LightCache: Efficient Inference for Transformers via KV Cache Compression in Feature Dimension

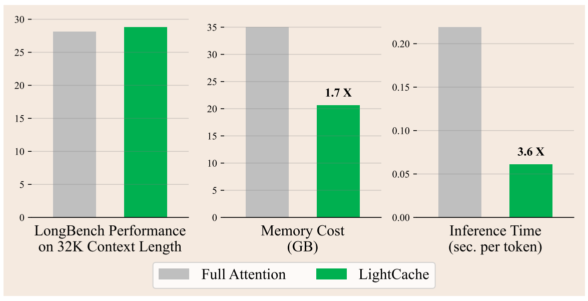
Anonymous ACL submission

Abstract

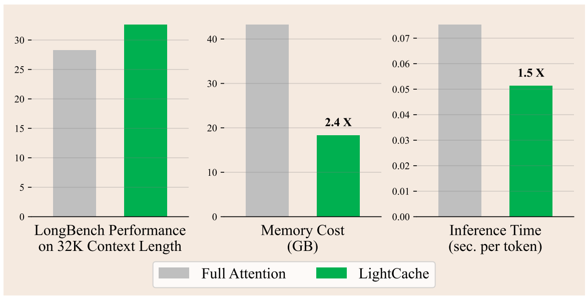
KV Cache Optimization is a crucial topic in improving the inference efficiency and length extrapolation of Transformer-based Large Language Models (LLMs). Previous KV Cache Optimization approaches often focus on pruning or compressing the sequence dimension, leading to an irreversible loss of contextual information. In this work, we propose LightCache, a novel KV Cache optimization approach that operates on the feature dimension. LightCache employs parameter-aware compression and full-context cache selection, allowing it to reduce memory usage and enhance computational efficiency without sacrificing contextual information. Importantly, LightCache enables a training-free integration with LLMs. Experiments demonstrate that LightCache outperforms classic extrapolation, quantization, and context-pruning methods in long-context evaluation. In terms of efficiency, LightCache reduces the KV Cache size by over 60% and achieves 1.7∼2.4× memory efficiency as well as 1.5∼3.6× speedup in 32K context length.

1 Introduction

Recently, Large Language Models (LLMs) based on Transformer architecture(Vaswani et al., 2017) have been proven to achieve excellent performance in Natural Language Processing(OpenAI, 2023; Touvron et al., 2023a,b), leading to higher demands for their applications. Among these, long context processing has been a widely discussed topic over the past year(An et al., 2023; Bai et al., 2023b; Zhang et al., 2024b), resulting in numerous works focusing on expanding the context of LLMs(Rozière et al., 2023; Liu et al., 2023; Xiong et al., 2023), including Linear PI(Chen et al., 2023), NTK(bloc97, 2023b,a), YaRN(Peng et al., 2023), SelfExtend(Jin et al., 2024), etc., which have extended the max-supported context of LLMs from 2K to up to 2M tokens(Liu et al., 2024a; Ding et al., 2024). Simultaneously, as the context lengths of



(a) Results on LLaMA2-7B in 32K context length.



(b) Results on LLaMA3-8B in 32K context length.

Figure 1: The comparison between LightCache and full attention with Dynamic NTK(bloc97, 2023a) on LLaMA2-7B(Touvron et al., 2023b) and LLaMA3-8B(Meta, 2024) in 32K context length from the perspective of LongBench performance(Bai et al., 2023b), memory cost and inference time. LightCache shows a consistent superiority in every dimension.

LLMs expand, researchers turn to focus on minimizing memory costs and enhancing computational efficiency while maintaining the long-context performance(Xiao et al., 2023b; Zhang et al., 2024a).

For a long-context Transformer-based LLM, the bottleneck of storage and computation cost primarily originates from the Key-Value (KV) Cache(Vaswani et al., 2017; Anagnostidis et al., 2024; Zhang et al., 2024c). For one transformer layer, the size of the KV Cache is 2 multiplied by the product of sequence length, number of attention heads, size of the feature dimension, and size of stored data type(Vaswani et al., 2017). From these aspects, different strategies for KV Cache Optimization have been derived. The mainstream methods for optimizing KV Cache mainly concentrate on dynamically pruning(Anagnostidis et al., 2024; Zhang et al., 2024c; Xiao et al., 2023b; Ge et al., 2023a; Dong et al., 2024) or compressing token representations along the sequence dimension(Ren et al., 2023; Pang et al., 2024; Zhang et al., 2024a); however, the operations along the sequence dimension often result in loss of contextual information, harming the downstream tasks(Dong et al., 2024; Lu et al., 2024). Another study focuses on optimization in data storage, namely quantization(Yue et al., 2024; Hooper et al., 2024; Liu et al., 2024b). Although most quantization methods commonly state lossless compression effects, they require the participation of a calibration set(Yue et al., 2024), which may introduce bias and cannot compress adaptively. Moreover, compressing along the attention head has also been a subject of persistent investigation, with representative works including MQA(Shazeer, 2019), GQA(Ainslie et al., 2023), and the recently emerging MLA(DeepSeek-AI, 2024). These approaches substantially reduce storage overhead and enhance inference speed but must be applied in pre-training and cannot offer plug-and-play compatibility.

In this work, we propose our KV Cache optimization method, LightCache. It is plug-and-play compatible and derived from the perspective of feature dimension. It surpasses the extrapolation effect of NTK methods as shown in Figure1 with much lower memory cost as well as a remarkable speedup. In contrast to quantization methods, we introduce parameter-aware compression based on matrix factorization, minimizing information loss mathmatically during efficient storage processes. Compared to context pruning and compression, we employ full-context token selection to ensure comprehensive coverage of crucial information within the complete context and demonstrate competitive performance on long-context tasks, showing superiority to NTK methods in testing. Our contributions can be summarized as follows.

