Large Language Models (LLMs): Quantization

Dwith Chenna[1] and Rahul Kavi

[1]AMD http://www.amd.com
dwith.chenna@ieee.org

I. INTRODUCTION

Large language models (LLMs) have demonstrated exceptional performance across a range of tasks, but their computational and memory demands are significant. These models, such as GPT-3 [1], which boasts 175 billion parameters, require substantial resources to operate—needing at least 350GB of memory to store and run in FP16. This setup demands multiple high-capacity GPUs, like 8×48GB A6000 or 5×80GB A100, merely for inference. Figure 1, show the comparison of LLM model size and GPU memory. The substantial computation and communication overhead often result in impractical latency for real-world applications.

One effective strategy to mitigate these challenges is quantization, which enhances the efficiency of LLMs by reducing their memory footprint and computational requirements. Quantization achieves this by representing weights and activations with low-bit integers, such as INT8 or INT4, thereby lowering GPU memory consumption and improving throughput, especially in operations like General Matrix Multiply (GEMM) in linear layers and Batch Matrix Multiply (BMM) in attention mechanisms. This approach can significantly decrease the cost of deploying LLMs, making them more feasible for practical applications [3-4]. Activations of LLMs are challenging to quantize due to observed large magnitude outliers, which leads to quantization error and degradation in accuracy [5]. This makes it difficult to have a quantization method that can work across models without significant degradation in model accuracy and maintaining performance.

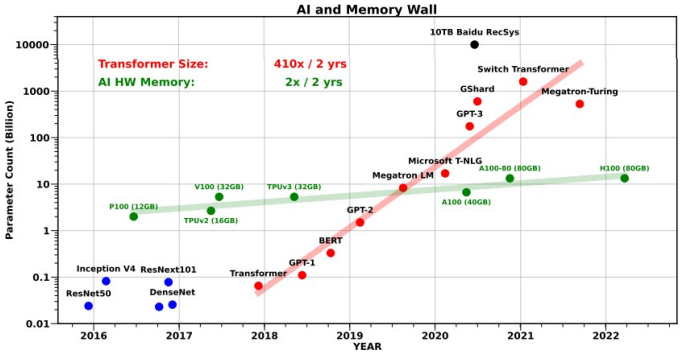

Fig. 1 AI model size over the years along with AI accelerator memory capacity [11]

In this article, we will review some of the popular quantization methods and its impact on accuracy and performance of popular models like OPT, Llama-2/3, understanding how hardware friendly and post training quantization methods for LLMs that can leverage support of low precision INT8/INT4 compute efficiency available in the hardware accelerators. More recently, we have seen the introduction of smaller Small Language Models (SLMs). Unlike LLMs that need hundreds of billions of parameters, these SLMs require a few billion parameters. However, these SLMs are trained on much cleaner data (textbook quality data), and are designed to be more efficient.

II. QUANTIZATION

In this section, we present the mathematical framework for the quantization scheme, which facilitates the efficient execution of integer arithmetic operations on quantized values. The transformation from real numbers r to quantized integers q is defined by equation for Asymmetric quantization, where S and Z, represent the scale and zero-point quantization parameters, respectively. For 8-bit quantization, q is an 8-bit integer, the scale is typically a floating-point value that is represented using a fixed-point format, and the zero-point is of the same type as the quantized value. A key constraint is placed on the zero-point to ensure that real zero values are quantized accurately, without error. The reverse mapping, from quantized values back to real values, is described by equation (2).

