Quantization Techniques for Efficient Deployment of Large Language Models: A Comprehensive Review

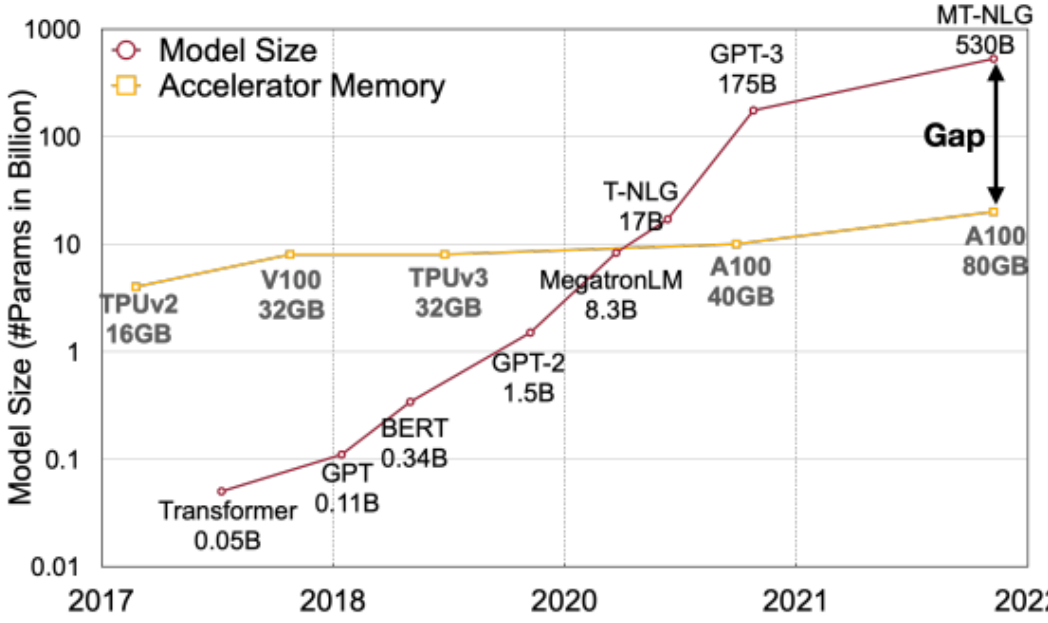
Dwith Chenna
MTS Product Engineer, AI Inference
Advanced Micro Devices (AMD)

Content

- Introduction
- Quantization
- Transformer architecture
- Transformer quantization
- Quantization methods
- Results and analysis
- Conclusion
- References

Introduction

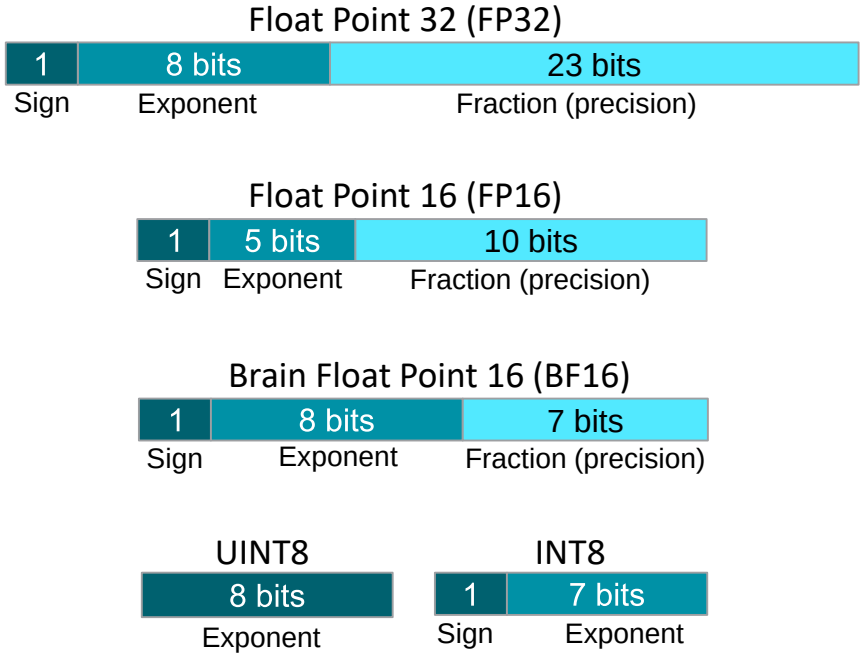
- LLMs are compute and memory intensive
- Quantization reduces model size and increases compute efficiency
- Maintaining accuracy can be challenging