- We find that in LLMs, the feature dimensions of the KV Cache are highly redundant. Particularly for the K Cache, the feature dimensions can be compressed to 1/16 of the original size while still preserving the original meaning.
- Building upon this observation, we propose LightCache, a training-free KV Cache optimization method. LightCache compresses the KV Cache along the feature dimension and selects a subset of the key entries to conduct self-attention.
- Experiments demonstrate that LightCache outperforms KV Cache pruning and quantization approaches in LongBench. Furthermore, it also exhibits better extrapolation performance than Dynamic NTK. Notably, LightCache achieves 1.7∼2.4× memory efficiency as well as 1.5∼3.6× speedup in 32K context length.

2 Method

The memory cost and inference latency of LLM mainly come from the storage and computational overhead of the KV Cache. To address this issue, LightCache proposes parameter-aware compression in the feature dimension and full-context selection in the sequential dimension and eventually addresses it with a training-free integration.

2.1 Parameter-Aware Compression

Compared to sequential optimization and cache quantization, feature dimension compression can retain complete contextual information while providing a parameter-aware compression scheme for KV Cache in every layer. Take the K Cache as an example. The K Cache $\mathcal{K}_{\text{cache}} = [\boldsymbol{k}_1, \boldsymbol{k}_2, \cdots, \boldsymbol{k}_t]$ is essentially a vector group obtained through the $\boldsymbol{W}_k$ projection. The vector space of $\boldsymbol{k}_s \in \mathbb{R}^{d_k}$, $s = 1, \cdots, t$, is determined by the singular vectors of $\boldsymbol{W}_k$, given the fact that

$$\boldsymbol{k}_s = \boldsymbol{h}_s \boldsymbol{W}_k^T, \quad s = 1, \cdots, t. \tag{1}$$

Therefore, by performing Singular Value Decomposition on $\boldsymbol{W}_k$, we can retain the projection matrix on principal dimensions to achieve high-quality compression of $\mathcal{K}_{\text{cache}}$.

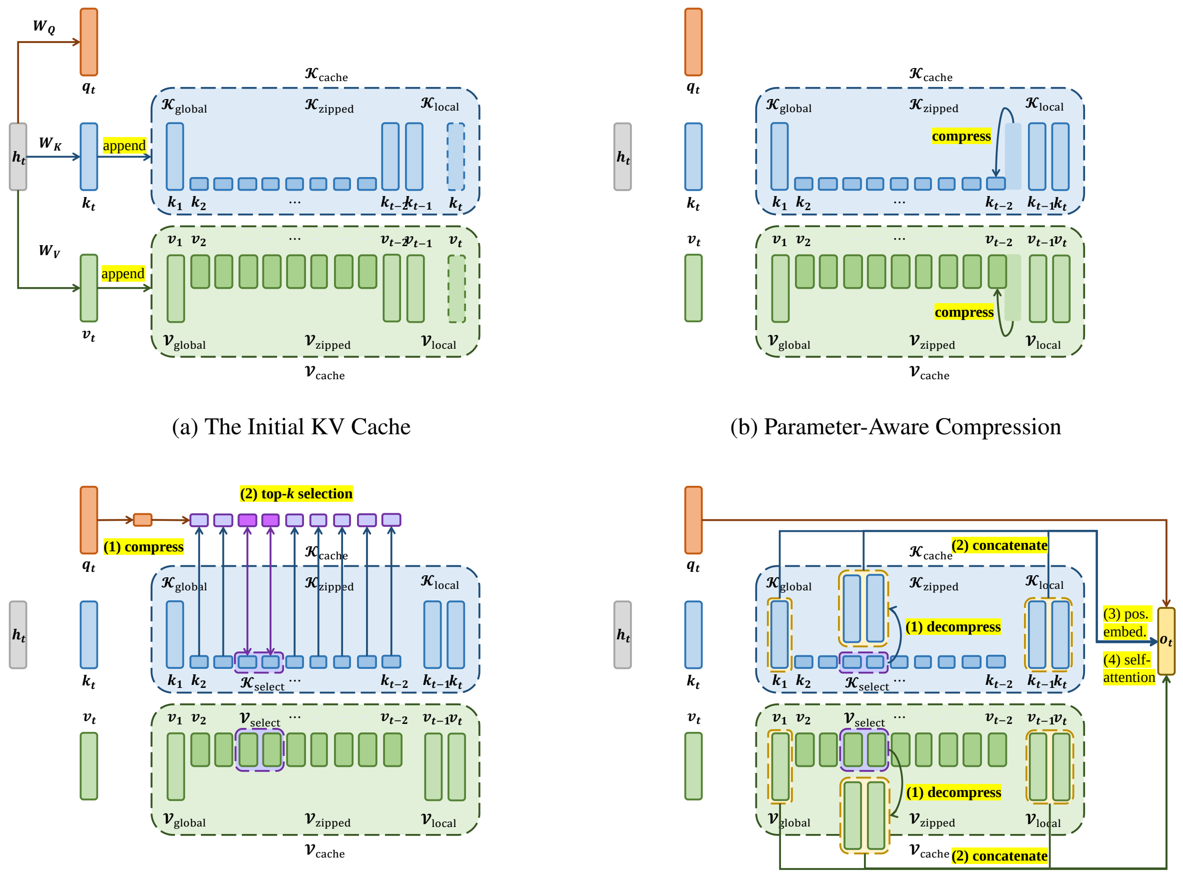
Specifically, we first decompose $\boldsymbol{W}_k$ into the product of three matrices, namely

$$\boldsymbol{W}_k = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T. \tag{2}$$

For the orthogonal matrix $\boldsymbol{U}_k \in \mathbb{R}^{d_k \times d_k}$, we only keep the first r_k row vectors corresponding to the principle components, and that is the compression matrix $\boldsymbol{P}_k$

$$\boldsymbol{U}_k' = \boldsymbol{U}_k[: r_k] \in \mathbb{R}^{r_k \times d_k}. \tag{3}$$

Therefore, for K Cache, projection by the pseudo-inverse(Moore, 1920) of $(\boldsymbol{U}_k')^T$, $\boldsymbol{P}_k$, can transform



