LaCache: Ladder-Shaped KV Caching for Efficient Long-Range Generation of Large Language Models

Anonymous Authors[1]

Abstract

Recent advancements in Large Language Models (LLMs) have spurred interest in numerous applications requiring robust long-range capabilities, essential for processing extensive input contexts and continuously generating extended outputs. As sequence lengths increase, the number of Key-Value (KV) pairs in LLMs escalates, creating a significant efficiency bottleneck by intensifying the long-range dependencies between tokens within the KV cache. To address this challenge, several KV cache reduction strategies have been proposed. These methods reduce the involvement of most KV pairs during inference, thereby lowering computational costs while striving to minimize information loss. Conceptually, existing methods typically fall into two categories: (1) *recency-based* approaches, which store only the initial and most recent tokens for storage and processing efficiency but suffer from *limited long-range capabilities* due to the exclusion of intermediate tokens; and (2) *retrieval-based* approaches, which store the entire KV cache and dynamically retrieve a subset of essential tokens for efficient and accurate generation but can encounter *storage inefficiency* and *limitations in continuous generation*. In this paper, we propose a new KV cache optimization paradigm called **LaCache**, a training-free method for efficient and accurate generative inference of LLMs. LaCache enables LLMs to simultaneously improve all three critical requirements when it comes to long-range LLMs: robust long-range capabilities, continuous generation, and efficient cache utilization. Specifically, LaCache integrates two key innovations: (1) a ladder-shaped KV cache pattern that stores KV pairs not only sequentially (left-to-right within each layer) but also across layers (from shallow to deep), providing an extended span for capturing long-range dependencies under a fixed storage budget, thereby boosting long-range capabilities; and (2) an iterative compaction mechanism that progressively compresses older caches, freeing up space for new tokens within a fixed cache size. This token distance-based dynamic compression enables more effective continuous generation under constrained cache budgets, facilitating optimal cache utilization. Experiments across various tasks, benchmarks, and LLM models consistently validate LaCache's effectiveness in enhancing LLMs' long-range capabilities. All code will be released upon acceptance.

1 Introduction

Large Language Models (LLMs) have significantly advanced natural language processing tasks (Achiam et al., 2023; Touvron et al., 2023a; Jiang et al., 2023; Team et al., 2024), yet they face significant challenges due to their substantial computational resource demands. The auto-regressive decoding mechanism, fundamental to current Transformer-based LLMs, exacerbates inefficiencies during inference by necessitating the sequential processing of tokens, which is particularly problematic in long-context tasks. Key-Value (KV) caching has been utilized to mitigate some of these challenges by storing intermediate attention keys and values, thereby avoiding the recomputation of KV states. However, this approach introduces significant memory overhead as requirements scale linearly with sequence length. For instance, a 30-billion-parameter model with a batch size of 128 and a sequence length of 1024 can require up to 180 GB of KV cache (Zhang et al., 2024), highlighting a critical bottleneck for memory efficiency.

Existing KV cache eviction strategies attempt to address these challenges by pruning cached KV states to enhance memory efficiency. Nevertheless, these strategies often struggle to simutaneously balance three critical requirements for long-range LLMs in real-world applications: robust long-range capabilities, continuous generation, and efficient cache utilization. For example, StreamingLLM (Xiao et al., 2023b) prioritizes continuous generation but compromises accuracy, while Quest (Tang et al., 2024) maintains high accuracy but at the expense of substantial memory usage due to the necessity of caching the entire KV cache.

In response to the aforementioned limitations, we propose

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review . Do not distribute.

LaCache, a training-free KV cache optimization framework featuring a ladder-shaped storage pattern for accurate and efficient generative inference of LLMs. LaCache integrates two key innovations: (1) *Ladder-shaped KV Cache Pattern*: This strategy stores KV pairs not only sequentially (left-to-right within each layer) but also across layers (from shallow to deep across layers). This configuration extends the span for capturing long-range dependencies under a constrained storage budget, thereby enhancing long-range capabilities. Specifically, it preserves the KV states of early tokens in earlier layers and then progressively shifts the focus to later tokens in the subsequent layers, forming a stepwise, ladder-like structure. (2) *Iterative Compaction*: This approach progressively compresses older caches, freeing up space for new tokens within a fixed cache size. It enables more aggressive compaction of older tokens while applying less intensive compaction to newer tokens, ensuring that recent and relevant information is prioritized. This optimizes memory usage while maintaining high generation accuracy over longer sequences. By integrating these approaches, LaCache alleviates the limitations of existing KV cache mechanisms, providing an efficient and robust solution for LLM generation. LaCache not only mitigates memory overhead but also enhances the model's ability to dynamically generate and utilize context, which is crucial for tasks involving complex, extended dialogues or document processing where context sensitivity is paramount.

The contributions of this paper are summarized as follows:

- **Ladder-Shaped KV Cache Storage Pattern:** We propose LaCache, introducing a novel ladder-shaped KV cache pattern designed to achieve accurate and cost-effective long-context generation. Specifically, it preserves the KV states of early tokens in earlier layers and then progressively shifts the focus to later tokens in the subsequent layers. This arrangement allows the KV cache to be condensed, reducing storage costs while maintaining high performance. Additionally, our ladder-shaped KV cache mechanism is orthogonal to existing KV cache strategies, making it possible to integrate it with state-of-the-art methods to achieve more accurate and efficient generative inference.
- **Iterative Compaction Mechanism:** We integrate LaCache with an innovative iterative compaction mechanism that periodically applies a ladder-based compression pattern to previously condensed KV states, freeing up space for new tokens. This method compresses older tokens more aggressively and newer tokens less so, allowing the model to prioritize recent information while efficiently managing memory for incoming tokens. This dynamic compression strategy supports continuous generation, even for infinite generation length without running out of memory (OOM).
- **Validation through Experiments and Ablation Studies:** We evaluate and validate the effectiveness of LaCache through a series of experiments and ablation studies. The results, obtained across multiple benchmarks, consistently demonstrate that LaCache enhances long-range capabilities, supports continuous generation, and optimizes cache utilization simultaneously in long-context generation tasks.

