Methods of Reducing Computational Requirements for Large Language Models

Oleksandr Kononov *

* *South East Technological University, Cork Road, Waterford, Ireland (e-mail: 20071032@mail.wit.ie).*

Abstract: TODO

Keywords: Artificial intelligence, Neural networks,

1. INTRODUCTION

1.1 Background

A Large Language Model (LLM) is neural network model which is capable at working with natural language tasks such as text generation, tmeasurmentext summarization, translation, and more. The novel Transformer architecture proposed in the "Attention Is All You Need" paper [22] has revolutionized the field by introducing more efficient multi-headed self-attention mechanism compared to Recurrent Neural Networks that came before. With the improvements in training times, better parallelization on GPUs and overall quality of output. Following this, OpenAI have used this novel Transformer architecture to design and develop their Generative Pre-Trained Transformer (GPT) LLMs the following years. In particular, the release of GPT-3 in 2020, has sparked a global interest in the continued development of LLMs from various companies such as Meta with LLaMa, Google with Gemma, Antropic with Claude, etc.

The AI researchers at Meta have developed a series of open-weight LLMs called LLaMa, ranging from 7B parameters to 65B parameters [20]. Their paper demonstrates, using various benchmark tests, we can draw a correlation between the increase in LLMs parameters and the scores that it can achieve on various benchmark tests such as HellaSwag [23], WinoGrande [17], ARC [2] and OpenBookQA [15]. There are challenges with regards to the computational requirements necessary for inferencing LLMs,"the compute and memory requirements of state-of-the-art language models have grown by three orders of magnitude in the last three years, and are projected to continue growing far faster than hardware capabilities" [1, p. 97].

Quantization and pruning are some of the strategies that can be used to help reduce computational requirement and memory footprint of LLMs. However applying these strategies often comes at the cost of increasing the LLM Perplexity (PPL), a metric for evaluation the uncertainty of a model in predicting a sequence. Quantization methods involve reducing the numerical precision of the model's weights, allowing 32-bit floating point value to be represented as an 8-bit floating point value or even lower. Popular quantization include GPT-Generated Unified Format (GGUF)[4, 5], Activation-Aware Weight Quantization (AWQ)[12], Vector Post-Training Quantization (VPTQ)[13] and others. Pruning involves removing parts of the model that have little effect on the output, this process could involve removing neurons or entire layers.

1.2 Problem Statement and Motivation

As mentioned in the previous section, the hardware requirements for medium to large sized LLMs make it difficult for consumers or small organisations to run their own local LLMs, often requiring to use third-party providers for access to modern powerful LLMs. This potentially reduces their privacy, security and accessibility, which could be otherwise achieved by running LLMs locally on their own hardware. For large businesses who might already be hosting their own models, this could be an opportunity to potentially reduce their running costs with regards to LLMs.

If in the future, smaller LLMs become more capable than they are today, it stands to reason that their larger counterparts would likewise become more capable. Therefore, finding efficient and cost effective methods of reducing hardware requirements for running large LLMs is holds meaningful significance to this researcher.

1.3 Research Scope and Limitations

This research will be using a select few foundational LLMs for testing and evaluation. The selected LLMs will be Gemma2 9B from Google[18], LLaMa 3.1 from Meta 8B [3] and Qwen2.5 7B from Alibaba[19]. These models were selected due to their research permissive licenses, LLM community popularity and reputability of the companies that have trained them.

The hardware for conducting this research will be limited to a single Nvidia RTX 4090 GPU with 24GB of VRAM, which will be sufficient to run the selected LLMs without any quantization. This will allow the researcher to establish baseline metrics and benchmark scores pre-quantization, which can be used to compare against post-quantized results of the selected LLMs.

The hardware used to demonstrate the effects and performance of LLM quantization will be a single Raspberry Pi 4b, which has quad-core Cortex-A72 @ 1.5GHz CPU and

8GB LPDDR4 RAM [16]. This device was chosen for it's limited hardware specifications, that should under normal circumstances be insufficient run any of the selected LLMs previously mentioned. As such, it qualifies to be a test device for this papers research and would serve as a baseline for future research evaluating more capable devices.

This research will be limited to exploring Post Training Quantization (PTQ) methods and not Quantization Aware Training (QAT) methods, as the later requires significant computational resources and time in order to carry out such research. PTQ methods are significantly less computationally intensive to perform on pre-trained LLMs and require less time to quantize LLM weights.

1.4 Research Questions

This paper aims to answer the following Research Questions (RQ):

RQ1: What quantization methods (GGUF, AWQ, VPTQ) are the most effective for reducing selected LLMs inferencing requirements while retaining most of it's output quality?

RQ2: What pruning strategies (block-wise, channel-wise, layer-wise) are the most effective for reducing selected LLMs inferencing requirements while retaining most of it's output quality?