W_Q
W_K
W_V
h_t
q_t
k_t
v_t
append
append
$\mathcal{K}_{\text{cache}}$
$\mathcal{K}_{\text{global}}$
$\mathcal{K}_{\text{zipped}}$
$\mathcal{K}_{\text{local}}$
$\mathcal{V}_{\text{global}}$
$\mathcal{V}_{\text{zipped}}$
$\mathcal{V}_{\text{local}}$
$\mathcal{V}_{\text{cache}}$
k_1 k_2 … k_{t-2} k_{t-1} k_t
v_1 v_2 … v_{t-2} v_{t-1} v_t
compress
compress
(1) compress
(2) top-k selection
$\mathcal{K}_{\text{select}}$
$\mathcal{V}_{\text{select}}$
(1) decompress
(1) decompress
(2) concatenate
(2) concatenate
(3) pos. embed.
(4) self-attention
o_t

(a) The Initial KV Cache

(b) Parameter-Aware Compression

(c) Full-Context Cache Selection

(d) Training-Free Integration

Figure 2: Visualization of the working principle of LightCache. In this example, we set the global preservation length to 1, the local preservation length to 2, the number of top-k selected segments $k = 2$, and the selected segment length $m = 1$. In Figure3a, similar to the normal KV Cache, the key-value vectors generated in the current step are first appended to the end. In Figure3b, the KV Cache entries that exceed the local part are compressed to a lower dimension, as detailed in Equation4. Specifically, for K Cache, we compress it to 1/16, while for V Cache, we only compress it to 1/2. In Figure2c, the query vector $\boldsymbol{q}_t$ is also compressed based on Equation5, and the top-k selection is performed on the reduced-dimension K Cache, as shown in Equation6, to find the critical K and their corresponding V. In Figure2d, finally, the extracted KV entries are decompressed based on Equation7, and then concatenated with the global and local preservation, with position embedding applied in sequence order before performing normal self-attention calculation as shown in Equation8.

it into the low-dimensional principal component space, thus achieving compression in feature dimension. On the other hand, V Cache compression follows the same method and may have different compressed dimensions r_v as follows,

$$\begin{aligned} \mathcal{K}_{\text{zipped}} &= \mathcal{K}_{\text{cache}} \boldsymbol{P}_k \in \mathbb{R}^{t \times r_k} \\ \mathcal{V}_{\text{zipped}} &= \mathcal{V}_{\text{cache}} \boldsymbol{P}_v \in \mathbb{R}^{t \times r_v} \end{aligned} . \tag{4}$$

Obviously, this compression strategy has the following advantages. It does not add any additional training parameters. After that, It is mathematically the compression scheme with the highest information density under the same order of magnitude. Compared to quantization, this compression does not require an additional calibration set and is adaptive to the parameters in each layer. Finally, the existence of the projection matrix $\boldsymbol{P}_k$ and its inverse makes it possible for the subsequent training-free integration. Additionally, to ensure the stability of the self-attention computation in LLM, inspired by StreamingLLM(Xiao et al., 2023b), LightCache retains the beginning and ending parts of the KV Cache, corresponding to the global and local context, without compression in feature dimension, to avoid disruption to the self-attention pattern.

2.2 Full-Context Selection

Since LightCache only performs compression in the feature dimension and does no pruning or compressing sequentially, it can attend to the full context of KV Cache during inference. However, since the training context of LLMs is limited and full attention is time-consuming, LightCache performs full-context cache selection before self-attention.

We first compress the query vector $\boldsymbol{q}_t \in \mathbb{R}_k^d$ in the current inference step in the same way as $\mathcal{K}_{\text{zipped}}$.

$$\boldsymbol{q}_t' = \boldsymbol{q}_t \boldsymbol{P}_k \in \mathbb{R}^{r_k}. \tag{5}$$

Then, based on the dot product between $\boldsymbol{q}_t'$ and $\mathcal{K}_{\text{zipped}}$, LightCache conducts a top-k selection and acquires the most critical K Cache entries, sorted in sequential order, along with their corresponding V Cache.

$$\begin{aligned} \text{Indices} &= \text{top-}k\left(\boldsymbol{q}_t' \mathcal{K}_{\text{zipped}}^T\right) \\ \mathcal{K}_{\text{select}} &= \mathcal{K}_{\text{zipped}}[\text{Indices}] \\ \mathcal{V}_{\text{select}} &= \mathcal{V}_{\text{zipped}}[\text{Indices}] \end{aligned} . \tag{6}$$

It's important to note that this dot product only considers semantic similarity, enabling information filtering irrespective of relative distance.

It's worth noting that for GQA, the feature dimension of $\boldsymbol{q}_t$ the feature dimension of key vectors, d_k, multiplied by the number of attention group, n_{group}. Therefore, LightCache needs first to reshape $\boldsymbol{q}_t'$ from a vector to a matrix in $\mathbb{R}^{n_{\text{group}} \times d_k}$, then compress it in the feature dimension and finally determine the selection is by voting across the attention groups. Furthermore, to improve the coherence of the selected entries and ensure complete coverage of important information, LightCache selects not just the critical cache entries, but the m neighboring entries as well with overlaps being deduplicated. This provides LightCache with greater flexibility compared to fixed-chunk efficient caching methods(Xiao et al., 2024; Lu et al., 2024).

