MidLLaMAI: Balancing Quantization with Pruning while compressing LLaMA 2

Anusha Alangar
aalanga1@asu.edu

Pranav Hegde
phegde7@asu.edu

Gokul Vasudeva
gvasude2@asu.edu

Abstract

GPT models being close-sourced prove prohibitive in adapting them to reduced-scope use cases. LLaMA is a family of autoregressive LLMs open-sourced by Meta, ranging from 7 billion to 70 billion parameters, with the authors finding that a 13B parameter LLaMA model outperformed 175B parameter GPT-3 in most NLP tasks [13]. The open weights nature of the LLaMA family of models encourages adaptation to specialized tasks, such as mobility-focused resource-constrained computing, enabling natural language generation on the edge. Looking into the future, we see a need for self-hosting to augment everyday tasks and personal or home assistants. This comes with various advantages, namely privacy and security, reduced and consistent latency, cost savings in the long run, the ability to have greater context windows and fine-tuning to your preference. Compression of model parameters gets these models closer than ever towards this goal, but there is ambiguity in the necessary tradeoffs to be made to proceed. We consider the application of offline quantization techniques like GPTQ and AWQ as well as runtime quantization via Bits and Bytes, in conjunction with GBLM-based pruning, seeking to find a suitable balance between VRAM usage, inference latency and accuracy.

1 Introduction

LLaMA models fall under a class of NLP models that are capable of achieving tasks through few-shot instructions or examples. These models are based on transformers, whereby input text is split into n-grams encoded as tokens and every token is vectorized through word table lookups in an embedding layer. A parallel multi-head attention mechanism is used to decipher the context for other tokens at each subsequent layer, resulting in boosted signals for more relevant tokens [14]. Unembedding layer converts the final vector representations back to a probability distribution over tokens, which results in an architecture that is capable of learnable textual generative tasks. The inception of these models arose from the assumption that scaling the size of these models would lead to better performance. However, more recent work has unearthed superior performance achieved when training a smaller model on vast amounts of data, given limited computational resources [8].

On this line of thinking, further reduction in model size while maintaining the size of the corpus of training data should prove to provide acceptable levels of performance for resource-constrained inference tasks, up to a threshold. There exist various techniques in compression that balance information retention with model size both in memory as well as on disk. We consider two techniques, with Quantization being the more popular and widely used of the two, where inferences are run by representing the weights and activations with low-precision data types, resulting in reduced computational and memory needs and offline quantization also resulting in reduced disk usage. Generally, we see an increase in throughput due to combining several multiplication steps by grouping weights in tensor multiplications.

The other compression technique we consider is pruning, which aims to reduce model size while minimizing loss. Pruning is a method of compression that involves removing weights from a trained model based on certain pruning criteria and heuristics. There are two ways to prune, zeroing out weights called weight-based pruning, or removing nodes, with the former being more popular as it is easier while having minimal degradation in accuracy [16]. Prior works mainly focused on applying one compression technique to maximally preserve accuracy, without balancing resource constraints and without considering the implications of applying multiple compression techniques simultaneously. Our work seeks to fill these gaps.

2 Background

The Transformer Architecture with Attention Mechanism [14] upheaved the field of Natural Language Processing, with models like BERT [3] and RoBERTa [10] achieving state-of-the-art performance in many Natural Language tasks. Further improvement was brought forth with the advent of Large

Language Models like GPT [11] and LLaMA [13]. These models used the same transformer architecture, but scaled to billions of parameters, achieving surprising generalized performance. While Large Language Models achieve impressive performance, their large number of parameters leads to large compute and memory requirements for inference. Several methods have been developed to reduce the size of these models while having minimal impact on performance. While these techniques applied individually attempt to maximally preserve accuracy, there has been little in the direction of trying to find a balance between model accuracy with inference throughput, memory footprint, and performance to VRAM usage. These are essential considerations when looking to reach ubiquity in local LLM hosting, especially considering that 8 GB of VRAM is the median in terms of home computers with dedicated GPUs as per the April 2024 Steam survey.