2 PRELIMINARIES

In autoregressive LLMs, causal attention mechanisms (Vaswani et al., 2017) are key components for language modeling, designed to ensure that each token only attends to previous tokens within a sequence, preventing information "leakage" from future tokens. This structure is fundamental in autoregressive models, where the task requires generating tokens one step at a time, based on prior context.

In the attention mechanism, each token x_i in a sequence $x = \{x_1, x_2, \ldots, x_n\}$ is transformed into a set of query (Q), key (K), and value (V) vectors. The attention score between two tokens i and j is calculated as:

$$\text{Attention}(x_i, x_j) = \text{softmax}\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) V_j$$

where d_k is the dimension of the key vectors. In practice, the KV cache stores the computed key and value vectors from prior tokens. This cache allows for efficient retrieval of prior context, enabling the model to skip recalculating keys and values for already processed tokens when generating new outputs. Formally, let $K_{\leq t}$ and $V_{\leq t}$ represent the stored keys and values up to time step t. For token x_{t+1}, we only compute the query vector Q_{t+1} and use the cached keys and values for attention calculation:

$$\text{Attention}(x_{t+1}) = \text{softmax}\left(\frac{Q_{t+1} K_{\leq t}^T}{\sqrt{d_k}}\right) V_{\leq t}$$

This approach significantly reduces computational overhead by avoiding redundant key and value calculations, especially in long sequences.

However, storing and processing the entire KV cache for long sequences results in substantial memory and computation overhead. Specifically, given a sequence length T, the memory complexity for storing keys and values for each token grows linearly, or $\mathcal{O}(T)$. As the sequence length increases, this requires substantial memory, especially for longer contexts. In terms of computational complexity, each new token t requires calculating attention with all previous

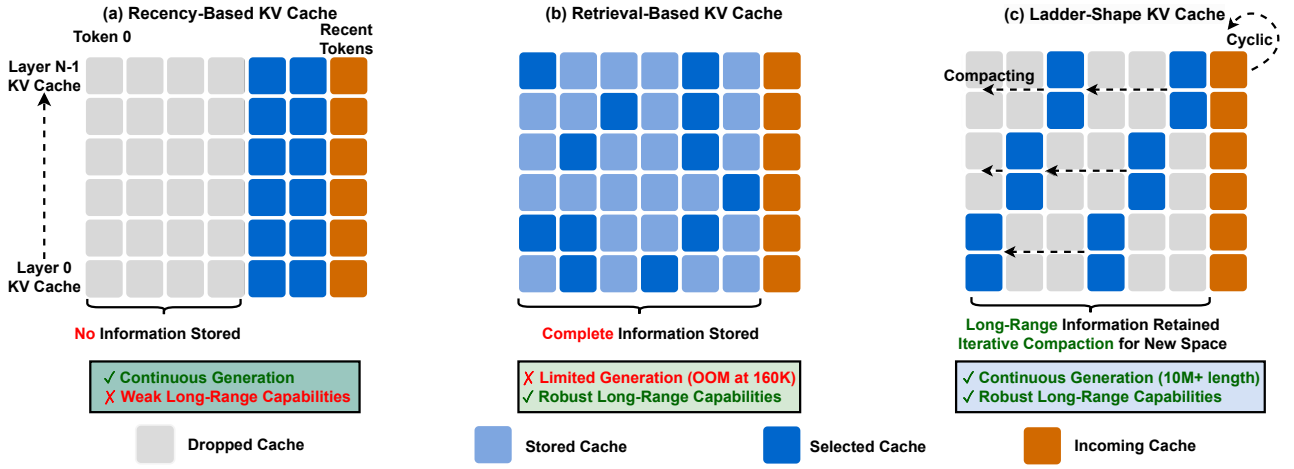

Figure 1. Illustrative comparisons among (a) the recency-based KV cache (Xiao et al., 2023b), (b) the retrieval-based KV cache (Tang et al., 2024), and (c) our proposed LaCache KV cache featuring a ladder-shaped pattern. Previous KV cache storage strategies struggle to simutaneously balance the needs for continuous generation, robust long-range capabilities, and efficient cache utilization. In contrast, the proposed LaCache allows LLMs to satisfy all three properties simultaneously. Here, "OOM" indicates out-of-memory.

tokens up to T. This involves T operations per token, resulting in a total computational complexity of $\mathcal{O}(T^2)$ for the entire sequence. This quadratic scaling in computation is resource-intensive and limits the model's ability to handle long contexts efficiently.

This challenge calls for KV cache reduction methods to enable efficient generation. For example, if the attention computation is limited to a fixed context window instead of all preceding tokens, the computational complexity per token becomes constant, keeping the total complexity linear with respect to T, or $\mathcal{O}(T)$. If only the KV cache of the most recent tokens within a fixed sliding window is maintained, the memory complexity can remain constant, i.e., $\mathcal{O}(1)$. This KV cache reduction approach is essential for enabling efficient processing of long sequences while controlling memory and computation costs.

3 The Proposed LaCache Framework

In this section, we present the proposed LaCache framework. We begin with an overview of LaCache in Sec. 3.1, followed by an introduction to the proposed ladder-shaped KV cache patterns in Sec. 3.2. Finally, to support efficient continuous generation without exhausting memory resources, even for tasks involving infinite-length generation, we further enhance LaCache with an iterative compaction strategy, as described in Sec. 3.3.

3.1 LaCache: Motivation and Methodology Overview

Limitations of existing methods. Existing efficient generation methods for LLMs face limitations in achieving both generation accuracy and memory efficiency, which are crucial for continuous long-context generation without running out of memory. Specifically, as shown in Fig. 1 (a), recency-based methods like StreamingLLM (Xiao et al., 2023b), which only maintain the KV cache of the latest tokens within a fixed-length sliding window with an $\mathcal{O}(1)$ memory complexity, can support infinite-length generation without running out of memory but may compromise generation accuracy. In contrast, retrieval-based methods like Quest (Tang et al., 2024), as shown in Fig. 1 (b), store the full KV cache of all tokens and retrieve the most relevant ones for each new token on the fly to improve computational efficiency. This strategy can achieve high accuracy across tasks due to the maintenance of the entire KV cache, but suffers from poor memory efficiency with an $\mathcal{O}(T)$ memory complexity, leading to potential OOM issues when handling long contexts.

