Efficient Low Rank Attention for Long-Context Inference in Large Language Models

Tenghui Li
Guangdong University of Technology
RIKEN AIP
tenghui.lee@foxmail.com

Guoxu Zhou [*]
Guangdong University of Technology
Key Laboratory of Intelligent Detection and the
Internet of Things in Manufacturing,
Ministry of Education, Guangzhou, CHINA
gx.zhou@gdut.edu.cn

Xuyang Zhao
RIKEN iTHEMS
RIKEN IMS
Chiba University
xuyang.zhao@riken.jp

Yuning Qiu
RIKEN AIP
yuning.qiu@riken.jp

Qibin Zhao
RIKEN AIP
qibin.zhao@riken.jp

Abstract

As the length of input text increases, the key-value (KV) cache in LLMs imposes prohibitive GPU memory costs and limits long-context inference on resource constrained devices. Existing approaches, such as KV quantization and pruning, reduce memory usage but suffer from numerical precision loss or suboptimal retention of key-value pairs. In this work, Low Rank Query and Key attention (LRQK) is introduced, a two-stage framework that jointly decomposes full-precision query and key matrices into compact rank-r factors during the prefill stage, and then employs these low-dimensional projections to compute proxy attention scores in $\mathcal{O}(lr)$ time at each decode step. By selecting only the top-k tokens and a small fixed set of recent tokens, LRQK employs a mixed GPU-CPU cache with a hit-and-miss mechanism where only missing full-precision KV pairs are transferred, thereby preserving exact attention outputs while reducing CPU-GPU data movement. Extensive experiments on the RULER and LongBench benchmarks with LLaMA-3-8B and Qwen2.5-7B demonstrate that LRQK matches or surpasses leading sparse-attention methods in long context settings, while delivering significant memory savings with minimal accuracy loss. Our code is available at https://github.com/tenghuilee/LRQK.

1 Introduction

Large language models (LLMs) have shown their remarkable capabilities across diverse tasks. Recent LLMs have extended their capabilities to support long-context processing, enabling them to leverage extended sequences for improved performance in applications such as document level understanding and long-form text generation [1–5]. Despite these advances, efficient processing of long-contexts presents substantial challenges. The primary constraint stems from computational resource limitations, particularly regarding memory consumption. The KV cache stores historical key-value pairs to prevent redundant computations. However, its size grows linearly with sequence length and quickly imposes prohibitive memory requirements when handling long inputs. Consequently, as context windows expand, the KV cache correspondingly increases in size, eventually becoming the system bottleneck.

[*] corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

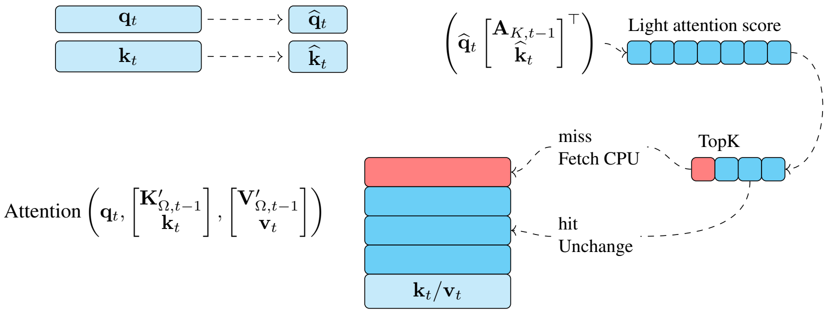

$\mathbf{q}_t$
$\mathbf{k}_t$
$\widehat{\mathbf{q}}_t$
$\widehat{\mathbf{k}}_t$
$\left(\widehat{\mathbf{q}}_t \begin{bmatrix} \mathbf{A}_{K,t-1} \\ \widehat{\mathbf{k}}_t \end{bmatrix}^\top\right)$
Light attention score
miss
Fetch CPU
TopK
hit
Unchange
Attention $\left(\mathbf{q}_t, \begin{bmatrix} \mathbf{K}'_{\Omega,t-1} \\ \mathbf{k}_t \end{bmatrix}, \begin{bmatrix} \mathbf{V}'_{\Omega,t-1} \\ \mathbf{v}_t \end{bmatrix}\right)$
$\mathbf{k}_t/\mathbf{v}_t$

Figure 1: Brief overview of the proposed Low Rank Query and Key attention (LRQK) method. Subscript Ω denotes the selected tokens, t denotes the current token. $\mathbf{q}_t, \mathbf{k}_t$ are the original query and key, $\widehat{\mathbf{q}}_t, \widehat{\mathbf{k}}_t$ are the approximated query and key. $\mathbf{A}_{K,t}$ is the low rank key matrix. $\mathbf{K}'_{\Omega,t-1}, \mathbf{V}'_{\Omega,t-1}$ are GPU cache $\mathbf{K}_{\Omega,t-1}, \mathbf{V}_{\Omega,t-1}$ merged with fetched CPU keys and values.

Existing approaches addressing the KV cache memory bottleneck can be systematically categorized into three fundamental methodologies: (1) quantization techniques [6–9], which reduce numerical precision while preserving semantic integrity; (2) pruning mechanisms [10–12], which selectively retain critical tokens while eliminating redundant ones; and (3) offloading strategies [13–15], which redistribute memory across heterogeneous storage hierarchies. Additionally, hybrid approaches [16] combine multiple techniques to maximize efficiency.

While these methods have made significant strides in addressing the KV cache memory bottleneck, they each come with inherent limitations. Quantization methods will sacrifice numerical precision, and the evincing techniques risk eliminating potentially crucial key-value pairs, particularly problematic when tokens deemed unimportant in current contexts later become critical for subsequent reasoning. Pruning methods may inadvertently discard important key-value pairs, leading to suboptimal performance when previously unimportant tokens become critical in subsequent reasoning. Offloading strategies introduce substantial latency overhead due to frequent PCIe data transfers, severely impacting real time processing capabilities.

Consequently, an ideal solution should balance the trade-offs between memory efficiency, numerical precision, and computational latency. A key observation in decoder-only transformer LLMs is the inherent sparsity of attention activation patterns during inference. It had been shown that only a small subset of tokens contributes significant attention weights at each generation step [11, 10, 16, 12]. This sparsity also shows that the possibility of offloading of inactive key-value pairs to cheaper CPU memory while retaining active pairs in GPU memory. However, a fundamental technical challenge emerges when implementing this strategy. Full attention computation requires complete access to key tokens, which are not efficiently accessible after offloading.

To mitigate these challenges, a hybrid attention mechanism is proposed to exploit the intrinsic sparsity of transformer attention patterns while preserving exact computation of attention outputs. At its core lies the observation that full query-key interaction matrix in decoder-only LLMs admits a close low rank approximation. Therefore, the query matrix and key matrix are jointly factorized into compact bases. In the decoding phase, a "proxy" attention score is computed via these factorized query and key matrices, and identifies the top-k relevant tokens in the GPU cache. Thus, the ground truth attention score can be carried out on the retrieved subset of $\mathbf{k}$ and $\mathbf{v}$, guaranteeing numerical fidelity. An overview of the proposed method is shown in Figure 1 and the time comparison is represented in Figure 2. The main contributions of this paper can be summarized as:

- **Joint Low Rank Approximation**: Rather than performing computationally expensive SVD operations on pre-RoPE keys as implemented in ShadowKV [16], the proposed method jointly optimizes low rank approximations of both query and key matrices, and reduces computational complexity while maintaining representation accuracy.

- **Precision-Preserving Attention Computation**: The low rank approximated keys and values serve independently as computational proxies for lightweight attention score estimation. Critically, the subsequent attention layer operations utilize the original query, key, and

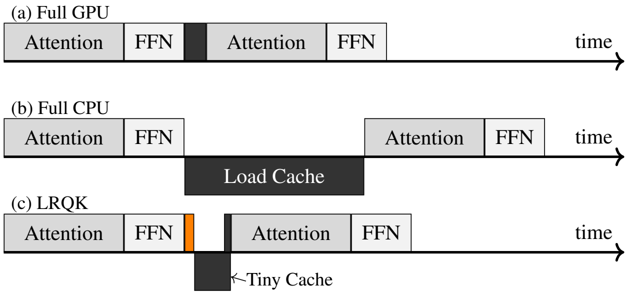


Figure 2: Time comparison with full GPU, full CPU and the proposed LRQK methods. The orange block is the selection operation of KV pairs. The black blocks are cache loading operations. The blocks above line mean GPU operations and the blocks below are CPU operations.

value vectors without any approximation or reconstruction, thereby preserving mathematical fidelity and model performance.

