Power-of-Two (PoT) Weights in Large Language Models (LLMs)

Mahmoud Elgenedy
Computer Science, Stanford
melgened@stanford.edu

Motivation and Proposed Solution

Problem statement:

- In Large Language Models (LLMs), the number of model parameters has grown exponentially in the past few years, for example, from 1.5 billion parameters in GPT2 to 175 billion in GPT3.
- This raises a significant challenge for implementation, especially for Edge devices where memory and processing power are very limited.

Proposed Solution:

- We investigate reducing LLM complexity with special type of **quantization**, **power of two** (**PoT**), for linear layers **weights** for both multi-layer perceptron **MLP** and **multi-head attention**.
- PoT not only provides **memory reduction** but more importantly provides significant computational reduction through converting **multiplication** to **bit-shifting**.
- PoT is investigated in few studies for CNN and general DNN [1]. Not avaialble clear studies for LLMs.

Summary of Results

For **124-M GPT-2** model, the PoT quantization results are shown to be very promising,

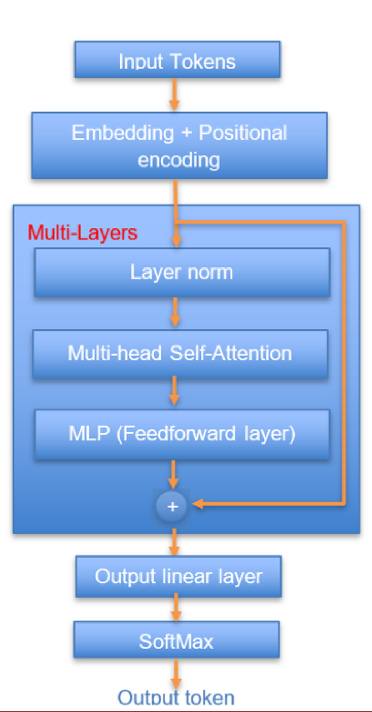
- Cross entropy loss degradation $\approx$[1.3-0.88] with number of bits range [4-6] to represent power levels.
- This results in **memory** reduction with up to a **factor of 8** and **processing power** reduction with a **factor of $\approx$5**.

LLM Model and Datasets

NanoGPT [2] is a simple repository for training and tuning Generative Pre-trained Transformer GPT. The design follows GPT-2 model and can reproduce GPT-2 on OpenWebText dataset [3].

1. **Shakespeare dataset**
   A small dataset ($\approx$ 1 MB) consists of texts from William Shakespeare's plays. Our preliminary trials considering a character-level small LLM model trained only on Shakespeare dataset. We partitioned the data as 80%-10%-10% for training-val-testing. Character-level Tokenizer, from nano-GPT, is used to encode/decode dataset.
2. **OpenWebText dataset**
   An open-source recreation of OpenAI's WebText dataset, which mainly used to train GPT-2. It is relatively large ($\approx$ 54 GB) dataset consists of news articles, blog posts and stories. We evaluated the performance of quantization on OpenWebText dataset using a pretrained GPT-2 model [4]. The evaluation dataset is 0.05% of the total dataset. A Tokenizer from "tiktoken" by OpenAI is used.

Figure 1. LLM Model illustration and parameters



Parameter	Character-level	GPT- 2
Block Size (Context size)	64	1024
Number of Layers	4	12
Number of heads	4	12
Embedding Dimension	128	768
Dropout	0.2	0.2
Batch size	12	12
Vocab size	65	50257
Headen Layer dimension	4x128	4x768

Model Parameters

Method

Quantization: numbers can be represented in lower precision to help reduce both memory and processing requirements [5].

- Different types of quantization including symmetric, asymmetric quantization and Power-of-Two (PoT) (Table 1).
- Restricting weights to Power-of-Two (PoT) can significantly reduce processing power as multiplication is converted into bit-shifting.
- PyTorch provides different quantization frameworks[6], including Eager Mode, FX Graph Mode, and PyTorch 2 Export Mode.
- Both symmetric and Asymmetric are supported in PyTorch frameworks. However, PoT is not supported, and part of this work is to add it to PyTorch framework.
- We choose to use most recent **PyTorch 2 Export Mode** [7] as it is more flexible and easily scalable.