Quantization involves reducing the number of bits used to represent the model weights. It can be divided into mainly two techniques: Quantization Aware Training (QAT) and Post Training Quantization (PTQ). In Quantization Aware Training the LLM is trained with Quantization in mind and the model and the LLM uses quantized representations of weights during training itself. In Post Training Quantization, the parameters of an LLM are quantized after training and might require some fine-tuning to overcome performance loss. It can also be divided into Offline Quantization where the quantized weights are stored before the model is loaded, and Online/Runtime Quantization where quantized weights are calculated dynamically at runtime when the model is loaded.

Pruning can be considered as "trimming the hedges" of a model, resulting in a size reduction. Edges or nodes are completely removed from the computation graph, however, there is a distinction in the taxonomy based on the criteria used for trimming. Pruning at a high level can be divided into Structured [4] and Unstructured [15] Pruning. Unstructured Pruning prunes the model without any regard for its structure, resulting in an irregular sparse representation and requires specialized compression techniques for storage. Structured Pruning removes entire components such as neurons, channels and layers from the model while maintaining the overall structure of the LLM.

3 Methodology

We focus our attention on a few techniques, as evaluating combinations of approaches will prove to be prohibitively expensive. For example, even just applying a fast quantization technique once takes half a dozen hours to complete on an Nvidia A100 or A6000. In conjunction with multiple quantization and pruning techniques, limiting scope becomes paramount.

3.1 GPTQ [5]

GPTQ is a post-training quantization offline technique where each row of the weight matrix is quantized independently to find a version of the weights that minimizes quantization error. These weights are quantized to 4-bit integers, but they're restored to 16-bit floating point on the fly during inference. This saves memory usage by 4 times because the integer weights are dequantized in a fused kernel. We also expect a speedup in inference as using a lower bit width takes less communication time. Figure 1 shows the working of GPTQ.

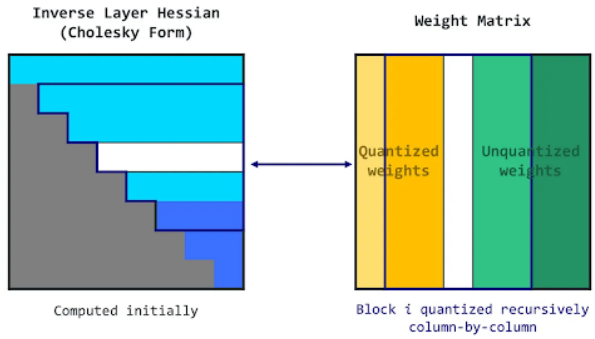

Figure 1: GPTQ

3.2 Activation Aware Weight Quantization [9]

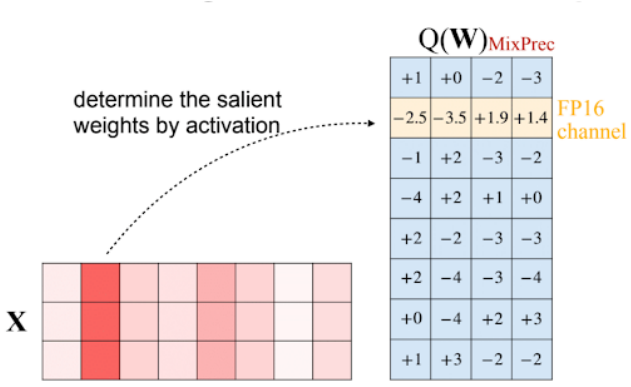

Figure 2: Activation Aware Weight Quantization

Activation Aware Weight Quantization (AWQ) is another offline quantization technique. It is an effective method for low-bit weight-only LLM compression, and it stems from the observation that not all weights are equally important in LLMs, with potentially needing to only protect roughly 1% of parameters. It performs per-channel activation aware scaling to reduce the quantization loss of salient weights. AWQ does not over-fit the validation set and preserves generality. Figure

2 shows the working of Activation Aware Weight Quantization.

