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, text summarization, translation, and more. The novel Transformer architecture proposed in the "Attention Is All You Need" paper [26] has revolutionized the field by introducing more efficient multi-headed self-attention mechanism compared to Recurrent Neural Networks that came before. Following this, OpenAI have used this novel Transformer architecture to design and develop their Generative Pre-Trained Transformer (GPT) LLMs in 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 [24]. Their paper has demonstrated, using various benchmark tests, that we can draw a correlation between the increase in the amount of LLMs parameters and the scores that it can achieve on benchmark tests such as HellaSwag [29], WinoGrande [21], ARC [2] and OpenBookQA [19]. However, this presents a number of challenges, especially with regards to computational requirements necessary to inference these large 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)[5, 6], Activation-Aware Weight Quantization (AWQ)[14], Vector Post-Training Quantization (VPTQ)[16] and others. Pruning involves removing parts of the model that have little effect on the output, this process could involve removing blocks 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 in helping to democratize access to powerful LLMs.

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[22], LLaMa 3.1 from Meta 8B [4] and Qwen2.5 7B from Alibaba[23]. 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 [20]. 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.

With regards to benchmark and evaluation tests, there exists a large pool of datasets curated for various purposes. This research will select a subset of popular and often used datasets from the categories such as **General Knowledge and Language Understanding**, **Reasoning Capabilities**, **Truthfulness** and **Instruction Following**.

1.4 Research Questions

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

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 [12]. 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 [7] (see figure 1).