RQ3: Deriving the best performing method (or combination of methods) from the previous questions, what is the best achievable performance of selected LLMs on a Raspberry Pi 4b?

2. PRELIMINARY LITERATURE REVIEW

This section will explore the supporting research literature that will cover the topics of various PTQ methods, pruning methods and what kind of benchmark tests exist for LLM evaluation that are used in the industry.

2.1 GGUF PTQ

One of the most well known LLM quantization method within the open-source/open-weight LLM community is GGUF. The initial library GPT-Generated Model Language (GGML) was developed by Georgi Gerganov in 2022, similar to other machine learning libraries such as PyTorch and Tensorflow, GGMLs purpose was to provide a minimal, lightweight and efficient tool for on-device LLM inference [11]. The successor to GGML format is the GGUF format which aimed to improve upon the previous GGML format by adding support for format versioning, model metadata and support for other architectures [6] (see figure 1).

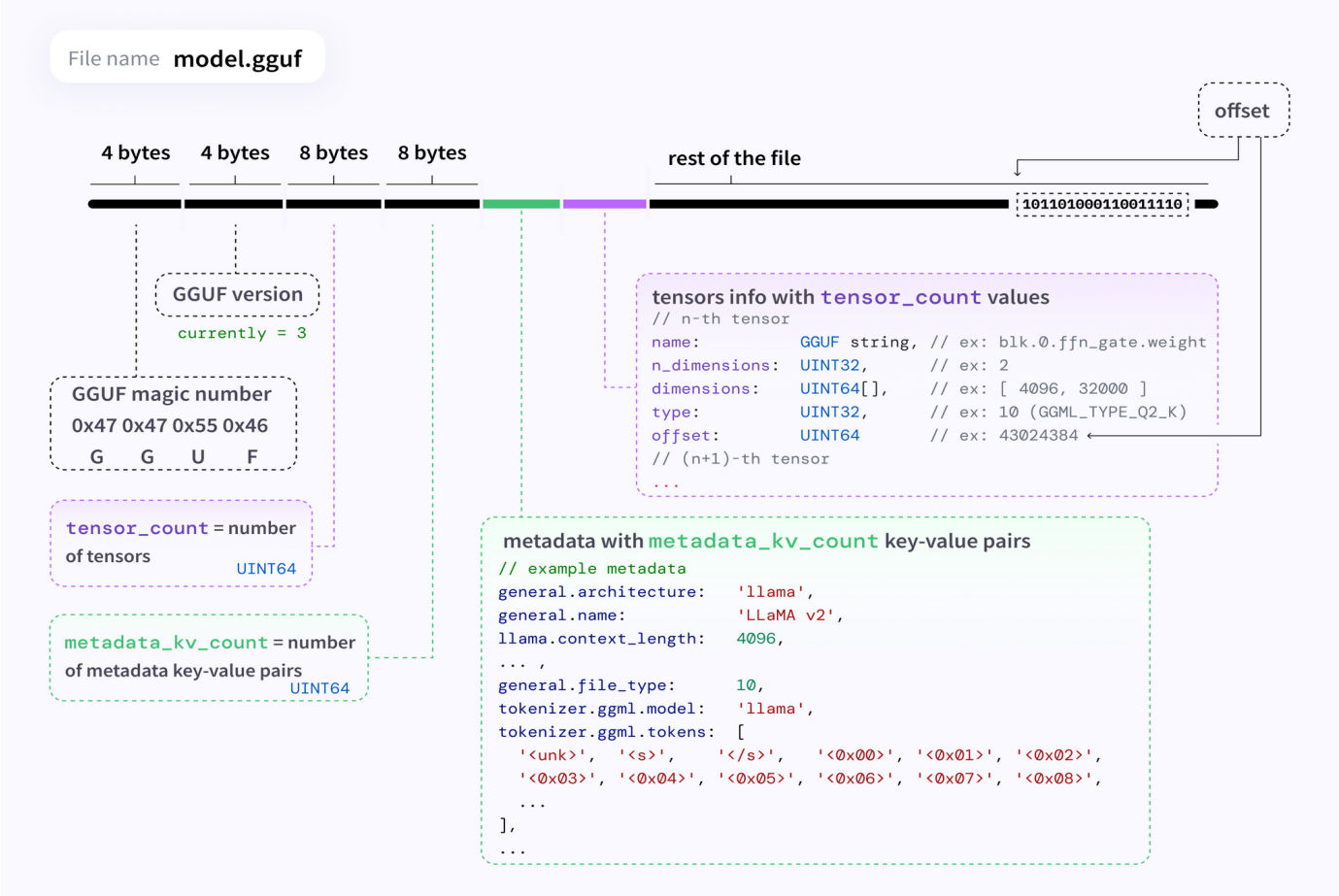

Fig. 1: GGUF Format Breakdown [6]