- **Mixed Cache Management**: A sophisticated hybrid GPU-CPU storage system featuring active token retention mechanisms and a hit/miss buffer architecture that minimizes cross device data transfer is implemented. Additionally, a specialized buffer maintains recently accessed keys and values, which empirical analysis confirms consistently receive high attention scores, further optimizing cache hit rates under operational conditions.

The paper is organized as follows. In Section 3, two key insights are presented: the low rank nature of the key matrix and the high attention scores between a current token and its neighbors. Section 4 introduces the Low Rank Query and Key (LRQK) method, detailing both prefill and decode stages. In Section 5, the performance of the LRQK method is assessed across various models and datasets.

2 Related Works

2.1 Token Eviction

Token eviction methods seek to bound the memory footprint of the KV cache by discarding less informative key-value pairs while preserving generation quality [11, 17, 10, 12, 18, 19]. Early work such as StreamingLLM [11], maintains a fixed "attention sink" of initial tokens together with a sliding window of recent tokens, stabilizing decoding for arbitrarily long sequences. Building on this, H_2O [10] formulates eviction as a dynamic submodule optimization problem to retain both recent tokens and "heavy hitters". Rather than relying on global scoring, SnapKV [12] leverages per-head clustering of attention patterns within a small observation window to select and compress the most relevant KV positions in one pass. More recently, Ada-KV [18] derives a theoretical upper bound on post-eviction attention error and proposes a head-wise adaptive budget allocation strategy, yielding consistent improvements. However, these existing eviction strategies universally trade off either long-range context fidelity or computational simplicity by relying on rigid windows, coarse heuristics, or expensive per-head clustering, therefore resulting in non-negligible approximation error or management overhead. Our approach adopts a joint low-rank proxy to select only the top-k full-precision KV pairs and a small hit/miss recency cache to load them, guaranteeing exact attention outputs and less CPU-GPU data movement.

2.2 Dynamic Sparse Attention

Dynamic sparse attention mechanisms represent a significant advancement in transformer based models by selectively computing attention scores on a subset of tokens while maintaining complete key-value cache storage, thereby substantially reducing computational requirements. Contemporary approaches implement diverse token selection strategies with varying degrees of efficacy: QUEST [20] employs a page based selection methodology, while Loki [21] leverages low-dimensional attention approximation through principal component analysis (PCA) on key matrices. PALU [22] and LPA [23] also utility the low rank capability in attention. Notably, two recent techniques, InfiniGen [14] and ShadowKV [16] utilize singular value decomposition (SVD) for key-value selection. InfiniGen pre-fetches essential entries using predefined projections via SVD, whereas ShadowKV offloads the entire $\mathbf{V}$ cache to CPU memory after SVD decomposition of the key matrix $(\mathbf{K})$. While effective, these SVD based approaches incur substantial computational overhead and fundamentally alter the original query/key representations, potentially compromising model fidelity. In contrast, the proposed method combines a joint low rank $\mathbf{Q}, \mathbf{K}$ projection and retrieves the relevant full-precision KV pairs on demand, ensuring exact attention computation with less approximation error.

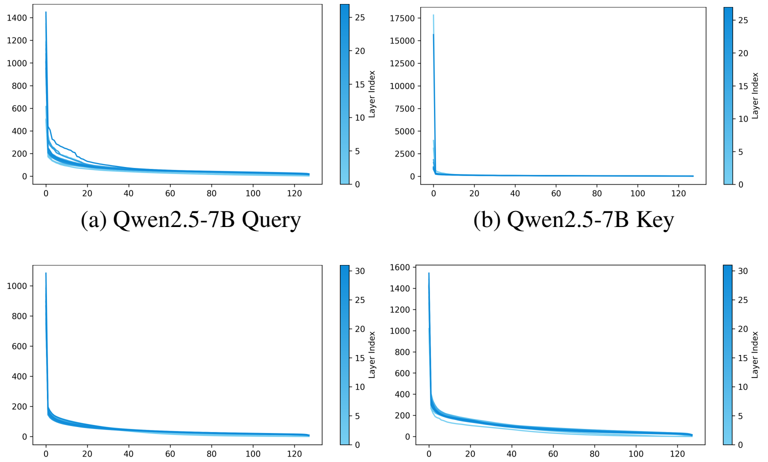
3 Preliminary

3.1 Low Rankness of Query and Key

As demonstrated in prior works [16, 14], the key matrix in decoder-only transformers exhibits a pronounced low rank structure. Motivated by this observation, the full-precision query and key matrices are jointly factorized into rank-r components, enforcing

$$\mathrm{rank}(\mathbf{Q}\mathbf{K}^\top) \leq \min\left(\mathrm{rank}(\mathbf{Q}), \mathrm{rank}(\mathbf{K}^\top)\right) = \min\left(\mathrm{rank}(\mathbf{Q}), \mathrm{rank}(\mathbf{K})\right). \tag{1}$$

This implies that if $\mathbf{K}$ is effectively of rank r, then its interaction with any query also lies close to an r-dimensional subspace. Consequently, it is possible to approximate $\mathbf{Q}\mathbf{K}^\top$ by a low rank factorization $\mathbf{A}_Q\mathbf{A}_K^\top$ with a low approximation error. An illustrative example of this phenomenon appears in Figure 3, which plots the average singular value spectrum of the per-head query and key matrix $\mathbf{K}$ on Qwen2.5-7B [2] and LLaMA-3-8B-1M [3, 24] under the Wikitext-2-v1 test set [25]. In both cases, the singular values decay rapidly beyond a small rank, confirming that $\mathbf{K}$ admits a low rank approximation with minimal loss.



(a) Qwen2.5-7B Query (b) Qwen2.5-7B Key

(c) LLaMA-3-8B-1M Query (d) LLaMA-3-8B-1M Key

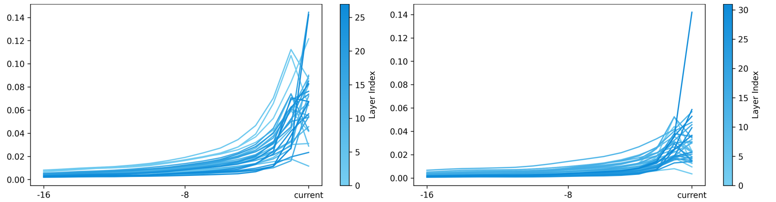
Figure 3: Examples of the mean of singular values of the query and key matrix over different layers on Qwen2.5-7B and LLaMA-3-8B-1M models. The singular values are summed over batches and attention heads.

3.2 Attention Scores Near Current Token

To quantify the locality of self-attention during decoding, let $\mathbf{q}_t$ be the query at step t and $\mathbf{k}_i$ the key at position i, so that the attention weights are given by

$$\mathrm{softmax}\left(\mathbf{q}_t \begin{bmatrix} \cdots & \mathbf{k}_{t-3} & \mathbf{k}_{t-2} & \mathbf{k}_{t-1} & \mathbf{k}_t \end{bmatrix}\right) = \begin{bmatrix} \cdots & a_{t-3} & a_{t-2} & a_{t-1} & a_t \end{bmatrix}.$$

The average per-head scores are evaluated on the Wikitext-2-v1 test set using both Qwen2.5-7B and LLaMA-3-8B-1M. Results are shown in Figure 4. In every case, the curves exhibit pronounced higher attention scores in the current token and its near neighbors, confirming a strong recency bias. This empirical finding is also observed by StreamingLLM [11], which motivates our inclusion of a compact recency buffer in the GPU cache that persistently holds the most recent tokens, thereby maximizing cache hit rates with minimal overhead.



(a) Qwen2.5-7B (b) Llama-3-8B-1M

Figure 4: Examples of the attention scores of the neighbors of current token. The window size is 16. The x-label "current" is the index of the current token, -8 means the tokens at $t-8$, and so on. The attention scores are averaged over batches and attention heads.