Quantization:

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

De-Quantization:

$$r = S\ (q - Z)$$

Symmetric Quantization:

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

Where "S" is the scale and "Z" is the zero points, which are determined from the original float distributions using:

$$S = (r\_max - r\_min) / (q\_max - q\_min)$$
$$Z = \text{round}\ (q\_max - r\_max / S)$$

III. Transformers Architecture

Transformers have emerged as a transformative technology in the realm of natural language processing (NLP), fundamentally altering how machines understand and generate human language. Unlike traditional models that process text sequentially, word by word, Transformers can analyze an entire sentence at once. This parallel processing capability allows them to grasp the nuances of language more effectively, leading to significant improvements in both speed and accuracy. The concept of the Transformer was first introduced in the seminal paper Attention Is All You Need [6]. Originally designed to tackle sequence-to-sequence tasks such as machine translation and text-to-speech, Transformers have since become the cornerstone of many advanced NLP applications.

A. Self-Attention

A key innovation of the Transformer architecture is the self-attention mechanism. This enables the model to assess the significance of each word in a sentence relative to all others, facilitating a deeper contextual understanding. By handling entire sentences simultaneously, Transformers not only expedite processing but also maintain a coherent understanding of context across long distances in text. This capability has revolutionized tasks like machine translation, content generation, and even the creation of human-like text, setting new benchmarks in NLP.

B. Encoder-Decoder Architecture

Transformers are built upon a two-part architecture: the encoder and the decoder. The encoder is responsible for reading and processing the input text, effectively distilling it into a form that the model can comprehend. This process involves breaking down a sentence into its core elements. The decoder, on the other hand, takes this processed information and generates the output sequence, such as translating the sentence into another language. This encoder-decoder interaction is crucial for tasks that require a nuanced understanding of context, like translation.

Within the encoder, multiple layers are employed, each consisting of self-attention mechanisms and feed-forward neural networks. The self-attention mechanism enables the encoder to weigh the importance of other words in the sentence when considering a specific word. This is mathematically facilitated by generating Query (Q), Key (K), and Value (V) vectors, which together allow the model to dynamically interpret the sentence's context. The decoder, starting

C. Positional Encoding

Since Transformers analyze all words in a sentence simultaneously, they require a mechanism to capture the order of words—this is achieved through positional encoding. Each word is assigned a unique positional code that represents its location in the sentence, ensuring the model understands the sequence and flow of language. This is essential for preserving the meaning and structure of sentences.

D. Multi-head Attention

A distinguishing feature of Transformers is the multi-head attention mechanism, which allows the model to focus on different parts of a sentence simultaneously. By applying multiple attention heads, the model can capture various