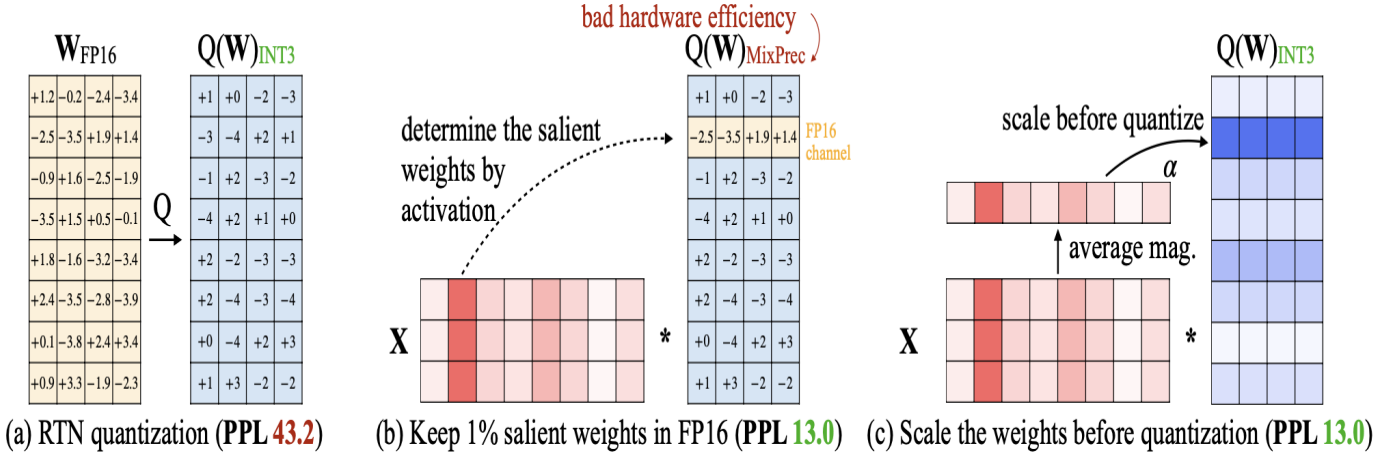

Fig. 2: AWQ method showing how scaling *salient* weights before quantizations shows comparable results to mixed precision weights but without the hardware inefficiencies mix precision introduces. [12]

The `llama.cpp` software developed by Georgi Gerganov and the open-source community is used to load the GGUF format. After it is loaded and processed based on the stored metadata, the underlying GGML library can then begin inferencing the LLM [7]. The GGUF format offers various quantization options, originally the quantization method was to simply split each layer into blocks of 256 weights and each block is then converted into their 256 quantized values, this was demoted with a **Q**. Additionally, there are two quantization type, "type-0" (**Q4_0**, **Q5_0**, etc) where weights w are calculated from quants q and block scale d using $w = d*q$. While "type-1" (**Q4_1**, **Q5_1**, etc) include an additional block minimum constant m such that $w = d * q + m$ [8, 10].

Later the community have developed the **K** Quants, which expand the quantization range to include 2-bit, 3-bit and 6-bit quantization and includes prioritization for certain weights over others (as denoted by suffixes like **Q3_K_S**, **Q3_K_M**, **Q3_K_L**) which leads to smaller models with less PPL increase [10]. The newest form of GGUF quants are the **I** Quants which added a new low bit quantization [9] which is based on the QuIP# paper [21], which suggests grouping even number of either positive or negative signed values into 8 quants, allowing sign information to be recorded using only 7 bits and if needed flipping the sign of the least important quants to maintain even count, while the magnitude of the 8 quant groups can be stored in a E8 lattice structure using 8 bits to record the grid point index [9].

2.2 AWQ PTQ

Similar to the **K** Quants of GGUF, the AWQ method proposes a similar approach of selectively quantizing LLM weights depending on their importance by analysing the model activation patterns [12]. This method identifies *salient* weights in the LLM that hold more importance to the LLM performance and withhold quantization for those weights, thereby avoiding significant performance degradation while reducing the model size. However, having mixed precision weights is not hardware-efficient, it was found that scaling the weights before quantization mitigates this issue while still preserving the benefits (see figure 2)

2.3 VPTQ PTQ

Similar to the **I** Quants of GGUF, the VPTQ method is a low bit LLM quantization method that claims better accuracy and for 1-2 bit LLM quantization compared to other conventional methods. It tries to achieve this by compressing vectors into indices by using lookup tables with additional refinement of weights using Channel- Independent Second-Order Optimization [13]. It differs from the previously covered AWQ method as instead of reducing precision of the weights, it builds an index that maps high-dimensional vectors to lower-dimensional vectors. According to the graph on figure 3, it's performance is comparable to QuIP# method which the **I** Quants of GGUF are based on.