3.3 NormalFloat Quantization [2]

NormalFloat (NF) Quantization was introduced as a part of the QLORA Algorithm for LLMs. It first normalizes the weights to have zero mean and unit variance. Weight ranges are then mapped into 4-bit values for storage. During runtime, these are transformed back into the original weight ranges for use in the model. Figure 3 shows the working of NormalFloat Quantization.

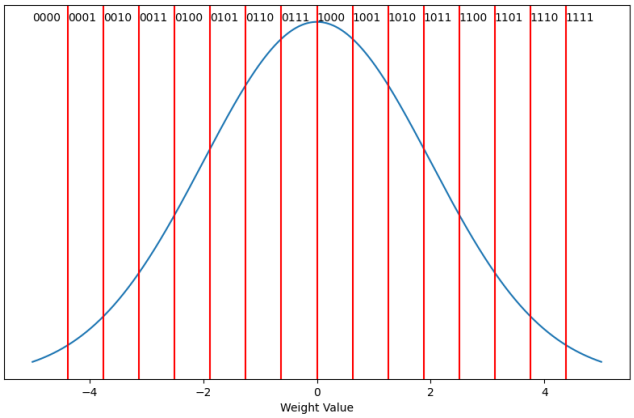

Figure 3: NormalFloat Quantization

3.4 LLM.int8() Quantization [1]

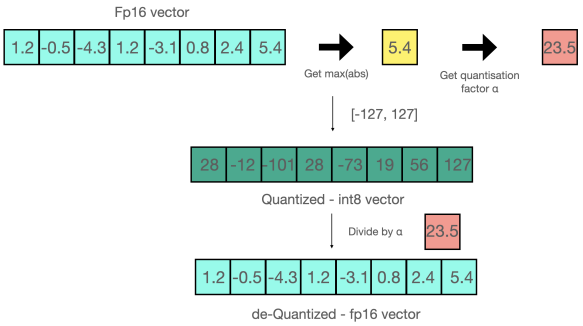

Figure 4: LLM.int8() Quantization

LLM.int8() Quantization multiplies outliers in 16-bit floating point with non-outliers in int8, before converting non-outlier values back to 16-bit floating point and adding them together to finally return the weights in FP16. This reduces the degradative effect outlier values have on a model's performance, with no on-disk storage reduction. Figure 4 shows the working of LLM.int8() Quantization.

3.5 Gradient-Based Language Model Pruning

Gradient-Based Language Model (GBLM) Pruning is a sparsity-centric method for pre-trained LLMs. It leverages the first-order term in the Taylor expansion of the optimal brain surgeon framework, operating in a training-free manner by harnessing normalized gradients from a few calibration samples to determine the pruning metric. Figure 5 shows the working of GBLM Pruning.

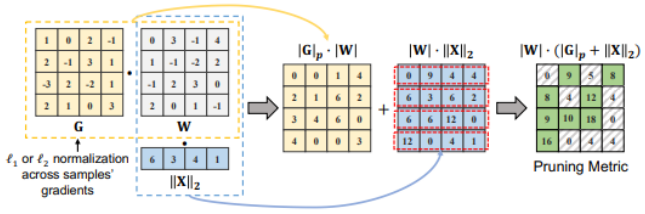

Figure 5: Gradient-Based Language Model Pruning

4 Evaluation Suite

We use the LLaMA 7B Chat model as the baseline for our evaluations. We compare this base model with the following five compressed models - 4-bit LLaMA 7B Chat quantized using GPTQ, 4-bit LLaMA 7B Chat quantized using NormalFloat Quantization, 4-bit LLaMA 13B Chat quantized using GPTQ, 4-bit LLaMA 13B Chat quantized using AWQ and finally 8-bit LLaMA 7B Chat quantized using LLM.int8() followed by GBLM Pruning.