In light of the limitations of both approaches, as shown in Fig. 1 (c), we propose LaCache, a training-free KV cache optimization featuring a ladder-shaped pattern, designed to balance accuracy and storage cost, enabling accurate and continuous generation without suffering from OOM.

Overview. Motivated by the need for both accurate and efficient KV caching, our proposed LaCache features a ladder-shaped KV cache compression and storage pattern. We illustrate LaCache's pattern in Fig. 2, where KV states are represented in a 2D array: the x-axis represents the generation timesteps (i.e., the token position index), and the y-axis represents the layer index of an LLM.

To achieve effective KV compression while preserving important information of past tokens, rather than uniformly

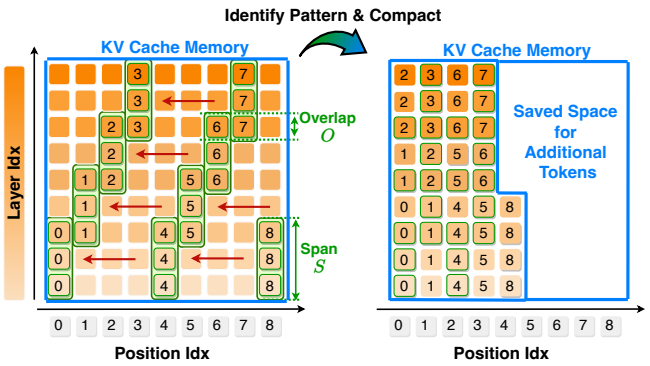

Figure 2. An illustration of LaCache's KV cache storage pattern. LaCache is leveraged to compact the original full KV cache into a compressed, ladder-shaped pattern, allowing for the storage of information from longer-range tokens compared to StreamingLLM (Xiao et al., 2023b) under the same KV cache budget, thereby providing stronger long-range sequence modeling.

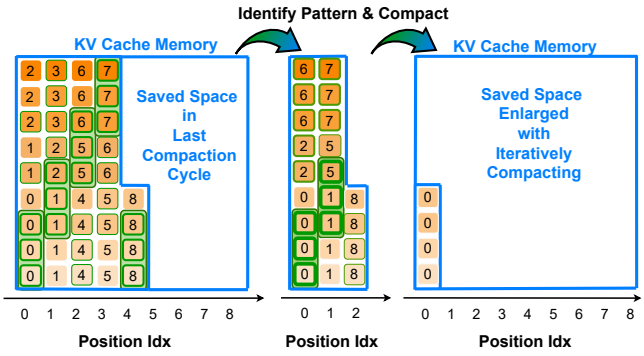

Figure 3. An illustration of LaCache's iterative compaction. Iterative compaction is introduced to support continuous generation without running out of memory, even for infinite-length generation. Once the KV cache reaches the predefined size, LaCache's ladder-shaped pattern is applied to the already-compacted KV cache, freeing up space for new tokens. This process is performed iteratively to maintain a constant KV cache size.

keeping the KV cache for the same set of tokens across all layers as in StreamingLLM (Xiao et al., 2023b), we retain the KV cache for different tokens across different layers using a ladder-shaped pattern, as highlighted by the green boxes in Fig. 2. Specifically, this is achieved by preserving the KV states of early tokens in earlier layers and then progressively shifting the focus to later tokens in the subsequent layers, forming a stepwise, ladder-like structure. By adopting this ladder-shaped pattern, with the KV cache maintained for varied tokens across different layers, our method can effectively retain the information of more past tokens within the same KV cache budget compared to StreamingLLM (Xiao et al., 2023b), thereby increasing the likelihood of retaining key information for more accurate generation.

Furthermore, to support continuous generation without suffering from OOM, even for infinite-length generation, we augment our LaCache with an iterative compaction strategy, as depicted in Fig. 3. Specifically, each time the KV cache reaches its capacity, we apply our LaCache with the ladder-shaped pattern to the already-compacted KV cache. This strategy ensures that older token information is progressively compressed further, while newer incoming tokens are compressed less. By iteratively applying LaCache, the occupied memory of the KV cache can remain constant with respect to the generation length, thus balancing generation accuracy and storage efficiency and enabling infinite-length generation without OOM.

In the following subsections, we will elaborate on the ladder-shaped pattern and the iterative compaction strategy, which are the two key enablers of our LaCache framework.

3.2 LaCache: Ladder-Shaped KV Cache Pattern

The key insight. Unlike StreamingLLM (Xiao et al., 2023b), which maintains the KV cache of the same set of most recent tokens across all layers, our key insight is that while the information from recent tokens is critical for generation accuracy, their KV cache can be maintained and processed by fewer layers. In other words, different layers can maintain the KV cache corresponding to different sets of tokens. The key advantage of this approach is that, under the same KV cache budget, more tokens can be retained in the KV cache, effectively enlarging the context length and preserving more past information.

The proposed ladder-shaped pattern. The aforementioned insight inspires the design of our ladder-shaped KV cache pattern. As shown in Fig. 1 (c), our LaCache adopts a simple yet effective strategy to cache the KV states of varied tokens across different layers: it preserves the KV states of early tokens in earlier layers and then progressively shifts focus to later tokens in the subsequent layers, aligning with the temporal and sequential processing nature. This approach results in our ladder-shaped pattern, ensuring both storage efficiency and the retention of essential information across past tokens. More specifically, as shown in Fig. 2, to implement the ladder-shaped pattern denoted by the green box, we remove the KV states falling outside this pattern and then condense the original 2D KV cache into a more compact structure with reduced cache size.

Principles and key design factors. To balance storage efficiency and generation accuracy, we need to ensure that our method properly eliminates redundancy in the stored KV states while accurately preserving past context information

by retaining sufficient past KV states.

