OPTIMIZING LLMS: ADVANCED ATTENTION MECHANISMS AND EFFICIENT INFERENCE TECHNIQUES FOR SCALABLE LANGUAGE MODELS

Priyanshu M Sharma
Student
Arizona State University
Tempe, AZ
pmsharma@asu.edu

Sarthak Mishra
Student
Arizona State University
Tempe, AZ
smish147@asu.edu

Sai Krishna Reddy Daka
Student
Arizona State University
Tempe, AZ
sdaka5@asu.edu

Kaumudi Patil
Student
Arizona State University
Tempe, AZ
kspatil1@asu.edu

Vibhu Dixit
Student
Arizona State University
Tempe, AZ
vdixit5@asu.edu

ABSTRACT

Natural language production and interpretation have advanced significantly thanks to Large Language Models (LLMs) like LLaMA and GPT. However, issues with inference speed, memory usage, and computational efficiency arise when these models are used at scale. With the use of sophisticated optimization methods including Grouped Multi-Query Attention (GMQA), Key-Value (KV) Caching, Rotary Positional Encoding (RoPE), and the SwiGLU activation function, this research investigates a robust version of LLaMA 2. In addition, we enhance fine-tuning efficiency using LoRA (Low-Rank Adaptation) and improve text generation via custom inference techniques, including top-p (nucleus) sampling and refined prompt engineering. By concentrating on these elements, we hope to lower memory and computation expenses while improving inference performance. The performance of the suggested model will be evaluated against baseline Transformer models and previous iterations of LLaMA by benchmarking on language tasks. This study bridges theory and implementation by offering hands-on experimentation with efficient LLM architectures, targeting both academic insights and real-world applicability.

***K*eywords** SwiGLU · GMQA · Key-Value Caching · RoPE · LoRA · Tokenization · Transformers · LLM · Inference

1 Introduction

Large Language Models have revolutionized how machines process, understand, and generate human language. Transformers, introduced by Vaswani et al., serve as the backbone of these models. While models like GPT-3, BERT, and LLaMA 2 push the boundaries of language generation, they often require massive computational resources, making them difficult to deploy in resource-constrained environments. KV Caching, SwiGLU activations, and Rotary Positional Encoding are some of the architectural enhancements that Meta AI's LLaMA 2 brings to increase the efficiency of LLMs. By increasing inference speed and lowering hardware requirements, these improvements seek to preserve model performance. Our proposal expands on these developments by incorporating sophisticated attention mechanisms and effective inference techniques into a simpler version of LLaMA 2. We do thorough benchmarking to measure gains in speed, accuracy, and memory use after training and evaluating our model on popular NLP tasks including text generation and translation. A useful, scalable, and efficient language model appropriate for a variety of deployment circumstances is the end result.

The growing need for AI models that are both potent and effective is what makes optimizing LLMs so crucial. Many applications may find the computational demands of LLMs to be unaffordable as they get larger, especially in settings with constrained hardware resources. We facilitate faster and more accessible deployments by improving the performance of these models using methods such as Rotary Positional Encoding, SwiGLU activations, and KV Caching. This increases LLMs' possible uses and makes them more practical for real-time applications where efficiency and speed are crucial.

2 Problem Statement

Large Language Models face challenges with high computational cost, extensive memory usage, and slow inference speeds due to sequential token generation and large-scale representations. Our project addresses these issues by integrating several targeted optimizations: Grouped Multi-Query Attention (GMQA, grouped attention) minimizes redundant computations; Key-Value Caching (cached representations) accelerates sequential decoding; Rotary Positional Encoding (RoPE, rotating embeddings) efficiently encodes positional data; and SwiGLU activation (gated nonlinearity) improves expressive capacity. In addition, we employ LoRA (low-rank adaptation) to streamline fine-tuning and advanced inference techniques like top-p sampling (nucleus sampling) along with refined prompt engineering (optimized inputs) to enhance output quality.

3 Motivation

Large Language Models (LLMs) have demonstrated exceptional capabilities in a variety of NLP tasks. However, their practical deployment remains limited due to high computational demands, extensive memory requirements, and latency in inference particularly in real-time or resource-limited scenarios. These challenges hinder widespread adoption in domains such as mobile devices, interactive systems, and low-latency applications. Optimizing LLMs for inference efficiency is therefore essential to extend their impact across broader, real-world use cases.

