Understanding Large Model Inference: Issues, Techniques, and Future Directions

李宇哲 PB21111653

摘要—This article surveys the basic processes of current large model inference, along with the relevant background knowledge. It examines typical scenarios in large model inference research, such as the potential issues encountered in the Prefill and Decode stages, KV-cache, and parallel inference.

Additionally, it addresses four categories of problems in large model inference: the low compute-memory ratio in the Decode stage, the high computational load and memory usage in the Prefill stage for long sequences, the bottleneck issues related to memory capacity and bandwidth in DNN training and inference devices like GPUs and NPUs, as well as the evolving scenarios of large model inference like MoE, Sora, LLM e.g.

By moving from the foundational inference processes to specific state-of-the-art problems, this review aims to gradually establish a complete knowledge framework for the field of large model inference.

***Index Terms*—Large Model Inference, Prefill, Decode, KV-cache, Parallel Inference, Computational Efficiency, Memory Usage, Mixture of Experts (MoE), Multimodal Models, Deep Learning**

I. Introduction

With the rapid development of AI and machine learning, large model inference has become an important research direction, encompassing not only algorithmic improvements but also various system optimization techniques. Among different large models across multiple domains, Transformer-based models have shown remarkable performance in NLP, CV, and other areas, garnering widespread attention. Therefore, the basic processes of large model inference and the key architecture of Transformers are significant subjects of this review.

As model sizes continue to expand, the computational and memory demands during the inference process also increase, leading to numerous challenges, which will be a primary focus of this survey.

This review aims to explore the basic processes of current large model inference and the relevant foundational knowledge, covering the fundamental Transformer architecture, as well as key stages such as prefill and decode, along with important technologies like KV-cache[2] and parallel inference[5] [16]. Additionally, I will analyze the main issues currently present in large model inference, including:

- The low compute-memory ratio problem in the decode stage
- The high computational load and memory usage in the prefill stage for long sequences
- The storage capacity and bandwidth bottlenecks on devices like GPUs and NPUs
- The evolution of different inference scenarios, such as Mixture of Experts (MoE)[8] [17]and multimodal models

II. Fundamentals of Large Model Inference

A. Definition and Characteristics of Large Models

Technically, language modeling (LM) is one of the major approaches to advancing language intelligence of machines.[22] In general, LM aims to model the generative likelihood of word sequences, so as to predict the probabilities of future (or missing) tokens. Researchers find that scaling PLM (e.g., scaling model size or data size) often leads to an improved model capacity on downstream tasks (i.e., following the scaling law [6]).Therefore, large language models have emerged.

These deep learning models, which possess vast parameters and high computational complexity, can handle complex tasks and rely on large-scale datasets for training to capture rich features and patterns. Such models are what we refer to as large models today, including GPT[1], LLaMA[19], Sora[10], and others.

Large models have the following characteristics:

- **Large parameter scale**: Large models typically contain parameters on the order of terabytes (TB) or even larger.
- **High computational resource requirements**: Training and inference require substantial computational resources, usually relying on high-performance GPU or TPU clusters. Moreover, various studies on DSA (Data-Selective Architecture) to accelerate large model training are emerging, such as Huawei's Ascend NPU, which is used to speed up critical operations of large models.
- **High data demands**: To effectively train large models, a large and diverse dataset is usually necessary to ensure the model has good generalization capabilities and does not suffer from overfitting.
- **Strong transfer learning capabilities**: Pre-trained large models can perform exceptionally well on a variety of downstream tasks, often requiring only minimal fine-tuning to achieve good performance on different tasks. Techniques proposed by pre-trained models like GPT and BERT[4] have wide applications in this regard.

B. Transformer Structure

Before discussing the specific process of inference, let's focus on a key structure in large models: the Transformer [20]. Since its introduction in 2017, this structure has been widely adopted in both NLP and CV tasks, with its self-attention mechanism serving as a crucial neural network block during the inference phase.