LLM model size vs GPU memory [2]

Quantization

- Quantization is the process of compressing real numbers to lower bit width representation
- Quantization can be broadly classified into two types:
  - Float to float
  - Float to integer
- Key factors:
  - Use case and application
  - Model complexity
  - Hardware compatablity



Quantization

- Float to float quantization relies on simplifying precision and adjusting float-point representation
  - Handling overflow and underflow
  - Mantissa truncation based on the rounding method
- Integer quantization is the process of mapping real numbers, denoted as "r", to quantized integers, represented as "q"
  - Symmetric Quantization

$$q = round(r/S)$$

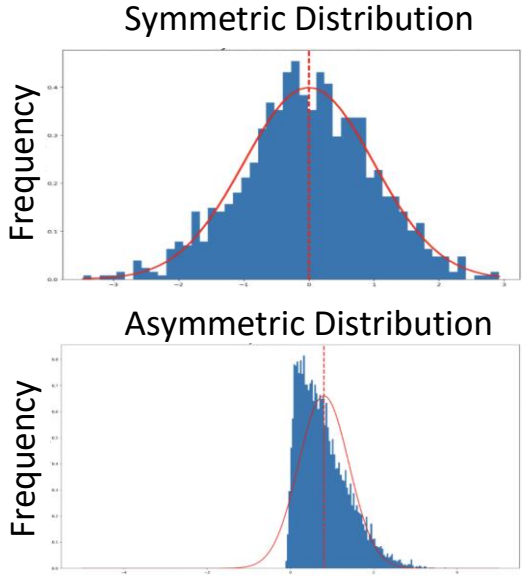
  - Asymmetric Quantization

$$q = round(r/S + Z)$$

where "S" is the scale and "Z" is the zero point

$$S = (r\_max - r\_min) / (q\_max - q\_min)$$

$$Z = round\ (q\_max - r\_max / S)$$



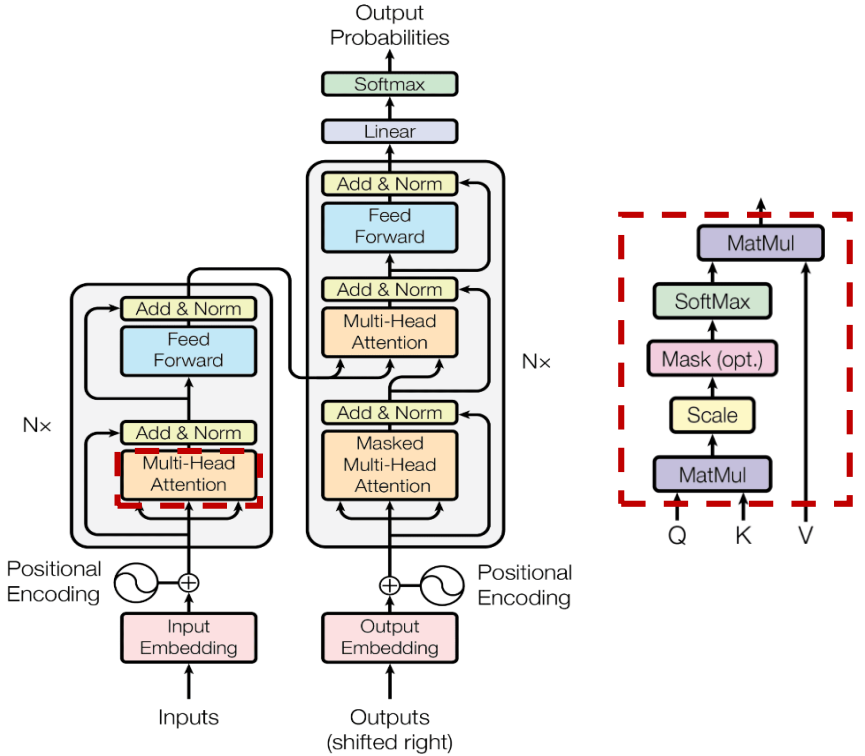
Symmetric and Asymmetric distribution

Quantization Challenges

- Quantization introduces noise in the weights and activations
- Can lead to significant degradation in model accuracy
- Effective quantization requires robust hardware and software support