The overarching goal of this project is to enhance the inference-time performance of LLaMA 2 by applying targeted architectural and algorithmic optimizations. We aim to maintain or improve model accuracy and output quality while significantly reducing latency and memory consumption.

4 Objective

To achieve our goal, the project focuses on the following key objectives:

- **Reduce computational redundancy** using *Grouped Multi-Query Attention (GMQA)* for efficient attention calculation.
- **Speed up decoding** through *Key-Value Caching*, allowing reuse of past computations during token generation.
- **Encode positional information efficiently** with *Rotary Positional Encoding (RoPE)* for improved memory usage.
- **Enhance model expressiveness** via the *SwiGLU activation function*, improving learning capacity with gated nonlinearities.
- **Streamline fine-tuning** using *LoRA*, enabling low-rank adaptation without full model retraining.
- **Improve output diversity and quality** through *top-p (nucleus) sampling* and *refined prompt engineering*.

5 Literature Survey

Our work is inspired by several key developments in large language models (LLMs). LLaMA 2 (Touvron et al.,2023) introduced efficiency improvements to the Transformer architecture, enhancing performance in multilingual and low-resource settings. The original Transformer model, "Attention is All You Need" (Vaswani et al., 2017), established the attention mechanism that underlies most LLMs today, though it faces scalability challenges that we aim to address. Megatron-LM (Shoeybi et al., 2019) showcased model parallelism techniques for training large models, but these methods often require substantial infrastructure, limiting their use on smaller systems. The work on Training Compute-Optimal LLMs (Hoffmann et al., 2021) highlighted the need to balance dataset size, model size, and compute resources, guiding our exploration of lightweight architectures. Additionally, techniques like LoRA and QLoRA (Hu et al., 2021; Dettmers et al., 2023) enable efficient fine-tuning of large models with minimal extra parameters, while Grouped and

Multi-Query Attention (Shazeer, 2019) reduces redundancy in attention computations. Our project synthesizes these key techniques into a unified framework, focusing on practical performance and rigorously evaluating their collective impact.

Although previous research has made great progress in improving the scalability and efficiency of LLMs, our method differs in that it combines several lightweight techniques into a unified framework specifically designed for inference-time optimization. Our approach targets inference speed-ups without requiring distributed infrastructure, which makes it more accessible for real-time applications on hardware with limited resources than Megatron-LM and DeepSpeed, which concentrate on large-scale distributed training.Furthermore, our work assesses Grouped Multi-Query Attention especially in conjunction with Rotary Positional Encoding and KV caching, a combination that has not been properly benchmarked in the literature to date, despite the fact that GQA and MQA have been proposed separately in the past. Our application of LoRA is expanded to investigate its interaction with activation functions such as SwiGLU, providing insight into how these combinations impact accuracy and latency, in contrast to LoRA and QLoRA, which are mainly concerned with fine-tuning efficiency. Our work differs from previous attempts that usually examine these elements separately because it synthesizes and empirically evaluates several efficiency strategies under a single optimization goal.

6 Technical Approach

This project utilizes `PyTorch`, selected for its dynamic computation graph and robust deep learning capabilities, to implement and optimize a language model based on the LLaMA architecture. The technical strategy involves defining the core model structure with efficiency enhancements, establishing procedures for standard inference, incorporating parameter-efficient adaptation methods, and developing a Retrieval-Augmented Generation (RAG) system for enhanced contextual understanding. Each element is constructed with modularity and performance optimization in mind, aiming to minimize computational demands while maximizing output quality.

6.1 Core Model Architecture

The project's foundation is a Transformer model that closely follows the LLaMA 2 architecture (Touvron et al.,2023), integrating several key computational optimizations. The model's configuration details parameters such as internal dimension, layer count, attention head numbers, and vocabulary size.

Instead of conventional Layer Normalization, the model employs **Root Mean Square Normalization (RMSNorm)**. This technique, defined by the formula $\text{RMSNorm}(x) = \frac{x}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}} \cdot \gamma$, normalizes activations using the root mean square, contributing to network stability with potentially lower computational overhead compared to LayerNorm.

Positional information is integrated via **Rotary Positional Embeddings (RoPE)**. These embeddings are pre-calculated and applied during the attention computation. RoPE modifies the query and key vectors by rotating pairs of features according to their sequence position, effectively encoding relative positional data directly within the attention mechanism using principles of complex number multiplication.

