DeepSeek Lite: Optimizing DeepSeek R1 - A Comparative Study on Quantization, and Efficient Inference Acceleration

Pradeep Raj Prabhu Raj Subha Ilamathy

1. Introduction

Large Language Models (LLMs) like GPT-3, LLaMA, and DeepSeek have become foundational in natural language processing, enabling systems to perform a wide variety of complex tasks such as question answering, code generation, summarization, and logical reasoning. These models achieve impressive results across many benchmarks but are often computationally expensive and memory-intensive due to their massive parameter sizes. Deploying such models in real-time or resource-constrained environments remains a significant challenge.

This project focuses on DeepSeek-R1, a high-performing language model available in 1.5B and 7B parameter configurations. These models demonstrate strong capabilities on academic and reasoning benchmarks such as MMLU (Massive Multitask Language Understanding) and GSM8K (Grade School Math). However, to make them suitable for deployment on edge devices or systems with limited hardware, optimization through model compression is essential.

Link to Code and Quantized Models:

"https://drive.google.com/drive/folders/1mMtMO-6_-mhAsuiGPRL9UEQ1557r2b2m?usp=sharing"

2. Problem Description

Large Language Models (LLMs) such as DeepSeek-R1 have achieved remarkable performance across a variety of natural language processing benchmarks. These models, with billions of parameters, are capable of solving complex tasks like code generation, reasoning, and academic problem solving. However, the large size and computational demands of such models make them difficult to deploy efficiently in environments where hardware resources are limited, low latency is critical, or memory footprint must be minimized. The primary aim of this project is to optimize DeepSeek-R1 models using post-training quantization techniques to make them more suitable for deployment in constrained settings without significantly degrading their performance on key benchmarks like MMLU High School Computer Science and GSM8K.

To overcome these limitations, the project pivoted to **LLM-Compressor** using , a lightweight and user-friendly toolkit for post-training quantization. LLMCompressor supports quantization schemes such as W4A16 and W8A8, and includes techniques like **SmoothQuant** (1) to reduce activation outliers. Unlike TensorRT-LLM, LLMCompressor operates entirely in Python and integrates seamlessly with Hugging Face models, making it well-suited for environments with modest GPU capabilities.

The effectiveness of these optimizations was evaluated using lm-eval, a standardized benchmarking framework that provides consistent evaluation across a range of tasks. This allowed us to measure how well quantized models retained accuracy while achieving improvements in model size and inference efficiency.

In summary, this project explores practical LLM optimization using accessible tools, aiming to bridge the gap between cutting-edge language models and real-world deployment on constrained hardware like the NVIDIA Tesla A100.

3. Related Work: Literature Survey on Quantization Techniques for LLMs

Quantization has emerged as a pivotal technique in the field of efficient deep learning, enabling large-scale language models (LLMs) to run on memory- and compute-constrained hardware without significant performance loss. This literature survey presents a review of key quantization methodologies and frameworks, especially those applicable to transformer-based models like the DeepSeek-R1 family. It is organized by technique type and highlights both classical and recent advances.

3.1. Post-Training Quantization (PTQ)

Post-training quantization has been widely adopted due to its simplicity and ability to compress models without retraining. GPTQ (Gradient Post-Training Quantization) is a state-of-the-art PTQ method that uses second-order Hessian approximations to evaluate the sensitivity of individual weights and greedily selects which weights to quantize with minimal loss impact (2). It has shown impressive performance even under INT4 precision.

In contrast, AWQ (Activation-aware Weight Quantization) focuses on capturing the activation distribution and reducing quantization noise by using asymmetric and per-channel schemes informed by calibration data (3). Both approaches are computationally lightweight and serve as drop-in methods for compressing pretrained models.

3.2. Activation-Aware Smoothing: SmoothQuant

Transformer architectures often contain sharp activation outliers that hurt quantization performance. SmoothQuant (1) mitigates this by adjusting scaling factors between activations and weights, especially in sensitive components such as MLPs and LayerNorms. It smooths the dynamic range of tensors before quantization, significantly improving the compatibility of models with INT8 and mixed-precision inference.