2.3 Training-Free Integration

Finally, LightCache only performs self-attention calculations on the preserved and selected cache. For the selected KV Cache with lower dimensions, LightCache uses the aforementioned $\boldsymbol{U}_k', \boldsymbol{U}_v'$ to perform decompression.

$$\begin{aligned} \mathcal{K}_{\text{select}}' &= \mathcal{K}_{\text{select}} \boldsymbol{U}_k'^T \\ \mathcal{V}_{\text{select}}' &= \mathcal{V}_{\text{select}} \boldsymbol{U}_v'^T \end{aligned} . \tag{7}$$

This ensures that the shape of the selected KV Cache matches the preserved KV Cache, allowing them to be directly concatenated. After applying positional encoding to the concatenated sequence, LightCache performs the self-attention directly.

$$\begin{aligned} \mathcal{K}_{\text{cache}}' &= \left[\mathcal{K}_{\text{global}}, \mathcal{K}_{\text{select}}', \mathcal{K}_{\text{local}}\right] \\ \mathcal{V}_{\text{cache}}' &= \left[\mathcal{V}_{\text{global}}, \mathcal{V}_{\text{select}}', \mathcal{V}_{\text{local}}\right] \\ \boldsymbol{o}_t &= \text{SelfAttn}\left(\boldsymbol{q}_t, \mathcal{K}_{\text{cache}}', \mathcal{V}_{\text{cache}}'\right) \end{aligned} . \tag{8}$$

This approach enables the training-free integration of LightCache with pre-trained LLMs. By controlling the number of selected KV Cache, k, LightCache achieves computational efficiency while limiting the self-attention calculation window to the pre-training context length. This avoids the OOD issue of position information and extends the actual context length that LLMs can handle. Additionally, in the prompt reading stage, LightCache processes the input chunk-wise and streamingly(Xiao et al., 2023b, 2024) to control GPU memory cost and self-attention window length.

3 Experiments

3.1 Setup

We conduct experiments on LLaMA2-7B-4k(Touvron et al., 2023b), LLaMA3-8B-8K(Meta, 2024), InternLM2-7B-200K(Cai et al., 2024), InternLM2-1.8B-32K(Cai et al., 2024), Qwen1.5-7B-128K(Bai et al., 2023a), Qwen1.5-1.8B-32K(Bai et al., 2023a). For all models, we set the global preservation length to 4, the number of top-k selected segments $k = 16$, the selected segment length $m = 32$, and the chunk size of streaming input to 1K. For LLaMA2-7B, we set the local context length to 2K, for LLaMA3B we set it to 4K, and for the rest of the models, we set it to 16K. The reason for this setting is that it allows the attention window size close to the pre-training context length, thus adapting LLM to the length distribution learned in the pre-training stage.

3.2 Main Results

We use the OpenCompass(Contributors, 2023c) to validate the effectiveness of our method on the commonly used long-context benchmark LongBench(Bai et al., 2023b), with an evaluation context length of 32k and middle truncation. For LLaMA2-7B(Touvron et al., 2023b) and LLaMA3-8B(Meta, 2024), we use Dynamic NTK(bloc97, 2023a) and StreamingLLM(Xiao et al., 2023b) as our baselines. For other LLMs with long-context capability, we compare our LightCache with Int8 and Int4. Except for the quantization experiments, other experiments take BFloat16 precision, and the quantization is supported by the

	Pretrained		LightCache (Ours)				
	Dynamic NTK	**StreamingLLM**	$v\times$**1/2**	$k\times$**1/8**	$k\times$**1/8** $v\times$**1/2**	$k\times$**1/16**	$k\times$**1/16** $v\times$**1/2**
Compressing Ratio	0.00%	**93.59%**	17.00%	39.34%	56.34%	43.07%	60.07%
LLaMA2-7B-4K	28.15	26.70	28.09	28.58	28.17	**28.80**	28.02
LLaMA3-8B-8K	28.27	31.58	**33.56**	33.13	32.94	32.67	32.58

Table 1: Evaluation results of LLaMA2-7B and LLaMA3-8B on LongBench(Bai et al., 2023b) as well as the cache efficiency, the percentage representing the amount of KV Cache reduced, in 32K context length. LightCache shows a better performance than Dynamic NTK(bloc97, 2023a) and StreamingLLM(Xiao et al., 2023b).

	Pretrained			LightCache (Ours)				
	BFloat16	**Int8**	**Int4**	$v\times$**1/2**	$k\times$**1/8**	$k\times$**1/8** $v\times$**1/2**	$k\times$**1/16**	$k\times$**1/16** $v\times$**1/2**
Compressing Ratio	0.00%	50.00%	**75.00%**	17.00%	39.34%	56.34%	43.07%	60.07%
InternLM2-7B-200K	42.93	**44.94**	43.76	42.45	42.53	42.47	42.42	42.41
InternLM2-1B-32K	**33.01**	32.73	32.19	32.66	32.74	32.68	32.70	32.63
Qwen1.5-7B-128K	**39.37**	34.63	34.41	39.10	39.06	39.04	39.07	39.05
Qwen1.5-1B-32K	**29.22**	25.66	24.60	29.20	29.11	29.19	29.16	29.20

Table 2: Evaluation results of InternLM2 and Qwen1.5 on LongBench(Bai et al., 2023b) as well as the cache efficiency in 32K context length. LightCache shows a better performance than Int8/Int4 quantization on average and a similar compression ratio.

LMDeploy(Contributors, 2023b); all experiments are accelerated with FlashAttention2(Dao, 2023) and conducted on one A100 GPU.

For LightCache, we compare the effects of compressing only V Cache to 1/2, compressing only K Cache to 1/8 or 1/16, and compressing both. While comparing the experimental effects, we also compare the compressing ratio at the 32k length, namely the percentage of KV Cache being reduced

$$\frac{\mathcal{M}_{\text{full}} - \mathcal{M}}{\mathcal{M}_{\text{full}}} \times 100\%.$$

Here, $\mathcal{M}$ is the size of preserved KV Cache for a certain optimization strategy while $\mathcal{M}_{\text{full}}$ is the size of KV Cache in full attention. For LightCache, the size of the projection matrix, $\mathcal{M}_{\text{proj}}$, should also be considered, so the compressing ratio is

$$\frac{\mathcal{M}_{\text{full}} - \mathcal{M} + \mathcal{M}_{\text{proj}}}{\mathcal{M}_{\text{full}}} \times 100\%.$$

As shown in Table 1, for the experiments on LLaMA2 and LLaMA3, LightCache significantly outperforms the StreamingLLM baseline in performance. Compared to the full attention with Dynamic NTK, our method can not only optimize 40%-60% of the KV Cache but also acquire a slight performance advantage. It should be noted that since LLaMA3 uses a larger rotary base(Meta, 2024), the NTK method has limited improvement on its long-text capability(Liu et al., 2023), and therefore, Dynamic NTK is not as good as StreamingLLM. In contrast, LightCache is independent of whether GQA or how much the rotary base is used and thus achieves stable downstream improvements on both LLaMA2 and LLaMA3.