To satisfy these principles, two key design factors help balance both aspects and achieve an optimal trade-off: (1) the **span** S of each layer, *i.e.*, the number of tokens with their KV states preserved in each layer, and (2) the **overlap** O across consecutive layers, *i.e.*, the number of layers used to preserve the KV state corresponding to the same token. The larger the **span** S, the more KV states are preserved by each layer, allowing for more accurate context recording at the cost of reduced storage efficiency. Similarly, the larger the **overlap** O, the more layers are used to record the context for the same token, thereby improving context preservation with an increased storage cost. As demonstrated in Sec. 4, we calibrate these two design factors for each task to minimize cache redundancy and maximize generation accuracy.

3.3 LaCache: Iterative Compaction for Continuous Infinite-Length Generation

To enable continuous generation in LLMs without encountering OOM issues, even with an infinite generation length, it is highly desirable to maintain a constant KV cache size. To achieve this, we need to augment our LaCache with an eviction mechanism that removes the KV states of past tokens when the predefined cache size is fully utilized.

Rationale and advantage of our method. In response to the aforementioned need, we propose the iterative compaction strategy. The rationale behind our method is simple: once the KV cache, already condensed using LaCache, is full, we apply LaCache again to further condense it.

The advantages of this approach include: (1) Thanks to LaCache's ladder-shaped pattern design introduced in Sec. 3.2, early KV caches are discarded first when applying LaCache on the already-condensed KV cache, as shown in Fig. 3. This use of larger/smaller compression ratios on early/late KV caches aligns with the design principles of recency-based methods (Xiao et al., 2023b); (2) From a deployment perspective, iterative compaction using LaCache provides a unified solution and a clean interface, facilitating wider use.

Implementation. We provide a more detailed illustration of how our iterative compaction works. As shown in Fig. 3, parts of the stored KV states are highlighted to demonstrate their changes after iterative compaction, while the non-highlighted parts are also occupied by other KV states. When the KV cache reaches its capacity, LaCache is applied to the stored KV states, which have already been condensed by LaCache when first enqueued into the KV cache. Consequently, KV states that fall outside the ladder-shaped pattern are removed, and the freed space is allocated for new incoming tokens. In iteration 2 of Fig. 3, older KV states are compressed more while newer ones are compressed less, thus better preserving recent information.

4 Experimental Results

4.1 Experimental Settings

Models. To validate LaCache's capability to generally improve long-range performance in target LLMs, we applied the proposed LaCache to five commonly used LLMs, including (1) Llama2-7B/13B (Touvron et al., 2023a), (2) Llama2-7B/13B-Chat (Touvron et al., 2023a), and (3) Llama3-8B (Dubey et al., 2024).

Datasets. To assess LaCache's ability to enhance long-range performance across different generation settings, we evaluated LaCache on two tasks: long-context modeling and long-context understanding. Specifically, (1) for the long-context modeling task, we used Wikitext-2 (Merity, 2016) and PG19 (Rae et al., 2019) to evaluate LaCache's language modeling capabilities; and (2) for the long-context understanding task, we utilized 21 datasets in LongBench (Bai et al., 2023) to thoroughly evaluate LaCache's performance.

Baselines. We compared our proposed LaCache against the standard full KV cache settings and prior KV cache compression methods, including StreamingLLM (Xiao et al., 2023a) and H2O (Zhang et al., 2024), under various text lengths and cache budgets.

Implementation Details. We implemented the proposed LaCache in PyTorch. For all benchmarked tasks, we used a batch size of 1 for evaluation. Specifically, for language modeling experiments, we follow the implementation in StreamingLLM (Xiao et al., 2023b) and H2O (Zhang et al., 2024) and employ regular token-by-token generation for Wikitext-2 and a sliding window approach with a window length of 256 tokens for PG-19, respectively. In all 21 datasets within the LongBench benchmark, following LongBench's default setting, we retained the first 128 tokens unchanged for both LaCache and baseline methods, as these initial tokens primarily consist of system prompts and questions. All experiments were conducted on NVIDIA A100 and A6000 GPUs.

4.2 Long-Context Modeling Benchmarks

Experimental Results on Wikitext-2 Benchmark. We first evaluate LaCache on language modeling tasks using the concatenated Wikitext-2-raw-v1 dataset in a standard token-by-token generation setting. Four models, Llama2-7B, Llama2-7B-Chat, Llama2-13B, and Llama3-8B, are tested with KV Cache sizes of 512 and 256. As summarized in Table 1, our experimental results demonstrate that the proposed LaCache consistently shows stronger capabilities in capturing long-range dependencies compared to the recency-based method StreamingLLM (Xiao et al., 2023b), under the same KV Cache budget, across decoding lengths ranging from 1K to 16K. Specifically, with a 512 cache size to store KV states from a 1K-length input, LaCache only

Table 1. Language Modeling experiments on the concatenated wikitext-2-raw-v1 dataset. We use decoding length across 1K to 16K for every tested model and cache budget. While decoding length exceeds the pre-training length, models using full cache encounter a perplexity explosion issue.

Decoding Length	1K	2K	4K	8K	16K	Decoding Length	1K	2K	4K	8K	16K
Llama2-7B-4K (Full Cache)	4.02	4.18	5.12	nan	nan	Llama2-7B-Chat-4K (Full Cache)	4.94	5.32	6.52	nan	nan
w/ StreamingLLM (Cache Size=512)	5.54	5.84	6.32	6.93	5.36	w/ StreamingLLM (Cache Size=512)	6.67	7.41	7.95	8.97	6.98
w/ LaCache (Cache Size=512)	**4.53**	**5.00**	**5.81**	**6.61**	**5.19**	w/ LaCache (Cache Size=512)	**5.20**	**6.01**	**7.06**	**8.35**	**6.64**
w/ StreamingLLM (Cache Size=256)	6.08	6.38	6.90	7.52	5.92	w/ StreamingLLM (Cache Size=256)	7.68	8.45	8.98	9.89	7.88
w/ LaCache (Cache Size=256)	**5.57**	**5.98**	**6.60**	**7.34**	**5.77**	w/ LaCache (Cache Size=256)	**7.16**	**7.91**	**8.47**	**9.57**	**7.53**
Llama2-13B-4K (Full Cache)	3.89	3.70	4.67	134.25	nan	Llama3-8B-8K (Full Cache)	4.28	4.39	5.82	6.16	109.94
w/ StreamingLLM (Cache Size=512)	4.92	5.02	5.66	6.28	4.82	w/ StreamingLLM (Cache Size=512)	5.46	5.33	6.73	6.99	5.52
w/ LaCache (Cache Size=512)	**4.40**	**4.68**	**5.42**	**6.09**	**4.69**	w/ LaCache (Cache Size=512)	**4.61**	**4.89**	**6.40**	**6.78**	**5.40**
w/ StreamingLLM (Cache Size=256)	5.64	5.52	6.17	6.79	5.29	w/ StreamingLLM (Cache Size=256)	6.39	5.97	7.40	7.66	6.06
w/ LaCache (Cache Size=256)	**5.22**	**5.17**	**5.88**	**6.50**	**5.08**	w/ LaCache (Cache Size=256)	**5.71**	**5.61**	**7.11**	**7.38**	**5.86**