Table 1. Types of Quantization

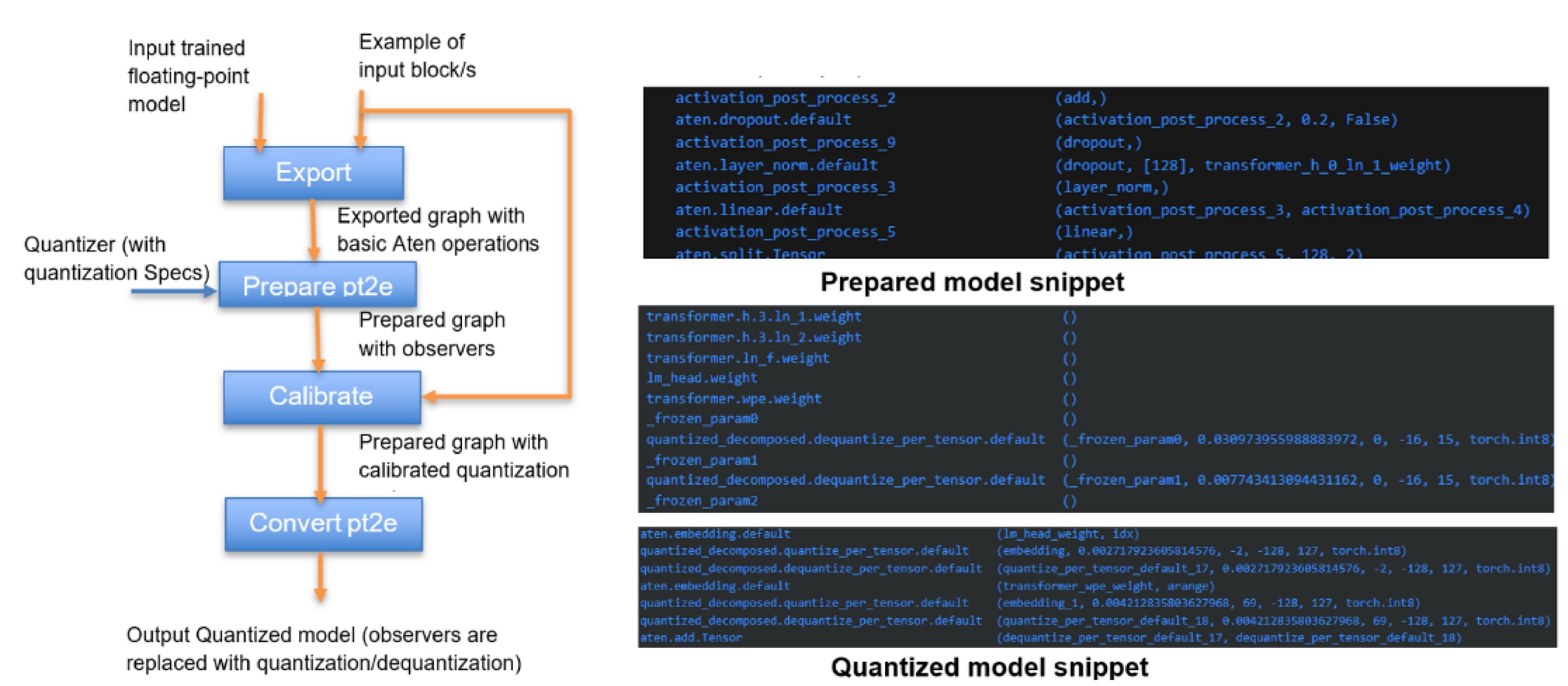
Quantization type	Conversion equation (input x, and output y)	Use case
Symmetric	$y = round(x/scale)$	Weights
Asymmetric (Affine)	$y = round(x/scale) + zp$	Activations
Power of Two (PoT)	$y = 2^{clip(round(\log_2(x/scale)))}$	Suggested in this proposal for Weights in MLP and Transformer

Quantized Model

Pytorch Steps to obtain quantized model[a], using **Post Training Quantization (PTQ)** approach,

1. **Export:** the trained floating-point model is broken down into a graph basic tensor operations.
2. **Prepare:** observers modules are added between operations (to estimate input/output range).
3. **Calibrate:** run the prepared model with realistic data so that observers can do calibration.
4. **Convert:** each observer is converted into quantize/dequantize operation based on calibration.
5. **PoT:** restrict weights and transformer LUTs to power of two[b].