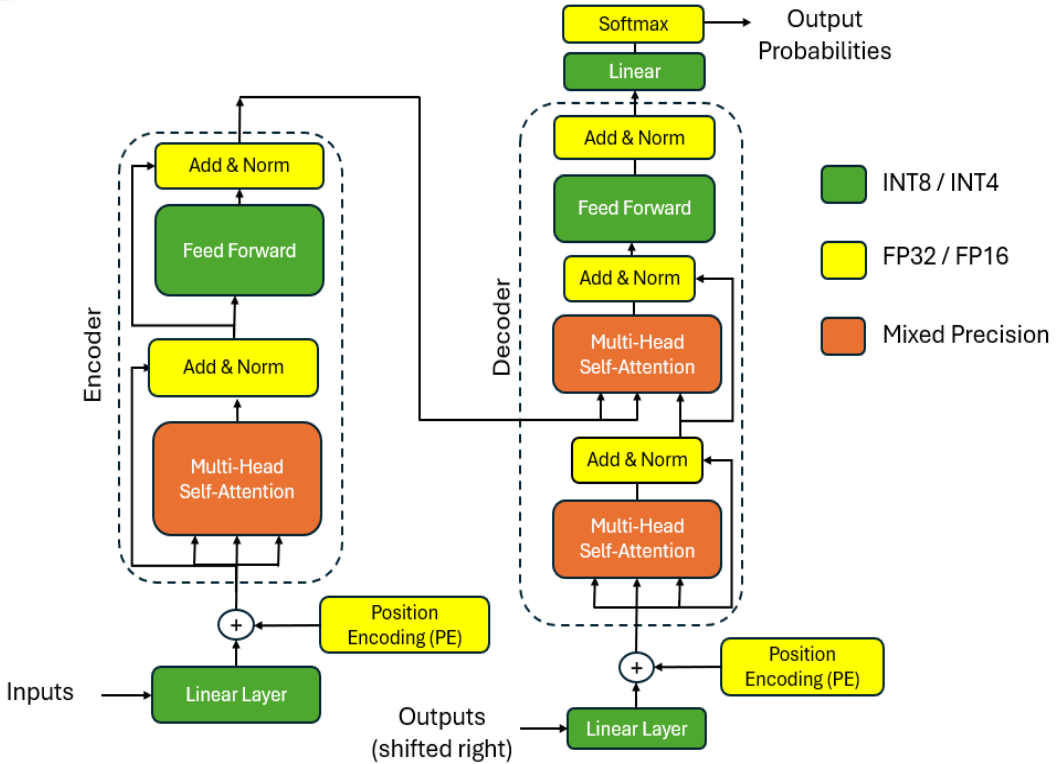
Transformer Architecture

- Compute (FLOPs, MACs) grows exponentially with model and input size
- Compute intensive operations:
  - Matrix Multiplication (MatMul)
  - Fully Connected (FC) / Linear layer
- Memory bound operations:
  - SoftMax
  - Add and Norm