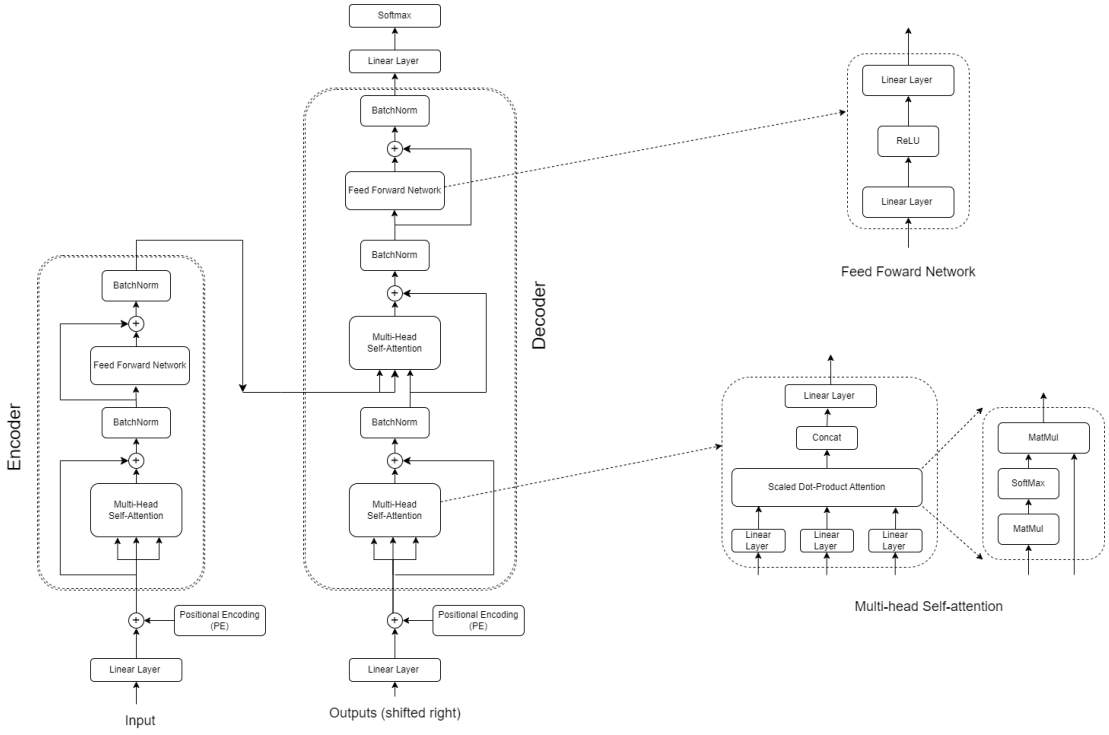

Fig. 2 Transformer architecture (left), Feed Forward Network (right) and Multi-head attention (right)

relationships and dependencies between words, leading to a richer and more nuanced understanding of the text. This parallel processing of attention layers is what gives Transformers their powerful capability to handle complex language tasks with ease.

IV. Transformer Quantization

The quantization methods used of transformer models can be broadly classified as i) Post Training Quantization (PTQ) and ii) Quantization Aware Training (QAT). The quantization parameters need to be adjusted to maintain the accuracy performance after quantization. The process of retraining the model to account for quantization is called Quantization Aware Training (QAT) or without retraining through Post Training Quantization (PTQ). A high-level comparison of two approaches is shown in Figure 3.

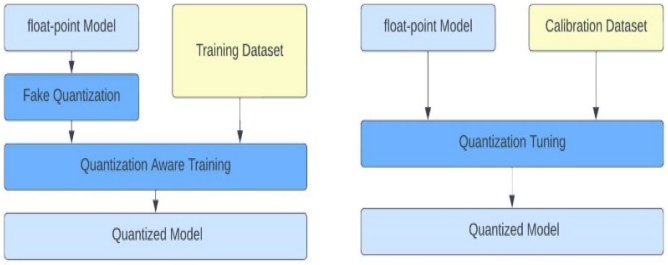

Fig. 3 Model quantization Quantization Aware Training (QAT) and Post Training Quantization (PTQ) [12]

In this article, we will be primarily focusing on PTQ which can be applied to the model trained with QAT. PTQ is the most popular technique because of its low compute requirement and its ability to quantize already trained models without the need for additional finetuning.

A. *Mixed Precision*

In practice, different levels of precision are applied selectively throughout the Transformer architecture. High-precision operations, such as FP16, are reserved for critical components where accuracy is paramount, and which are not compute intense like SoftMax activation and elementwise operations. These operations are sensitive to small numerical changes, and maintaining high precision ensures that the model can capture the intricate relationships within the data. On the other hand, lower-precision formats, like INT8 or even INT4, are used to optimize performance by reducing the computational load and memory usage. This is particularly advantageous for real-time applications or when operating in resource-constrained environments, though it may come with a slight compromise in accuracy. The decision on where to apply different levels of precision is driven by the specific needs of the application. In scenarios where speed and efficiency are more important, lower precision can be utilized to achieve faster inference times and reduce power consumption. By carefully managing these trade-offs, developers can tailor Transformer models to deliver the optimal balance between accuracy and performance for their intended use cases.

B. *Quantization Granularity*

Quantization granularity refers to the level at which quantization is applied within a Transformer model, impacting both the precision and the efficiency of computations. There are several approaches to quantization granularity, each suited to different aspects of the model's architecture.

