LLM Compression: How Far Can We Go in Balancing Size and Performance?

Sahil Sk[*], **Debasish Dhal**[*], **Sonal Khosla**[*], **Sk Shahid**[*], **Sambit Shekhar**[*], **Akash Dhaka**[†], **Shantipriya Parida**[†], **Dilip K. Prasad**[‡], **Ondřej Bojar**[§]

[*]Odia Generative AI, India; [†]AMD Silo AI, Finland
[‡]The Arctic University of Norway, Norway
[§]Charles University, MFF, ÚFAL, Czech Republic

Correspondence: akash.dhaka@amd.com

Abstract

Quantization is an essential and popular technique for improving the accessibility of large language models (LLMs) by reducing memory usage and computational costs while maintaining performance. In this study, we apply 4-bit Group Scaling Quantization (*GSQ*) and Generative Pretrained Transformer Quantization (*GPTQ*) to *LLaMA 1B*, *Qwen 0.5B*, and *PHI 1.5B*, evaluating their impact across multiple NLP tasks. We benchmark these models on *MS MARCO* (Information Retrieval), *BoolQ* (Boolean Question Answering), and *GSM8K* (Mathematical Reasoning) datasets, assessing both accuracy and efficiency across various tasks. The study measures the trade-offs between model compression and task performance, analyzing key evaluation metrics namely: accuracy, inference latency, and throughput (total output tokens generated per second), providing insights into the suitability of low-bit quantization for real-world deployment. Using the results, a user can then make a suitable decision based on the specifications that need to be met. We discuss the pros and cons of GSQ and GPTQ techniques on models of different sizes, which also serve as a benchmark for future experiments.

1 Introduction

The increasing demand for high-performing LLMs has driven the development of transformer architectures with billions of parameters, capable of achieving state-of-the-art results and unlocking new capabilities in various language understanding tasks such as reasoning, proof-checking, and automated software development. However, the size and complexity of these models often pose significant challenges, including high computational costs in terms of floating point operations per second, memory requirements, and energy consumption or limited throughput. These limitations hinder the deployment of such models on resource-constrained devices such as mobile phones, Internet of Things (IoT) devices, and edge computing platforms.

Quantization is one of the techniques that has gained prominence lately; it reduces the precision of model weights and activations, for efficient deployment on resource-constrained hardware. It reduces the number of bits used to represent each parameter (e.g., from 32-bit floating-point, FP32, to 8-bit integer, INT8, or lower), thereby enabling lower memory usage and faster inference.

Quantization can also support deployment on low resource/power devices like FPGA, Neural Processing Units (NPU) and System on a chip (SOCs) and can be combined easily with other compression techniques like knowledge distillation and pruning and multiply compression effects.

In this paper, we explore the effectiveness of two compression techniques, GSQ and GPTQ, in computing the resource requirements of a model as well as their impact on various performance metrics.

Based on our experiments, our key findings are as follows:

- We observe that whether to use quantization and the choice of technique will ultimately depend on the user's requirements in terms of tasks and the model they decide to use.
- 4-bit quantization schemes used in this work had little to no impact on latency and throughput, supporting their practical deployment on production. In some cases, there was a noticeable overhead due to their implementation.

2 Related Work

Some of the Post-training quantization (PTQ) techniques are: static quantization (converts both weights and activations to a lower-bit format (e.g.,

Proceedings of Recent Advances in Natural Language Processing, pages 1183–1187
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-136

INT8; (Montestruque and Antsaklis, 2007)), dynamic quantization (only weights are quantized (e.g., FP32 → INT8), while activations remain in FP32; (Montestruque and Antsaklis, 2007)), weight-only quantization (only model weights are quantized; (Kim et al., 2023)), GSQ which splits model weights into small groups and applies different scaling factors per group (Zeng et al., 2025), GPTQ which quantizes a LLM one layer after another (Sharify et al., 2024), KL Divergence Based Quantization (Xie et al., 2016) and Smooth Quantization (Xiao et al., 2024).

3 Methodology

3.1 Quantization Techniques

In this section, we discuss the mechanism behind the two techniques used in our experiments.

3.1.1 Generative Pretrained Transformer Quantization (GPTQ)

GPTQ is a one-shot quantization technique that reduces the model size by converting weights to a lower bit representation (such as 8-bits or 4-bits) from the original 32 bit or 64 bit precision (Frantar et al., 2023). Since this could lead to a loss of model accuracy, GPTQ minimizes quantization errors using a dynamic error correction technique that adjusts subsequent weights to compensate for previous errors during inference.