All of the evaluations were done using a system with an RTX 4070 GPU with 8GB VRAM. We used the A100 GPU provided by the ASU Sol Supercomputer to evaluate the base LLaMA 7B model as it would not fit on an 8GB GPU. Key performance metrics such as the time taken by the models for evaluation on benchmark datasets, GPU VRAM utilization and compression ratio of the algorithms are captured. These metrics combined with the performance on benchmark datasets provide a holistic comparison of different compression techniques and their effect.

We evaluate these models on the following three standard NLP benchmarks.

4.1 WikiText-2 Perplexity

The Wikitext-2 dataset consists of over 100 million tokens gathered from the set of Good and Featured articles from Wikipedia. Perplexity is a measure of how uncertain a model is in predicting the distribution of tokens in a corpus. Low perplexity scores indicate that the model is better at predicting sequences in that dataset.

	LLaMA Chat 7B				LLaMA Chat 13B	
	Base	**4-bit NF**	**4-bit GPTQ**	**LLM.int8() + GBLM**	**4-bit GPTQ**	**4-bit AWQ**
Size on Disk (MB)	13803.52	13312	**3788.8**	13312	6963.2	6963.2
GPU VRAM Use (MB)	13803.52	**4856**	4786	7522	7770	7772
WikiText-2 Perplexity	95.74	94.23	118.11	114.62	86.077	**61.01**
WikiText-2 Time (s)	1606.26	1100	892.5	**651.83**	1687.87	2790.1
MMLU Score	0.3428	0.4646	0.4439	0.3428	0.5221	**0.529**
MMLU Time (s)	522.92	552.9	480.33	394.53	868.85	634.73
BBH Score	0.3741	0.3629	0.3518	0.3148	**0.4481**	-

Table 1: Compressed Models Evaluation Results

4.2 Massive Multitask Language Understanding [6]

The Massive Multitask Language Understanding (MMLU) Dataset evaluates models in zero-shot and few-shot settings to benchmark the knowledge acquired during pre-training. It consists of 57 subjects covering fields such as STEM, the humanities, the social sciences, and more. Due to limited resources, we evaluate on a 50% subset of the dataset, with examples chosen randomly.

4.3 BIG-Bench Hard [12]

BIG-Bench is a collaborative suite of over 200 benchmarks that focuses on tasks believed to be beyond the capabilities of current Large Language Models. BIG-Bench Hard is a subset consisting of 23 tasks where prior language models were unable to outperform humans. Similar to MMLU, due to resource constraints, we evaluate on 10% of this dataset

5 Results and Discussion

Table 1 contains the results obtained for the evaluation suite mentioned in the previous section. In general, we observe that compressed models take up less GPU VRAM and have a faster inference time, as expected. The compressed models also obtain reduced scores on the benchmark dataset, the magnitude of reduction varying based on the algorithm used. Comparatively, NormalFloat Quantization performs the best, even beating the base model in some tasks, while LLM.int8() with GBLM Pruning has the most degradation in performance.

Another key observation is that the quantized versions of the larger LLaMA 13B Model performed the best across all benchmarks, beating the Base LLaMA 7B model while also having less VRAM usage. However, their inference speed was slower than the base LLaMA 7B model.

Figure 6 plots the VRAM usage against benchmark time for all models. Even though the LLM.int8() with GBLM model had the lowest score on the benchmarks, it has the fastest inference speed, thus making this method enticing for latency-critical applications. Both the GPTQ and NormalFLoat models have similar VRAM usage and Inference Speed. As expected, the quantized versions of the 13B model were the largest and slowest (but still smaller than the base 7B model).