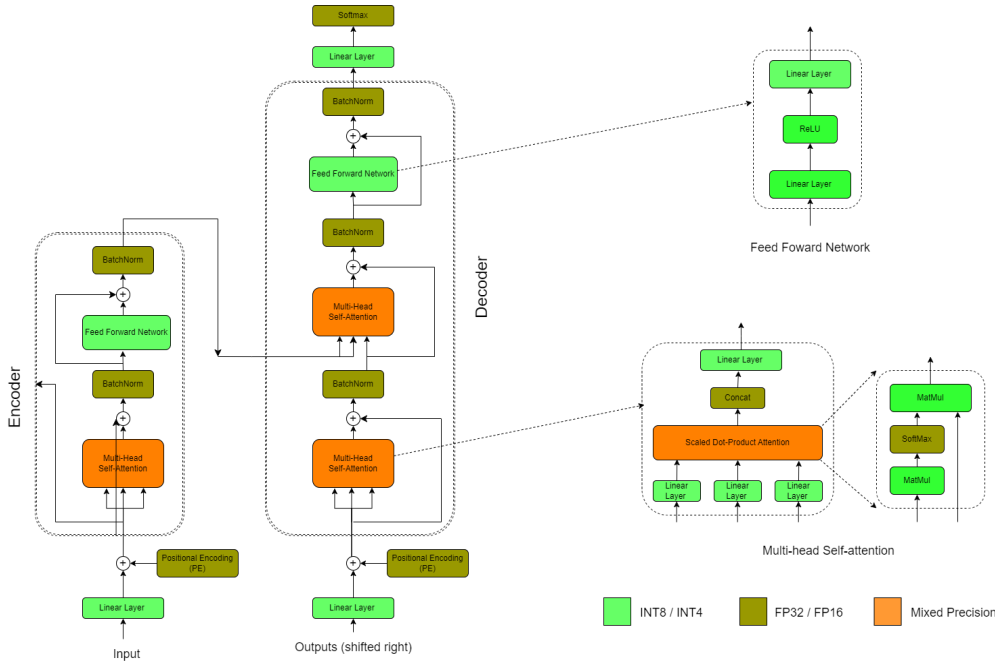

Fig. 4 Quantization precision mapping for Transformers [6]

Per-Tensor Quantization is the most straightforward approach, where a single quantization coefficient is applied across an entire tensor. While this method is computationally efficient, it may lead to a loss in accuracy, especially in complex models like Transformers where the dynamic range of values can vary significantly across different parts of the tensor.

Per-Group Quantization offers a finer level of control by applying quantization across groups of rows or columns within a tensor. For example, M rows (for activations) or K columns (for weights) might correspond to a single quantization coefficient, with K often set to values like 64. This method balances the trade-off between computational efficiency and maintaining accuracy by allowing different parts of the tensor to be quantized differently, depending on their significance.

Per-Channel or Per-Token Quantization provides the highest granularity, applying a separate quantization coefficient to each individual channel or token. In the case of per-token quantization for activations (denoted as X), each row of the activation matrix receives its own quantization coefficient. Similarly, for per-channel quantization of weights (denoted as W), each column is assigned a distinct quantization coefficient. This approach allows for the most precise adjustments, preserving the nuances in data processing but at the cost of increased computational complexity.

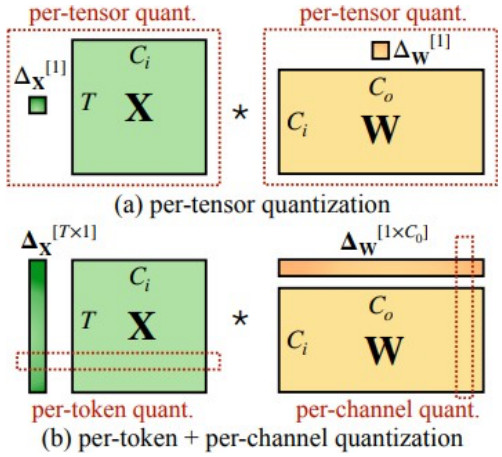

Fig. 5 Quantization granularity for Transformers [1]

By choosing the appropriate quantization granularity, developers can finetune the balance between the model's accuracy and the computational resources required for its deployment. Per-channel and per-token quantization methods are useful when maintaining high accuracy is crucial, while per-group and per-tensor quantization are favored for optimizing performance in resource-constrained environments.