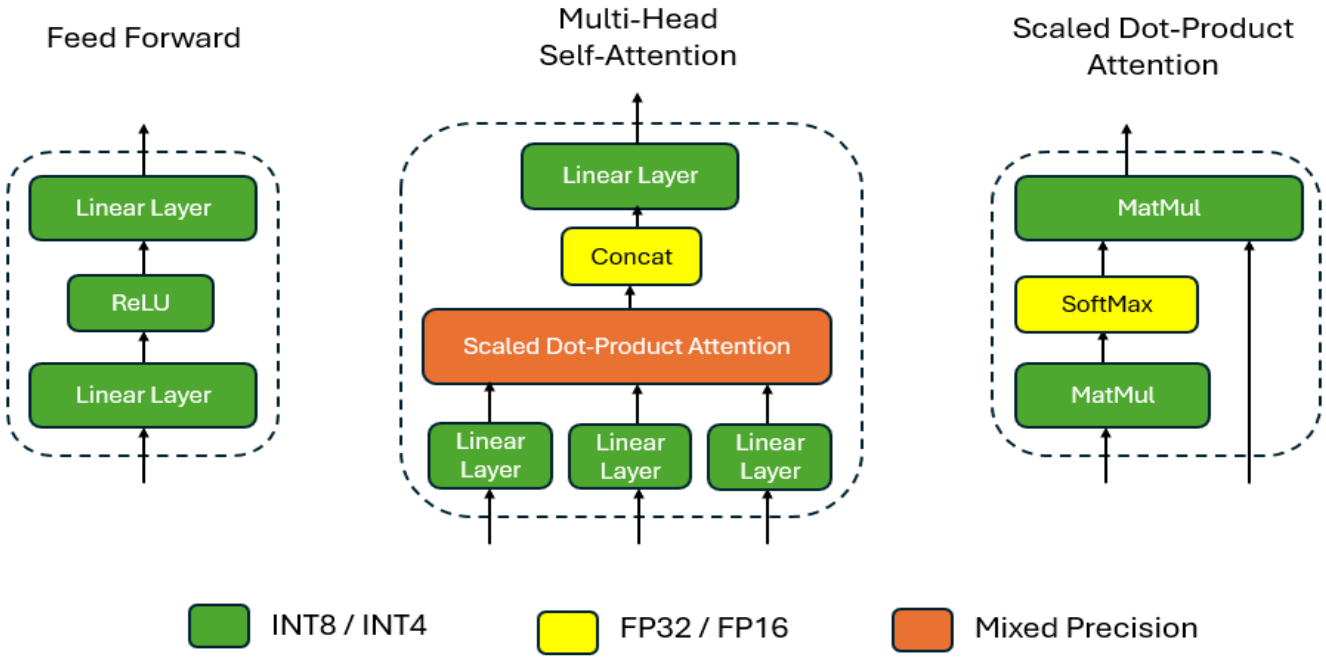
Transformer Quantization

- Quantization precision
  - Different operations need different precision
- Trade-offs in precision
  - Accuracy and performance
- Accuracy:
  - Different operations use different precision
- Performance:
  - Avoid quantization overheads



Quantization precision mapping for Transformers [4]

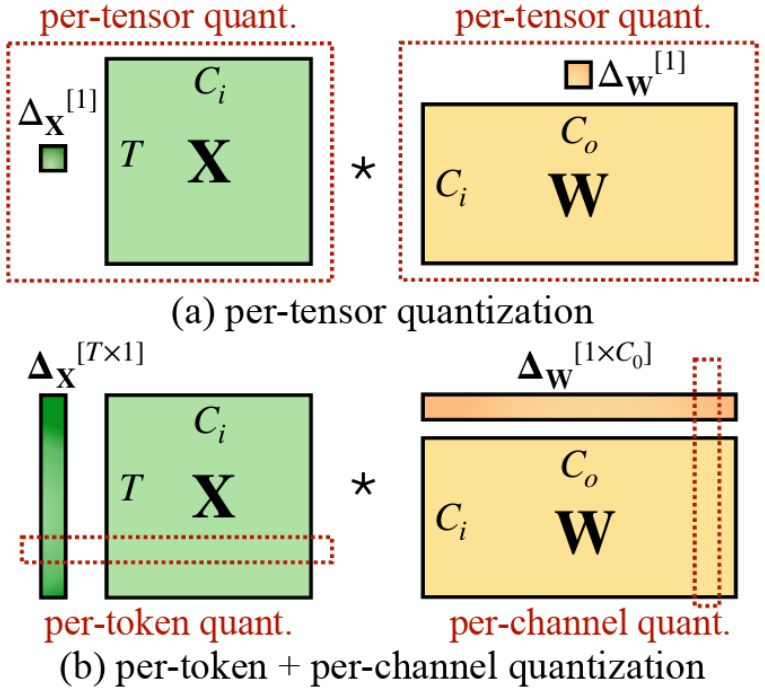
Transformer Quantization



Quantization precision mapping for Transformers [4]