As shown in Table 2, for the experiments on InternLM and Qwen1.5, LightCache can outperform the quantization methods on average with similar compression strength and is closer to the original performance of full attention. Except for InternLM2-7B, where the quantization method can surpass LightCache and even the original full attention, for other models, especially the Qwen1.5 series, the quantization method presents a significant decline, proving that it suffers from biases introduced by parameter adaptation and calibration sets. Moreover, the compression of K Cache by 1/16 in LightCache is equivalent to 1-bit quantization in terms of storage, so we have achieved compression effects superior to the quantization methods in this sense.

Further analysis of the results in Table 6 and Table 2 reveals that the performance degradation from compressing V Cache is more severe than compressing K Cache. Under the condition of compressing K Cache to 1/16 of its original size, Light-

	GPU Memory (GB)					Inference Time (sec. per token)				
	4K	8K	16K	32K	100K	4K	8K	16K	32K	100K
Full Attention	<u>15.7</u>	18.4	24.0	35.0	OOM	<u>0.043</u>	0.070	0.126	0.219	OOM
StreamingLLM	**15.3**	**15.3**	**15.3**	**15.3**	**15.3**	**0.036**	**0.035**	**0.035**	**0.035**	**0.036**
AutoCompressor*	17.7	22.6	32.3	51.7	OOM	0.087	0.134	0.224	0.478	OOM
LongLlama*	18.2	21.9	34.2	OOM	OOM	0.079	0.190	0.436	OOM	OOM
Activation Beacon*	21.7	21.3	21.4	21.6	<u>21.6</u>	0.071	0.121	0.237	0.473	1.494
LightCache (ours)	16.7	<u>17.2</u>	<u>18.3</u>	<u>20.6</u>	30.4	0.051	<u>0.050</u>	<u>0.054</u>	<u>0.061</u>	<u>0.092</u>

Table 3: Evaluation of inference time and GPU memory cost. Both are measured by the average value of 20 inference passes with FlashAttention2(Dao, 2023) enabled. The starred results of AutoCompressor(Chevalier et al., 2023), LongLLaMA(Tworkowski et al., 2023) and Activation Beacon(Zhang et al., 2024a) is acquired from Zhang et al. (2024a). LightCache shows slower memory growth and a much faster speed compared with full attention.

Cache can still achieve results close to or even better than that of full attention. This is also why LightCache can achieve over 60% compression, and approach 75% compression in infinite input length. However, the compression of V Cache is very limited, and on the LLaMA series, compressing V Cache will lead to certain performance degradation, which is consistent with the conclusion from some matrix compression research that the V projection matrix is more difficult to compress(Peng et al., 2024). In the following experiments, we will continue to use the default setting of 1/16 optimization for the K Cache and 1/2 optimization for the V Cache.

3.3 Efficiency

We use the commonly used long-text dataset PG19(Rae et al., 2019) to evaluate the efficiency of our method in processing long contexts and compare the results of LightCache with that of full attention, as well as commonly used context pruning methods such as StreamingLLM(Xiao et al., 2023b), AutoCompressor(Chevalier et al., 2023), LongLLaMA(Tworkowski et al., 2023), and Activation Beacon(Zhang et al., 2024a).

As shown in Table 3, LightCache can save over 60% of memory cost at a 32K context length and achieve over $3\times$ speedup in inference, only inferior to StreamingLLM, which discards a large number of tokens. Although at 100K input length, the memory cost of feature dimension compression still grows slowly with sequence length and is not as good as the token compression method of Activation Beacon, its inference speed is still very fast, almost 15 $\times$ its counterpart.

4 Analysis

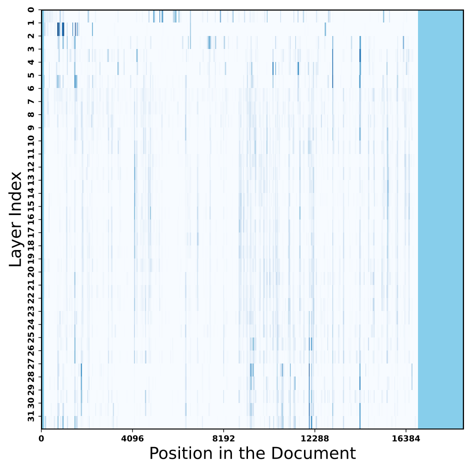
4.1 Case Study

To verify that the top-k operation can help select the key information from the previous text, we conduct a case study on the QA and summarization tasks from the LongBench, using data from the NarrativeQA(Kočiskỳ et al., 2018) and GovReport(Huang et al., 2021). We track the token selection during the text generation process at all layers in LLaMA2-7B with LightCache. The results are shown in Figure3. The blue area represents the global and local contexts stored without compression as well as the cache decompressed after top-k selection. The more blue the color, the more times the corresponding token was selected to attend to during self-attention.

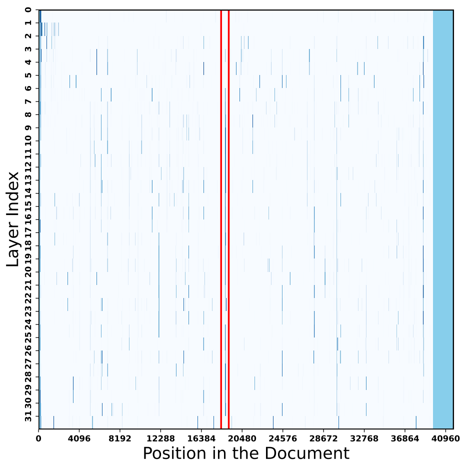
For the summarization task, we found that LightCache could select the key cached fragments regardless of the distance and thus generally cover the information in the context during the generation. For the QA task, we find that LightCache can identify the key information in the context, highlighted in the red box, and pay continuous attention to it, especially in the layers closer to the output, thereby answering the questions correctly. This proves that the process of compression in feature dimension, top-k selection, decompression, and concatenation for attention in LightCache is reasonable and effective.