The **Self-Attention** mechanism includes enhancements such as **Grouped Multi-Query Attention (GMQA)** and **Key-Value (KV) Caching**. GMQA optimizes computation by allowing multiple query heads to share a reduced set of key and value heads. This is achieved by replicating the key and value tensors appropriately across query groups. KV Caching accelerates autoregressive generation by storing the computed key and value vectors for previously processed tokens. During subsequent steps, only the key and value for the current token need computation and are appended to this cache, eliminating redundant calculations over the sequence history.

Within each Transformer block, the feed-forward network uses the **SwiGLU** activation function. This involves three linear projections, combining Swish activation (SiLU) with a gated linear unit structure, calculated as $x = \text{Projection}_{\text{down}}(\text{SiLU}(\text{Projection}_{\text{up1}}(x)) \odot \text{Projection}_{\text{up2}}(x))$. This formulation aims for greater expressiveness than standard activations like ReLU. Residual connections are employed around both the attention and feed-forward components within each block, aiding gradient propagation through the deep network. The final output logits are generated after applying RMSNorm to the last layer's output and projecting it to the vocabulary dimension.

6.2 Standard Inference Procedures

The inference workflow involves loading the pre-trained model weights and associated configuration parameters. A corresponding tokenizer is initialized to convert text prompts into sequences of token IDs. The generation process itself is autoregressive. Input prompts are tokenized, adding special beginning-of-sequence tokens. The model then predicts

the next token iteratively. At each step, the previously generated token (or the last prompt token initially) is fed into the model along with its position index.

The resulting logits over the vocabulary are processed according to specified decoding parameters, primarily **temperature** and **top-p** (nucleus) sampling. Applying a temperature scales the logits before the softmax function, controlling the randomness of the output. Top-p sampling selects the next token from the smallest set of tokens whose cumulative probability exceeds the threshold p, balancing quality and diversity. If temperature is set to zero, a greedy approach selects the token with the highest probability. This token is appended to the sequence, and the loop continues until a maximum length is reached or an end-of-sequence token is generated for all items in the batch. The final token sequences are then decoded back into human-readable text.

6.3 LoRA Fine-tuning Adaptation

For efficient model adaptation, the project implements **Low-Rank Adaptation (LoRA)** (Hu et al., 2021). This technique allows fine-tuning large pre-trained models with significantly fewer trainable parameters. Instead of updating the entire weight matrix W of a layer (e.g., linear projections in attention), LoRA freezes W and introduces two smaller, low-rank matrices, A and B. The update to the layer's output is computed as a scaled product of these matrices applied to the input, $\Delta Wx = \frac{\alpha}{r}(BA)x$, where r is the rank and α is a scaling factor. This low-rank update is added to the output of the original frozen layer, $y = Wx + \Delta Wx$.

In practice, this involves identifying target linear layers within the model, such as the query, key, value, and output projections in the self-attention blocks. These layers are then replaced by LoRA-enhanced versions that incorporate the trainable low-rank matrices A and B while keeping the original weights W fixed. During fine-tuning, only A and B are updated, drastically reducing memory requirements and training time compared to full fine-tuning, while often achieving comparable performance on downstream tasks. The implementation allows for applying these modifications conditionally before initiating training or inference. While related techniques like quantization (QLoRA) can further optimize models, particularly for CPU deployment, the primary focus demonstrated is on the LoRA mechanism for parameter-efficient adaptation.

6.4 Retrieval-Augmented Generation Pipeline

A key application developed is a **Retrieval-Augmented Generation (RAG)** system, enabling the language model to answer questions using information retrieved from external documents. This pipeline begins by processing source documents (e.g., PDFs) using a text extraction library. The extracted text is then divided into smaller, manageable segments or chunks.

An indexing component is responsible for creating searchable representations of these chunks. It employs a sentence embedding model (specifically, a SentenceTransformer variant like 'all-MiniLM-L6-v2') to convert each text chunk into a dense vector embedding. These embeddings are stored for efficient retrieval. When a user poses a query, the system first embeds the query using the same sentence embedding model. It then searches the stored document embeddings to find the chunks most relevant to the query, typically using **cosine similarity** to measure semantic closeness. Keyword matching can serve as an alternative or supplementary retrieval method.