Many quantization algorithms try to use different trade-off quantization schemes, mixed precision and quantization granularity to achieve the highest accuracy for the best performance efficiency. We discuss a few popular quantization techniques applied to transformer based LLMs mainly i) GPTQ ii) AWQ iii) SmoothQuant and iv) Block Quantization. We deep dive into different trade-offs for these different methods and analyze the effect on accuracy for popular models.

C. GPTQ

GPTQ (Gradient Post-Training Quantization) is an efficient algorithm for layerwise quantization of large language models (LLMs), designed to reduce the computational footprint while maintaining model accuracy. The algorithm converts floating-point weight parameters into quantized integers, aiming to minimize errors at the output level. The process begins by performing a Cholesky decomposition of the Hessian inverse matrix, which helps guide the quantization by understanding how weight changes affect the model's output. GPTQ operates in batches to run efficiently on GPUs, where each batch contains a subset of weight matrix columns.

For each column, the algorithm performs the following steps:

i. Quantizes the weights by converting floating-point values into lower-precision integers.

ii. Calculates the quantization error, which represents the difference between the original and quantized values.

iii. Updates the weights within the current block to account for the error, ensuring better approximation.

After processing a batch, GPTQ further updates all the remaining weights, compensating for any errors introduced in the quantized block. This iterative process ensures that the entire weight matrix is adjusted, allowing the model to retain high accuracy despite reduced precision. By processing weights in isolation and updating errors dynamically, GPTQ enables significant model compression with minimal accuracy loss.

D. Activation-aware Weight Quantization (AWQ)

AWQ (Activation-Weighted Quantization) [7] is an advanced quantization technique that leverages activation information to enhance the precision of weight quantization in Transformer models. Unlike traditional methods that primarily rely on the magnitude of weights to guide quantization, AWQ focuses on the sensitivity of weights based on activation patterns. Even minor contributions ranging from 0.1% to 1% can significantly improve the overall quantization results. One of the core principles of AWQ is the use of activation-based scaling, which has proven to be more effective than scaling based solely on weight magnitude. This approach ensures that the scaling factors are aligned with the actual importance of weights in the context of activations, leading to a more accurate quantization. To determine the optimal scaling factors, a small calibration dataset is used, allowing the model to adjust the scales in a way that minimizes quantization error.

Additionally, AWQ adopts a hardware-friendly approach by utilizing integer scales rather than floating-point ones. This minimizes quantization error and is more compatible with the low-precision arithmetic used in modern hardware accelerators.

Empirical heuristics suggest that scaling values less than or equal to 2 yield the best results, producing the least quantization error. AWQ typically employs a 4-bit quantization for weights and a 16-bit quantization for activations (W4A16). This configuration balances between reducing the model size and computational load while maintaining a high level of accuracy, particularly in scenarios where activation sensitivity plays a crucial role. By combining these techniques, AWQ provides a robust solution for deploying high-performance Transformer models in environments where resources are limited.

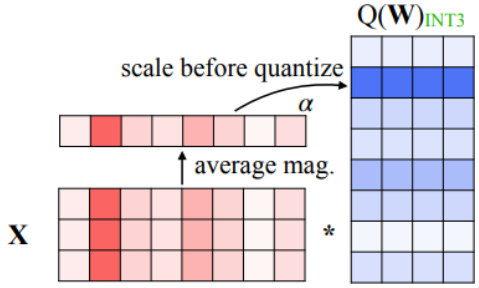

Fig. 6 Overview of AWQ Quantization, scaling weights before quantization [7]