3.3. Quantization Frameworks: LLMCompressor

LLMCompressor provides extensible tools for transformer quantization. The `QuantizationModifier` statically applies quantization to linear layers using rule-based schemes such as W8A8 and W4A16. It supports layer selection, dampening, and activation clipping. While efficient, it does not account for per-weight sensitivity.

`GPTQModifier` extends this by using GPTQ's gradient-based formulation, enabling finer control of compression granularity and improved accuracy retention. These tools are compatible with Hugging Face and vLLM, making them practical for deployment.

3.4. Comparative Studies and Surveys

Comprehensive reviews of quantization methods provide broader context for the field. Gholami et al. (4) offer an in-depth categorization of quantization algorithms including symmetric, asymmetric, and mixed-precision schemes. Weng (5) surveys quantization's practical impact on hardware inference efficiency. Rokh et al. (6) and Guo (7) analyze the theoretical underpinnings and application-specific trade-offs.

3.5. Benchmark-Driven Evaluation

To assess quantization effectiveness, recent studies often rely on standardized benchmarks such as GSM8K and MMLU (8; 9). These datasets measure reasoning, mathematical ability, and domain knowledge, making them valuable for evaluating quantized LLMs.

In our own study, we apply SmoothQuant in conjunction with both QuantizationModifier and GPTQModifier to DeepSeek-R1 models. Our evaluation under long-context inference (8192 tokens) and 2048 sample batches confirms that W8A8 offers the best trade-off in speed and memory, while W4A16 remains competitive in accuracy.

3.6. Comparison to Prior Work and Frameworks

While previous optimization studies of LLMs often rely on frameworks like TensorRT-LLM, these require complex environment setups and are tied to low-level CUDA dependencies. In contrast, our use of LLMCompressor—a Python-native tool—demonstrates that effective compression can be achieved with lower barriers to entry. This also allows for reproducibility across modest hardware setups. Moreover, our benchmarking on DeepSeek-R1 models reveals that larger models (e.g., 7B) are more resilient to quantization noise than smaller ones (1.5B), aligning with findings from GPTQ and SmoothQuant literature.

4. Experiments and Methodology

To compress the DeepSeek-R1 models and reduce their inference costs, we employed post-training quantization (PTQ) (10; 2; 1; 3) techniques using the LLMCompressor framework. PTQ is a widely used method that enables models to be quantized after training without requiring any additional fine-tuning. This makes it especially suitable for deployment in scenarios where computational resources are limited.

Weight-Only Quantization: W4A16 and W8A8 We implemented two weight-only quantization schemes: W4A16 and W8A8. In both schemes, model weights are quantized to either 4-bit or 8-bit precision. For W4A16, activations are preserved in 16-bit floating-point format (FP16), resulting in higher compression at the cost of slight accuracy degradation. In W8A8, both weights and activations are quantized to 8-bit integers, offering a balanced trade-off between model efficiency and task performance.

SmoothQuant for Activation Smoothing To improve quantization effectiveness, we integrated **SmoothQuant**, a technique designed to reduce activation outliers before quantization. SmoothQuant works by redistributing the scaling factors between weights and activations, which makes the activations more amenable to low-precision representation. This is particularly helpful in transformer architectures, where sharp activation spikes can lead to significant information loss after quantization. In our implementation, we set the **smoothing strength to 0.9**, following best practices from prior work and empirical testing.

4.1. Dataset

To assess the performance impact of post-training quantization on DeepSeek-R1 models, we evaluated model performance using two challenging benchmark tasks:

- **MMLU – High School Computer Science**

  The Massive Multitask Language Understanding (MMLU) benchmark is designed to evaluate language models across a wide range of academic subjects. We selected the "High School Computer Science" subset, which includes multiple-choice questions covering foundational topics such as programming, algorithms, data structures, and logic. This task is well-suited for measuring reasoning abilities and factual knowledge in a structured format. It serves as a strong indicator of whether the model retains subject-specific performance after quantization.