4 Proposed Method

The inference process of LLM can be divided into two distinct phases: prefill and decode. When a prompt is provided by the user, the long prompt is first prefilled once by the LLM, and then the token is decoded autoregressively. Specific cache mechanisms are required for these two phases, the LRQK will be introduced for prefill in Section 4.2 and decode in Section 4.3, respectively.

The preliminaries and formulation used in the proposed method are clarified below. In practical implementations, transformer models typically process input tensors with a shape of (batch size, num heads, sequence length, hidden size). However, when methods such as GQA [26] are employed, the number of attention heads used for the query tensor is usually different from those used for the key and value tensors. Therefore, repeated key/value (repeat KV) mechanism is utilized, ensuring that the number of heads for the query and the key/value tensors are aligned. In addition, to simplify notation, 'batch size' and 'num heads' dimensions are also omitted, as computations across different batches and heads are independent. Thus, the query, key, and value matrices as $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{l \times d}$, where l denotes the sequence length and d the hidden size of each head. Moreover, since the prefill and decode process is identical across all transformer layers, the layer indices are omitted for clarity.

4.1 Mixed Caching

To address memory constraints and optimize runtime efficiency, a mixed caching strategy is adopted that leverages both a fast yet limited GPU KV cache and a larger but slower CPU cache. To further reduce the data transfer overhead between GPU and CPU, a hit-and-miss cache mechanism is introduced. In this scheme, when the required tokens are already present in the GPU cache (i.e., a cache hit), direct access is provided without incurring additional transfer cost. When the required cache is missing in the GPU, only the missing tokens are fetched from the CPU cache, therefore avoiding redundant data movement and improving overall system efficiency.

Based on the observation that recent tokens are more likely to receive high attention scores (as shown in Figure 4), a dedicated buffer is incorporated to store the most recent tokens. The GPU KV cache is structured as [active tokens lite tokens], where the 'active tokens' are the top-k tokens corresponding to the set Ω_k with the approximated highest attention scores selected by the low rank approximations of query and key matrices, and the 'lite tokens' corresponding to the set Ω_l capture a fixed number of neighboring recent tokens. The total window size of the GPU KV cache is thus the sum of the active and lite tokens, allowing for a balance between relevance and recency in cache composition.

4.2 Prefill

At this stage, the input consists of a long prompt represented as a matrix $\mathbf{X} \in \mathbb{R}^{l \times d}$, where l denotes the sequence length and d the hidden dimension. By applying linear projections to $\mathbf{X}$, the query, key, and value matrices: $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{l \times d}$ will be obtained.

According to (1), the attention score matrix is usually low rank, thus can be approximated using a jointly low rank decomposition:

$$\mathbf{Q}\mathbf{K}^\top \approx \mathbf{A}_Q \mathbf{A}_K^\top, \tag{2}$$

where $\mathbf{A}_Q, \mathbf{A}_K \in \mathbb{R}^{l \times r}$ and $r \ll d$ denotes matrix rank. In addition to the joint low rank approximation, query and key matrices are approximated for memory efficiency:

$$\mathbf{Q} \approx \mathbf{A}_Q \mathbf{B}_Q, \quad \mathbf{K} \approx \mathbf{A}_K \mathbf{B}_K, \tag{3}$$

where $\mathbf{B}_Q, \mathbf{B}_K \in \mathbb{R}^{r \times d}$. Note that the approximation for attention score $\mathbf{Q}\mathbf{K}^\top$ and $\mathbf{Q}, \mathbf{K}$ shares the same column/row space spanned by $\mathbf{A}_Q$ and $\mathbf{A}_K$. For solving (2) and (3), it can be formulated as

$$\underset{\mathbf{A}_Q, \mathbf{B}_Q, \mathbf{A}_K, \mathbf{B}_K}{\arg\min} \ \frac{1}{2} \left\| \mathbf{Q}\mathbf{K}^\top - \mathbf{A}_Q \mathbf{A}_K^\top \right\|_\mathrm{F}^2, \text{s.t. } \mathbf{Q} = \mathbf{A}_Q \mathbf{B}_Q, \mathbf{K} = \mathbf{A}_K \mathbf{B}_K. \tag{4}$$

To solve the constraint optimization problem (4), it is relaxed by deriving a Lagrangian:

$$\mathcal{L}_{pre} = \frac{1}{2} \left\| \mathbf{Q}\mathbf{K}^\top - \mathbf{A}_Q \mathbf{A}_K^\top \right\|_\mathrm{F}^2 + \frac{\lambda_{pQ}}{2} \left\| \mathbf{Q} - \mathbf{A}_Q \mathbf{B}_Q \right\|_\mathrm{F}^2 + \frac{\lambda_{pK}}{2} \left\| \mathbf{K} - \mathbf{A}_K \mathbf{B}_K \right\|_\mathrm{F}^2, \tag{5}$$

where λ_{pQ} and λ_{pK} are two scaling factors indicating the importance of the low rank approximation of the query and key matrices, respectively.

By solving the Lagrangian (5), with setting the partial derivative to be zero, the solution of the query and key matrices can be obtained (see Appendix A.1 for details):

$$\begin{aligned} \mathbf{A}_Q &= \mathbf{Q} \left(\mathbf{K}^\top \mathbf{A}_K + \lambda_{pQ} \mathbf{B}_Q^\top \right) \left(\mathbf{A}_K^\top \mathbf{A}_K + \lambda_{pQ} \mathbf{B}_Q \mathbf{B}_Q^\top \right)^{-1} \\ \mathbf{A}_K &= \mathbf{K} \left(\mathbf{Q}^\top \mathbf{A}_Q + \lambda_{pK} \mathbf{B}_K^\top \right) \left(\mathbf{A}_Q^\top \mathbf{A}_Q + \lambda_{pK} \mathbf{B}_K \mathbf{B}_K^\top \right)^{-1} \\ \mathbf{B}_Q &= (\mathbf{A}_Q^\top \mathbf{A}_Q)^{-1} \mathbf{A}_Q^\top \mathbf{Q} \\ \mathbf{B}_K &= (\mathbf{A}_K^\top \mathbf{A}_K)^{-1} \mathbf{A}_K^\top \mathbf{K}. \end{aligned} \tag{6}$$

From (6), it is notice that the operation of matrix inverse is required. The matrix inverse operation is expensive $O(r^3)$. However, since the rank r is a small number, and $\mathbf{A}_K^\top \mathbf{A}_K, \mathbf{A}_Q^\top \mathbf{A}_Q$, $\mathbf{B}_K \mathbf{B}_K^\top, \mathbf{B}_Q \mathbf{B}_Q^\top$ are all with shape (r, r), which are small matrices. Therefore, the computation costs of matrix inverse are relatively low. In addition, when solving $\mathbf{A}_Q$ and $\mathbf{A}_K$, the memory cost can further be reduced by changing the multiplication order. For example, by changing the order of operations from $(\mathbf{Q}\mathbf{K}^\top)\mathbf{A}_K$ to $\mathbf{Q}(\mathbf{K}^\top \mathbf{A}_K)$, the computational complexity is reduced from $\mathcal{O}(l^2 d)$ to $\mathcal{O}(rld)$.

The iterative procedure in Algorithm 1 can be interpreted as a Block Coordinate Descent (BCD) method, where each step optimizes the Lagrangian with respect to a subset of variables while holding the others fixed.