4.2 Discussion

We then discussed the compression of the V cache, the hyperparameters for the token selection in the middle, and the size of the local context. We report the changes in the average score of LongBench when taking different model settings, as shown in Table 4 and Table 5.



(a) Case study on GovReport



(b) Case study on NarrativeQA

Figure 3: Visualization of KV Cache selected during the inference at each layer represented by color blocks. For NarrativeQA, the paragraph containing the answer is highlighted with a red box.

Method Setting	LongBench Avg.
LightCache	**28.02**
$\boldsymbol{v}\times 1/4$	27.46
$\boldsymbol{v}\times 1/4$, $\boldsymbol{k}\times 1/16$	27.71
$\boldsymbol{v}\times 1/8$	27.05
$\boldsymbol{v}\times 1/8$, $\boldsymbol{k}\times 1/16$	27.21
k=4, m=128	27.86
k=64, m=8	27.97
k=512, m=1	26.66

Table 4: Comparison based on LLaMA2-7B of the different compression ratios of V Cache and the different top-k selection settings on LongBench.

Method Setting	LongBench Avg.
LightCache	**42.41**
local size l=8192	40.40
local size l=4096	37.71
local size l=2048	33.22

Table 5: Comparison based on InternLM2-7B of the different local attention size on LongBench.

Compression of V Cache In this work, the compression of V is significantly less than that of K. The results obtained by further compressing V are shown in Table 4. It can be found that further compressing V will have an obvious negative impact on the downstream tasks. This is mainly because, as reported in some studies, the projection matrix $\boldsymbol{W}_V$ has a relatively high rank compared with $\boldsymbol{W}_K$(Peng et al., 2024) and V Cache is more essential in self-attention transform(Liu et al., 2024b). Therefore, it is not appropriate to apply excessive compression.

Number of Token Selection For the hyperparameters for the token selection in the middle, namely the number of top-k selected segments k and the selected segment length m, we compare the effects of different k, m pairs while keeping $k \times m$ the same. We find that our default setting with $k = 16, m = 32$ achieved the best results. For the setting, $k = 1, m = 512$, where only a single entry is selected each time, we found that although theoretically, it can acquire the most entries from compressed KV Cache, the entries are too scattered to form continuous semantics. Therefore the results show a significant decline. Additionally, for the setting that continuously selects more entries, like $k = 4, m = 128$, the targeted entries are easy to miss, which also leads to a decline.

Size of Local Context: In this paper, we use a local size of 16K for InternLM2 and Qwen1.5 models, much larger than that for LLaMA2 and LLaMA3. We compare the downstream performance of InternLM2-7B under different local sizes, since the context length of LLaMA2 and LLaMA3 are limited, and find that when local size is reduced, the downstream task performance of the models declines. This is mainly because these models have already been trained on longer contexts, and have adapted to the position information features of longer lengths(Liu et al., 2023). If a smaller local size is retained, it is detrimental to their downstream task performance, and increasing the intermediate selection tokens will also incur a certain computational overhead.

5 Related Work

5.1 KV Cache Optimization

KV Cache Optimization is a critical research area in LLMs based on Transformer architecture and is crucial for improving efficiency and capabilities in processing long contexts. It can be categorized into optimization from the perspectives of layer(Yen et al., 2024; Sun et al., 2024), sequence length, feature dimensions, and storage data types.

optimization from the sequence dimension includes different implementation methods such as pruning, compression, rolling, and retrieval. Nawrot et al. (2024); Zhang et al. (2024c); Xiao et al. (2023b); Ge et al. (2023a) enables LLMs to automatically prune the context during inference with key information being preserved. Though these works can enhance the inference speed greatly, the pruned context may lead to a performance decline. Besides, Mu et al. (2024); Ren et al. (2023); Pang et al. (2024); Zhang et al. (2024a) compress and replace context segments with special tokens, which can prune up to 99% of the context, but generally require fine-tuning and lack evaluation on long-context. Besides, Chevalier et al. (2023) and Ge et al. (2023b), borrow the idea of chunked input and memory from Dai et al. (2019); Bulatov et al. (2022) to improve the LLM efficiency in processing long contexts. Finally, Mohtashami and Jaggi (2024); Xiao et al. (2024); Lu et al. (2024) apply KV chunk representations, CPU offload, and partial recall to reduce memory cost, but need to suffer the additional CPU-GPU traffic.

Additionally, quantization methods optimize the KV Cache from the data type perspective, Xiao et al. (2023a); Hooper et al. (2024); Yue et al. (2024); Liu et al. (2024b) have proposed fine-designed optimization strategies tailored to the KV Cache features, achieving up to 2-bit optimization. While these methods claim to achieve nearly lossless optimization, they still suffer from biases introduced by parameter adaptation and calibration sets. From the head perspective, MQA(Shazeer, 2019) and MHA(Vaswani et al., 2017) share KV Cache among the queries to achieve efficiency. Recently, MLA(DeepSeek-AI, 2024) combines the minor dimensions of MHA and the major dimensions of MQA to balance efficiencye an performancd. However, these works can only be used in pre-training. Past research lacks the exploration of KV Cache optimization in the feature dimension. A few works, such as SparQ(Ribar et al., 2023), have discussed the potential of low-dimensional mapping to filter a small number of tokens into self-attention to accelerate, but do not mention its potential in memory saving.

5.2 Length Extrapolation

Long text processing is a key application of KV Cache compression, while length extrapolation is an important issue in long text processing. Length extrapolation means making LLMs maintain stable downstream performance in long contexts despite being trained in short contexts(Press et al., 2022). Extrapolation research stems from Linear PI(Chen et al., 2023), compressing unseen position information into the training window. Similarly, ReRoPE(Su, 2023), SelfExtend(Jin et al., 2024), and ChunkLLaMA(An et al., 2024) use relative position limits. In contrast, bloc97 (2023a) proposes the lossless NTK method that amplifies rotation angles during inference. Subsequent work combines enlarging the rotary base(Su et al., 2021) with fine-tuning on longer context as the mainstream approach to extend model context(Rozière et al., 2023; Liu et al., 2023; Xiong et al., 2023). Extending context increases storage and computation, leading to work on extending input length, from parallel windows to LM-Infinite(Han et al., 2023) and concurrent StreamingLLM(Xiao et al., 2023b), which are part of KV Cache Optimization. Our work extends model context while improving efficiency, surpassing NTK on long text tasks.