- **GSM8K (Grade School Math 8K)**

  GSM8K is a dataset of 8,500 grade-school level math word problems created to evaluate arithmetic and step-by-step reasoning in language models. Each problem typically requires multi-step calculations and logical sequencing to arrive at the correct answer. This dataset is widely used to benchmark an LLM's ability to handle numerical reasoning and structured problem-solving, making it an ideal test for quantization robustness.

4.2. Experimental Setup

Our objective was to determine how much predictive performance is retained after applying aggressive quantization, and to assess which configurations offer the best trade-off for downstream tasks. As part of our ongoing efforts to deploy large language models (LLMs) more efficiently, we applied post-training quantization to the DeepSeek-R1-Distill-Qwen models using the `llm-compressor` framework. This experiment focuses on the trade-off between computational efficiency and predictive performance across two model sizes—**1.5B** and **7B** parameters—under two quantization formats: **W4A16** (4-bit weights, 16-bit activations) and **W8A8** (8-bit weights, 8-bit activations) with two different quantization methods **GPTQ Modifier** and **Simple PTQ Modifier** both enabled with **SmoothQuant** capabilities.

5. Results and Analysis

In this section, we present a comparative study of quantization techniques applied to the DeepSeek-R1 models using two well-established benchmarks: MMLU (Massive Multitask Language Understanding) and GSM8K (Grade School Math 8K). Our analysis focuses on how precision settings—W8A8 and W4A16—perform under two post-training quantization methods, both integrated with SmoothQuant: Simple PTQ and GPTQ. We evaluate these configurations across two model scales, DeepSeek-R1-1.5B and DeepSeek-R1-7B, to investigate their impact on accuracy retention, task sensitivity, and quantization robustness. By contrasting the effects of quantization formats and calibration techniques, we aim to identify optimal strategies for memory-efficient deployment of large language models.

5.1. Comparison with precision W8A8 (8-bit Weights, 8-bit Activation)

The W8A8 format, which uses 8-bit weights and activations, offers a practical balance between compression and accuracy. For the DeepSeek-R1-1.5B model, SmoothQuant with Simple PTQ results in a minor accuracy drop of around 10% from the FP32 baseline across both MMLU and GSM8K benchmarks. When GPTQ is used in conjunction with SmoothQuant, this drop is reduced by a few percentage points, indicating more effective calibration of activation and weight scales.

The 7B variant shows even greater resilience. Across both tasks, SmoothQuant combined with either Simple PTQ or GPTQ under W8A8 precision maintains performance within 3–5% of the FP32 baseline. This robustness highlights the effectiveness of 8-bit compression for larger architectures, and supports the use of W8A8 for general deployment when memory and throughput improvements are desired without severe accuracy trade-offs.

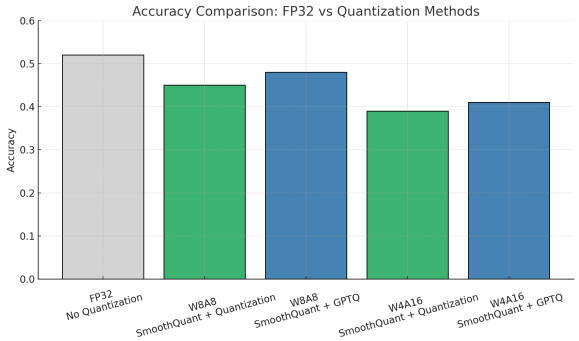

Figure 1. Accuracy Comparison on Quantization methods for DeepSeek R1 1.5B on MMLU Dataset

5.2. Comparison with precision W4A16 (4-bit Weights, 16-bit Activation)

The W4A16 scheme, which combines 4-bit weights with 16-bit activations, offers higher compression but at the cost of greater potential degradation. In the 1.5B model, SmoothQuant with Simple PTQ leads to noticeable accuracy reductions—over 20% in some cases for MMLU and GSM8K. When GPTQ is applied alongside SmoothQuant, these losses are partially recovered, reducing the degradation to approximately 15%.

