S_L$
  return q_{out}, S_{out} $\triangleright$ $q_{out} S_{out} \approx \text{erf}(x)$
end function

function I-GELU(q, S) $\triangleright$ $qS = x$
  $q_{\text{erf}}, S_{\text{erf}} \leftarrow$ I-ERF$(q, S/\sqrt{2})$
  $q_1 \leftarrow \lfloor 1/S_{\text{erf}} \rfloor$
  $q_{out}, S_{out} \leftarrow q(q_{\text{erf}} + q_1), S S_{\text{erf}}/2$
  return q_{out}, S_{out} $\triangleright$ $q_{out} S_{out} \approx \text{GELU}(x)$
end function

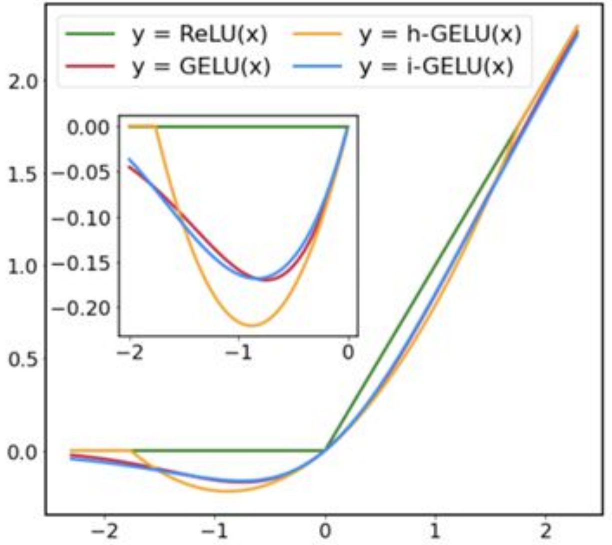


Table 1. Comparison of different approximation methods for GELU. The second column (Int-only) indicates whether each approximation method can be computed with integer-only arithmetic. As metrics for approximation error, we report L^2 and L^∞ distance from GELU across the range of [-4, 4].

	Int-only	L^2 dist	L^∞ dist
$x\sigma(1.702x)$	✗	0.012	0.020
h-GELU	✓	0.031	0.068
i-GELU (Ours)	✓	**0.0082**	**0.018**

Integer-only Softmax (1)

Softmax normalized an input vector and maps it to a probability distribution.

$$\mathrm{Softmax}(\mathbf{x})_i := \frac{\exp x_i}{\sum_{j=1}^{k} \exp x_j}, \quad \text{where } \mathbf{x} = [x_1, \ldots, x_k].$$

Appx the Softmax layer with integer arithmetic is challenging: exp func.

Solution

$$\mathrm{Softmax}(\mathbf{x})_i = \frac{\exp(x_i - x_{\max})}{\sum_{j=1}^{k} \exp(x_j - x_{\max})},$$

- Limiting approx. range: subs max value
- Decompose non-positive value: $\exp(\tilde{x}) = 2^{-z}\exp(p) = \exp(p) >> z,$ $\quad \tilde{x} = (-\ln 2)z + p,$
  - Already used in Itanium 2 machine in 2004 (but with LUT)
- Find coeff of polynomial $\quad L(p) = 0.3585(p + 1.353)^2 + 0.344 \approx \exp(p).$
- Results: (largest gap 1.9e-3)

$$\text{i-exp}(\tilde{x}) := L(p) >> z$$

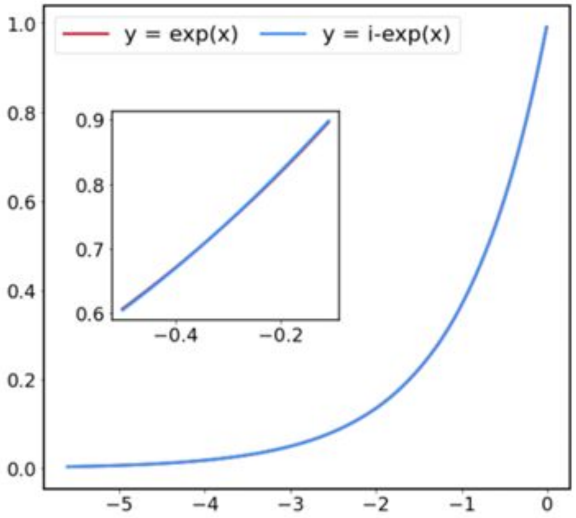
Integer-only Softmax (2)

Algorithm 3 Integer-only Exponential and Softmax

Input: q, S: quantized input and scaling factor
Output: q_{out}, S_{out}: quantized output and scaling factor

function I-EXP(q, S) ▷ $qS = x$
  $a, b, c \leftarrow 0.3585, 1.353, 0.344$
  $q_{\ln 2} \leftarrow \lfloor \ln 2/S \rfloor$
  $z \leftarrow \lfloor -q/q_{\ln 2} \rfloor$
  $q_p \leftarrow q + zq_{\ln 2}$ ▷ $q_p S = p$
  $q_L, S_L \leftarrow$ I-POLY(q_p, S) with a, b, c ▷ Eq. 13
  $q_{out}, S_{out} \leftarrow q_L >> z, S_L$
  return q_{out}, S_{out} ▷ $q_{out}S_{out} \approx \exp(x)$
end function

function I-SOFTMAX(q, S) ▷ $qS = x$
  $\tilde{q} \leftarrow q - \max(q)$
  $q_{\exp}, S_{\exp} \leftarrow$ I-EXP$(\tilde{q}, S)$
  $q_{out}, S_{out} \leftarrow q_{\exp}/\mathrm{sum}(q_{\exp}), S_{\exp}$
  return q_{out}, S_{out} ▷ $q_{out}S_{out} \approx \mathrm{Softmax}(x)$
end function



Integer-only LayerNorm

LayerNorm: for normalizing the input activation across the channel dimension

$$\tilde{x} = \frac{x - \mu}{\sigma} \quad \text{where} \quad \mu = \frac{1}{C}\sum_{i=1}^{C} x_i \quad \text{and} \quad \sigma = \sqrt{\frac{1}{C}\sum_{i=1}^{C}(x_i - \mu)^2}.$$

For NLP, mean and std are calculated dynamically during runtime

std dev requires the square-root function.