E. Smooth Quant

SmoothQuant [8] is a technique designed to address the challenges of quantizing Transformer models, particularly focusing on the difficulty of quantizing activations due to their wide dynamic range. This wide range often results in a significant limitation on quantization precision, making it challenging to maintain model accuracy. In contrast, weights typically exhibit a more uniform distribution, making them easier to quantize with higher precision. To mitigate the impact of quantization on activations, SmoothQuant proposes a novel scaling method that redistributes quantization error from activations to weights. This redistribution is controlled by a parameter known as "migration strength," which determines the extent to which quantization error is shifted. A migration strength of 0 indicates that all quantization error remains in the activations, reflecting the original distribution. Conversely, a migration strength of 1 moves all the quantization error to the weights. Through experimentation, it has been observed that a migration strength within the range of 0.4 to 0.5 offers an optimal balance, minimizing the overall quantization error.

SmoothQuant utilizes an 8-bit quantization scheme for both weights and activations (W8A8), a configuration that helps achieve a balance between model efficiency and accuracy. The flexibility of this method is further enhanced by its orthogonality to other quantization schemes, meaning it can be integrated with various existing quantization approaches without being constrained by them. This versatility makes SmoothQuant a powerful tool for improving the performance of Transformer models in resource-constrained environments while maintaining high accuracy.

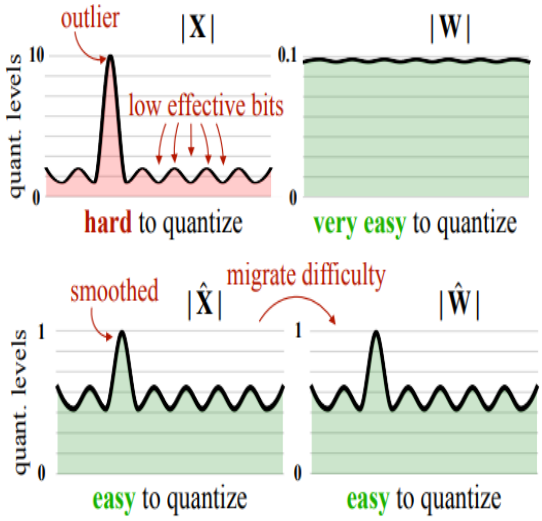

Fig. 7 SmoothQuant showing the migration factors that transfers variability in activations to weights

V. Results and Analysis

In this section, we will be evaluating the results for different quantization techniques on popular models like OPT and Llama, across their different variants in sizes.

A. Model Evaluation

Perplexity is a metric used to evaluate the performance of language models, measuring how well a model predicts a sample of text. Using the WikiText-2 dataset containing a diverse collection of Wikipedia articles, perplexity is calculated by determining the inverse probability of the test words normalized by the number of words. A lower perplexity indicates that the model has a better understanding of the language and predicts the next word more accurately. Perplexity helps in comparing different models and assessing their ability to generate coherent and contextually relevant text.

B. GPTQ

GPTQ can accurately compress some of the largest publicly available models down to 3 and 4 bits. Tables show the perplexity measurement on the wikitest-2 database on different sizes of the OPT models, showing consistent results even for large model sizes. Similarly, Table 2 shows the perplexity metrics for the Llama-2 family of models. These results are generated using the AutoGPTQ library [10].

TABLE I: Perplexity metrics on WikiText-2 for AutoGPTQ on OPT models

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 II: Perplexity metrics on WikiText-2 for AutoGPTQ on Llama-2 models

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

C. AWQ

Activation-aware Weight Quantization (AWQ) is an effective way for low-bit 4 bit weight quantization. Table 1. shows the perplexity measurement on the wikitest-2 dataset on different variants of OPT based models, showing consistent results across model sizes from 1.3B to 30B. Similarly, we look at the perplexity metrics for the Llama-2 family of models showing consistent results with 4-bit quantization.