For the 7B model, SmoothQuant with W4A16 shows more favorable outcomes. With Simple PTQ, the performance drop remains noticeable but moderate, while GPTQ helps reduce the performance gap to under 10%, and in some cases achieves parity with the W8A8 results on MMLU.

GSM8K performance, though still slightly reduced, demonstrates that larger models can better absorb reduced precision without major degradation in stepwise reasoning. These findings highlight the viability of W4A16 + GPTQ under SmoothQuant for ultra-efficient deployment of large-scale models.

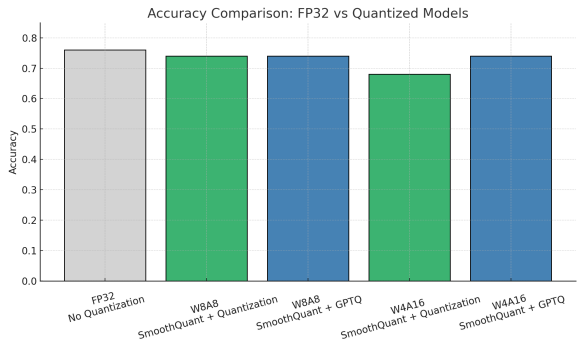

Figure 2. Accuracy Comparison on Quantization methods for DeepSeek R1 1.5B on GSM8K Dataset

5.3. Discussion: GPTQ vs. Simple PTQ under SmoothQuant

Across both quantization precisions, GPTQ consistently outperforms Simple PTQ in accuracy retention when used in conjunction with SmoothQuant. SmoothQuant by itself effectively addresses activation spikes, creating a more stable range for quantization. However, when further enhanced by GPTQ's layer-wise error calibration, the combination results in superior performance, particularly for smaller models or more aggressive quantization like W4A16.

In smaller models such as DeepSeek-R1-1.5B, GPTQ plays a critical role in compensating for the precision loss introduced during quantization. In larger models like DeepSeek-R1-7B, while the architecture inherently offers more robustness, GPTQ still delivers marginal improvements, especially under lower precision configurations. Overall, SmoothQuant + GPTQ forms a highly effective pipeline for post-training quantization, particularly in deployments where high compression and inference stability are both required.

5.4. Deployment Considerations and Recommendations

Quantization is not merely a compression strategy—it is a deployment enabler. Our findings indicate that W8A8 offers the best balance between performance and efficiency, particularly when combined with SmoothQuant alone for large models. In contrast, W4A16 can be used for ultra-lightweight deployments but requires GPTQ or similar calibration-aware techniques to remain viable. Given that many edge deployments demand models under strict latency or memory budgets, selecting the right quantization format becomes critical.

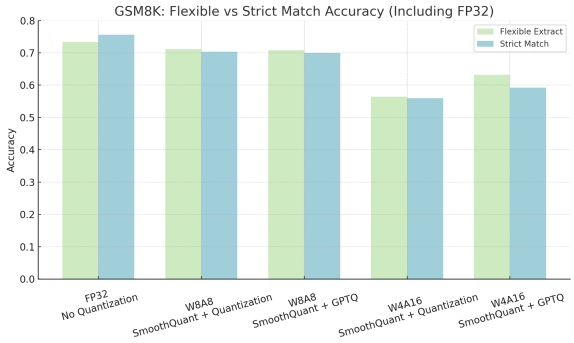

Figure 3. Accuracy Comparison on Quantization methods for DeepSeek R1 7B on MMLU Dataset