This also allows for faster computation during inference (Rajput and Sharma, 2024), as lower-precision arithmetic operations (e.g., 8-bit multiplications) are more computationally efficient than high-precision operations.

3.1.2 Group Scaling Quantization (GSQ)

GSQ, is based on Activation Weight Quantization (AWQ) (Lin et al., 2024) introduces an innovative technique that prioritizes activation-aware scaling, GSQ divides the weight matrix into groups and assigns a shared scaling factor. This ensures that all quantized values fit within the INT4 range, minimizing precision loss. Instead of selecting weights based on magnitude, GSQ identifies important weights by quantifying their impact on activation. GSQ has been shown to preserve more fine-grained information and potentially yield higher post-quantization accuracy. The working of the technique is shown in Figure 1.

3.2 Experiment Setup and Metrics

In this section, we describe our experimental configuration and the benchmark datasets used. We evaluated three language models to assess the impact of quantization and group size on various downstream tasks:

- LLaMA 3.2-1B (Touvron et al., 2023): Fine-tuned on GSM8K (Cobbe et al., 2021); validated on BOOLQ (Clark et al., 2019), MS MARCO (Nguyen et al., 2016), GSM8K (5,000 samples).
- Qwen 0.5B (qwe, 2024): Fine-tuned on; validated on BOOLQ, MS MARCO (5,000 samples).
- Phi 1.5B (Li et al., 2023): Fine-tuned on GSM8K; validated on BOOLQ, MS MARCO, GSM8K test split(5,000 samples).

Results are reported for both pre-quantization and post-quantization stages.

1. Pre-quantization (Unquantized) Evaluation
2. Post-quantization (Quantized) Evaluation: We then apply quantization, maintaining the same validation setup to measure changes in accuracy and other metrics.

To thoroughly compare pre-quantization and post-quantization model performance, we used a combination of accuracy based metrics like perplexity, and accuracy and efficiency based metrics like inference latency, throughput, and memory usage. Accuracy evaluates the proportion of correct predictions in binary classification tasks as in dataset like BoolQ, while perplexity measures the log-likelihood score at each generation step.

Inference Latency is the average time needed to generate a response or process a batch of inputs (Hasan, 2024). By comparing pre and post-quantization latency, we determine the impact of quantization on real time or near-real-time deployment scenarios, taking into account any effect due to the overhead involved. Throughput measures how many samples (or tokens) a model can process per second (et al., 2017). We compare the rate at which models can handle input data in both unquantized and quantized forms, illustrating the trade-offs between speed and accuracy. Memory usage tracks the RAM and GPU memory footprint during inference (Frantar et al., 2023). Lower memory usage can enable deployment on resource-constrained devices. We quantify the difference in memory usage before and after quantization to show whether quantization yields tangible resource savings.

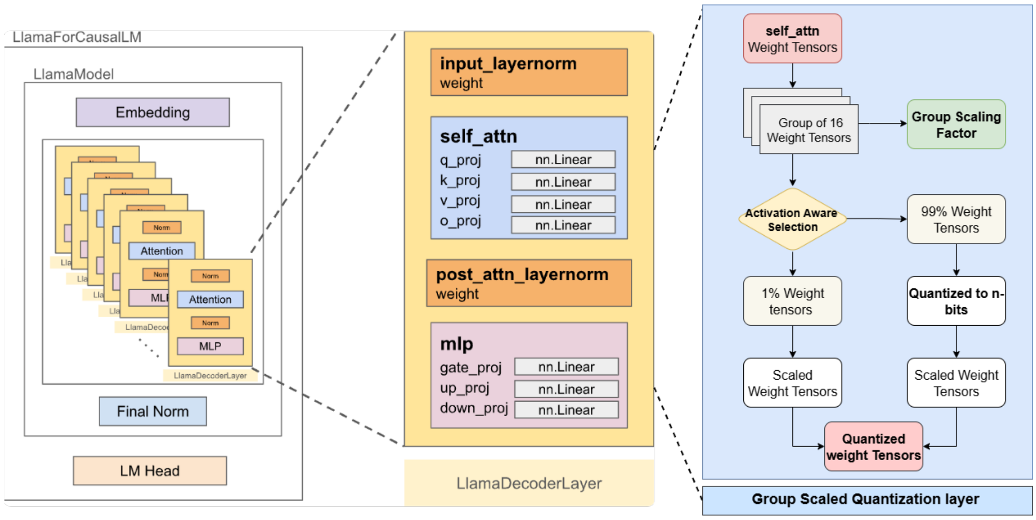

Figure 1: Detailed view of the architecture behind GSQ Quantization Process.