Transformer Quantization

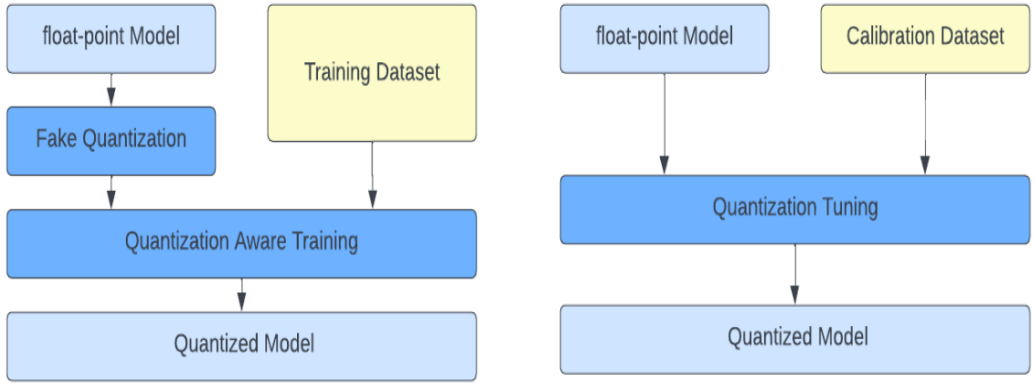
- Quantization granularity
  - Per-tensor
  - Per-group
  - Per-token
  - Per-channel
- Per-group: M rows (for activations) or K columns (for weights) (e.g., K=64) correspond to one quantization coefficient
- Per-token for activation X: each row corresponds to a quantization coefficient
- Per-channel for weight W: each column corresponds to a quantization coefficient



Quantization granularity for Transformers [2]

Quantization Methods

- Model quantization can be divided into:
  - Quantization Aware Training (QAT)
  - Post Training Quantization (PTQ)
- QAT has been proven to work effectively:
  - Fake-quantization nodes
  - Increased computational costs on LLMs
- PTQ techniques for LLM quantization:
  - **GPTQ**
  - **SmoothQuant**
  - **AWQ**



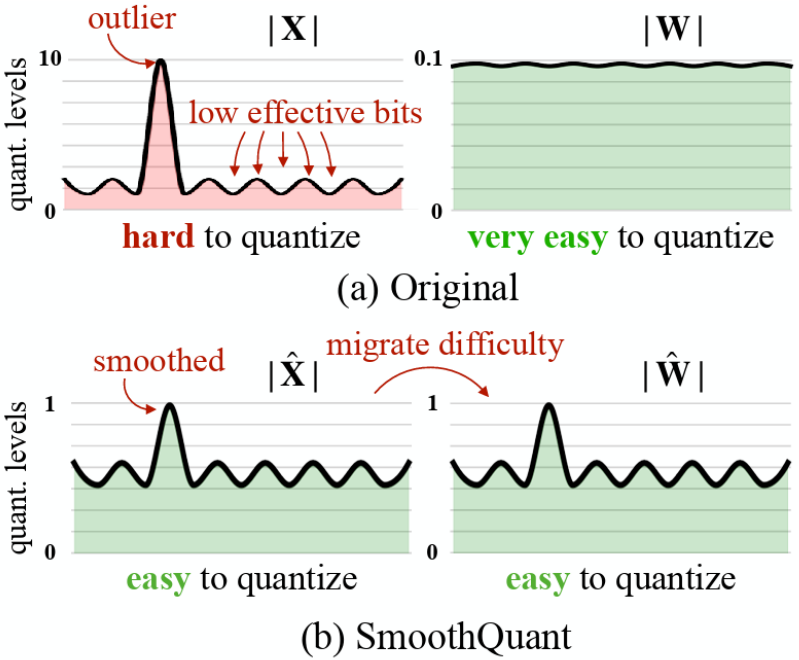
Model quantization i) QAT and ii) PTQ

GPTQ (Gradient Post-Training Quantization)

- Efficient algorithm for layer-wise quantization of large language models (LLMs)
- Algorithm performs the following steps:
  - Quantizes the weights by converting floating-point values into lower-precision integers
  - Calculates the quantization error, the difference between the original and quantized values
  - Updates the weights in current block to account for the error, ensuring better approximation
- Iterative process ensures that the entire weight matrix is adjusted, allowing the model to retain high accuracy despite reduced precision