Algorithm 1 Prefill of LRQK, alternating updates for $\mathbf{A}_Q, \mathbf{A}_K, \mathbf{B}_Q, \mathbf{B}_K$

```
Require: $\mathbf{Q}, \mathbf{K}$, rank $r$, $\lambda_{pQ}, \lambda_{pK}$
Initialize $\mathbf{A}_Q, \mathbf{A}_K \sim \mathcal{N}(0, 1)$
while $i <$ max iteration or not converge do
    Update $\mathbf{B}_Q \leftarrow (\mathbf{A}_Q^\top \mathbf{A}_Q)^{-1} \mathbf{A}_Q^\top \mathbf{Q}$
    Update $\mathbf{B}_K \leftarrow (\mathbf{A}_K^\top \mathbf{A}_K)^{-1} \mathbf{A}_K^\top \mathbf{K}$
    Update $\mathbf{A}_K \leftarrow \mathbf{K} \left( \mathbf{Q}^\top \mathbf{A}_Q + \lambda_{pK} \mathbf{B}_K^\top \right) \left( \mathbf{A}_Q^\top \mathbf{A}_Q + \lambda_{pK} \mathbf{B}_K \mathbf{B}_K^\top \right)^{-1}$
    Update $\mathbf{A}_Q \leftarrow \mathbf{Q} \left( \mathbf{K}^\top \mathbf{A}_K + \lambda_{pQ} \mathbf{B}_Q^\top \right) \left( \mathbf{A}_K^\top \mathbf{A}_K + \lambda_{pQ} \mathbf{B}_Q \mathbf{B}_Q^\top \right)^{-1}$
end while
return $\mathbf{A}_Q, \mathbf{A}_K, \mathbf{B}_Q, \mathbf{B}_K$
```

4.3 Decode

Following the prefilling phase, the language model generates the remaining tokens sequentially, adhering to an autoregressive decoding process where each token is predicted based on the previously generated context. Suppose the current stage is t, and the input is a row vector $\mathbf{x}_t \in \mathbb{R}^{1 \times d}$ and its projections are $\mathbf{q}_t \in \mathbb{R}^{1 \times d}$ and $\mathbf{k}_t \in \mathbb{R}^{1 \times d}$. In the decode stage, the goal is to compress the $\mathbf{q}_t$ and $\mathbf{k}_t$

to much smaller ones $\widehat{\mathbf{q}}_t \in \mathbb{R}^{1\times r}$ and $\widehat{\mathbf{k}}_t \in \mathbb{R}^{1\times r}$, therefore the memory footprint of KV cache can be greatly mitigated.

From the previous stage, the low rank approximations of the query and key matrices are obtained in the form of left and right factors: $\mathbf{A}_Q, \mathbf{A}_K \in \mathbb{R}^{l\times r}$, and $\mathbf{B}_Q, \mathbf{B}_K \in \mathbb{R}^{r\times d}$. To maintain consistency in the modeling setup for new tokens, the following optimization problem is formulated for estimating the approximated query and key vectors, $\widehat{\mathbf{q}}_t$ and $\widehat{\mathbf{k}}_t$, respectively,

$$\begin{aligned} &\arg\min_{\widehat{\mathbf{q}}_t, \widehat{\mathbf{k}}_t} \frac{1}{2}\left\|\widehat{\mathbf{q}}_t \mathbf{B}_{Q,t-1} - \mathbf{q}_t\right\|_{\mathrm{F}}^2 + \frac{1}{2}\left\|\widehat{\mathbf{k}}_t \mathbf{B}_{K,t-1} - \mathbf{k}_t\right\|_{\mathrm{F}}^2, \\ &\text{s.t. } \widehat{\mathbf{q}}_t \widehat{\mathbf{k}}_t^\top = \mathbf{q}_t \mathbf{k}_t^\top, \widehat{\mathbf{q}}_t \mathbf{A}_{K,\Omega,t-1}^\top = \mathbf{q}_t \mathbf{K}_{\Omega,t-1}^\top. \end{aligned} \tag{7}$$

The matrix $\mathbf{A}_{K,\Omega,t-1}$ denotes a submatrix of $\mathbf{A}_K$ that includes only the rows indexed by Ω. Similarly, $\mathbf{K}_{\Omega,t-1}$ is the corresponding submatrix of $\mathbf{K}$ formed by selecting the rows indexed by Ω. Here, Ω represents the index set of KV cache retained in GPU memory at time step $t-1$.

The Lagrangian of (7) can be written as:

$$\begin{aligned} \mathcal{L}_{\text{dec}} = &\frac{1}{2}\left\|\widehat{\mathbf{q}}_t \mathbf{B}_{Q,t-1} - \mathbf{q}_t\right\|_{\mathrm{F}}^2 + \frac{1}{2}\left\|\widehat{\mathbf{k}}_t \mathbf{B}_{K,t-1} - \mathbf{k}_t\right\|_{\mathrm{F}}^2 \\ &+ \frac{\lambda_{d1}}{2}\left(\widehat{\mathbf{q}}_t \widehat{\mathbf{k}}_t^\top - \mathbf{q}_t \mathbf{k}_t^\top\right)^2 + \frac{\lambda_{d2}}{2}\left\|\widehat{\mathbf{q}}_t \mathbf{A}_{K,\Omega,t-1}^\top - \mathbf{q}_t \mathbf{K}_{\Omega,t-1}^\top\right\|_{\mathrm{F}}^2, \end{aligned} \tag{8}$$

where λ_{d1} and λ_{d2} are regularization parameters. For more details, please refer to the Appendix A.2.

By solving the Lagrangian (8), update rules for $\widehat{\mathbf{q}}_t$ and $\widehat{\mathbf{k}}_t$ can be obtained as:

$$\begin{aligned} \widehat{\mathbf{q}}_t &= \mathbf{m}_{lq,t}\mathbf{M}_{rq,t}^{-1}, \\ \widehat{\mathbf{k}}_t &= \left(\mathbf{k}_t \mathbf{B}_{K,t-1}^\top + \lambda_{d1}\mathbf{k}_t \mathbf{q}_t^\top \widehat{\mathbf{q}}_t\right)\left(\mathbf{B}_{K,t-1}\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\widehat{\mathbf{q}}_t^\top \widehat{\mathbf{q}}_t\right)^{-1} \\ \mathbf{m}_{lq,t} &:= \left(\mathbf{q}_t \mathbf{B}_{Q,t-1}^\top + \lambda_{d1}\mathbf{q}_t \mathbf{k}_t^\top \widehat{\mathbf{k}}_t + \lambda_{d2}\mathbf{q}_t \mathbf{K}_{\Omega,t-1}^\top \mathbf{A}_{K,\Omega,t-1}\right) \\ \mathbf{M}_{rq,t} &:= \left(\mathbf{B}_{Q,t-1}\mathbf{B}_{Q,t-1}^\top + \lambda_{d1}\widehat{\mathbf{k}}_t^\top \widehat{\mathbf{k}}_t + \lambda_{d2}\mathbf{A}_{K,\Omega,t-1}^\top \mathbf{A}_{K,\Omega,t-1}\right). \end{aligned} \tag{9}$$

Similar to prefill stage, since the matrices require taking inverse is of size (r, r), which is small, and the computation cost is relatively small.

For $\mathbf{B}_{Q,t}$ and $\mathbf{B}_{K,t}$, one step gradient descent is applied to op the objective function. The update rules are given by:

$$\mathbf{B}_{Q,t} \leftarrow \mathbf{B}_{Q,t-1} - \eta_Q^* \nabla \mathbf{B}_{Q,t-1}, \quad \mathbf{B}_{K,t} \leftarrow \mathbf{B}_{K,t-1} - \eta_K^* \nabla \mathbf{B}_{K,t-1}, \tag{10}$$

where η_Q and η_K are the learning rates for $\mathbf{B}_{Q,t-1}$ and $\mathbf{B}_{K,t-1}$, respectively. Two learning rates η_Q^* and η_K^* are computed as:

$$\eta_Q^* = \frac{\left(\widehat{\mathbf{q}}_t \mathbf{B}_{Q,t-1} - \mathbf{q}_t\right)\left(\widehat{\mathbf{q}}_t \nabla \mathbf{B}_{Q,t-1}\right)^\top}{\left(\widehat{\mathbf{q}}_t \nabla \mathbf{B}_{Q,t-1}\right)\left(\widehat{\mathbf{q}}_t \nabla \mathbf{B}_{Q,t-1}\right)^\top}, \tag{11}$$

$$\eta_K^* = \frac{\left(\widehat{\mathbf{k}}_t \mathbf{B}_{K,t-1} - \mathbf{k}_t\right)\left(\widehat{\mathbf{k}}_t \nabla \mathbf{B}_{K,t-1}\right)^\top}{\left(\widehat{\mathbf{k}}_t \nabla \mathbf{B}_{K,t-1}\right)\left(\widehat{\mathbf{k}}_t \nabla \mathbf{B}_{K,t-1}\right)^\top}. \tag{12}$$

Algorithm 2 summarizes the update rules for $\widehat{\mathbf{q}}_t$, $\widehat{\mathbf{k}}_t$, $\mathbf{B}_{Q,t}$, and $\mathbf{B}_{K,t}$, the computation of proxy attention, and the update for selected key and value. The graphical representation of the thorough decoding process of LRQK is shown in Figure 1. To avoid random initialization, the initial guess of $\widehat{\mathbf{k}}_t$ is utilized by setting $\lambda_{d1} = 0$. The algorithm iterates until convergence or reaches the maximum number of iterations. The attention is computed with selected key and value, Attention $(\mathbf{q}_t, \mathbf{K}_{\Omega,t}, \mathbf{V}_{\Omega,t})$.