degrades perplexity by 5% compared to using the full cache. In contrast, StreamingLLM (Xiao et al., 2023b), under the same cache budget, results in a 35% degradation in perplexity. This indicates that with 2x compression, LaCache reduces performance degradation in language modeling by up to 7× compared to StreamingLLM (Xiao et al., 2023b).

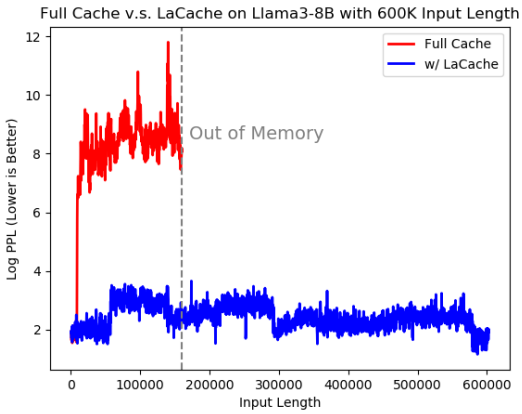

Figure 4. Evaluate LaCache on the first ten books of the concatenated PG19 dataset, which equals a 600K input length. The experimental results validate the effective continuous generation and efficient KV Cache storage of LaCache.

Experimental Results on PG19 Benchmark. To further assess LaCache's capabilities on language modeling tasks with extremely long inputs, we compare it against full cache and StreamingLLM on the concatenated PG19 dataset, which comprises 100 books totaling 10 million tokens. Specifically, we adopt a sliding window of 256 tokens for higher efficiency, following the settings in (Wolf, 2019). The compacted KV Cache output from each window is then passed to generate subsequent tokens, allowing us to evaluate the model's long-range capabilities. As shown in Figure 4, after an 8K input length, the perplexity of the Llama3-8B model using a full cache quickly escalates; after a 160K input length, an out-of-memory issue arises on a single NVIDIA A100 (80G) GPU due to the large KV Cache requirements. In contrast, with our proposed LaCache, the Llama3-8B model supports continuous generation with up to a 600K input length while maintaining reasonable perplexity.

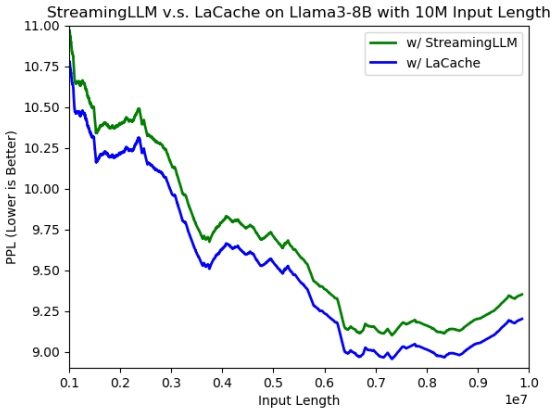

Figure 5. Evaluate LaCache on the entire concatenated PG19 dateaset, which equals to a 10 million input length. The experimental results showcases the stronger capabilities of LaCache in capturing long-range information from extremely long inputs.

Furthermore, Figure 5 summarizes the comparison between our proposed LaCache and the prior work StreamingLLM (Xiao et al., 2023b) on the full concatenated PG19 dataset. The consistent improvements achieved by the proposed LaCache further validate its strong long-range capabilities with inputs exceeding 10 million tokens.

4.3 Long-Context Understanding Benchmarks

Experimental Results on LongBench Benchmark. The LongBench benchmark (Bai et al., 2023) assesses the bilingual long-context understanding capabilities of large language models, with most task lengths averaging between

Table 2. LaCache for Llama2-13B-Chat model on 21 LongBench datasets under 50% and 25% KV Cache budget.

Budget	100%	50%		25%	
Method	Full Cache	StreamingLLM	LaCache	StreamingLLM	LaCache
HotpotQA	38.86	37.09	38.08	37.16	36.49
2WikiMultihopQA	34.19	32.30	34.11	32.15	31.17
MuSiQue	14.19	12.12	13.67	11.99	12.97
DuReader	27.34	20.03	24.07	21.07	20.78
MultiFieldQA-en	36.63	27.03	31.73	24.56	26.09
MultiFieldQA-zh	34.13	24.90	27.49	23.11	25.44
NarrativeQA	19.38	17.50	19.30	14.78	17.95
Qasper	26.84	23.02	24.51	21.44	21.05
GovReport	26.21	23.89	23.76	21.78	21.54
QMSum	20.12	19.03	19.61	18.76	19.16
MultiNews	26.06	25.38	25.33	23.71	23.79
VCSUM	16.89	15.53	16.03	14.37	13.87
TriviaQA	88.45	88.25	88.56	85.78	85.95
SAMSum	36.77	36.50	36.99	36.01	35.85
TREC	68.5	65.50	66.50	62.00	60.00
LSHT	20.25	20.00	19.00	18.75	18.75
PassageRetrieval-en	9.00	8.50	10.00	7.50	7.00
PassageCount	3.92	3.50	2.50	2.00	3.50
PassageRetrieval-zh	14.50	13.00	10.50	8.00	9.00
LCC	39.31	39.22	40.12	39.00	39.09
RepoBench-P	42.98	41.14	41.7	39.46	38.33
Average on 21 Datasets	30.69	28.30	**29.22**	26.82	**27.04**

5k and 15k tokens. Evaluation results on the 21 LongBench datasets for Llama2-7B/13B-chat models are presented in Table 2 and Table 3. We compare the proposed LaCache with the recency-based cache storage method, StreamingLLM, in these tables. Our experimental results demonstrate that LaCache, requiring no additional computation or storage costs, outperforms StreamingLLM under the same KV Cache budgets. For instance, with a 50% KV Cache budget, LaCache reduces the average performance degradation from StreamingLLM's 2.4 to 1.5 on the Llama2-13B-Chat model (Table 2) and from 2.5 to 1.7 on the Llama2-7B-Chat model (Table 3).

