Project Report

Group Members: Abhinaya Anil Menon AA5536, Ali Bauyrzhan AB5867, Dacia John DMJ2149

Research Paper Chosen: QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs
(https://arxiv.org/pdf/2404.00456)

Paper Summary

QuaRot is a quantization method that enables full end-to-end 4-bit inference in large language models by eliminating outlier values in weights, activations, and the KV cache. It achieves this by applying randomized Hadamard transformations that "rotate" these components, dispersing extreme values into a more uniform distribution. This rotation is fused into the weight matrices using computational invariance, ensuring that the overall model output remains unchanged, and allowing the use of standard quantization methods like GPTQ or round-to-nearest.On LLAMA2-70B model, the 4-bit quantized version incurs only a 0.47 increase in WikiText-2 perplexity and retains 99% of the original zero-shot performance.

Methodology

Data set: The data set that the QuaRot was tested on are Wikitext2, C4 and PTB however we extended the paper to work with Pile and Wikitext103 for more robust testing. These datasets were available via hugging face.

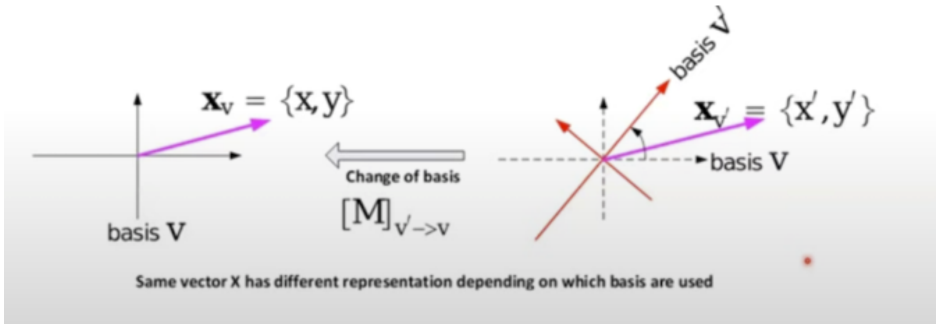
Model chosen: Even though the paper tested out three different models LLaMA-2 7B, LLaMA-2 13B, LLaMA-2 70B due to limitations in hardware the LLaMA-2 7B was used.

System/Hardware Requirements: In recreating the research paper, we came up with the following requirements to run the tests for the Llama-2-7B model:

CUDA >12.0; VRAM >26 GB, TFLOPS per GPU > 12.

Evaluation method: Perplexity was calculated based on the negative log-likelihood and used for evaluating the Post training Quantization loss.

First Extension: Rotation Matrix Analysis


$\mathbf{x}_V = \{x, y\}$
basis V
Change of basis
$[M]_{V'\to V}$
basis V'
$\mathbf{x}_{V'} = \{x', y'\}$
basis V
Same vector X has different representation depending on which basis are used

By applying rotation in the space, the vectors preserve their length, but the values of each entry changes in the way that outlier values may spread over the other entries. With the correct rotation of weights the model will have identical functionality, but the activation layers inside it will have reduced outliers. Nevertheless, some rotation matrices might not work well or even generate more outliers for the activations, making the quantization performance worse. In the paper itself, authors show that random Hadamard matrices usually perform much better compared to the fully random orthogonal matrices. We believe that it is possible to improve the performance of the methodology by picking an optimized rotation matrix. We will focus on the outliers in the activations and build our optimization algorithms based on the outliers of the activation layers.

First, we needed to define what the outliers are. If we compute the absolute Z-scores of every entry of the activation layer, we can consider the outliers to be numbers that are higher than 3 or any other threshold value. Only picking the values that are above the threshold value and taking the mean would give us outlier scores but it would not be differentiable due to the if-else statements in this function. We decided to divide all absolute Z-score values by the threshold, so that any value in the bounds of the threshold would be less than 1 and others are higher than 1. By taking some power (2 for example) of every entry, we are decreasing the non-outlier values and increasing outlier values. Thus the mean of such processed entries would be differentiable analogue for the outlier scoring and serve as the loss function in the optimization algorithms.

$$outlier(Y) \ = \left(\frac{|Y - mean(Y)|}{std*THRESHOLD}\right)^{POWER}$$
$$loss(Y', \, Y) \ = mean(outlier(Y')) - mean(outlier(Y))$$

Here Y is the tensor, an activation layer in the original model. Y’ is the activation layer of the rotated model. By comparing the outlier scores of both tensors we are measuring how well the matrix performed in a cycle.