All experiments were conducted on one NVIDIA A100 SXM4 40GB GPU using PyTorch and the Transformers libraries.

4 Results and Analysis

The results are summarized and analyzed in Table 1. For the LLaMA 1B model, GSQ actually improved accuracy on MS MARCO (81.12% $\rightarrow$ 84.04%) with minimal impact on latency, memory, or throughput. GPTQ maintained high accuracy, with a small reduction in memory usage after quantization. On BoolQ, GSQ significantly increased throughput (69.57 $\rightarrow$ 364.91), while accuracy remained mostly unchanged for both. On GSM8K, GSQ showed a slight accuracy drop (1.21% $\rightarrow$ 1.14%) with a small increase in memory use.

Overall, GPTQ outperforms GSQ in accuracy for MS MARCO and BoolQ, while GSQ maintains better stability in latency and throughput but underperforms on GSM8K.

For the *QWEN 0.5B* model, GSQ's accuracy declined (19.54% $\rightarrow$ 14.90%), while GPTQ improved (6.84% $\rightarrow$ 10.66%). On BoolQ, accuracy changes were minor, indicating low sensitivity to quantization. Latency increased for both methods (0.1105 s $\rightarrow$ 0.2459 s). Throughput dropped notably across GSM8K, MS MARCO, and BoolQ.

In summary, quantization affects throughput and latency more than accuracy, with GPTQ offering better accuracy overall, and GSQ delivering more stable runtime performance.

The *Phi* model is a small-scale language model capable of general NLP tasks and QA tasks. Although they are trained on GSM8K, the accuracy is low and also there is hardly any difference after quantization. The model struggles with GSM8K regardless of quantization. Quantization had minimal impact on inference latency, and memory remained almost unchanged across models.

However, it is worth noting is that there is a significant drop in throughput across all datasets in the GPTQ method, although the GSQ method still does not have a major drop in throughput. While *4-bit quantization* at a *16-group size* can lead to higher accuracy retention, the above experiments revealed the following trade-offs when compared to larger-group quantization:

- Improved Accuracy: The structured quantization approach preserves crucial model parameters more effectively, enhancing accuracy on validation tasks.
- Reduced model size: The quantization methods are able to achieve up to a 13-fold reduction in model size with minimal drop in performance across all benchmarks.
- Increased Latency: Processing smaller groups introduces additional overhead, resulting in slightly higher per-inference execution time.

These observations underscore the importance of balancing group size, bit precision, and task requirements. For deployments that prioritize accuracy, a 16-group size may be ideal despite the higher cost in latency, throughput, and memory usage.

Perplexity has slightly increased after quantization across all three models for Wikitext, MS

Model	Dataset	Baseline Score (%)	Best Quantized Score (%)	Memory Reduction	Key Observations
LLama 1B	GSM8K	1.21	1.14 (GSQ)	-391 MB†	GPTQ performance score is very low
	MS MARCO	81.12	**99.86 (GPTQ)**	+2691 MB†	GSQ improves baseline
	BoolQ	40.15	**62.17 (GPTQ)**	+2240 MB†	Significant improvement
QWEN 0.5B	GSM8K	–	0.00 (GPTQ)	No change	Both methods fail/missing
	MS MARCO	19.54	**14.90 (GSQ)**	No change	All scores degraded
	BoolQ	56.21	**55.81 (GSQ)**	No change	Minimal degradation
Phi 1.5B	GSM8K	2.58	2.50 (GSQ)	+527 MB†	GPTQ performance score is very low
	MS MARCO	99.80	**99.82 (GSQ)**	-2742 MB	Slight improvement
	BoolQ	–	40.18 (GPTQ)	-437 MB	Baseline missing

Critical Issues	Description
GPTQ Math Failure	GPTQ achieves very low accuracy on GSM8K across ALL models
Memory Paradox	Quantized models often use MORE memory than baseline (marked with †)
Missing Data	Extensive GSQ data missing, particularly for BoolQ datasets
Inconsistent Benefits	Quantization benefits vary dramatically by model-dataset combination

Table 1: Executive Summary: Quantization Performance Across Models. † Negative memory reduction (quantized uses more memory) suggests experimental issues or inefficient implementation. Best performing quantization method shown in bold. The memory footprint for each dataset is different due to different input token lengths.