Algorithm 2 Decode of LRQK

```
Require: $\mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t, \mathbf{B}_{Q,t-1}, \mathbf{B}_{K,t-1}, \lambda_{d1}, \lambda_{d2}, \mathbf{A}_{K,\Omega,t-1}, \mathbf{K}_{\Omega,t-1}, \mathbf{V}_{\Omega,t-1}$
Initial Guess $\widehat{\mathbf{k}}_t \leftarrow \left(\mathbf{k}_t \mathbf{B}_{K,t-1}^\top\right)\left(\mathbf{B}_{K,t-1}\mathbf{B}_{K,t-1}^\top\right)^{-1}$
while $i <$ max iteration or not converged do
    Update $\widehat{\mathbf{q}}_t \leftarrow \mathbf{m}_{lq,t}\mathbf{M}_{rq,t}^{-1}$
    Update $\widehat{\mathbf{k}}_t \leftarrow \left(\mathbf{k}_t \mathbf{B}_{K,t-1}^\top + \lambda_{d1}\mathbf{q}_t\mathbf{k}_t^\top\widehat{\mathbf{q}}_t\right)\left(\mathbf{B}_{K,t-1}\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\widehat{\mathbf{q}}_t^\top\widehat{\mathbf{q}}_t\right)^{-1}$
end while
$\mathbf{B}_{Q,t} \leftarrow \mathbf{B}_{Q,t-1} - \eta_Q^* \nabla \mathbf{B}_{Q,t-1}$, with $\eta_Q^*$ in (11)
$\mathbf{B}_{K,t} \leftarrow \mathbf{B}_{K,t-1} - \eta_K^* \nabla \mathbf{B}_{K,t-1}$, with $\eta_K^*$ in (12)
$\mathbf{A}_{K,t} \leftarrow \left[\mathbf{A}_{K,t-1}; \widehat{\mathbf{k}}_t\right]$                ▷ concatenate
top-$k$ index $\Omega_k \leftarrow$ top $\left(\mathbf{A}_{K,t}\widehat{\mathbf{q}}_t^\top, k\right)$       ▷ compute proxy attention
Fetch missing $\{\mathbf{k}_i\}, \{\mathbf{v}_i\}$ from CPU $\mathbf{K}, \mathbf{V}$ according to $\Omega_k$
$\mathbf{K}'_{\Omega,t-1}, \mathbf{V}'_{\Omega,t-1} \leftarrow$ merge fetched $\{\mathbf{k}_i\}, \{\mathbf{v}_i\}$ with GPU $\mathbf{K}_{\Omega,t-1}, \mathbf{V}_{\Omega,t-1}$
$\mathbf{K}_{\Omega,t} \leftarrow \left[\mathbf{K}'_{\Omega,t-1}; \mathbf{k}_t\right]$, $\mathbf{V}_{\Omega,t} \leftarrow \left[\mathbf{V}'_{\Omega,t-1}; \mathbf{v}_t\right]$
Async $\mathbf{k}_t, \mathbf{v}_t$ to CPU
return $\mathbf{K}_{\Omega,t}, \mathbf{V}_{\Omega,t}$.
```

5 Experiments

In this section, a series of experiments are conducted to rigorously evaluate the effectiveness and performance of our proposed method. The experiments are performed on NVIDIA A100 GPUs. The number of max iteration and the tolerance introduced in Algorithm 1 and 2 are chosen as 2 and 0.01, respectively. All scaling parameters are set as $\lambda_{pQ} = \lambda_{pK} = \lambda_{d1} = \lambda_{d2} = 1$. All LLM evaluations are empowered by OpenCompass [27]. Additional results can be found in Appendix B, the time cost analysis is presented in Appendix C, and a guideline for hyperparameter tuning is provided in Appendix D.

5.1 Accuracy Evaluation

The accuracy of the proposed method is evaluated on the RULER dataset [28], using a long-context setting with a sequence length of 128K tokens. For the low rank approximation, the rank is set to $r = 32$. The number of top-k tokens selected based on attention scores is set to 2048 (1.56% of 128K), while the number of lite tokens (the most recently generated tokens) is set to 64. The backbone language model used in the evaluation is LLaMA-3-8B-1M [24]. This method is compared against four recent dynamic sparse attention baselines: Loki [21], InfiniGen [14], Quest [20], and ShadowKV [16]. Results are shown in Table 1.

Table 1: Comparison of different models and methods on RULER (left) and LongBench (right).

Methods	S1	S2	MK1	MK2	MQ	MV	QA-1	QA-2	VT	FWE	NQA	MQA	GRep	SAM	PRetr	LCC
Llama-3-8B-1M	100.00	100.00	98.96	98.96	98.96	95.57	75.00	48.96	78.54	71.85	18.98	41.84	34.18	35.96	81.50	56.07
Loki	18.75	1.04	2.08	0.00	1.56	0.78	4.17	13.54	26.04	25.35	2.26	10.19	28.97	7.84	40.52	31.44
InfiniGen	100.00	98.96	84.38	53.13	63.28	54.95	65.63	48.96	**81.67**	50.35	14.39	31.46	27.38	21.97	74.30	38.58
Quest	100.00	100.00	**98.96**	77.08	97.65	93.49	60.42	50.00	77.08	65.63	**20.13**	36.63	27.11	35.63	79.00	53.64
ShadowKV	100.00	100.00	97.92	**98.96**	96.88	95.83	72.92	52.08	**81.67**	**72.57**	17.17	39.73	**31.62**	**35.87**	80.00	63.93
Proposed	81.00	100.00	97.00	42.00	**99.25**	**98.00**	**75.00**	**53.00**	80.20	69.67	16.52	**40.48**	20.40	26.35	**89.00**	**66.13**

Table 1 presents the comparative performance of various models and methods on both the RULER and LongBench benchmarks. Our proposed method demonstrates competitive results across a wide range of tasks, particularly excelling in several key domains.

On the RULER benchmark, our model achieves perfect accuracy on the S2 (NIAH Single 2) task, matching the performance of larger models like ShadowKV and Quest. Notably, our method outperforms all baselines on QA-1 (QA SQuAD) and QA-2 (QA HotpotQA). In the MQ (NIAH MultiQuery) and MV (NIAH MultiValue) tasks, which evaluate a model's ability to process multiple queries or values within a single instance, our method even surpasses the original LLaMA-3-8B-1M model, achieving results of 99.25% and 98.00%, respectively.

For the LongBench evaluation, our approach demonstrates notable gains in tasks such as PRetr (Passage Retrieval) and LCC, outperforming all other methods with 89.00% and 66.13%, respectively. These tasks are particularly demanding due to their emphasis on retrieving and reasoning over long-range dependencies, suggesting that our method is well-suited for these kinds of tasks.

While some tasks like MK2 (NIAH MultiKey 2) and NQA (NarrativeQA) show slightly lower performance compared to leading baselines, the overall consistency and high scores across multiple complex tasks underscore the effectiveness of our approach. These results affirm the potential of our method in addressing long-context language understanding challenges.

The proposed method is further evaluated on two backbone models: LLaMA-3-8B-1M [24] and Qwen2.5-7B-Instruct [2], using the same configuration of rank $r = 16$, top-$k = 256$ (6.25% of 4K), and 16 lite tokens on a subset of RULER-4K. As shown in Table 2, our method consists the performance of both models.