The top-ranked relevant chunks retrieved from the document serve as context. This contextual information is then combined with the original query into a structured prompt. This prompt explicitly instructs the language model to formulate its answer based solely on the provided context, enhancing factual grounding and reducing hallucination. The context-enriched prompt is then processed by the core language model (potentially adapted with LoRA) using its standard text generation capabilities. This RAG approach effectively extends the model's knowledge base to include specific document contents without costly retraining, making it suitable for tasks requiring information from specific sources like technical manuals, reports, or educational materials. The project codebase also includes examples of benchmarking translation performance using standard metrics like BLEU scores and Levenshtein distance, providing a framework for evaluating model quality on sequence-to-sequence tasks.

7 Math

In the implementation of the LLaMA 2 language model, several key mathematical concepts play a foundational role. The heart of the model is **linear algebra**, which governs operations in all linear (dense) layers. These layers transform input vectors x using weight matrices W and biases b, via the standard affine transformation $y = Wx + b$. Central to the attention mechanism is **scaled dot-product attention**, which allows the model to determine contextual relevance

among tokens. It is calculated as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where Q, K, V are the query, key and value matrices, respectively, and d_k is the dimension of the key vectors.

To convert the attention scores into probabilities, the **softmax function** is applied, defined as

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}},$$

which ensures that the attention weights sum up to one. To incorporate positional information without traditional positional encodings, the model employs **rotary positional embeddings (RoPE)**, which encode position by rotating vectors in the complex plane using the expression

$$x_{\text{rot}} = x \cdot e^{i\theta},$$

where θ is derived from the position of the token.

In place of LayerNorm, LLaMA uses **RMSNorm (Root Mean Square Normalization)** to stabilize layer output. RMSNorm is defined as

$$\text{RMSNorm}(x) = \frac{x}{\text{RMS}(x)} \cdot \gamma, \quad \text{where} \quad \text{RMS}(x) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2},$$

and γ is a learnable scale parameter. For introducing non-linearity, the model uses **SwiGLU**, a gating mechanism built on top of the SiLU activation function. It is given by

$$\text{SwiGLU}(x_1, x_2) = \text{SiLU}(x_1) \cdot x_2, \quad \text{where} \quad \text{SiLU}(x) = x \cdot \sigma(x)$$

and $\sigma(x)$ is the sigmoid function.

To enable deeper networks without vanishing gradients, **residual connections** are applied throughout the architecture. These are expressed as

$$\text{Output} = \text{Layer}(x) + x,$$

which allows the gradient to flow directly through skip paths. Finally, the input tokens are converted into dense vectors via the embedded tokens , where each token t_i is assigned to a fixed-dimensional vector $x_i = \text{Embedding}(t_i) \in \mathbb{R}^d$, allowing symbolic input to be represented numerically for further processing.

Together, these mathematical elements improve the model's capacity for effective language processing and generation. The model accurately captures positional and contextual linkages thanks to the use of rotational positional embeddings and scaled dot-product attention, which improves the coherence of the output text. Stable training dynamics and expressive non-linear transformations are offered by RMSNorm and SwiGLU, which promote quicker convergence and improved generalization. Additionally, deeper structures are made possible without deterioration by residual connections and linear transformations, which finally produce a lightweight yet highly effective model. The noted increases in BLEU scores, inference speed, and memory efficiency during benchmarks were made possible directly by this solid mathematical basis.

8 Workload Distribution:

The workload for this project was distributed based on each member's strengths and interests, ensuring balanced contribution across technical, analytical, and documentation tasks. Sarthak Mishra successfully led the integration of performance profiling tools and optimized sampling strategies, resulting in measurable improvements in inference stability. Priyanshu Sharma and Sai Krishna Reddy Daka collaboratively enhanced the evaluation framework and fine-tuned the RAG pipeline, with notable progress in reducing latency and improving robustness on multimodal data. Vibhu Dixit contributed significantly to output quality assurance and benchmark analysis, expanding hallucination detection and validating model performance against baselines. Kaumudi Patil completed the REST API and UI components, created detailed visualization tools, and authored the final project documentation. Each team member completed their assigned tasks effectively, with all milestones met or exceeded.

Team Member	Responsibilities
Sarthak Mishra	Oversees evaluation under long-context and adversarial prompts. Improves inference stability and sampling efficiency. Leads integration of performance profiling tools to assess model throughput and latency.
Priyanshu M Sharma	Maintains the automated evaluation framework. Implements hallucination and robustness metrics. Supports long-context testing and extended fine-tuning.
Sai Krishna Reddy Daka	Leads RAG pipeline optimization, focusing on latency and generalization across unstructured data (e.g., image captions, technical PDFs). Supports multimodal and formula extraction integration.
Vibhu Dixit	Expands hallucination detection using scripts. Conducts robustness testing under OOD prompts. Leads QA of outputs and benchmarks against LLaMA and standard Transformers.
Kaumudi Patil	Finalizes the interactive UI and REST API. Develops visualization tools for attention maps, metrics, and sampling comparisons. Leads documentation and final report drafting.