SmoothQuant

- Observations show that activations are more difficult to quantize due to their wide range
- Significantly limiting the quantization precision
- Weights tend to have a more uniform distribution
- Proposes a scaling method to move the quantization error to weights
- Uses 8 bits for both weight and activation quantization (W8A8)



Migration of scale variance from Activations to Weights [2]

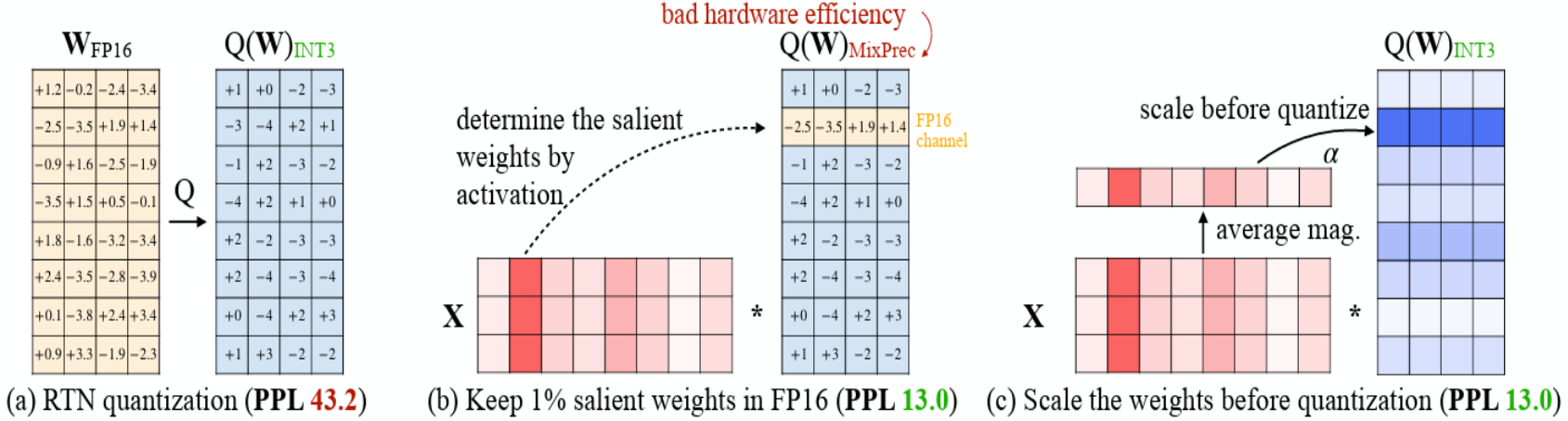
SmoothQuant

- Uses parameter called “migration strength”
  - 0 – indicating all the quantization error is in activation (original)
  - 1 – indicating all the quantization error is moved to weights
  - [0.4 – 0.5] seems to be the sweet spot for minimum quantization error

- This is orthogonal to the quantization scheme, i.e., this technique can be used with different quantization schemes

Activation-aware Weight Quantization (AWQ)

- Uses activations to decide the sensitive weights, even 0.1% to 1% weight contribution leads to improved quantization
- Activation based scaling of weights is more effective than weight magnitude-based scaling

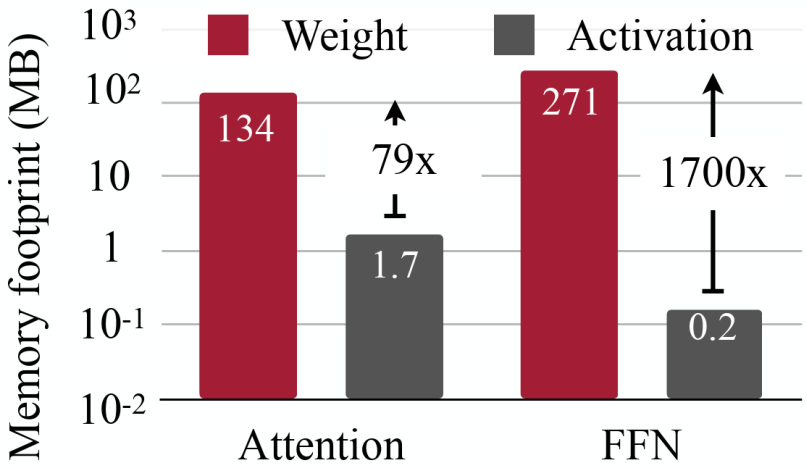


(a) RTN quantization (**PPL 43.2**) (b) Keep 1% salient weights in FP16 (**PPL 13.0**) (c) Scale the weights before quantization (**PPL 13.0**)