The basic Transformer model adopts an encoder-decoder architecture. The encoder consists of six stacked layers, each containing a multi-head attention mechanism and a feed-forward neural network, and

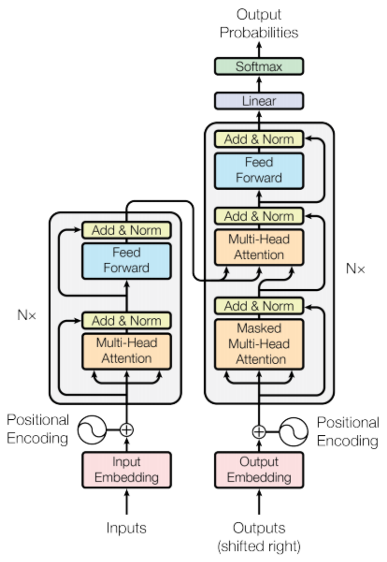

图 1. Transformer Model Structure

operates in a self-regressive manner. The decoder is similarly composed of six stacked layers, each incorporating a masked multi-head attention mechanism followed by residual connections and normalization, and then passing into another multi-head attention mechanism. After being processed through the feed-forward network (FFN), the output is passed through a softmax function to produce appropriate probabilities. For both the encoder and decoder, the input and output are encoded with positional encodings, preserving the contextual relationships of long sequences. Unlike traditional RNN structures, this encoder-decoder architecture does not rely on a single context vector to maintain connections between past and future information, thus avoiding issues related to fixed lengths.

From this structure, we can observe that if we treat the Transformer as a key operator in the inference of large models, this operator involves both element-wise operations such as exponential and addition, as well as reduction operations like softmax, which are typically memory-bound. There are also computation-intensive operations like matrix multiplication. Modern specialized processors may be able to complete element-wise and reduction-type operations in a single clock cycle on vector cores, while the computational pressure primarily comes from matrix multiplications.

Understanding the characteristics of these operators can facilitate deeper insights into optimizing inference, addressing aspects such as computation, storage, and the structure of the Transformer itself.

C. Overview of the Inference Process

The inference process is a key step in generating outputs in large models for practical applications. This process typically includes the following basic stages:

- **Data Input**: At the beginning of the inference stage, the model needs to receive external input data. This data can be in the form of text, images, or other types of information, depending on the application scenario of the model. The quality and format of the input data have a direct impact on the subsequent inference results, so appropriate preprocessing is required.
- **Prefill Stage**: In this stage, the model generates an initial state representation using the input data. Typically, the model employs self-attention mechanisms to capture the relationships between different parts of the input data. For example, in decoder-only models like LLaMA, the attention mechanism is an important operator layer during the prefill stage. By processing the entire input sequence, the model can establish contextual information, preparing for the subsequent decoding stage.
- **Decode Stage**: This stage is the core process of generating outputs. The model typically uses a self-regressive approach to generate the next word or output step by step until a termination condition is met. During the decoding process, the model relies on previously generated outputs while considering the contextual information from the input data to ensure the coherence and consistency of the generated results.
- **Output Generation**: After completing the decoding, the final results generated by the model are converted into a readable format. This may involve transforming the generated sequence into text, images, or other types of output, depending on application requirements. At this point, the model's output can be directly used for downstream tasks such as translation, question answering, image recognition, and more.

Through these stages, the inference process transforms the input data into useful outputs, fully leveraging the capabilities of large models.

D. Key Concepts in Inference

In the realm of large model inference, several key concepts play a crucial role in understanding how models function and perform. Below are some of these fundamental terms:

- **Context Window**: The context window refers to the number of tokens (words or subwords) that a model can process at one time. In transformer models, the size of the context window significantly influences the model's ability to capture long-range dependencies within the input data. A larger context window allows the model to consider more information, which can enhance the quality of the generated outputs but also increases computational demands.
- **Computational Complexity**: This term describes the amount of computational resources required to execute a model's inference process. Computational complexity is often expressed in terms of time complexity and space complexity. Time complexity measures how the execution time of the model grows with the size of the input, while space complexity refers to the amount of memory required. Understanding computational complexity is vital for optimizing model performance and ensuring efficient resource utilization.
- **Memory Usage**: Memory usage indicates the amount of RAM consumed by the model during inference. This includes the memory required to store the model parameters, intermediate computations, and the input data. High memory usage can lead to bottlenecks, especially on devices with limited memory capacity, such as GPUs. Efficient memory management techniques, such as model

pruning and quantization, can help reduce memory consumption without significantly sacrificing performance.