Table 2: Results on subset of RULER-4K of two models

Methods	QA-1	QA-2	VT	Methods	QA-1	QA-2	VT
LLaMA-3-8B-1M	82.00	58.00	99.00	Qwen2.5-7B	90.00	64.00	99.20
+ LRQK	84.00	57.00	98.80	+ LRQK	91.00	65.00	96.60

5.2 Impact of Rank and Top-k Selection

In this subsection, the impact of two key hyperparameters in the proposed method are investigated: the rank r used in the low rank approximation, and the number of top-k active tokens selected based on attention scores.

Rank Selection. To isolate the effect of the rank, we fix the number of top-k tokens at 256 and the number of lite tokens at 16. The rank are vary over the set $\{16, 24, 32\}$, allowing us to examine how the capacity of the low rank representations affects model performance.

Top-k Selection. To analyze the influence of the number of active tokens, we fix the rank at 8 and the number of lite tokens at 16. The top-k parameter is varied across $\{256, 512, 1024\}$ to evaluate how the size of the active set impacts retrieval effectiveness and downstream task accuracy. All experiments are conducted on a representative subset of the RULER-4K dataset using the LLaMA-3-8B-1M model as the backbone.

Tables 3 and 4 show that increasing the rank r generally improves performance on QA-1 and VT, with $r = 32$ even surpassing the original LLaMA-3-8B-1M on QA-1. In contrast, QA-2 performs best at a lower rank $r = 16$, suggesting some tasks benefit from more compact representations. For top-k, larger values consistently lead to better accuracy across all tasks. The configuration with $k = 1024$ achieves the highest scores, indicating that a broader active token window improves context modeling. However, larger top-k tokens will increase both computation and memory costs, necessitating a trade-off between performance and resource usage. Overall, while higher ranks and larger top-k values enhance performance, smaller settings (e.g., $r = 16$, $k = 512$) still offer strong results with reduced resource usage.

Table 3: Accuracy across different rank values with top-k 256 and 16 lite tokens

Models	QA-1	QA-2	VT
Llama-3-8B-1M	82.00	58.00	99.00
$r = 8$	79.00	50.00	56.80
$r = 16$	80.00	60.00	98.80
$r = 24$	83.00	56.00	98.80
$r = 32$	84.00	57.00	99.00

Table 4: Accuracy across different top-k values with $r = 8$ and 16 lite tokens

Models	QA-1	QA-2	VT
Llama-3-8B-1M	82.00	58.00	99.00
top-256	79.00	50.00	56.80
top-512	78.00	61.00	95.60
top-1024	83.00	63.00	100.00

5.3 Miss Rates

The performance of the proposed strategy for memory management is evaluated by computing the miss rate. The rate is a ratio $c_{\text{miss}}/c_{\text{total}}$, where c_{miss} denotes the number of KV cache rows that must be transferred from CPU to GPU, and c_{total} is the number of all selected indices. This metric quantifies the efficiency of the token selection mechanism in reducing memory transfers.

Experiments are conducted on the summarization task using the wikitext-2-v1 test set [25]. A grid search is performed over different hyperparameters, including the rank $r \in \{8, 16, 32, 64\}$, the number of active tokens $\in \{128, 256, 512\}$, and the number of lite tokens $\in \{4, 8\}$. The distribution of miss rates is illustrated in Figure 5, which reveals an approximately Gaussian shape with a mean miss rate of around 0.40 and a standard deviation of approximately 0.10. This implies that, on average, the proposed hit-and-miss mechanism reduces the amount of data transferred from CPU to GPU by about 60% in average.

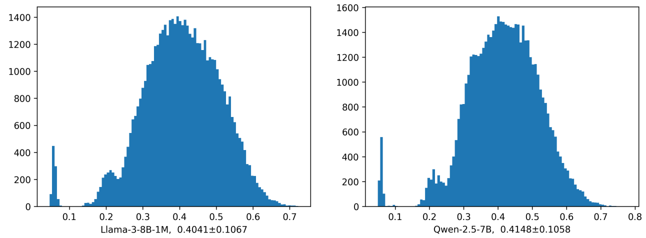


Figure 5: Histogram of miss rates on wikitext-2-v1.

6 Conclusion

Low Rank Query and Key (LRQK) attention, a two stage inference algorithm that enables long-context processing through joint low rank decomposition of query and key matrices combined with mixed GPU-CPU cache management is presented. By computing proxy attention scores on compact rank-r factors and selectively fetching only the top-k active tokens and most recent tokens' full-precision key-value pairs, LRQK preserves exact attention outputs while reducing CPU-GPU data transfers.

Extensive evaluations on RULER (up to 128K tokens) or LongBench benchmarks demonstrate that LRQK matches or exceeds state-of-the-art sparse attention methods across diverse long-context tasks. The method achieves substantial memory savings, enabling processing of contexts that would otherwise cause out-of-memory errors, without significant accuracy degradation. Ablation studies confirm the effectiveness of each component: low rank approximation, active token selection, and lite token retention all contribute to the method's robust performance.

Limitations. While LRQK significantly reduces data transfer overhead, further observation reveals that CPU-side indexing operations constitute the primary performance bottleneck rather than PCIe bandwidth limitations, and hoping can be addressed in future work. Additionally, LRQK's hyperparameters require task-specific tuning. While a practical guidelines and default configurations are provided, optimal settings still demand empirical validation for new domains.

Societal Impact. LRQK's reduced inference costs could enable broader access to long-context language models, benefiting research and education. However, this efficiency may also lower barriers to deploying harmful applications such as large-scale disinformation campaigns. We believe democratized access to advanced AI capabilities provides net benefits when coupled with appropriate safeguards and responsible deployment practices.

Acknowledgments and Disclosure of Funding

This study was supported in part by Natural Science Foundation of China under Grant 62203124, JSPS KAKENHI Grant Number JP23K28109 and JP24K20849, and Moonshot R&D Grant Number JPMJMS2239. Yuning Qiu was supported by the RIKEN Special Postdoctoral Researcher Program. Tenghui Li was supported by RIKEN's IPA Program.

References

[1] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

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

[3] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

[4] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

[5] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[6] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length LLM inference with KVcache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

[7] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

[8] Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. KVcache is 1 bit per channel: Efficient large language model inference with coupled quantization. *Advances in Neural Information Processing Systems*, 37:3304–3331, 2024.

[9] Shibo Jie, Yehui Tang, Kai Han, Zhi-Hong Deng, and Jing Han. Specache: Speculative key-value caching for efficient generation of llms. *arXiv preprint arXiv:2503.16163*, 2025.

[10] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

[11] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Representation Learning*, volume 2024, pages 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

[12] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

[13] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single GPU. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

[14] Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 155–172, 2024.

[15] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. IEEE, 2022.

[16] Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=oa7MYAO6h6.

[17] Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. CAKE: Cascading and adaptive KV cache eviction with layer preferences. *arXiv preprint arXiv:2503.12491*, 2025.

[18] Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. *arXiv preprint arXiv:2407.11550*, 2024.

[19] Minghui Liu, Tahseen Rabbani, Tony O'Halloran, Ananth Sankaralingam, Mary-Anne Hartley, Brian Gravelle, Furong Huang, Cornelia Fermüller, and Yiannis Aloimonos. Hashevict: A pre-attention KV cache eviction strategy using locality-sensitive hashing. *arXiv preprint arXiv:2412.16187*, 2024.

[20] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

[21] Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. *Advances in Neural Information Processing Systems*, 37:16692–16723, 2024.

[22] Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S Abdelfattah, and Kai-Chiang Wu. Palu: Compressing KV-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

[23] Xingtai Lv, Ning Ding, Kaiyan Zhang, Ermo Hua, Ganqu Cui, and Bowen Zhou. Scalable efficient training of large language models with low-dimensional projected attention. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14588–14599, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.808. URL https://aclanthology.org/2024.emnlp-main.808/.

[24] Leonid Pekelis, Michael Feil, Forrest Moret, Mark Huang, and Tiffany Peng. Llama 3 gradient: A series of long context models, 2024. URL https://gradient.ai/blog/scaling-rotational-embeddings-for-long-context-language-models.

[25] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016. URL http://arxiv.org/abs/1609.07843.

[26] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 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, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298.

[27] OpenCompass. Opencompass: A universal evaluation platform for foundation models, 2023. URL https://github.com/open-compass/opencompass.