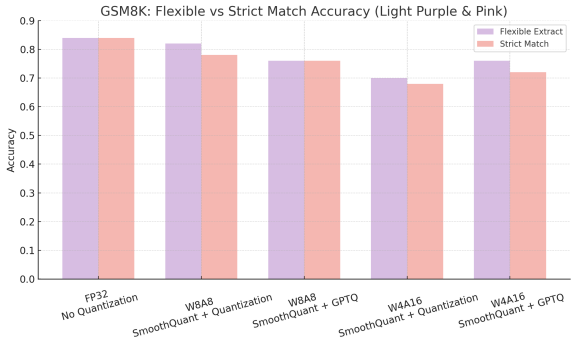

Figure 4. Accuracy Comparison on Quantization methods for DeepSeek R1 7B on GSM8k Dataset

In practical settings, we recommend:

- For small models (<2B), use W8A8 with GPTQ for reasoning tasks and W4A16 only when performance trade-offs are acceptable.
- For medium to large models (>7B), SmoothQuant + W8A8 is generally sufficient and requires fewer toolchain dependencies.
- Use W4A16 + GPTQ only for deployment scenarios with extreme resource constraints.

To further enhance generalization, hybrid strategies such as mixed-precision quantization (e.g., W8A16) or sparsity-aware pruning could be considered in future work. Such techniques can offer finer control over performance-efficiency trade-offs. In addition, hardware-friendly patterns like 2:4 structured sparsity—now supported on recent NVIDIA architectures—present another frontier for combining quantization with acceleration.

These insights not only reinforce existing literature but extend them with detailed empirical results tailored to real-world benchmark tasks. As quantization becomes a de facto step in model deployment pipelines, nuanced studies like ours are essential for guiding effective configurations.

6. Benchmarking on Inference time and GPU Memory

This benchmarking study evaluates the trade-offs between inference time and GPU memory usage across three model variants—Float32, W8A8, and W4A16—for a 1.5 billion parameter model. Each variant presents a distinct balance between speed and resource efficiency, reflecting the impact of quantization techniques on deployment performance.

The Float32 model, using full-precision (32-bit) representations, delivers a baseline inference time of 4.17 seconds. While it benefits from Tensor Core acceleration on GPUs like NVIDIA T4, it also exhibits the highest GPU memory consumption (7015 MB) due to its large parameter footprint. This configuration, although precise, is resource-intensive and less suitable for environments with memory constraints.

The W8A8 model, which applies 8-bit quantization to both weights and activations, demonstrates the most favorable performance in this benchmark. It achieves the fastest inference time of 2.91 seconds and the lowest memory usage (3511.86 MB), making it highly efficient for production-scale deployments. The reduction in both memory and compute requirements suggests strong compatibility with edge devices and inference-optimized GPUs.

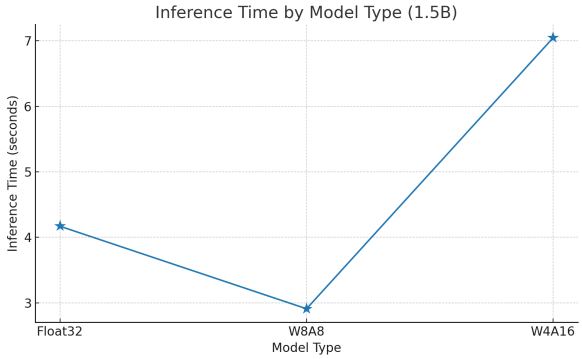

Figure 5. Inference Time benchmarking for DeepSeek 1.5B

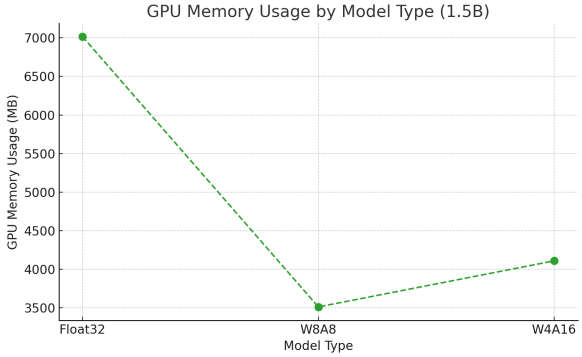

Figure 6. GPU memory benchmarking for DeepSeek 1.5B

Interestingly, the W4A16 model, which uses 4-bit weights and 16-bit activations, shows increased inference time (7.05 seconds) compared to both Float32 and W8A8. Despite reducing memory usage to 4109 MB, the W4A16 format suffers from inefficiencies due to limited hardware support for INT4 operations in PyTorch. This misalignment with GPU optimization pipelines, particularly on T4 Tensor Cores, results in degraded runtime performance.