- **Attention Mechanism**: The attention mechanism is a core component of transformer architectures that allows models to focus on different parts of the input sequence when generating outputs. It calculates attention scores based on the relationships between tokens, enabling the model to weigh the importance of each token relative to others. This mechanism is crucial for capturing contextual information and improving the overall quality of inference.
- **Self-Attention**: Self-attention is a specific type of attention mechanism where the input sequence is compared against itself. Each token in the sequence attends to every other token, allowing the model to create a rich representation of the input. This technique enhances the model's ability to understand context and relationships within the data.
- **Batch Processing**: Batch processing refers to the method of feeding multiple input samples into the model simultaneously during inference. This technique can significantly speed up the inference process by leveraging parallel processing capabilities of modern hardware. However, it also requires careful management of memory and computational resources to avoid exceeding device limits.

Understanding these key concepts is essential for grasping the intricacies of large model inference and optimizing the performance of models in various applications.

III. Key Technologies in Large Model Inference

In large model inference, several key technologies play a crucial role, such as the Prefill and Decode stages, the KV Cache used for accelerating inference, and parallel inference techniques.

A. Prefill and Decode

For large models, the inference process can be technically divided into two phases to achieve more efficient reasoning. Researchers have found [15] that the complete inference process can be separated into the Prompt Phase and the Token Generation Phase. The former processes a large chunk of user input, while the latter generates tokens output by token.

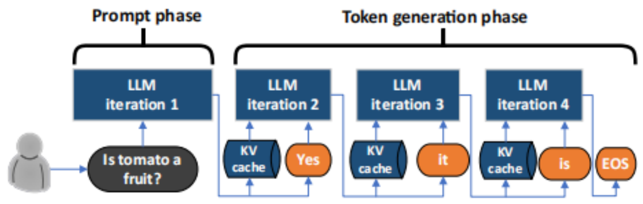

图 2. Prefill & Decode

The differences in the objectives of these two phases lead to significant disparities in memory and computation characteristics. From a memory perspective, the Token Generation Phase has much higher memory requirements than the Prompt Phase. In contrast, the computation demands are quite different: the Prompt Phase is far more computation-intensive than the Token Generation Phase.

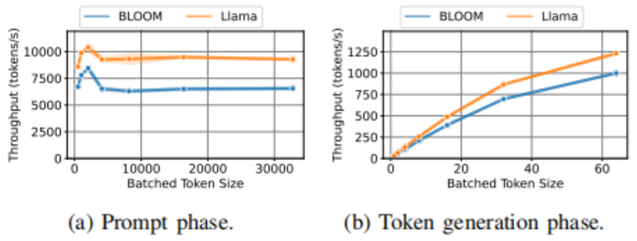

图 3. Computation

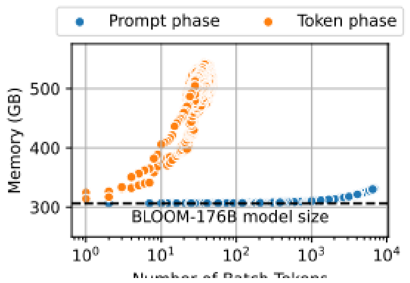

图 4. Memory

Thus, it can be observed that although both phases belong to the same task, they differ greatly in memory and computational characteristics. The Prompt Phase is typically a computation-limited scenario, where the bottleneck is dominated by computation, whereas the Token Generation Phase is a memory-limited scenario, where the bottleneck is dominated by memory access. The Prompt Phase can also be referred to as the prefill stage. During the prefill stage, the large model computes and stores the raw input token's key-value (KV) cache, generating the first output token. In the Token Generation Phase (decode stage), the model utilizes the KV cache to output tokens one by one and updates the cache with new key-value pairs generated from the newly generated tokens.

Efficient allocation of computation and memory resources can help achieve high-performance large model inference.

B. KV-cache (Key-Value Cache)

KV Cache is a crucial technique in large model inference. It is based on the Q, K, and V matrices from the attention mechanism. The attention mechanism formula is:

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

For the Decode phase, the queries (Q), keys (K), and values (V) are defined as follows:

$$Q = W_q x_j$$

$$K = W_k[x_1, x_2, ..., x_j]$$

$$V = W_v[x_1, x_2, ..., x_j]$$

Due to the autoregressive nature of large models, when calculating a new token, we need to use the previously computed key and value vectors. However, since tokens are generated one by one, we do not need to recompute the vectors for all tokens. Instead, we can reuse the already generated keys and values, significantly reducing the computational overhead.

This caching and reuse of key and value vectors is known as KV Cache, which updates only when a new token is computed. Compressing KV caches to reduce storage overhead and minimizing network transmission costs during communication are also active research areas [21].