[28] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[29] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the two-stage LRQK framework, namely, the joint low rank decomposition of query/key matrices and the hierarchical GPU–CPU cache with hit-and-miss token selection—and claim both exact attention preservation and substantial memory reduction. These points are rigorously supported by the derivations in Sections 3 and 4, the complexity analysis, and the empirical results on RULER and LongBench, with full details in the appendix. Thus, the stated contributions match the theoretical development, algorithms, and experiments without overstatement.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated "Limitations" paragraph that acknowledges two key drawbacks: the need to carefully tune the rank r and active set size k, whose optimal values can vary across different models and tasks, and the inference is slightly slow.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All theoretical results—including the low-rank prefill objective and the decoding Lagrangian are stated with their full assumptions (e.g. rank-r factorizations, invertibility of $r \times r$ matrices) and are followed by complete, step-by-step derivations of the closed-form update rules in Appendices.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper gives full details for reproduction, including backbone models (LLaMA-3-8B, Qwen2.5-7B), datasets (RULER, LongBench, WikiText-2), context lengths (128K, 4K), hyperparameter settings (rank r, top-k, lite tokens), iteration counts and tolerances, hardware setup (NVIDIA A100 GPUs; NVIDIA 3090s), and the evaluation toolkit (OpenCompass). It also provides clear algorithm pseudocode and detailed derivations in the appendix, ensuring all components needed to replicate the experiments are specified.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The authors release a public GitHub repository (`https://github.com/tenghuilee/LRQK`) containing all implementation codes, and evaluation scripts. The supplemental material includes detailed environment specifications (library versions), exact command-line invocations for evaluation, and processing instructions for datasets, ensuring that others can faithfully reproduce the reported results.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and Appendix C give complete details: hardware (NVIDIA A100 GPUs), backbone models (LLaMA-3-8B-1M, Qwen2.5-7B), datasets and context lengths (128K for RULER/LongBench, 4K subset, WikiText-2 for summarization), hyperparameter settings (rank r, top-k, lite tokens, max iterations = 2, tolerance = 1e-2), grid-search ranges, and evaluation toolkit (OpenCompass). Algorithms pseudocode provide full pseudocode for reproducing the inference procedure, ensuring all necessary information is specified.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper provides extensive experimental results across multiple datasets, models, and settings, which offer a broad statistical perspective. While it may not include explicit error bars, the consistent performance trends and comprehensive comparisons serve as strong empirical evidence.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports that all experiments were conducted on NVIDIA A100 GPUs, specifies the batch sizes 1, context lengths (4K, 128K), and hardware setup in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: This work proposes an algorithmic optimization for LLM inference without involving human subjects, personal data, or ethically sensitive applications. It presents transparent methods and evaluations, poses no privacy or fairness risks, and adheres to NeurIPS ethical guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The proposed method is a technical improvement in LLM inference, and does not involve any sensitive content, human data, or high-risk applications. The proposed method reduces inference costs could enable border access to long-context language models. However, this efficiency may also lower barriers to deploying applications such as large-scale disinformation campaigns. We believe democratized access to advanced AI capabilities provides net benefits when coupled with appropriate safeguards and responsible deployment practices.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper does not release new high-risk models or datasets. It builds on existing pretrained models (e.g., Qwen, LLaMA) and standard datasets, which do not require additional safeguards beyond those already established by their original authors. However, the proposed method reduces the cost of LLM inference, and it may be wrongly used.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper properly credits the original sources of all models, datasets, and methods used, including citations to Qwen, LLaMA and others. There is no indication of license violations, and usage aligns with standard academic practice.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The newly proposed method, LRQK, is well documented. The paper provides theoretical foundations, implementation details, and extensive experimental settings. The released code includes instructions to reproduce results, ensuring the new assets are accessible and usable.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve any crowdsourcing experiments or research with human subjects, so this question is not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The study does not involve human participants or subject data, so there are no associated risks or requirements for IRB approval.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Although the paper evaluates methods on pretrained LLMs such as LLaMA and Qwen, these are standard models used for benchmarking. The core methodological contribution does not involve original or non-standard usage of LLMs. We use LLM only for writing, editing purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, so no special declaration is required.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

Appendix

A Details of Derivatives

A.1 Prefill Derivatives

The Lagrangian (5) is formulated as:

$$\mathcal{L}_{pre} = \frac{1}{2}\left\|\mathbf{Q}\mathbf{K}^\top - \mathbf{A}_Q\mathbf{A}_K^\top\right\|_\mathrm{F}^2 + \frac{\lambda_{pQ}}{2}\left\|\mathbf{Q} - \mathbf{A}_Q\mathbf{B}_Q\right\|_\mathrm{F}^2 + \frac{\lambda_{pK}}{2}\left\|\mathbf{K} - \mathbf{A}_K\mathbf{B}_K\right\|_\mathrm{F}^2,$$

Compute partial derivative:

$$\begin{aligned}
\frac{\partial\mathcal{L}_{pre}}{\partial\mathbf{A}_Q} &= -\left(\mathbf{Q}\mathbf{K}^\top - \mathbf{A}_Q\mathbf{A}_K^\top\right)\mathbf{A}_K - \lambda_{pQ}\left(\mathbf{Q} - \mathbf{A}_Q\mathbf{B}_Q\right)\mathbf{B}_Q^\top \\
\frac{\partial\mathcal{L}_{pre}}{\partial\mathbf{A}_K} &= -\left(\mathbf{Q}\mathbf{K}^\top - \mathbf{A}_Q\mathbf{A}_K^\top\right)^\top\mathbf{A}_Q - \lambda_{pK}\left(\mathbf{K} - \mathbf{A}_K\mathbf{B}_K\right)\mathbf{B}_K^\top \\
\frac{\partial\mathcal{L}_{pre}}{\partial\mathbf{B}_Q} &= -\lambda_{pQ}\mathbf{A}_Q^\top\left(\mathbf{Q} - \mathbf{A}_Q\mathbf{B}_Q\right) \\
\frac{\partial\mathcal{L}_{pre}}{\partial\mathbf{B}_K} &= -\lambda_{pQ}\mathbf{A}_K^\top\left(\mathbf{K} - \mathbf{A}_K\mathbf{B}_K\right)
\end{aligned} \tag{13}$$

Setting the partial derivative to be zero,

$$\frac{\partial\mathcal{L}_{pre}}{\partial\mathbf{A}_Q} := 0, \quad \frac{\partial\mathcal{L}_{pre}}{\partial\mathbf{A}_K} := 0, \quad \frac{\partial\mathcal{L}_{pre}}{\partial\mathbf{B}_Q} := 0, \quad \frac{\partial\mathcal{L}_{pre}}{\partial\mathbf{B}_K} := 0. \tag{14}$$

Solving the equations (14), we get:

$$\begin{aligned}
\mathbf{A}_Q &= \mathbf{Q}\left(\mathbf{K}^\top\mathbf{A}_K + \lambda_{pQ}\mathbf{B}_Q^\top\right)\left(\mathbf{A}_K^\top\mathbf{A}_K + \lambda_{pQ}\mathbf{B}_Q\mathbf{B}_Q^\top\right)^{-1} \\
\mathbf{A}_K &= \mathbf{K}\left(\mathbf{Q}^\top\mathbf{A}_Q + \lambda_{pK}\mathbf{B}_K^\top\right)\left(\mathbf{A}_Q^\top\mathbf{A}_Q + \lambda_{pK}\mathbf{B}_K\mathbf{B}_K^\top\right)^{-1} \\
\mathbf{B}_Q &= (\mathbf{A}_Q^\top\mathbf{A}_Q)^{-1}\mathbf{A}_Q^\top\mathbf{Q} \\
\mathbf{B}_K &= (\mathbf{A}_K^\top\mathbf{A}_K)^{-1}\mathbf{A}_K^\top\mathbf{K}
\end{aligned}$$