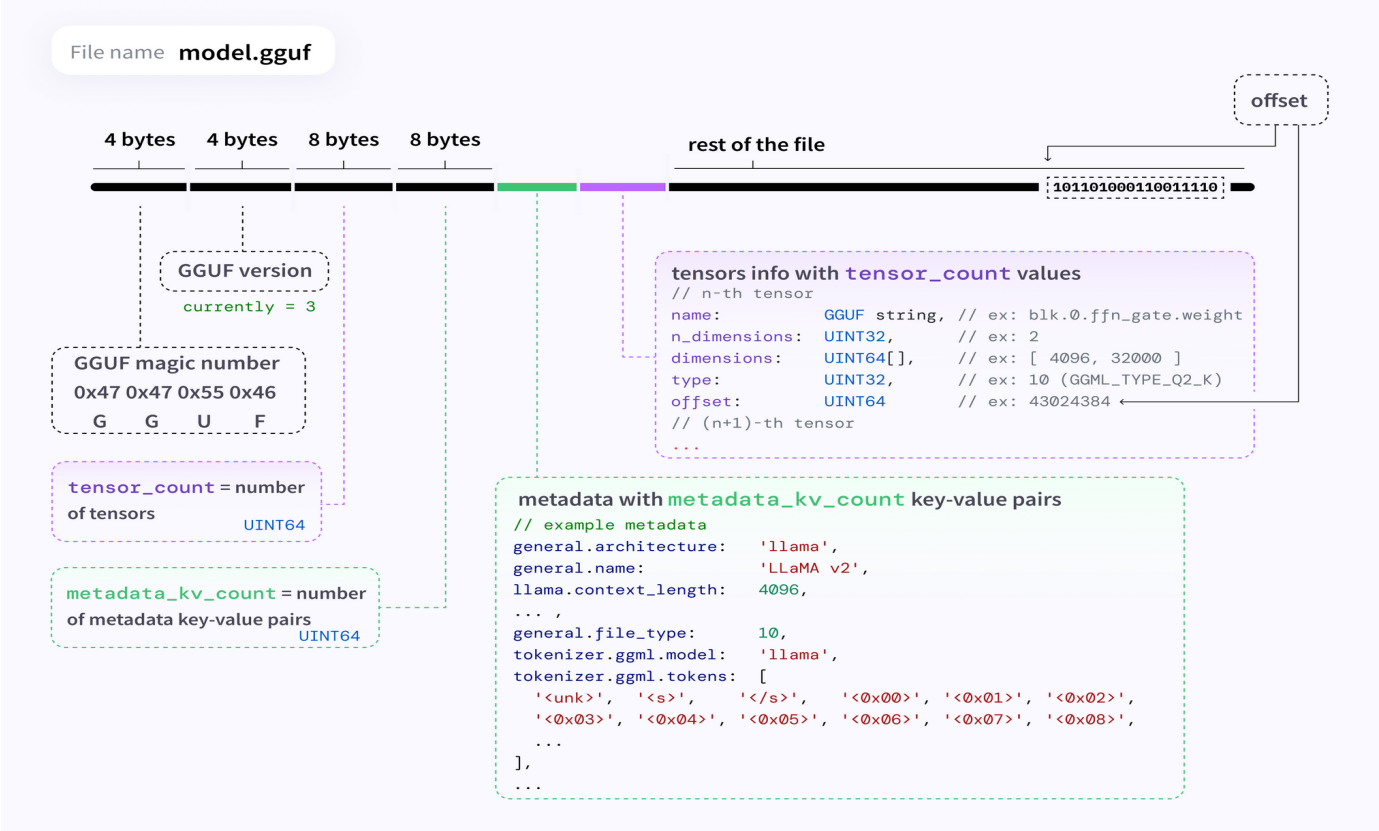

Fig. 1: GGUF Format Breakdown [7]

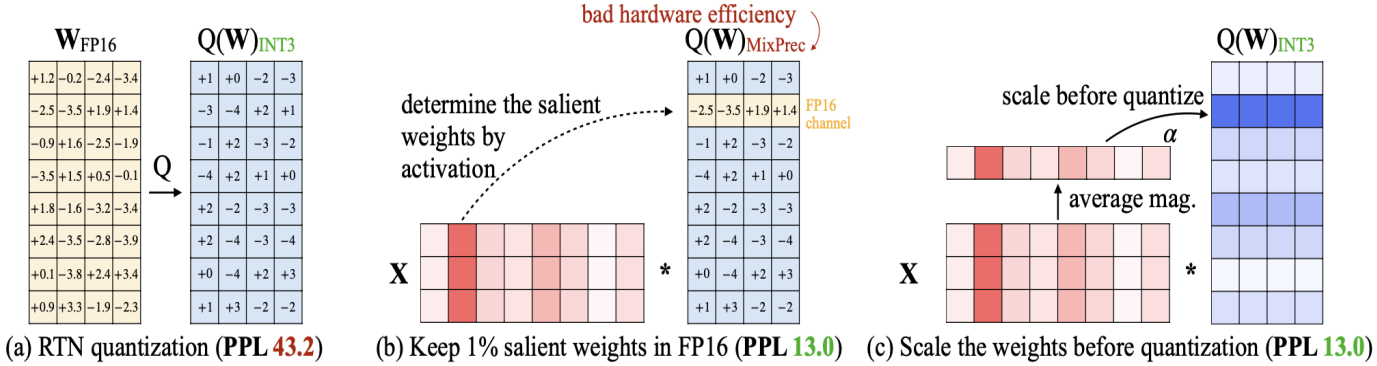

Fig. 2: AWQ method showing how scaling *salient* weights before quantizations shows comparable results to mixed precision weights but without the hardware inefficiencies mix precision introduces [14]

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 [8]. 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 $w = d * q + m$ [9, 11].

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 [11]. The newest form of GGUF quants are the **I** Quants which added a new low bit quantization [10] which is based on the QuIP# paper [25], 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 [10].

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 [14]. 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 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 [16]. 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.