6 Conclusion

We propose LightCache, a novel, training-free framework to improve the efficiency of LLMs in long contexts. Based on the parameter-aware compression of the feature dimension of the KV Cache, LightCache significantly enhances the storage efficiency while achieving lossless storage of contextual information. Through global top-k selection and subsequent decompression, we achieve computational efficiency, extended context window, and plug-and-play capability. Experiments demonstrate that LightCache can achieve performance on par with or even exceeding full attention, and is competitive compared to quantization and StreamingLLM approaches. Our method provides a new approach to KV Cache optimization and length extrapolation for LLMs.

Limitations

The main limitations of this work are as follows. First, We do not evaluate LightCache on the Needle-In-a-Haystack benchmark(Contributors, 2023a), so there is a lack of comprehensive verification on whether compression on the feature dimension can truly preserve contextual information without loss, and whether top-k selection can achieve distance-free retrieval in LLMs. Besides, From the performance perspective, we lack comparisons to works like Landmark Attention(Mohtashami and Jaggi, 2024), InfLLM(Xiao et al., 2024) and LongHeads(Lu et al., 2024). From the efficiency perspective, we lack comparisons to quantization methods. From the evaluation perspective, we lack evaluation on context over 100K(Zhang et al., 2024b), mainly due to limited resources and time. Additionally, LightCache only conducts compression on the feature dimension rather than the sequence dimension, so when the sequence length reaches a certain point, it will still exceed the GPU memory limit. For example, for LLaMA2-7B with LightCache, when the context window reaches 400K, a single A800 GPU cannot support the inference. Finally, We did not integrate LightCache with efficient frameworks like vLLM(Kwon et al., 2023), which is planned for our future work.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.

Chenxin An, Shansan Gong, Ming Zhong, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *CoRR*, abs/2307.11088.

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. 2024. Training-free long-context scaling of large language models. *arXiv preprint arXiv:2402.17463*.

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. 2024. Dynamic context pruning for efficient and interpretable autoregressive transformers. *Advances in Neural Information Processing Systems*, 36.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023a. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023b. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

bloc97. 2023a. Dynamically scaled rope further increases performance of long context llama with zero fine-tuning.

bloc97. 2023b. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report. *Preprint*, arXiv:2403.17297.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *CoRR*, abs/2306.15595.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846.

Contributors. 2023a. Needle in a haystack - pressure testing llms. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`.

LMDeploy Contributors. 2023b. Lmdeploy: A toolkit for compressing, deploying, and serving llm. `https://github.com/InternLM/lmdeploy`.

OpenCompass Contributors. 2023c. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*, abs/2307.08691.

DeepSeek-AI. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*.

Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. 2024. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. *arXiv preprint arXiv:2402.09398*.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023a. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023b. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *CoRR*, abs/2308.16137.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. 2024a. World model on million-length video and language with ringattention. *arXiv preprint arXiv:2402.08268*.

Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. 2023. Scaling laws of rope-based extrapolation. *CoRR*, abs/2310.05209.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*.

Yi Lu, Xin Zhou, Wei He, Jun Zhao, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longheads: Multi-head attention is secretly a long context processor. *arXiv preprint arXiv:2402.10685*.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*.

Amirkeivan Mohtashami and Martin Jaggi. 2024. Random-access infinite context length for transformers. *Advances in Neural Information Processing Systems*, 36.

Eliakim H Moore. 1920. On the reciprocal of the general algebraic matrix. *Bulletin of the american mathematical society*, 26:294–295.

Jesse Mu, Xiang Li, and Noah Goodman. 2024. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M Ponti. 2024. Dynamic memory compression: Retrofitting llms for accelerated inference. *arXiv preprint arXiv:2403.09636*.

OpenAI. 2023. Gpt-4 technical report. Technical report.

Jianhui Pang, Fanghua Ye, Derek F Wong, and Longyue Wang. 2024. Anchor-based large language models. *arXiv preprint arXiv:2402.07616*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *CoRR*, abs/2309.00071.

Runyu Peng, Yunhua Zhou, Qipeng Guo, Yang Gao, Hang Yan, Xipeng Qiu, and Dahua Lin. 2024. Data-freeweight compress and denoise for large language models. *arXiv preprint arXiv:2402.16319*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Siyu Ren, Qi Jia, and Kenny Zhu. 2023. Context compression for auto-regressive transformers with sentinel tokens. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12860–12867.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. 2023. Sparq attention: Bandwidth-efficient llm inference. *arXiv preprint arXiv:2312.04985*.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.

Jianlin Su. 2023. Rerope: Rectified rotary position embeddings.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. 2024. You only cache once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2023. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Song Han, and Maosong Sun. 2024. Infllm: Unveiling the intrinsic capacity of llms for understanding extremely long sequences with training-free memory. *arXiv preprint arXiv:2402.04617*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023a. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023b. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. 2023. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. *arXiv preprint arXiv:2402.16617*.

Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. 2024. Wkvquant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024a. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, et al. 2024b. Inifinitebench: Extending long context evaluation beyond 100k tokens. *arXiv preprint arXiv:2402.13718*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2024c. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36.

A Appendix

The detailed results of LightCache on LongBench are shown in Table6.