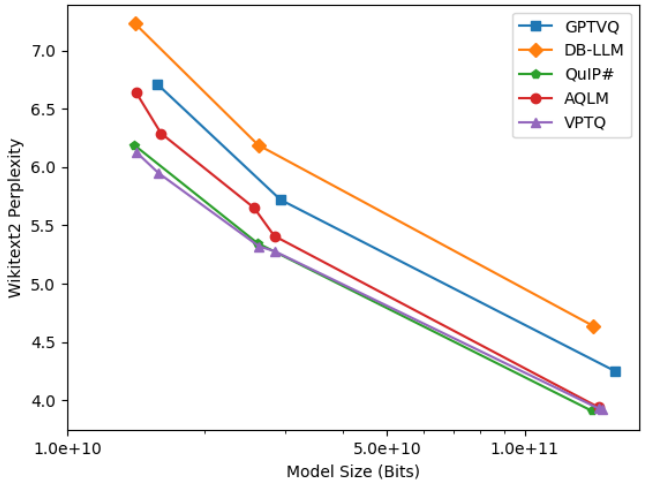

Fig. 3. Graph showing PPL of Wikitext2 dataset compared to model size using various PTQ methods [14]

2.4 LLM Pruning

2.5 LLM Benchmarking

3. WORKING THEORY

4. RESEARCH DESIGN

4.1 Introduction

4.2 Design

5. CONCLUSION

REFERENCES

[1] Rishi Bommasani, Drew A. Hudson, et al. On the opportunities and risks of foundation models, 2022. https://arxiv.org/abs/2108.07258.

[2] Peter Clark, Isaac Cowhey, et al. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. https://arxiv.org/abs/1803.05457.

[3] Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. https://arxiv.org/abs/2407.21783.

[4] Georgi Gerganov. ggml, 2022. https://github.com/ggerganov/ggml.

[5] Georgi Gerganov. llama.cpp, 2023. https://github.com/ggerganov/llama.cpp.

[6] Georgi Gerganov. Ggml, 2024. https://github.com/ggerganov/ggml/blob/master/docs/gguf.md.

[7] Georgi Gerganov. Gguf, 2024. https://github.com/ggerganov/ggml.

[8] Georgi Gerganov. Gguf quantize, 2024. https://github.com/ggerganov/llama.cpp/tree/master/examples/quantize.

[9] Georgi Gerganov and Kawrakow. Gguf i quants pull request, 2024. https://github.com/ggerganov/llama.cpp/pull/4773.

[10] Georgi Gerganov and Kawrakow. Gguf k quants pull request, 2024. https://github.com/ggerganov/llama.cpp/pull/1684.

[11] Georgi Gerganov and Xuan Son Nguyen. Introduction to ggml, 2024. https://huggingface.co/blog/introduction-to-ggml.

[12] Ji Lin, Jiaming Tang, et al. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024. https://arxiv.org/abs/2306.00978.

[13] Yifei Liu, Jicheng Wen, et al. Vptq: Extreme low-bit vector post-training quantization for large language models, 2024. https://arxiv.org/abs/2409.17066.

[14] Microsoft. Vptq: Extreme low-bit vector post-training quantization for large language models, 2024. https://github.com/microsoft/VPTQ/blob/main/README.md.

[15] Todor Mihaylov, Peter Clark, et al. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. https://arxiv.org/abs/1809.02789.

[16] Raspberry Pi Ltd. *Raspberry Pi 4 Model B*, 4 2024. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf.

[17] Keisuke Sakaguchi, Ronan Le Bras, et al. Winogrande: An adversarial winograd schema challenge at scale, 2019. https://arxiv.org/abs/1907.10641.

[18] Gemma Team, Morgane Riviere, et al. Gemma 2: Improving open language models at a practical size, 2024. https://arxiv.org/abs/2408.00118.

[19] Qwen Team. Qwen2.5: A party of foundation models, September 2024. https://qwenlm.github.io/blog/qwen2.5/.

[20] Hugo Touvron, Thibaut Lavril, et al. Llama: Open and efficient foundation language models, 2023. https://arxiv.org/abs/2302.13971.

[21] Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks, 2024. https://arxiv.org/abs/2402.04396.

[22] Ashish Vaswani, Noam Shazeer, et al. Attention is all you need, 2017. https://arxiv.org/abs/1706.03762.

[23] Rowan Zellers, Ari Holtzman, et al. Hellaswag: Can a machine really finish your sentence?, 2019. https://arxiv.org/abs/1905.07830.

Appendix A. APPENDIX A