To further compare with the importance-based KV Cache eviction method, We use NVIDIA A6000 GPU to evaluate the trade-offs between throughput and F1 score, as well as latency per request and F1 score, on the LongBench NarrativeQA dataset using Llama-2-7B and Llama-2-13B models. Figure 6 shows that the importance-based KV Cache eviction method H2O (Zhang et al., 2024) maintains a good F1 score but suffers from low throughput and high latency, and the recency-based method StreamingLLM with higher efficiency suffers from lower performance. However, Our method demonstrates the optimal balance between throughput/latency and F1 score performance compared with these baselines.

4.4 Ablation Studies

Hyperparameter Span and Overlap. In Table 4, we present ablation experiments on the Llama2-7B-Chat model and the Wikitext-2 dataset under a 256 KV cache budget to examine the impact of the hyperparameters span and overlap on model performance. Note that the span should be a positive integer less than the number of model layers, while overlap should be a non-negative number less than the span. As shown in Table 4, varying the span and overlap in a ladder configuration significantly affects model performance. When overlap is fixed, we found that a span of 8 typically yields good results. Conversely, when span is fixed, the

Table 3. LaCache for Llama2-7B-Chat model on 21 LongBench datasets under 50% and 25% KV Cache budget.

KV Cache Budget	100%	50%		25%	
Method	Full Cache	StreamingLLM	LaCache	StreamingLLM	LaCache
HotpotQA	33.84	29.98	32.62	30.74	30.6
2WikiMultihopQA	26.83	24.75	26.22	24.99	25.19
MuSiQue	8.82	8.48	7.72	6.58	7.94
DuReader	24.06	18.30	22.64	19.11	21.05
MultiFieldQA-en	35.72	27.02	31.55	24.74	25.34
MultiFieldQA-zh	33.32	23.84	25.69	20.13	22.55
NarrativeQA	16.78	15.97	16.27	13.46	15.18
Qasper	17.33	15.79	16.55	15.90	16.1
GovReport	26.25	22.54	22.84	20.73	20.47
QMSum	20.89	19.72	20.34	19.30	19.58
MultiNews	25.83	25.07	25.16	23.32	23.27
VCSUM	14.33	13.12	13.16	12.31	12.18
TriviaQA	83.01	83.13	83.28	80.31	81.09
SAMSum	41.28	83.22	39.47	38.46	38.86
TREC	64.50	62.50	65.00	59.00	58.00
LSHT	17.75	17.25	16.25	15.00	15.75
PassageRetrieval-en	11.50	6.50	5.00	6.50	6.50
PassageCount	4.50	5.00	5.50	5.00	5.00
PassageRetrieval-zh	12.00	7.50	7.60	7.00	5.50
LCC	47.74	47.97	47.97	47.18	45.59
RepoBench-P	44.35	43.44	43.41	43.82	43.54
Average	29.08	26.56	**27.34**	25.41	**25.68**

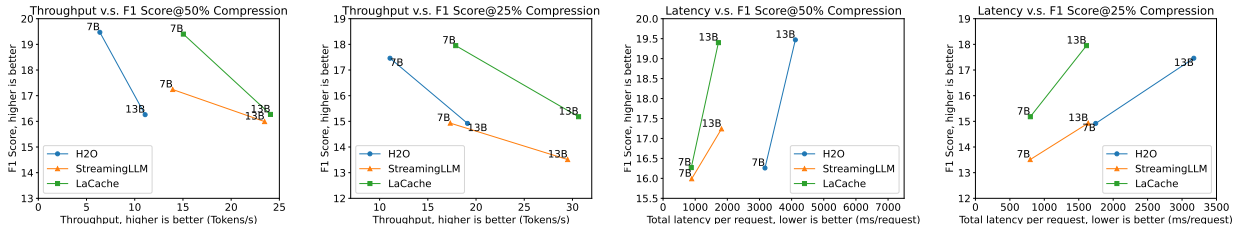

Figure 6. The throughput v.s. F1 score and the latency per request v.s. F1 score on LongBench NarrativeQA dataset, with different Llama-2 model sizes (7B and 13B). The proposed LaCache achieved the best balance between the throughput/latency and F1 score performance. H2O (Zhang et al., 2024) maintains a good F1 score but suffers from low throughput and high latency. StreamingLLM (Xiao et al., 2023b) with higher efficiency suffers from lower performance. LaCache demonstrates the balance between efficiency and performance

optimal overlap may vary. For instance, with a span of 4, the best performance is achieved when overlap reaches its highest possible value, 3. However, with a span of 8, optimal performance occurs when overlap is at its lowest possible value, 0. Additionally, with a span of 16, most overlap values did not notably affect model performance.

Choices of Patterns for Compaction In addition to the hyperparameters for span and overlap, we also conduct ablation studies on the Llama3-7B model and the PG-19

Table 4. Ablation studies on hyperparameter span and overlap. Perplexity is reported in the table. **Bold** indicates the config with the best performance while underline indicates the config with the second best performance.

Span/Overlap	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	10.04	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	9.96	10.07	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	9.89	9.72	9.79	9.69	-	-	-	-	-	-	-	-	-	-	-	-
8	**9.44**	9.59	9.59	9.57	9.63	9.46	9.71	9.64	-	-	-		-		-	-
16	9.57	9.63	9.63	9.63	9.63	9.63	9.63	9.63	9.63	9.63	9.63	9.64	9.64	9.67	9.67	9.75

Table 5. Ablation studies on patterns applied for compacting. We compare six different patterns as visualized in Figure 4.4 and find ladder-shaped patterns simple yet effective.