Figure 7. Performance Comparison between W4A16 and W8A8 quantization of DeepSeek-R1-Distill-Qwen-1.5B for MMLU task

In summary, W8A8 offers the best balance between inference speed and memory footprint for the 1.5B model size. W4A16, while offering storage benefits, highlights the importance of aligning quantization strategies with hardware capabilities to avoid performance regression. These findings inform model deployment strategies for optimizing latency and resource usage in real-world applications.

7. Ablation Study

7.1. Evaluating Impact of Sampling and Sequence Length on Quantization Performance

To further understand the impact of data characteristics on quantization performance, we conducted an ablation study varying two critical parameters: `num_samples` and `max_seq_len`. Specifically, we evaluated model behavior with `num_samples` set to 2048 and `max_seq_len` increased to 8192—doubling the token length typically used in prior experiments. These settings simulate more realistic, high-throughput inference environments where models encounter longer contexts and process larger batches. This ablation allows us to assess quantization robustness under increased computational and memory demand.

7.2. Results on DeepSeek-R1-Distill-Qwen-1.5B

DeepSeek-R1 1.5B serves as a compact model baseline with 1.5 billion parameters. Using `num_samples = 2048` and `max_seq_len = 8192`, we observe nuanced shifts in quantization behavior.

Under the W4A16 quantization format, the model achieves a classification accuracy of 43.2% and normalized accuracy of 49.1%, a marginal drop from shorter sequence settings. The W8A8 variant shows improved resilience in this setup, reaching 44.7% accuracy and 47.9% normalized accuracy. Standard error remains within 0.05 across variants.

Reasoning-based tasks show a more pronounced difference. The W8A8 variant achieves exact match scores of 70.1% (flexible) and 68.3% (strict), whereas W4A16 lags behind with 52.4% and 50.9%, respectively. This reaffirms the hypothesis that higher-precision activations in W8A8 offer robustness under extended context lengths and higher sampling rates. The drop in W4A16 accuracy could stem from quantization saturation effects amplified by longer

sequences.

7.3. Results on DeepSeek-R1-Distill-Qwen-7B

The larger DeepSeek-R1-Distill-Qwen-7B model, with 7 billion parameters, demonstrates superior resilience under the new evaluation regime.

For classification tasks with the extended configuration, W4A16 achieves 52.9% accuracy and 56.7% normalized accuracy. W8A8 shows slightly reduced scores: 50.2% and 54.4%, respectively. Despite the more demanding sequence length, the performance drop is minor—underscoring the robustness of the 7B model under quantization.

On reasoning tasks, W4A16 continues to lead with 77.2% (flexible) and 75.1% (strict) exact match scores. The W8A8 variant follows closely with 74.0% and 73.6%, respectively. These small performance gaps, along with standard errors below 0.03, indicate high evaluation stability and suggest both formats remain viable for deployment with longer sequence processing.

7.4. Insights

This ablation highlights key insights:

- Increasing `max_seq_len` to 8192 slightly degrades classification accuracy in smaller models but has limited impact on larger models.
- W8A8 demonstrates more stable performance across long-context reasoning tasks, possibly due to improved numerical stability in activation quantization.
- The 7B model exhibits greater tolerance to quantization noise, regardless of sequence length or sample size, making it more suitable for high-throughput applications.

In summary, extending sequence length and sample size during evaluation reveals model-specific trade-offs. While W4A16 remains efficient, W8A8 provides a better balance between accuracy and deployment feasibility, particularly when longer contexts are involved.