		Single-Doc QA	Multi-Doc QA	Sum	Few-shot	Syn.	Code	Text	Avg.
LLaMA2-7B	Dynamic NTK	16.14	7.89	11.61	50.58	3.68	63.34	18.77	28.15
	StreamingLLM	14.03	7.85	11.03	46.91	3.31	61.69	17.37	26.70
	$\boldsymbol{v}\times 1/2$	16.61	8.61	11.19	49.71	4.56	62.64	18.88	28.09
	$\boldsymbol{k}\times 1/8$	16.53	8.33	12.79	50.52	4.14	63.60	19.25	28.58
	$\boldsymbol{k}\times 1/8$, $\boldsymbol{v}\times 1/2$	16.67	8.01	11.56	50.18	4.54	62.79	18.94	28.17
	$\boldsymbol{k}\times 1/16$	16.46	8.55	12.62	50.41	5.31	64.02	19.41	**28.80**
	$\boldsymbol{k}\times 1/16$, $\boldsymbol{v}\times 1/2$	15.87	8.21	11.15	50.24	4.05	63.08	18.67	28.02
LLaMA3-8B	Dynamic NTK	16.28	11.59	17.74	58.14	8.58	46.96	23.29	28.27
	StreamingLLM	13.65	9.04	16.25	56.94	5.70	70.58	21.18	31.58
	$\boldsymbol{v}\times 1/2$	15.63	9.93	20.11	61.50	7.42	70.02	23.84	**33.56**
	$\boldsymbol{k}\times 1/8$	14.10	9.70	18.96	60.82	8.52	70.10	23.27	33.13
	$\boldsymbol{k}\times 1/8$, $\boldsymbol{v}\times 1/2$	14.46	9.71	18.64	60.20	7.45	70.32	22.97	32.94
	$\boldsymbol{k}\times 1/16$	14.15	9.48	18.81	59.20	7.40	70.15	22.67	32.67
	$\boldsymbol{k}\times 1/16$, $\boldsymbol{v}\times 1/2$	14.30	9.64	18.62	58.61	6.65	70.57	22.46	32.58
InternLM2-7B	bfloat16	41.62	35.21	22.88	57.30	34.67	59.61	38.48	42.93
	int8	40.02	36.68	23.59	60.28	40.33	62.79	40.18	**44.94**
	int4	39.27	36.13	23.57	60.58	33.00	62.38	38.79	43.76
	$\boldsymbol{v}\times 1/2$	41.62	35.00	22.83	56.18	33.00	59.42	37.92	42.45
	$\boldsymbol{k}\times 1/8$	41.58	34.89	22.73	56.57	33.17	59.58	37.98	42.53
	$\boldsymbol{k}\times 1/8$, $\boldsymbol{v}\times 1/2$	41.52	34.16	22.76	56.34	33.33	59.99	37.80	42.47
	$\boldsymbol{k}\times 1/16$	41.29	34.92	22.84	56.23	33.00	59.53	37.86	42.42
	$\boldsymbol{k}\times 1/16$, $\boldsymbol{v}\times 1/2$	41.01	34.28	22.54	56.66	33.50	59.80	37.77	42.41
InternLM2-1B	bfloat16	29.74	25.58	20.20	51.10	3.41	54.96	27.16	**33.01**
	int8	27.44	26.13	20.29	51.13	3.44	54.78	26.85	32.73
	int4	27.44	24.08	19.95	50.44	3.54	54.62	26.21	32.19
	$\boldsymbol{v}\times 1/2$	28.89	24.82	20.17	50.43	3.65	54.96	26.72	32.66
	$\boldsymbol{k}\times 1/8$	28.90	25.21	20.04	50.62	3.65	54.96	26.81	32.74
	$\boldsymbol{k}\times 1/8$, $\boldsymbol{v}\times 1/2$	28.94	24.95	20.16	50.42	3.62	54.92	26.75	32.68
	$\boldsymbol{k}\times 1/16$	28.81	25.27	20.09	50.43	3.61	54.93	26.77	32.70
	$\boldsymbol{k}\times 1/16$, $\boldsymbol{v}\times 1/2$	28.79	24.92	20.05	50.40	3.61	54.94	26.68	32.63
Qwen1.5-7B	bfloat16	37.64	22.52	24.68	60.85	6.50	68.45	31.62	**39.37**
	int8	22.37	14.38	22.21	59.89	4.60	67.79	25.79	34.63
	int4	21.50	14.02	22.18	59.90	4.97	67.48	25.60	34.41
	$\boldsymbol{v}\times 1/2$	36.62	22.34	24.58	60.84	6.39	68.22	31.34	39.10
	$\boldsymbol{k}\times 1/8$	36.65	22.32	24.42	60.67	6.50	68.23	31.28	39.06
	$\boldsymbol{k}\times 1/8$, $\boldsymbol{v}\times 1/2$	36.51	22.35	24.38	60.78	6.50	68.16	31.28	39.04
	$\boldsymbol{k}\times 1/16$	36.73	22.42	24.30	60.77	6.33	68.23	31.29	39.07
	$\boldsymbol{k}\times 1/16$, $\boldsymbol{v}\times 1/2$	36.59	22.25	24.25	60.96	6.33	68.28	31.26	39.05
Qwen1.5-1B	bfloat16	24.77	15.22	20.41	44.35	3.00	54.45	22.50	**29.22**
	int8	13.60	8.37	18.84	42.04	3.50	54.18	18.05	25.66
	int4	12.68	8.36	18.29	40.99	3.74	50.99	17.56	24.60
	$\boldsymbol{v}\times 1/2$	24.76	15.18	20.39	44.19	2.99	54.55	22.44	29.20
	$\boldsymbol{k}\times 1/8$	24.79	15.26	20.31	43.83	3.00	54.38	22.37	29.11
	$\boldsymbol{k}\times 1/8$, $\boldsymbol{v}\times 1/2$	24.78	15.13	20.31	44.24	3.00	54.53	22.44	29.19
	$\boldsymbol{k}\times 1/16$	24.77	15.04	20.32	44.46	2.99	54.31	22.46	29.16
	$\boldsymbol{k}\times 1/16$, $\boldsymbol{v}\times 1/2$	24.77	15.17	20.32	44.34	3.00	54.45	22.46	29.20

Table 6: Citation commands supported by the style file. The style is based on the natbib package and supports all natbib citation commands. It also supports commands defined in previous ACL style files for compatibility.