Alg4

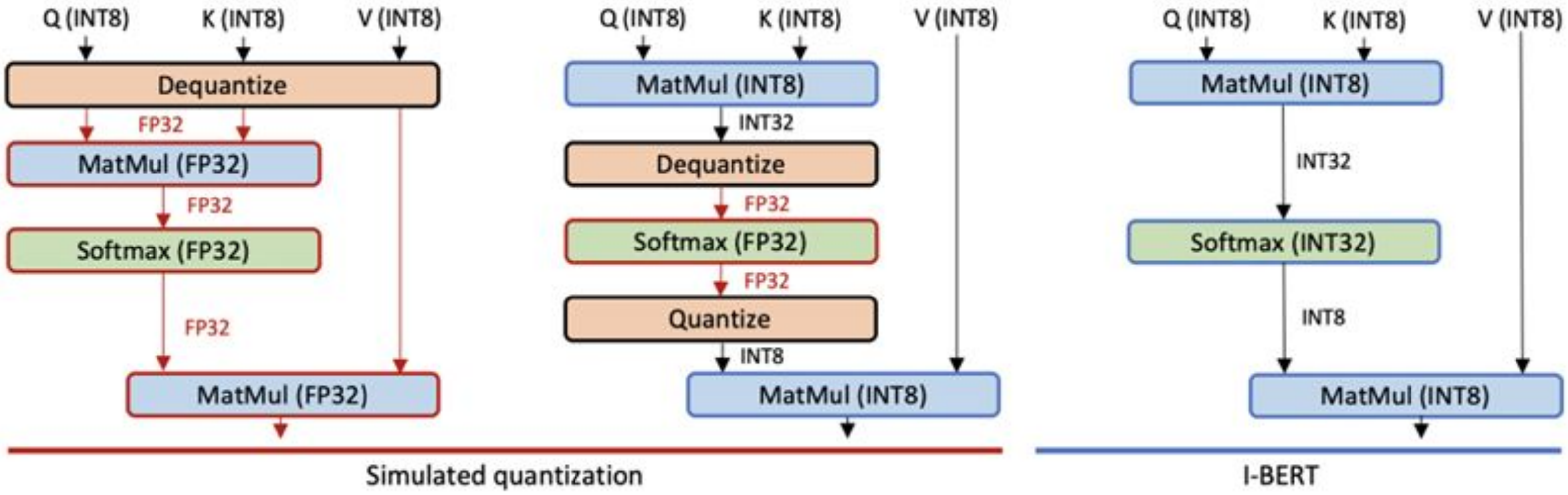
Lightweight, converges at most 4 steps

Algorithm 4 Integer-only Square Root

Input: n: input integer
Output: integer square root of n, i.e., $\lfloor\sqrt{n}\rfloor$

function I-SQRT(n)
  if $n = 0$ **then return** 0
  Intialize x_0 to $2^{\lceil Bits(n)/2 \rceil}$ and i to 0
  repeat
    $x_{i+1} \leftarrow \lfloor (x_i + \lfloor n/x_i \rfloor)/2 \rfloor$
    if $x_{i+1} \geq x_i$ **then return** x_i
    else $i \leftarrow i + 1$
end function

New design for integer-only quantization for Transormer



Results

Integer-only quantization result for Roberta-Base and Robert-Large

(a) RoBERTa-Base

	Precision	Int-only	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
Baseline	FP32	✗	**87.8**	**87.4**	**90.4**	**92.8**	94.6	61.2	**91.1**	90.9	78.0	86.0
I-BERT	INT8	✓	87.5	**87.4**	90.2	**92.8**	**95.2**	**62.5**	90.8	**91.1**	**79.4**	**86.3**
Diff			-0.3	0.0	-0.2	0.0	+0.6	+1.3	-0.3	+0.2	+1.4	+0.3

(b) RoBERTa-Large

	Precision	Int-only	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
Baseline	FP32	✗	90.0	89.9	92.8	94.1	96.3	68.0	**92.2**	91.8	86.3	89.0
I-BERT	INT8	✓	**90.4**	**90.3**	**93.0**	**94.5**	**96.4**	**69.0**	**92.2**	**93.0**	**87.0**	**89.5**
Diff			+0.4	+0.4	+0.2	+0.4	+0.1	+1.0	0.0	+1.2	+0.7	+0.5

Resutls

Inference latency speedup of INT8 inference with respect to FP32 inference

- BERT-base, BERT-Large

SL	128				256				Avg.
BS	1	2	4	8	1	2	4	8	
Base	2.42	3.36	3.39	3.31	3.11	2.96	2.94	3.15	3.08
Large	3.20	4.00	3.98	3.81	3.19	3.51	3.37	3.40	3.56

Results

Accuracy of models that use GELU, h-GELU and i-GELU for GELU computation

	Int-only	QNLI	SST-2	MRPC	RTE	Avg.
GELU	✗	94.4	96.3	92.6	85.9	92.3
h-GELU	✓	94.3	96.0	92.8	84.8	92.0
i-GELU	✓	**94.5**	**96.4**	**93.0**	**87.0**	**92.7**

Links

Paper: https://arxiv.org/pdf/2101.01321

Code: https://github.com/kssteven418/I-BERT