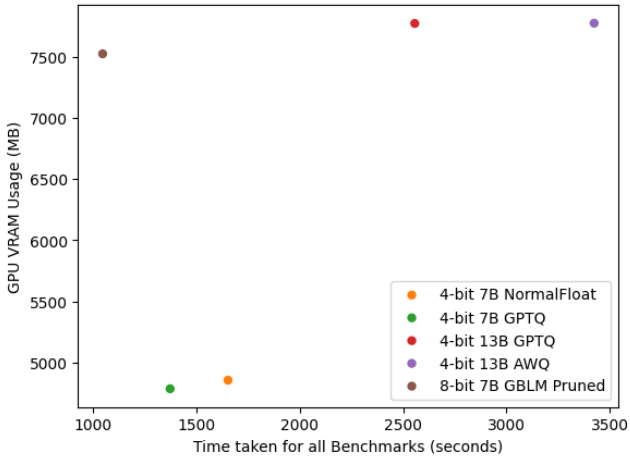

Figure 6: GPU VRAM Usage (MB) vs Benchmark Time (s)

We also plot the compression ratio of the algorithms in Figure 7, which was calculated by dividing the base model VRAM usage by the quantized model VRAM usage. An interesting trend to note is that the compression ratio is higher for the 13B Quantized models, indicating that we might be able to achieve a larger reduction in VRAM usage for bigger models.

On average, the NormalFloat Quantization algorithm performed the best, with minimal loss in accuracy and an impressive reduction in VRAM usage and inference time. Quantization in conjunction with pruning seemed to reduce model performance quite a bit but also obtained impressive gains in inference speed, indicating the usefulness of pruning algorithms in improving latency.

6 Conclusion and Future Work

We compared and contrasted several compression techniques for LLMs using the popular LLaMA 2 7B and 13B models for our evaluations. Each technique tested has its advantages

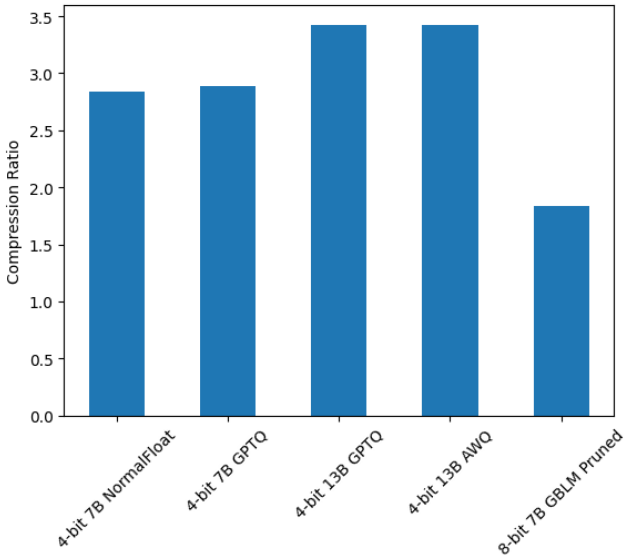

Figure 7: Compression Ratios

and disadvantages, with NormalFloat Quantization being the best on average, and pruning being the best for increasing inference speed. We also noticed that compression versions of bigger models outperformed the uncompressed versions of smaller models in benchmark scores and VRAM usage. Thus in VRAM-constrained environments, we suggest using compressed versions of bigger models over uncompressed smaller models.

In the future we plan to evaluate more models and compression techniques as we were limited this time due to resource constraints. One avenue could be exploring the effect of Knowledge Distillation techniques [7]. A thorough study on the effects of combining different compression methods like Quantization, Pruning and Knowledge Distillation would also prove useful, as we currently only evaluated one model that combined Quantization and Pruning.

References

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

[2] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms, 2023.

[3] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Fang, G., Ma, X., Song, M., Mi, M. B., and Wang, X. Depgraph: Towards any structural pruning, 2023.

[5] Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023.

[6] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding, 2021.

[7] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015.

[8] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022.

[9] Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024.

[10] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach, 2019.

[11] Radford, A., and Narasimhan, K. Improving language understanding by generative pre-training.

[12] Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022.

[13] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023.

[15] Zhang, T., Ye, S., Zhang, K., Tang, J., Wen, W., Fardad, M., and Wang, Y. *A Systematic DNN Weight Pruning Framework Using Alternating Direction Method of Multipliers*. Springer International Publishing, 2018, p. 191–207.

[16] Zhu, X., Li, J., Liu, Y., Ma, C., and Wang, W. A survey on model compression for large language models, 2023.