C. Parallel Inference

Parallel inference is used to run large model inference in parallel across multiple device nodes to speed up the process. Currently, three classic methods can be used: data parallelism [9], model parallelism [18], and pipeline parallelism [5].

Data parallelism involves splitting large datasets into smaller batches, distributing them across multiple independent nodes that each run a copy of the model, and then aggregating the results. This method is simple in structure but limits the model' s parameter size to fit within the capacity of a single node.

Model parallelism, on the other hand, involves distributing large layers or modules of a model that cannot fit within a single node across multiple nodes. This method can significantly reduce the memory and computational requirements of a single node, but it is relatively complex to implement.

Pipeline parallelism involves splitting different layers of the same model across different nodes and running them in parallel like a pipeline. This allows for efficient handling of large models while maintaining good communication between nodes. However, the computation and communication time between pipeline stages need to be carefully managed.

In practice, these three parallel methods are often combined in a flexible manner to optimize the performance of large model inference.

IV. Analysis of Issues in Large Model Inference and Solutions

This section introduces the four main issues currently encountered in large model inference, along with the solutions proposed by both industry and academia.

A. Low Compute-Memory Ratio in Decode Stage

LLM inference is an autoregressive, serial decoding model, where each input request involves generating one token at a time, and the operation is a matrix-vector multiplication. The memory demands during the decode stage significantly outweigh the computational requirements.

In the decoding process, the model gradually generates each token in the output sequence (for example, generating words or characters in a language model). This process depends on previously generated tokens, and with each new token, the model typically performs a computation to predict the next most likely output. However, as more tokens are generated during the decode stage, the model needs to continuously reference and read the context of previously generated tokens, which leads to a rapid increase in memory requirements. Consequently, although the computational demand for generating each new token is relatively low, memory access becomes the primary bottleneck, resulting in a low compute-memory ratio in the decode stage.

Feasible solutions include batch processing with multiple requests, such as PD separation. PD separation divides the computation-intensive prefill phase from the memory-intensive decode phase to avoid batch interference during both phases [15], as well as parallel decoding with speculative inference, which aims to speed up inference [11].

B. High Computational and Memory Demands of Long Sequence Prefill

During the prefill phase, as the input sequence length increases, both computational and memory demands grow significantly, leading to a decrease in inference efficiency and potentially exceeding hardware capacity.

The prefill phase involves performing self-attention calculations on the input sequence, and the computational cost of this step grows quadratically with the sequence length. Additionally, transformers need to store embeddings for each position and intermediate results such as attention matrices during each layer's computation. As the network's depth increases and the sequence length grows, memory usage increases dramatically, leading to high computational and memory demands.

To address the challenges posed by long sequences in the prefill phase, feasible solutions in the industry include techniques like KV cache reuse across multiple requests, such as prefix caching, multi-level parallelism for long sequence prefill, and sparse attention mechanisms.

Prefix caching [7] allows for the sharing of KV Cache across requests that share the same prefix, which is useful in scenarios like system prompts and multi-turn dialogues.

C. GPU Memory Capacity and Computational Bottlenecks

As model size grows, the number of parameters increases rapidly. During inference, model weights, activations, and intermediate results require substantial memory, which may exceed the GPU's capacity. Furthermore, during inference, layers need to frequently read parameters, activations, and intermediate results from GPU memory. If the GPU's memory bandwidth is insufficient, the read speed becomes a bottleneck, leading to delays in the computation process.

To address these issues, several solutions have been proposed in the industry, such as optimizing memory allocation mechanisms, optimizing attention operators like Flash Attention[3], using compression techniques like quantization and pruning, as well as optimizing the attention mechanism itself with approaches such as GQA[12], MQA[14], and MLA[13].

Flash Attention leverages the memory architecture of GPUs, which consists of two layers: HBM (High Bandwidth Memory) and SRAM (Static Random Access Memory). The former has high capacity but limited bandwidth, while the latter offers high bandwidth but limited capacity. All intermediate computation results must eventually be written back to the GPU's HBM. For the Q, K, and V matrices in the attention mechanism, we can use techniques such as tiling to partition the matrices and load them into SRAM for computation and reuse. Then, the results can be combined through transformations to form the final output, addressing the bandwidth issues of both memory-intensive operations and compute-intensive operations.