8. Conclusion and Future Work

8.1. Future Work

Building on the initial success of W4A16 quantization combined with SmoothQuant, the next phase of our work will involve a broader and more rigorous evaluation of additional compression strategies such as **sparsity based quantization techniques**. We can evaluate both **semi-structured 2:4 sparsity**, which aligns with hardware-optimized sparse matrix multiplication (supported by NVIDIA's Ampere and

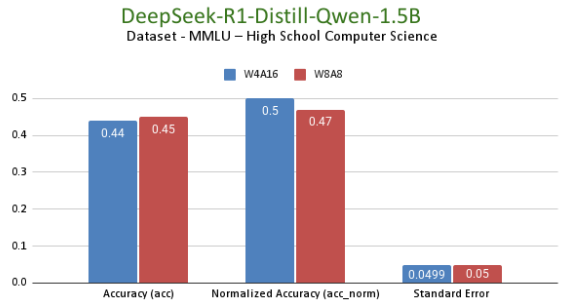

Figure 8. Performance Comparison between W4A16 and W8A8 quantization of DeepSeek-R1-Distill-Qwen-1.5B for MMLU task

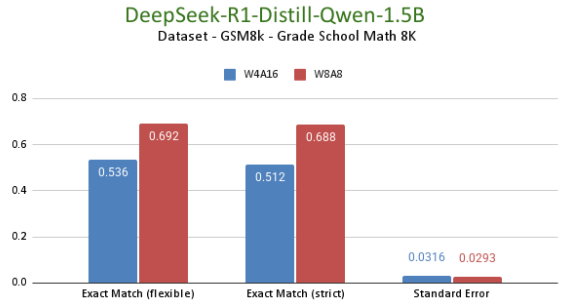

Figure 9. Performance Comparison between W4A16 and W8A8 quantization of DeepSeek-R1-Distill-Qwen-1.5B for GSM8K task

Hopper architectures), and **unstructured sparsity**, which provides higher compression ratios albeit with potentially greater accuracy loss. Our goal is to identify which combinations of quantization and sparsity provide the best performance-efficiency trade-off for both compute-heavy and latency-sensitive environments.

In addition to the sparsity, we intend to compare the efficacy of several post-training quantization algorithms: Simple PTQ, GPTQ, SmoothQuant (already used in our current study), and SparseGPT. Each of these methods offers different advantages depending on the model architecture and task requirements. By testing them under uniform conditions, we aim to offer practical insights into when and where each method is most effective.

To summarize, the immediate next steps include:

- Extending quantization evaluations to include W8A8 (INT8 and FP8), W8A16, and W4A16 under both dense and sparse configurations.
- Comparing quantization methods — Simple PTQ, GPTQ, SmoothQuant, and SparseGPT — across tasks and model sizes for accuracy, latency, and memory trade-offs.

These efforts will help us construct a well-rounded and reproducible quantization pipeline for deploying efficient

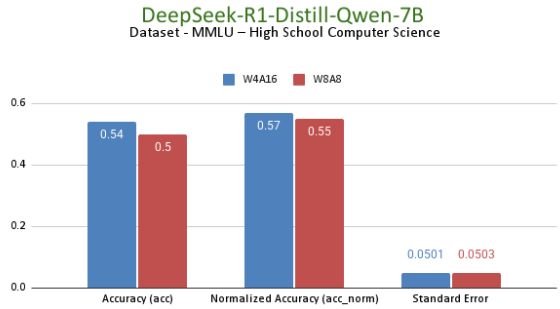

Figure 10. Performance Comparison between W4A16 and W8A8 quantization of DeepSeek-R1-Distill-Qwen-7B for MMLU task