MARCO, BoolQ, and GSM8K. The increase in perplexity is more pronounced in GPTQ than in GSQ. In edge cases, GPTQ gives us an almost two-fold increase in perplexity, while for GSQ, no such event was observed.

In summary, we examined the effects of *GSQ-based 4-bit quantization* and *GPTQ-based 4-bit quantization* on three different language models: *LLaMA 3.2–1B*, *Qwen 0.5B*, and *Phi 1.5B* across various tasks. The *16-group size* configuration generally preserved higher accuracy at the expense of increased latency, lower throughput, and elevated memory usage. The metrics such as *perplexity*, *accuracy*, *inference latency*, *throughput*, and *memory usage* provided the trade-offs faced when compressing large language models.

The experiments for GSQ were also tried with a group size of 64 and a maximum sequence length of 512. The throughput has increased across all models. Worth noting is that there is a drop in latency post quantization in all models, though the drop is very small for LLama.

5 Limitations

We highlight key limitations of GSQ (and partially GPTQ) observed during our experiments:

- *Group Size Constraint:* GSQ requires the last tensor dimension to be divisible by the group size (e.g., a (256, 100) tensor fails with group size 32 due to misalignment).
- *Lack of Layer-wise Flexibility:* A fixed group size across layers restricts GSQ's applicability to models with varying layer dimensions.
- *Sensitivity to Fine-Tuning:* Fine-tuned models often introduce sparsity or minor structural changes. Large group sizes tend to fail; smaller ones (≤ 16) work better.
- *No Fallback Handling:* GSQ lacks mechanisms to detect or adapt to incompatible shapes, leading to runtime failures.

6 Conclusion and Future work

Our results show that LLaMA 1B benefits from quantization, even outperforming the base model on MS MARCO, while smaller models like Qwen 0.5B suffer significant accuracy loss. BoolQ remains largely unaffected by quantization, whereas GSM8K, a math-focused dataset, demonstrates sensitivity due to precision loss. Efficiency metrics reveal minimal impact on latency and throughput, suggesting that 4-bit quantization is a viable compression technique for real-world deployment. Future work includes a layer-wise analysis of the effects of 4-bit quantization.

Acknowledgement

This work was supported by the Research Council of Norway Project (nanoAI, Project ID: 325741).

References

2024. Qwen2 technical report.

Norman P. Jouppi et al. 2017. In-datacenter performance analysis of a tensor processing unit. *arXiv preprint arXiv:1704.04760*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Jahid Hasan. 2024. Optimizing large language models through quantization: A comparative analysis of ptq and qat techniques.

Young Jin Kim, Rawn Henry, Raffy Fahim, and Hany Hassan Awadalla. 2023. Finequant: Unlocking efficiency with fine-grained weight-only quantization for llms. *arXiv preprint arXiv:2308.09723*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration.

Luis Antonio Montestruque and Panos J Antsaklis. 2007. Static and dynamic quantization in model-based networked control systems. *International Journal of Control*, 80(1):87–101.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Saurabhsingh Rajput and Tushar Sharma. 2024. Benchmarking emerging deep learning quantization methods for energy efficiency. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*, pages 238–242.

Sayeh Sharify, Utkarsh Saxena, Zifei Xu, Wanzin Yazar, Ilya Soloveychik, and Xin Wang. 2024. Post training quantization of large language models with microscaling formats. *arXiv preprint arXiv:2405.07135*.

Hugo Touvron, Louis Martin, Kevin Stone, and et al. 2023. Llama 2: Open foundation and fine-tuned chat models.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2024. Smoothquant: Accurate and efficient post-training quantization for large language models.

Feng-Long Xie, Frank K Soong, and Haifeng Li. 2016. A kl divergence and dnn-based approach to voice conversion without parallel training sentences. In *Interspeech*, pages 287–291.

Chao Zeng, Songwei Liu, Shu Yang, Fangmin Chen, Xing Mei, and Lean Fu. 2025. Gqsa: Group quantization and sparsity for accelerating large language model inference.