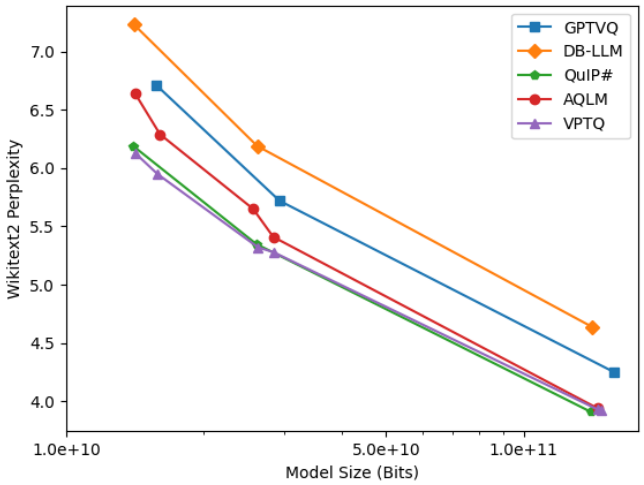

Fig. 3. Graph showing PPL of Wikitext2 dataset compared to model size using various PTQ methods [18]

2.4 LLM Pruning

Model pruning is a way of compressing the size of a model by removing certain components from the model while avoiding severe damage to how the model functions. This is achieved by removing redundant or unimportant singular or groups of neurons [13]. There are two type

of pruning methods that can be performed on a model, structured pruning and unstructured pruning, the former involves simplifying the model by removing entire structural weights such as channels or layers while maintaining the network structures, while the latter focuses on removing redundant neurons or links [13]. Between the two approaches, structured pruning has much better hardware compatibility compared to unstructured pruning which maybe require additional software or hardware treatment to complete the task [13].

LLM-Pruner is a tool developed to perform structural pruning on an LLM using gradient-based optimization processes for structure selection [17]. The LLM-Pruner uses an algorithm to detect dependencies within a model, this allows them to select optimal structured groups for pruning based on their importance estimation. Like with most pruning methods, a post pruning re-training is required, which can be called a "recovery" phase, where it's trained on a small dataset in order to help maintain it's performance [17].

Unlike LLM-Pruner, the Bonsai pruning method does not use a gradient-based structured pruning and is designed for more typical consumer grade hardware to execute [3]. It instead generates sub-models and evaluates their performance to give a more holistic view, which claim to outperform State Of The Art (SOTA) "gradient-based structured pruning methods like LLM-Pruner" [3, p. 2] on 4/6 evaluation tests. Similar to LLM-Pruner, the Bonsai method performs post pruning operations to help recover some lost performance, but unlike LLM-Pruner, it uses distillation from the original model to the pruned model. However, in some cases this post pruning adaptation may not be necessary due to the robustness of the LLM as well as redundancy of its modules [3].

2.5 LLM Benchmarking

There are many benchmark dataset available that are designed to test for various capabilities of LLMs, and as mentioned in the research scope section, this research will focus on a subset of datasets from three categories of **General Knowledge and Language Understanding**, **Reasoning Capabilities**, **Truthfulness** and **Instruction Following**. Starting with Massive Multitask Language Understanding (MMLU) Pro dataset which is an iterative improvement over the original MMLU dataset by removing trivial and noisy questions [27]. It's a dataset with a diverse fields such as mathematics, computer science, physics, chemistry, biology, business and more to produce over 12,000 questions [27]. The dataset is constructed as series of multiple choice questions with up to 10 possible response options, compared to only 4 in the original MMLU dataset, which should reduce random guessing score [28].

Next dataset to look at is HellaSwag, which similar to MMLU part of the **General Knowledge and Language Understanding** category. This dataset evaluates an LLMs capability of answering questions in a contextually appropriate and common-sense manner, referred to as *common-sense natural language inference* [29]. Similar to MMLU the dataset is structured as multiple choice questions.

Moving onto the **Reasoning Capabilities** category, AGIEval dataset uses various human-centric exams to evaluate a models reasoning capabilities [30]. The dataset is constructed with a particular focus on human-level cognitive tasks and real-world scenarios following a wide range of exam papers like mathematics exams, lawyer qualification tests, college admission tests and other qualification exams [30, p. 5]. The format of the dataset is comprised of multiple choice with an addition of some fill-in-the-blank questions that employ exact matching strategy during evaluation [30, p. 6].