Another issue here is that we needed to preserve orthogonality of the rotation matrices we are going to change during the process. To keep the constraints, Instead of searching in the R^n space, we needed to search in the Stiefel manifold, a space of orthogonal matrices with the specific sizes we need. That is why, classical SGD algorithms would not be beneficial, since we need to ensure the orthogonality of a matrix at each step. Here, we have used Cayley-SGD that makes special steps on a matrix, and given the initial matrix is orthogonal, the further changed matrices will be orthogonal all the time.

We have run tests with several values of threshold and power, full 4 bit quantization on all activations, keys, values and the weights. We have used the wikitext2 dataset specifically for this methodology. Due to resource limitations, we have only tested the methodology on only one activation layer generating stable results, but still the perplexities of the same level as in the original QuaRot. In the table below the T is the **threshold** and P is the **power** and the corresponding values are the perplexities of the quantized model rotated with the matrix generated with these hyperparameters.

T=1.5 P=2	T=2 P=2	T=2.5 P=2	T=3 P=2	Original (avg)
6.37	6.36	6.37	6.34	6.34

Rotated model using these hyperparameters. (T is the threshold, P is the power, and the values are the perplexities of the quantized)

The experiments show that the threshold value of 3 brings more consistent results. Even though the matrix does not beat the best cases of random Hadamard transformations, it keeps more consistent perplexity after quantization and is scalable for frequent model loading mechanisms.

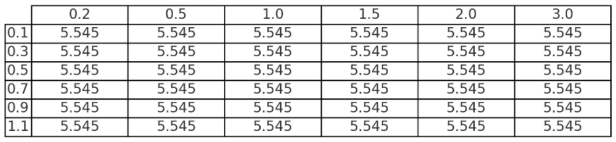
Second Extension: Skew Based Quantization Implementation

QuaRot uses symmetric quantization for all activation layers by default. Its implementation is static in which you choose either symmetric or asymmetric quantization at runtime, and that choice is applied uniformly across every activation layer. While symmetric quantization is more memory-efficient, as it only requires a scale parameter, it assumes the distribution is centered around zero. In contrast, asymmetric quantization can better capture distributions that are offset from zero, since it introduces both a scale and a zero-point (offset), allowing the quantization range to shift and more accurately represent non-zero-centered data.

In Post-Training Quantization (PTQ), calibration involves collecting min/max values or histogram statistics from activations and weights. However, each activation layer can exhibit a different distribution and applying a single quantization method uniformly across all layers overlooks this variation.To address this, we proposed a dynamic approach in which the quantization method (symmetric or asymmetric) is selected per layer based on the level of skewness in its activation distribution during calibration.

The skew analysis was performed before quantization, during the calibration forward pass. By sweeping over different skewness and mean threshold values, measuring perplexity for each configuration tuning the heuristics. This process quantified the sensitivity of performance to these thresholds and helped identify an optimal range where asymmetric quantization is applied only when it offers a genuine benefit.

Perplexity Scores for Skew and Mean/Std Threshold Combinations



	0.2	0.5	1.0	1.5	2.0	3.0
0.1	5.545	5.545	5.545	5.545	5.545	5.545
0.3	5.545	5.545	5.545	5.545	5.545	5.545
0.5	5.545	5.545	5.545	5.545	5.545	5.545
0.7	5.545	5.545	5.545	5.545	5.545	5.545
0.9	5.545	5.545	5.545	5.545	5.545	5.545
1.1	5.545	5.545	5.545	5.545	5.545	5.545

References

Meng, Zijian, Heechul Yun, Daniel Scherer, Zixuan Zhang, and Tal Ben-Nun. “QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs.” *arXiv preprint* arXiv:2404.00456, 2024. https://arxiv.org/abs/2404.00456.

Li, Jun, Li Fuxin, and Sinisa Todorovic. “Efficient Riemannian Optimization on the Stiefel Manifold via the Cayley Transform.” *arXiv preprint* arXiv:2002.01113, 2020. https://arxiv.org/abs/2002.01113.

Meta AI. *LLaMA-2 7B Model*. Hugging Face, 2023. https://huggingface.co/meta-llama/Llama-2-7b.

SPCL Group. *QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs (Code)*. GitHub repository, 2024. https://github.com/spcl/QuaRot.

Repository with our code: https://github.com/K1ta141k/QuaRot_extended

Appendix

Fisher’s moment coefficient of skewness formula used in

$$\gamma_1 := \tilde{\mu}_3 = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu_3}{\sigma^3} = \frac{\mathrm{E}[(X-\mu)^3]}{(\mathrm{E}[(X-\mu)^2])^{3/2}} = \frac{\kappa_3}{\kappa_2^{3/2}}$$