Ladder Pattern	Old-First	Recent-First	Evenly Sampling	Reverse Ladder	Ladder	Stacked Ladder
Perplexity	16.08	11.44	11.36	11.33	11.32	11.22

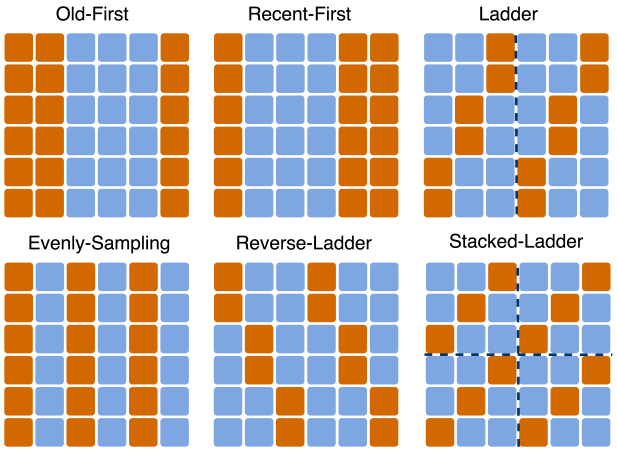

Figure 7. Different patterns in our ablation study. *Ladder*-based patterns achieve better performance compared with other patterns.

dataset to explore the effects of applying patterns with different shapes. As shown in Table 5, our observations include: (1) Applying location-aggregated patterns, such as recent-first (StreamingLLM) and old-first (a reverse version of StreamingLLM, where we retain most of the old caches while discarding newly generated tokens), results in weaker long-range capabilities than location-scattered patterns, such as even sampling and ladder-shaped patterns. This is reasonable since location-scattered patterns capture more KV states under a fixed KV cache budget. (2) Additionally, ladder-based patterns outperform even sampling, likely because ladder-shaped patterns span the longest under the same KV cache budget. (3) Stacking multiple ladders vertically further improves LaCache's performance. We hypothesize that this is due to cache states from the same token being more dispersed when stacking ladder patterns.

5 Related Work

5.1 Large Language Models

The development of LLMs marks a substantial advancement in transformer-based architectures. By leveraging the attention mechanism (Vaswani et al., 2017) to process intricate input patterns, these models excel at understanding and predicting token relationships within complex sequences. This architecture's inherent scalability (Qin et al., 2023; Kaplan et al., 2020; Biderman et al., 2023) has enabled LLMs to expand massively in both model size and data requirements, unlocking advanced in-context learning abilities in zero-shot and few-shot scenarios. Pioneering models like GPT-3 (Brown et al., 2020), BLOOM (Workshop et al., 2022), the Llama family (Touvron et al., 2023a;b), and later versions such as GPT-4 (Achiam et al., 2023) have pushed model complexity and parameter counts to new heights. While LLMs hold transformative potential, their extensive parameterization imposes significant memory and computational challenges, especially for long-context generation tasks, limiting their deployment across various platforms.

5.2 KV Cache Eviction

The substantial size of the KV cache introduces significant overhead for LLMs during inference, particularly in long-context generation tasks. KV cache eviction techniques manage the removal of non-essential tokens, with approaches like those in (Xiao et al., 2023b; Han et al., 2024) suggesting the static retention of initial and recent tokens to preserve performance. To leverage dynamic information during inference, several methods (Adnan et al., 2024; Liu et al., 2024; Zhang et al., 2024; Oren et al., 2024; Wan et al., 2024) utilize attention weights to identify important tokens. For example, (Liu et al., 2024) proposes detecting repeti-

tive attention patterns, allowing future token importance to be estimated based on historical attention weights. Meanwhile, (Nawrot et al., 2024) automatically learns memory compression ratios for each head and layer. Token merging techniques (Wang et al., 2024; Shi et al., 2024; Yu et al., 2024) offer an alternative approach to reducing KV cache size by identifying regions for merging and combining tokens accordingly. In addition to memory capacity reduction methods, SparQ Attention (Ribar et al., 2023) addresses the issue by minimizing data transfer requirements. However, there remains room for improvement in balancing storage with long-context accuracy during continuous generation. The proposed LaCache tackles this by dynamically compacting and managing the KV cache through a ladder-shaped pattern that adapts based on the timestep, ensuring both cost-effective storage and high accuracy for extended contexts.

5.3 Long-Context Modeling

The demand for long-context modeling has grown significantly due to its capability to handle complex, multi-step tasks and support coherent interactions. Consequently, a substantial body of research has focused on extending long-context generation capabilities (Li et al., 2023; Peng et al., 2023), allowing models to process more tokens in a single forward pass. Approximate attention mechanisms (Beltagy et al., 2020; Kitaev et al., 2020; Wang et al., 2020) have improved efficiency, albeit with some performance trade-offs. Recently, significant advances have been made in extending the context window using positional embeddings, with techniques such as position interpolation and fine-tuning (Chen et al., 2023; Peng & Quesnelle, 2023). However, long input sequences still pose efficiency challenges during inference. The proposed LaCache mitigates these challenges by enabling LLMs to maintain high efficiency during long-context generation, even for continuous or infinite generation tasks.

5.4 Efficient Inference

Beyond KV cache eviction techniques, other common strategies for accelerating model inference include model compression and system-level optimizations. Model compression methods, such as pruning (Zhang et al., 2022; Xia et al., 2022; Tao et al., 2023), quantization (Tao et al., 2022; Frantar et al., 2022; Yao et al., 2022; Xiao et al., 2023a), and distillation (Zhang et al., 2023; Gu et al., 2024; Wu et al., 2024), aim to reduce the number of parameters or computational costs, thereby speeding up inference. At the system level, optimizations typically focus on inference engines and serving systems. Inference engine improvements include operator optimizations, such as FlashAttention (Dao et al., 2022), and memory offloading solutions (Sheng et al., 2023; He & Zhai, 2024). Serving system optimizations, on the other hand, target batch inference and scheduling (Yu et al., 2022; Kwon et al., 2023; Holmes et al., 2024). Notably, our LaCache design focuses on optimizing KV cache management through dynamic compression, making it orthogonal to the techniques mentioned above.