For the **Truthfulness** category there exists a TruthfulQA dataset that evaluates a model for providing accurate and unbiased information. The benchmark consists of over 800 questions that covers 38 categories which include law, health, finance and politics [15]. This dataset was designed to help address the concerns of accidental and/or malicious LLM misuse. It relies heavily on true or false questions and uses Wikipedia to source the factual information. The questions in the dataset were designed to be adversarial such that "questions test a weakness to imitative falsehoods: false statements with high likelihood on the training distribution" [15, p. 4]

Finally, for the **Instruction Following** there is the Instruction-Following Eval (IFEval) dataset which was designed to evaluate how well a model follows a set of instructions in the prompt using verifiable instructions. A verifiable instruction is a specific, measurable and objective directive that, upon completion, can be checked and confirmed for adherence [31]. The IFEval has identified 25 types of verifiable instructions and the dataset is comprised of 500 prompts based on those types. An example of such instruct prompt is "write at least 25 sentence" which can be automatically and objectively verified [31]. This dataset is quite valuable as it stands to reason that a model that can follow instructions well will do better at answering multiple choice questions, which most of other previously mentioned datasets use, which require a certain level of structure in the response to be evaluated correctly.

2.6 Summary

This preliminary literature review section has explored some of the existing LLM PTQ methods such as GGUF, AWQ and VPTQ. All of the discussed methods operate based on similar principals of reducing the precision of the numeric values within the model weights. The GGUF, being a community driven open-source project has more options for quantization available to it in the form of **K** quants (which selectively quantize certain weights over others) and **I** quants (which is based on QuIP# method involving clever sign grouping and E8 lattice structure storage). Following from this, the LLM pruning methods LLM-Pruner and Bonsai function by removing parts of the model which has little impact on it's quality. The differing feature between the two approaches lies with it's selection process for the structured groups (gradient-based versus sub-model evaluation). Finally, benchmarking review covered various datasets that evaluate specific model capabilities such as **General Knowledge and Language Understanding**, **Reasoning Capabilities**, **Truthfulness** and **Instruction Following**.

3. WORKING THEORY

Following from the preliminary literature review, this section will define Theoritical Propositions (TPs) based on the previously reviewed literature which will help guide this papers RQs.

TP1: *The effectiveness of PTQ techniques in reducing computational requirements is influenced by the LLM architecture, where certain architectures benefit more than others.*

TP2: *The combination of PTQ and pruning can achieve significant reduction in hardware requirements compared to using only one approach over the other.*

TP3: *The impacts of PTQ and pruning on LLM performance can vary across different types of tasks (e.g. Instruction Following, Reasoning, Knowledge) .*

TP4: *An LLMs reasoning tasks will be more strongly impacted compared to knowledge retrieval tasks after applying a PTQ or pruning method.*

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] Lucio Dery, Steven Kolawole, Jean-François Kagy, Virginia Smith, Graham Neubig, and Ameet Talwalkar. Everybody prune now: Structured pruning of llms with only forward passes, 2024. https://arxiv.org/abs/2402.05406.

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

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

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

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

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

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

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

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

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

[13] Hanjuan Huang, Hao-Jia Song, and Hsing-Kuo Pao. Large language model pruning, 2024. https://arxiv.org/abs/2406.00030.

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

[15] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. https://arxiv.org/abs/2109.07958.

[16] 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.

[17] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models, 2023. https://arxiv.org/abs/2305.11627.

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

[19] 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.

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

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

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

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

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

[25] 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.

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

[27] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark (published at neurips 2024 track datasets and benchmarks), 2024. https://arxiv.org/abs/2406.

01574.
[28] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, et al. Mmlu-pro huggingface, 2024. https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro.
[29] Rowan Zellers, Ari Holtzman, et al. Hellaswag: Can a machine really finish your sentence?, 2019. https://arxiv.org/abs/1905.07830.
[30] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, et al. Agieval: A human-centric benchmark for evaluating foundation models, 2023. https://arxiv.org/abs/2304.06364.
[31] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, et al. Instruction-following evaluation for large language models, 2023. https://arxiv.org/abs/2311.07911.

Appendix A. APPENDIX A