Reference [3]

Activation-aware Weight Quantization (AWQ)

- Needs a small calibration dataset to determine the optimal scale
- The method uses hardware friendly approach by using scaling that minimizes quantization error
- Scale values <= 2, show best results or least quantization error
- Uses 4 bits for weights and 16 bits for activations (W4A16)



Memory footprint for Weights and Activation [3]

Results and Analysis

- Perplexity is a commonly used metric for evaluating language models, including large language models (LLMs)
  - A lower perplexity score means the model makes better predictions
- Perplexity is a measure of how well a language model predicts a set of words
- In these tests, WikiText-2 dataset is used, which consists of a diverse collections of Wikipedia articles

GPTQ: Results

- Perplexity measurement on the wikitest-2 database on different sizes of the OPT / Llama-2 models
- Quantization configuration (INT4-g128):
  - INT4 : Weight quantization precision
  - g128: Using 128 group size
- These results are generated using the AutoGPTQ library [1]

PPL	OPT-1.3B	OPT-2.7B	OPT-6.7B	OPT-13B
FP16	14.62	12.47	10.86	10.13
GPTQ (INT4-g128)	16.15	12.84	11.05	10.21

Table 1. Perplexity metrics on WikiText-2 for AutoGPTQ on OPT models [4]

PPL	Llama-2 7B	Llama-2 13B	Llama-2 70B
FP16	5.47	4.88	3.32
GPTQ (INT4-g128)	5.87	4.97	3.52

Table 2. Perplexity metrics on WikiText-2 for AutoGPTQ on Llama-2 models [4]

SmoothQuant: Results

- SmoothQuant [2] shows reliable results with 8-bit quantization

PPL	OPT-1.3B	OPT-2.7B	OPT-6.7B	OPT-13B
FP16	14.62	12.47	10.86	10.13
SmoothQuant (A8W8)	14.82	12.50	10.86	10.14

Table 5. Perplexity metrics for FP16 and SmoothQuant(A8W8) on OPT models [4]

- Quantization configuration:
  - W8: Weight quantization precision
  - A8: Activation quantization precision

PPL	Llama-2 7B	Llama-2 13B	Llama-2 70B
FP16	5.474	4.95	3.32
SmoothQuant (A8W8)	5.515	4.929	3.359

Table 6. Perplexity metrics on WikiText-2 for FP16 and SmoothQuant (A8W8) [4]

AWQ: Results

- These results are generated using the AWQ implementation [3]
- Quantization configuration:
  - INT4: Weight quantization precision
  - g128: Uses 128 group size

PPL	OPT-1.3B	OPT-2.7B	OPT-6.7B	OPT-13B
FP16	14.62	12.47	10.86	10.13
AWQ (INT4-g128)	14.92	12.70	10.92	10.22

Table 3. Perplexity metrics on WikiText-2 for AWQ on OPT models [4]

PPL	Llama-2 7B	Llama-2 13B	Llama-2 70B
FP16	5.47	4.88	3.32
AWQ (INT4-g128)	5.6	4.97	3.41

Table 4. Perplexity metrics on WikiText-2 for AWQ on Llama-2 models [4]

Conclusion

- Quantization emerges as a popular technique, but maintaining model accuracy is challenging
- Mixed precision 4/8/16-bit is critical to maintain accuracy and performance trade-off on LLMs
- Model weights tend to be more uniform allowing lower precision 4/8-bit quantization
- Activations are more challenging to quantize, requires representative calibration dataset
- We explored advanced quantization approaches such as GPTQ, SmoothQuant, and AWQ
- Practical results of applying these quantization techniques on popular LLMs models: OPT / Llama

References

1. GPTQ: https://arxiv.org/abs/2210.17323
2. SmoothQuant: https://arxiv.org/pdf/2211.10438
3. AWQ: https://arxiv.org/abs/2306.00978
4. Results: https://github.com/cyndwith/llm-quantization

LinkedIn: https://www.linkedin.com/in/dwithchenna/

Thank you

Dwith Chenna