Figure 2. PyTorch Quantization Framework



[a] This is not fully quantized model but simulates with good accuracy quantization loss
[b] This is additional step not in originally supported in Pytorch but added as part of this work

Performance Metrics

We use **cross-entropy loss** and **perplexity** to measure the performance degradation due to quantization.

To measure cross-entropy, we average over multiple iterations of calling the model (both floating and quantized) with input batches captured from testing set.

The cross-entropy $l_{ce}(t, y)$ with input $k-$dimension logits $t \in \mathbb{R}^k$ and target class y is defined as follows,

$$l_{ce}(t, y) = -\log P(y; t) = -\log \left(\frac{\exp(t_y)}{\sum_s \exp(t_s)} \right)$$

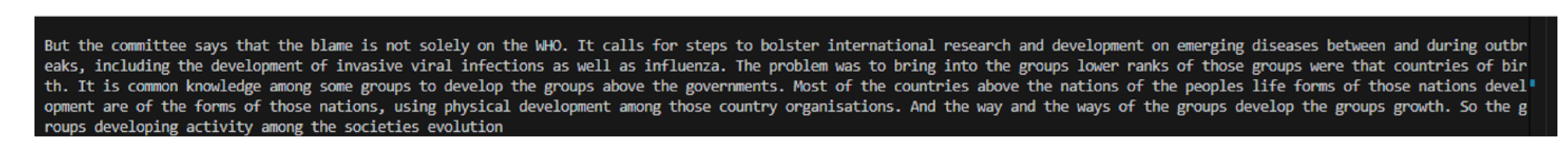
, and perplexity is just the exponential of the cross entropy defined as $\exp(l_{ce})$

Main Results

Table 2. Performance summary of Normal quantization versus PoT

	Normal quantization			Power of Two (PoT)		
Integer range	No. of bits	l_{ce}	Perplexity	No. of bits	l_{ce}	Perplexity
$[-2^5, 2^5]$	6 bits	7.7	2225.86	4 bits	9.563	14158.84
$[-2^7, 2^7]$	8 bits	3.329	27.92	4 bits	4.5	90
$[-2^{15}, 2^{15}]$	16 bits	3.1888	24.25	5 bits	4.3	73.7
$[-2^{20}, 2^{20}]$	32 bits	3.1883	24.24	6 bits	4.23	68.7
$[-2^{30}, 2^{30}]$	32 bits	3.187	24.23	6 bits	4.08	59.14

Figure 3. Output generated text with 4 bits PoT



- The PoT quantized model has very promising performance even at very low number of bits, **4-bits** (15 levels $[2^{-7}, 2^7]$). This results in huge reduction of memory requirements (a factor of **32/4=8**).
- Interestingly, at same number of bits (4-bits), uniform quantization shows a big failure (cross entropy $\approx$ 7.7) and completely off generated text, while PoT shows only $\approx$ 4.5 cross entropy loss, with very good quality output text shown in figure 3. This can be expected as PoT is **non-uniform** (log) quantization which can fit better the data with more levels for smaller values and less levels for outliers.
- Finally, PoT reduces all linear and transformer operations by factor of $\approx$5 since multiplication requires approximately 5 clock cycles while bit shifting is just one clock cycle.

Future work

- Try **Quantization Assisted Training (QAT)** instead of Post Training Quantization (PTQ), as training quantized model can significantly enhance the accuracy of quantized model.
- Try relatively bigger models recently introduced to edge devices, e.g., **Llama** lightweight (1B).

References

[1] D. Przewlocka-Rus, S. S. Sarwar, H. E. Sumbul, Y. Li, and B. De Salvo, "Power-of-two quantization for low bitwidth and hardware compliant neural networks," *arXiv preprint arXiv:2203.05025*, 2022.
[2] Karpathy, "Karpathy-nanogpt: The simplest, fastest repository for training/finetuning medium-sized gpts.."
[3] A. Gokaslan, "Dataset card for "openwebtext"."
[4] OpenAI-community, "Gpt-2 model."
[5] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-power computer vision*, pp. 291–326, Chapman and Hall/CRC, 2022.
[6] "Quantization, quantization - pytorch 2.6 documentation."
[7] "Pytorch 2.0 export post training static quantization."