6 Limitations and Future Work

While our LaCache demonstrates its advantages in accurate and efficient long-context generation for LLMs, its limitations present opportunities for future exploration.

First, the ladder-shaped KV cache pattern, though effective, may not be the optimal structure for all scenarios, and alternative patterns could further enhance memory efficiency and performance. Future work will explore alternative KV storage configurations, guided by our core insight: while recent tokens are crucial for maintaining generation accuracy, fewer layers may be sufficient to process and store their KV cache effectively.

Second, LaCache is implemented in a training-free setting for efficiency and ease of deployment. However, incorporating a fine-tuning phase could potentially improve performance by adapting the KV cache pattern to specific tasks. Future research will focus on extending LaCache to a fine-tuning setting and benchmarking it against other methods that require training, providing a direct comparison of its adaptability across various application contexts.

7 Conclusion

In this paper, we introduced LaCache, a novel training-free KV cache optimization framework aimed at enhancing both the efficiency and efficacy of LLMs in long-context generation tasks. LaCache addresses the critical limitations of existing KV caching methods by incorporating a ladder-shaped KV cache storage pattern and an iterative compaction mechanism. These innovations enable LLMs to better capture long-range dependencies, optimize memory usage, and sustain continuous generation, even under fixed storage constraints. Specifically, by sequentially storing KV pairs both within layers and across layers, the ladder-shaped structure allows the model to preserve crucial information across varying levels of context, thus bolstering its ability to process and retain extensive sequences. Furthermore, the iterative compaction mechanism dynamically manages memory, ensuring that essential information is prioritized while older, less relevant data is compressed. Our results demonstrate that LaCache can effectively enhance memory efficiency and maintaining generation quality, outperforming baseline methods across benchmarks. By offering a scalable, storage-efficient solution, LaCache has the potential to advance the capability of LLMs to handle extended contexts and continuous generation and inspire further innovations in long-range LLM optimization.

REFERENCES

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Adnan, M., Arunkumar, A., Jain, G., Nair, P., Soloveychik, I., and Kamath, P. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6:114–127, 2024.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Gu, Y., Dong, L., Wei, F., and Huang, M. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Han, C., Wang, Q., Peng, H., Xiong, W., Chen, Y., Ji, H., and Wang, S. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3991–4008, 2024.

He, J. and Zhai, J. Fastdecode: High-throughput gpu-efficient llm serving using heterogeneous pipelines. *arXiv preprint arXiv:2403.11421*, 2024.

Holmes, C., Tanaka, M., Wyatt, M., Awan, A. A., Rasley, J., Rajbhandari, S., Aminabadi, R. Y., Qin, H., Bakhtiari, A., Kurilenko, L., et al. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference. *arXiv preprint arXiv:2401.08671*, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Li, D., Shao, R., Xie, A., Sheng, Y., Zheng, L., Gonzalez, J., Stoica, I., Ma, X., and Zhang, H. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36, 2024.

Merity, S. The wikitext long term dependency language modeling dataset. *Salesforce Metamind*, 9, 2016.

Nawrot, P., Łańcucki, A., Chochowski, M., Tarjan, D., and Ponti, E. M. Dynamic memory compression: Retrofitting llms for accelerated inference. *arXiv preprint arXiv:2403.09636*, 2024.

Oren, M., Hassid, M., Adi, Y., and Schwartz, R. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*, 2024.

Peng, B. and Quesnelle, J. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation, 2023.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Qin, Z., Li, D., Sun, W., Sun, W., Shen, X., Han, X., Wei, Y., Lv, B., Yuan, F., Luo, X., et al. Scaling transnormer to 175 billion parameters. *arXiv preprint arXiv:2307.14995*, 2023.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

Ribar, L., Chelombiev, I., Hudlass-Galley, L., Blake, C., Luschi, C., and Orr, D. Sparq attention: Bandwidth-efficient llm inference. *arXiv preprint arXiv:2312.04985*, 2023.

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Chen, B., Liang, P., Ré, C., Stoica, I., and Zhang, C. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pp. 31094–31116. PMLR, 2023.

Shi, D., Tao, C., Rao, A., Yang, Z., Yuan, C., and Wang, J. Crossget: Cross-guided ensemble of tokens for accelerating vision-language transformers. In *Forty-first International Conference on Machine Learning*, 2024.

Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., and Han, S. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Tao, C., Hou, L., Zhang, W., Shang, L., Jiang, X., Liu, Q., Luo, P., and Wong, N. Compression of generative pre-trained language models via quantization. *arXiv preprint arXiv:2203.10705*, 2022.

Tao, C., Hou, L., Bai, H., Wei, J., Jiang, X., Liu, Q., Luo, P., and Wong, N. Structured pruning for efficient generative pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10880–10895, 2023.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wan, Z., Wu, X., Zhang, Y., Xin, Y., Tao, C., Zhu, Z., Wang, X., Luo, S., Xiong, J., and Zhang, M. D2o: Dynamic discriminative operations for efficient generative inference of large language models. *arXiv preprint arXiv:2406.13035*, 2024.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wang, Z., Jin, B., Yu, Z., and Zhang, M. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks. *arXiv preprint arXiv:2407.08454*, 2024.

Wolf, T. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Workshop, B., Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Wu, T., Tao, C., Wang, J., Zhao, Z., and Wong, N. Rethinking kullback-leibler divergence in knowledge distillation for large language models. *arXiv preprint arXiv:2404.02657*, 2024.

Xia, M., Zhong, Z., and Chen, D. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023a.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023b.

Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., and He, Y. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35: 27168–27183, 2022.

Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.-G. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 521–538, 2022.

Yu, H., Yang, Z., Li, S., Li, Y., and Wu, J. Effectively compress kv heads for llm. *arXiv preprint arXiv:2406.07056*, 2024.

Zhang, C., Song, D., Ye, Z., and Gao, Y. Towards the law of capacity gap in distilling language models. *arXiv preprint arXiv:2311.07052*, 2023.

Zhang, Q., Zuo, S., Liang, C., Bukharin, A., He, P., Chen, W., and Zhao, T. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International conference on machine learning*, pp. 26809–26823. PMLR, 2022.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.