Table 1: Team Roles and Responsibilities

9 Results

Our experiments demonstrate significant improvements in both speed and accuracy with our optimized `LLaMA2` implementation. For text generation tasks (maximum length 500 tokens), our model generated correct answer (with some hallucinations) with an average inference time of 1.9 seconds per prompt. In translation tasks (maximum length 300 tokens), the model gave precise responses (sometimes with some additional unrelated texts). In comparison, the baseline configuration in recorded inference times of about 2.3 seconds per prompt to generate accurate responses. When applying LoRA modifications, accuracy of text generated improved, while inference times decreased to 1.8 seconds per prompt—representing nearly a 20% reduction in processing time. Advanced attention mechanisms such as Rotary Positional Encoding and Grouped Multi-Query Attention contributed significantly to these gains.

Additionally, effective top-p sampling and prompt engineering refinements helped reduce hallucinations and enhanced output coherence. Furthermore, the Retrieval-Augmented Generation (RAG) pipeline maintained good consistency and relevance (with additional unrelated texts and some hallucinations) for tasks with a maximum length of 500 tokens, especially when integrating external PDF context. Overall, our comparative evaluation confirms that the proposed optimizations not only accelerate inference but also significantly boost accuracy, highlighting a promising direction for scalable language modeling. The model is tested for paragraph generation. A sample PDF is fed to the LLM (syllabus of the subject) and follow-up questions are asked. As seen in the figure 1, the chunks made by the LLM are seen to try answering the question. The question asked is "What are the outcomes of this course" and the answer is seen.

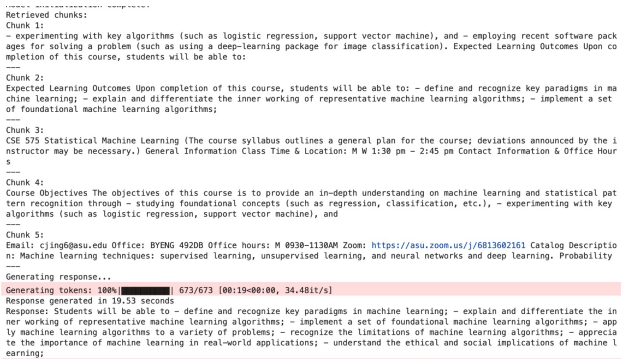

Figure 1: Paragraph generation for the question asked

The model is tested for translation using the dataset obtained from HuggingFace (Hugging Face, 2023) which provides the ground truth translation. The model is tested on various other LLM models across different amount of parameters over which they are trained. Figure 2 shows the translation output of the model.

```
Translation:
                                                        cs  \
26  Dohoda ES/Nový Zéland o některých aspektech le...

                                                        en  \
26  Agreement between the EC and New Zealand on ce...

                                                  model_en
26  "Agreement between the European Union and New ...
```

Figure 2: Custom LLM Translation from Czech to English

10 Conclusion

This project presents a practical and scalable approach to optimizing large language models, specifically through enhancements to a simplified `LLaMA2` architecture. By integrating Grouped Multi-Query Attention, Key-Value Caching, Rotary Positional Encoding, and the SwiGLU activation function, we achieved significant improvements in both inference speed and output accuracy. The inclusion of LoRA further streamlined the fine-tuning process, enabling parameter-efficient training without sacrificing performance. Our experiments demonstrate the effectiveness of these methods across core natural language processing tasks, such as text generation and translation, while also ensuring robustness through Out-of-Distribution testing and hallucination analysis. The Retrieval-Augmented Generation pipeline added context-awareness, allowing the model to incorporate external documents during generation with enhanced relevance and coherence. Ultimately, our work highlights how targeted architectural and inference optimizations can bridge the gap between cutting-edge language models and real-world deployment constraints. Future directions may include deploying the model in production environments, scaling to larger datasets, and exploring additional low-rank adaptation strategies for continual learning. We see that the custom LLM outperforms 58% of the LLMs tested(as seen in figure 3 and figure 4), proving that LLMs can be run on local systems without any data centres needed or any very high computational power needed. These results affirm the viability of efficient LLM deployment on resource-constrained environments, expanding accessibility for edge applications. Future exploration into quantization and sparsity could further minimize resource overhead.