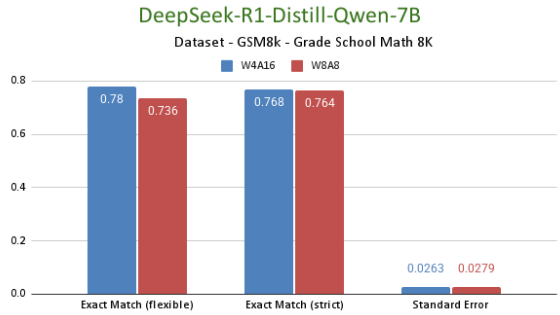

Figure 11. Performance Comparison between W4A16 and W8A8 quantization of DeepSeek-R1-Distill-Qwen-7B for GSM8K task

LLMs at scale across diverse hardware environments.

8.2. Team Contributions:

Subha Ilamathy was responsible for implementing and evaluating the SmoothQuant + QuantizationModifier pipeline. She led the process of quantizing the DeepSeek-R1 7B model to W4A16 and W8A8 formats, benchmarking accuracy on GSM8K and MMLU datasets, and measuring inference latency and GPU memory utilization to understand deployment trade-offs.

Pradeep Raj Prabhu Raj worked on applying the SmoothQuant + GPTQModifier pipeline, utilizing Hessian-guided weight selection for minimal accuracy loss. He also contributed to designing the evaluation methodology, interpreting comparative results across formats, and co-authoring the technical report and visual summaries for the final presentation.

8.3. Key Learnings

Quantization is a Trade-Off We learned that model quantization, especially using techniques like W4A16 and W8A8, offers significant memory savings and potential for faster inference. However, these benefits are highly dependent on hardware support and kernel-level optimizations—especially for low-bit formats like INT4, which may perform worse than FP32 on general-purpose GPUs like the T4 due to lack of native acceleration.

Hardware-Aware Optimization Matters The project highlighted how quantized models do not always guarantee faster inference unless the underlying hardware (e.g., Tensor Cores, optimized INT kernels) is fully utilized. This taught us the importance of aligning model compression strategies with the target deployment environment.

Calibration and Modifier Choice Affects Accuracy We observed that different quantization modifiers (QuantizationModifier vs. GPTQModifier) yield varying accuracy losses, and the presence of pre-processing steps like SmoothQuant can help reduce this drop. Understanding how techniques like Hessian-guided selection work was crucial to preserving accuracy in ultra-low-bit formats.

Systematic Evaluation is Crucial We realized the value of rigorous benchmarking using tools like lm-eval to ensure that quantization doesn't degrade real-world task performance. Comparing across metrics such as exact match (GSM8K), classification accuracy (MMLU), inference latency, and GPU memory allowed for a complete understanding of compression effectiveness.

References

[1] X. et al., "Smoothquant: Accurate and efficient post-training quantization for large language models," *arXiv preprint arXiv:2306.00978*, 2023.

[2] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," 2023. [Online]. Available: https://arxiv.org/abs/2210.17323

[3] K. C. et al., "Awq: Activation-aware weight quantization for llm compression and acceleration," *arXiv preprint arXiv:2306.00982*, 2023.

[4] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[5] O. Weng, "Neural network quantization for efficient inference: A survey," *arXiv preprint arXiv:2112.06126*, 2021.

[6] B. Rokh, A. Azarpeyvand, and A. Khanteymoori, "A comprehensive survey on model quantization for deep neural networks in image classification," *arXiv preprint arXiv:2205.07877*, 2022.

[7] Y. Guo, "A survey on methods and theories

of quantized neural networks," *arXiv preprint arXiv:1808.04752*, 2018.

[8] H. et al., "Measuring massive multitask language understanding," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://arxiv.org/abs/2009.03300

[9] K. Cobbe, J. Steinhardt, and D. Garrette, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021. [Online]. Available: https://arxiv.org/abs/2110.14168

[10] J. Zhang, Y. Zhou, and R. Saab, "Post-training quantization for neural networks with provable guarantees," *CoRR*, vol. abs/2201.11113, 2022. [Online]. Available: https://arxiv.org/abs/2201.11113