TABLE III: PERPLEXITY METRICS ON WIKITEXT-2 FOR AWQ ON OPT MODELS

PPL	OPT-1.3B	OPT-2.7B	OPT-6.7B	OPT-13B
FP16	14.62	12.47	10.86	10.13
AWQ (INT3-g128)	16.32	13.58	11.39	10.56
AWQ (INT4-g128)	14.92	12.70	10.92	10.22

TABLE IV: PERPLEXITY METRICS ON WIKITEXT-2 FOR AWQ ON LLAMA-2 MODELS

PPL	Llama-2 7B	Llama-2 13B	Llama-2 70B
FP16	5.47	4.88	3.32
AWQ (INT3-g128)	6.24	5.32	3.74
AWQ (INT4-g128)	5.6	4.97	3.41

D. SmoothQuant

SmoothQuant shows reliable results with 8-bit quantization for different variants of popular model i.e. OPT / Llama-2. Table 4 compares the perplexity metric measured on WikiText-2 dataset for FP16 and SmoothQuant models, it shows consistent results with minimal drop in accuracy across different variants of the OPT/Llama-2 models.

TABLE V: COMPARISON OF PERPLEXITY METRICS FOR FP16 AND SMOOTHQUANT(A8W8) ON OPT MODELS

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 VI: COMPARISON OF PERPLEXITY METRICS ON WIKITEXT-2 FOR FP16 AND SMOOTHQUANT (A8W8)

PPL	Llama-2 7B	Llama-2 13B	Llama-2 70B
FP16	5.47	4.88	3.32
SmoothQuant (A8W8)	5.515	4.929	3.359

VI. CONCLUSION

In conclusion, as the demand for LLM applications grows, efficient deployment strategies become increasingly critical. Quantization stands out as a key solution, enabling significant reductions in computational and memory overhead while addressing the pressing concerns of cost, environmental impact, and data privacy at the edge. By exploring and implementing advanced quantization techniques like AWQ, SmoothQuant, and Block Quantization, we can unlock the full potential of large language models in resource-constrained environments.

Along with quantization techniques, choosing quality data based on the intended end-use can greatly improve performance. This presentation will provide valuable insights into the practical application of these techniques, highlighting their benefits and trade-offs and ultimately guiding the path toward more sustainable and efficient GenAI deployments.

REFERENCES

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 1877-1901. CurranAssociates, Inc., 2020a. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[2] Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.

[3] Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[4] Yao, Z., Aminabadi, R. Y., Zhang, M., Wu, X., Li, C., and He, Y. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers, 2022. URL https://arxiv.org/abs/2206.01861.

[5] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, S. Han, Smoothquant: Accurate and efficient post-training quantization for large language models, in: *ICML, Vol. 202 of Proceedings of Machine Learning Research, PMLR*, 2023, pp. 38087-38099. 2, 20

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[7] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han, "Awq: Activation-aware weight quantization for llm compression and acceleration," 2023.

[8] Phi-3 Microsoft, "Technical Report: A Highly Capable Language Model Locally on Your Phone", 2024.

[9] Multi-task Language Understanding on MMLU, 2024.

[10] AutoGPTQ:https://github.com/AutoGPTQ/AutoGPTQ/blob/main/examples/quantization/bas ic_usage_wikitext2.py

[11] Gholami, A., Yao, Z., Kim, S., Hooper, C., Mahoney, M. W., and Keutzer, K. Ai and memory wall. *IEEE Micro*, pp. 1-5, 2024.

[12] Dwith Chenna, "Quantization of Convolutional Neural Networks: A Practical Approach", *International Journal of Science & Engineering Development Research*, Vol.8, Issue 12, page no.181 - 192, December-2023, Available :http://www.ijrti.org/papers/IJRTI2312025.pdf

[13] AWQ, Github: https://github.com/mit-han-lab/llm-awq

[14] SmoothQuant, Github: https://github.com/mit-han-lab/smoothquant