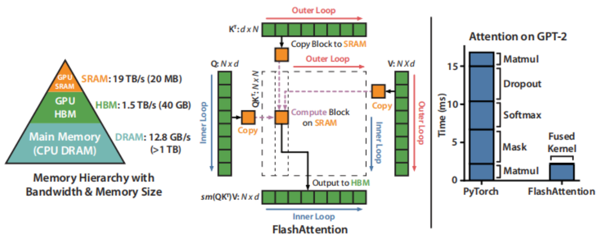

图 5. Flash Attention

D. *Evolution of Inference Scenarios for Large Models*

Traditional large model inference, such as GPT and BERT, focuses primarily on unimodal tasks like text. However, multimodal inference has emerged, with models such as Sora, text-to-image, and text-to-video generation models rapidly proliferating. Understanding the evolution of different large model inference scenarios has become one of the key research areas.

Mixture of Experts (MoE) is a large-scale machine learning model architecture designed to handle complex tasks by introducing multiple "expert" models (typically deep neural networks). During inference, only a subset of experts is activated, reducing computational overhead while maintaining performance, thereby improving inference efficiency.

Retrieval-Augmented Generation (RAG) is an advanced model architecture that combines retrieval and generation to enhance the capabilities of generative models, such as language generation models. By integrating external knowledge bases, RAG improves the inference performance of generative models. This architecture allows generative models to access knowledge not present in the training data and enhances their performance in handling rare or domain-specific information.

V. Conclusion

This survey first summarizes the inference scenarios of large models, including their basic process and key structure, the Transformer. It then focuses on three critical aspects of inference: Prefill and Decode, KV Cache, and Parallel Inference. Finally, the survey explores four research hotspot issues in both industry and academia, along with the associated solutions.

VI. Learning Insights and Reflections

As I write this literature review, I hope it can serve as a solid index for my entry into the field of large model inference. It should provide an introduction to the basic processes and structures of large model inference, covering various types of large models such as MoE, LLMs, and multimodal models. Additionally, it should reveal some key issues in inference and potential research directions, both algorithmic and systemic, clarifying my future learning path.

Of course, while writing this review, there are parts I haven't fully understood; my knowledge of these issues is still superficial, limited mainly to recognizing that these problems exist and understanding some potential solutions. My familiarity with relevant papers is also restricted to their abstracts. However, I plan to gradually delve deeper into each direction and ultimately focus on the areas I am most passionate about. Currently, I am particularly interested in Flash Attention, multimodal models especially those related to Sora and PD classification, which presents a systematic viewpoint. These will be my focal points for future research.

The reason why I chose to write in English is that I want to try in generative language models to tes

The reason I chose to write in English is that I want to see the capabilities of generative language models in refining English text and how it compares to my own writing. In an era where many papers require GPT for editing, I believe that simply prohibiting the use of GPT is not a reasonable demand. Crafting suitable logic on my own and then having GPT refine it has become a common practice in the research community for writing papers.

参考文献

[1] Tom B. Brown, J. Manns, N. Ryder, and et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[2] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[3] Tri Dao, Daniel Y. Fu, Anish Athalye, Omer Levi, Hafeez Anwar, and H. T. Kung. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, pages 1–12. NeurIPS, 2022.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics, 2019.

[5] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019.

[6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[7] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023.

[8] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020.

[9] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: Experiences on accelerating data parallel training, 2020.

[10] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, and Lichao Sun. Sora: A review on background, technology, limitations, and opportunities of large vision models, 2024.

[11] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS ' 24, page 932–949. ACM, April 2024.

[12] Author Name and Author Name. Global query attention: Improving long-range dependency capture in transformers. *Journal of Machine Learning Research*, 24(7):1–20, 2023.

[13] Author Name and Author Name. Multi-level attention for large-scale transformers. In *Proceedings of the ICML Conference*, pages 1234–1245. ICML, 2023.

[14] Author Name and Author Name. Multi-query attention: Efficient parallel attention for large models. *Proceedings of the NeurIPS Conference*, 37:30–40, 2024.

[15] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting, 2024.

[16] Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. Mesh-tensorflow: Deep learning for supercomputers, 2018.

[17] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[18] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

[19] Hugo Touvron, Piotr Bojanowski, Tomas Mikolov, and et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[20] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[21] Jiayi Yuan, Hongyi Liu, Shaochen Zhong, Yu-Neng Chuang, Songchen Li, Guanchu Wang, Duy Le, Hongye Jin, Vipin Chaudhary, Zhaozhuo Xu, Zirui Liu, and Xia Hu. Kv cache compression, but what must we give in return? a comprehensive benchmark of long context capable approaches, 2024.

[22] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2024.