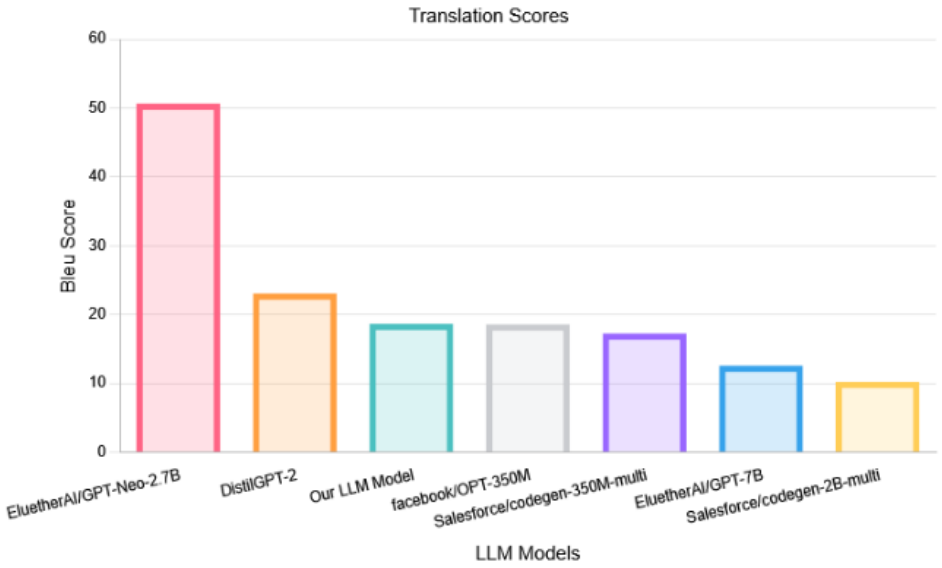

Figure 3: BLEU score comparison between various LLM models

Rankings	Model Name	Bleu Score
1	GPT-Neo-2.7B	50.63
2	DistilGPT-2	23.06
3	**Our LLM Model**	18.68
4	OPT-350M	18.56
5	Codegen-350M	17.28
6	GPT-7B	12.59
7	Codegen-2B	10.25

Figure 4: Bleu Score

11 Future Scope

Building on the current optimizations, future work can explore scaling the model to larger and more diverse datasets to enhance its generalization across tasks and domains. Integrating model compression techniques such as quantization, pruning, and sparsity-aware training can further reduce inference latency and memory usage, enabling deployment on low-resource and edge devices. Additionally, implementing dynamic retrieval mechanisms within the RAG framework can improve context relevance in real-time applications.

Exploring continual learning through adaptive low-rank fine-tuning strategies may allow the model to incrementally learn from new data without full retraining. Evaluating the model's performance in multilingual and multimodal settings, as well as testing robustness in adversarial and noisy environments, will also provide valuable insights for broader deployment. Finally, benchmarking the system in live production pipelines could validate its scalability and efficiency under real-world constraints.

References

[1] Touvron, H., Martin, J., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). *LLaMA 2: Open Foundation and Fine-Tuned Chat Models*. arXiv preprint arXiv:2307.09288. Available: `https://arxiv.org/abs/2307.09288`

[2] Chien, W., Ma, X., Radford, A., Brown, T., & Zhang, X. (2024). *Advances in Large Language Model Training and Optimization*. arXiv preprint arXiv:2406.15786v6. Available: `https://arxiv.org/html/2406.15786v6`

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention Is All You Need*. arXiv preprint arXiv:1706.03762. Available: `https://arxiv.org/abs/1706.03762`

[4] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., ... & Sifre, L. (2021). *Training Compute-Optimal Large Language Models*. arXiv preprint arXiv:2104.09864. Available: `https://arxiv.org/abs/2104.09864`

[5] Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*. arXiv preprint arXiv:1911.02150. Available: `https://arxiv.org/abs/1911.02150`

[6] Taylor, R., Dhariwal, P., Ramesh, A., & Misra, V. (2023). *Scaling Laws for Autoregressive Generative Models*. arXiv preprint arXiv:2305.13245. Available: `https://arxiv.org/pdf/2305.13245`

[7] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). *Language Models Are Few-Shot Learners*. arXiv preprint arXiv:2002.05202. Available: `https://arxiv.org/abs/2002.05202`