A.2 Decode Derivatives

The Lagrangian (8) is formulated as:

$$\begin{aligned}
\mathcal{L}_\mathrm{dec} = &\frac{1}{2}\left\|\widehat{\mathbf{q}}_t\mathbf{B}_{Q,t-1} - \mathbf{q}_t\right\|_\mathrm{F}^2 + \frac{1}{2}\left\|\widehat{\mathbf{k}}_t\mathbf{B}_{K,t-1} - \mathbf{k}_t\right\|_\mathrm{F}^2 \\
&+ \frac{\lambda_{d1}}{2}\left(\widehat{\mathbf{q}}_t\widehat{\mathbf{k}}_t^\top - \mathbf{q}_t\mathbf{k}_t^\top\right)^2 + \frac{\lambda_{d2}}{2}\left\|\widehat{\mathbf{q}}_t\mathbf{A}_{K,\Omega,t-1}^\top - \mathbf{q}_t\mathbf{K}_{\Omega,t-1}^\top\right\|_\mathrm{F}^2.
\end{aligned}$$

Compute partial derivative:

$$\begin{aligned}
\frac{\partial\mathcal{L}_\mathrm{dec}}{\partial\widehat{\mathbf{q}}_t} &= \left(\widehat{\mathbf{q}}_t\mathbf{B}_{Q,t-1} - \mathbf{q}_t\right)\mathbf{B}_{Q,t-1}^\top \\
&\quad + \lambda_{d1}\left(\widehat{\mathbf{q}}_t\widehat{\mathbf{k}}_t^\top - \mathbf{q}_t\mathbf{k}_t^\top\right)\widehat{\mathbf{k}}_t + \lambda_{d2}\left(\widehat{\mathbf{q}}_t\mathbf{A}_{K,\Omega,t-1}^\top - \mathbf{q}_t\mathbf{K}_{\Omega,t-1}^\top\right)\mathbf{A}_{K,\Omega,t-1}. \\
\frac{\partial\mathcal{L}_\mathrm{dec}}{\partial\widehat{\mathbf{k}}_t} &= \left(\widehat{\mathbf{k}}_t\mathbf{B}_{K,t-1} - \mathbf{k}_t\right)\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\left(\widehat{\mathbf{q}}_t\widehat{\mathbf{k}}_t^\top - \mathbf{q}_t\mathbf{k}_t^\top\right)^\top\widehat{\mathbf{q}}_t
\end{aligned} \tag{15}$$

By setting the partial derivative to be zero, we get:

$$\widehat{\mathbf{q}}_t\left(\mathbf{B}_{Q,t-1}\mathbf{B}_{Q,t-1}^\top + \lambda_{d1}\widehat{\mathbf{k}}_t^\top\widehat{\mathbf{k}}_t + \lambda_{d2}\mathbf{A}_{K,\Omega,t-1}^\top\mathbf{A}_{K,\Omega,t-1}\right) = \mathbf{q}_t\mathbf{B}_{Q,t-1}^\top + \lambda_{d1}\mathbf{q}_t\mathbf{k}_t^\top\widehat{\mathbf{k}}_t + \lambda_{d2}\mathbf{q}_t\mathbf{K}_{\Omega,t-1}^\top\mathbf{A}_{K,\Omega,t-1},$$

$$\widehat{\mathbf{q}}_t = \left(\mathbf{q}_t\mathbf{B}_{Q,t-1}^\top + \lambda_{d1}\mathbf{q}_t\mathbf{k}_t^\top\widehat{\mathbf{k}}_t + \lambda_{d2}\mathbf{q}_t\mathbf{K}_{\Omega,t-1}^\top\mathbf{A}_{K,\Omega,t-1}\right)\left(\mathbf{B}_{Q,t-1}\mathbf{B}_{Q,t-1}^\top + \lambda_{d1}\widehat{\mathbf{k}}_t^\top\widehat{\mathbf{k}}_t + \lambda_{d2}\mathbf{A}_{K,\Omega,t-1}^\top\mathbf{A}_{K,\Omega,t-1}\right)^{-1}.$$

Similarly, we have:

$$\begin{aligned}
\widehat{\mathbf{k}}_t\mathbf{B}_{K,t-1}\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\widehat{\mathbf{k}}_t\widehat{\mathbf{q}}_t^\top\widehat{\mathbf{q}}_t &= \mathbf{k}_t\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\mathbf{k}_t\mathbf{q}_t^\top\widehat{\mathbf{q}}_t \\
\widehat{\mathbf{k}}_t\left(\mathbf{B}_{K,t-1}\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\widehat{\mathbf{q}}_t^\top\widehat{\mathbf{q}}_t\right) &= \left(\mathbf{k}_t\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\mathbf{k}_t\mathbf{q}_t^\top\widehat{\mathbf{q}}_t\right) \\
\widehat{\mathbf{k}}_t &= \left(\mathbf{k}_t\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\mathbf{k}_t\mathbf{q}_t^\top\widehat{\mathbf{q}}_t\right)\left(\mathbf{B}_{K,t-1}\mathbf{B}_{K,t-1}^\top + \lambda_{d1}\widehat{\mathbf{q}}_t^\top\widehat{\mathbf{q}}_t\right)^{-1}
\end{aligned}$$

Update $\mathbf{B}_{Q,t-1}$ and $\mathbf{B}_{K,t-1}$. First, compute the partial derivative with respect to $\mathbf{B}_{Q,t-1}$ and $\mathbf{B}_{K,t-1}$,

$$\begin{aligned}
\frac{\partial\mathcal{L}_{\text{dec}}}{\partial\mathbf{B}_{Q,t-1}} &= \widehat{\mathbf{q}}_t^\top\left(\widehat{\mathbf{q}}_t\mathbf{B}_{Q,t-1} - \mathbf{q}_t\right) = \widehat{\mathbf{q}}_t^\top\widehat{\mathbf{q}}_t\mathbf{B}_{Q,t-1} - \widehat{\mathbf{q}}_t^\top\mathbf{q}_t, \\
\frac{\partial\mathcal{L}_{\text{dec}}}{\partial\mathbf{B}_{K,t-1}} &= \widehat{\mathbf{k}}_t^\top\left(\widehat{\mathbf{k}}_t\mathbf{B}_{K,t-1} - \mathbf{k}_t\right) = \widehat{\mathbf{k}}_t^\top\widehat{\mathbf{k}}_t\mathbf{B}_{K,t-1} - \widehat{\mathbf{k}}_t^\top\mathbf{k}_t.
\end{aligned}$$

Since the outer product of row vector, $\widehat{\mathbf{q}}_t^\top\widehat{\mathbf{q}}_t \in \mathbb{R}^{r\times r}$ is a rank 1 matrix, it is not invertible. Therefore, there is no closed-form solution for both $\mathbf{B}_{Q,t-1}$ and $\mathbf{B}_{K,t-1}$.

To update $\mathbf{B}_{Q,t-1}$ and $\mathbf{B}_{K,t-1}$, gradient descent is utilized. The gradient of $\mathcal{L}_{\text{dec}}$ with respect to $\mathbf{B}_{Q,t-1}$ and $\mathbf{B}_{K,t-1}$ are:

$$\begin{aligned}
\nabla\mathbf{B}_{Q,t-1} &= \widehat{\mathbf{q}}_t^\top\left(\widehat{\mathbf{q}}_t\mathbf{B}_{Q,t-1} - \mathbf{q}_t\right) \\
\nabla\mathbf{B}_{K,t-1} &= \widehat{\mathbf{k}}_t^\top\left(\widehat{\mathbf{k}}_t\mathbf{B}_{K,t-1} - \mathbf{k}_t\right).
\end{aligned} \tag{16}$$

The gradient descent update rules are:

$$\begin{aligned}
\mathbf{B}_{Q,t} &\leftarrow \mathbf{B}_{Q,t-1} - \eta_Q\nabla\mathbf{B}_{Q,t-1}, \\
\mathbf{B}_{K,t} &\leftarrow \mathbf{B}_{K,t-1} - \eta_K\nabla\mathbf{B}_{K,t-1},
\end{aligned} \tag{17}$$

where η_Q and η_K are the learning rates for $\mathbf{B}_{Q,t-1}$ and $\mathbf{B}_{K,t-1}$, respectively. Plugging the update rule of $\mathbf{B}_{Q,t-1}$ into the Lagrangian (8), we have,

$$\underset{\eta_Q}{\arg\min}\ \frac{1}{2}\left\|\widehat{\mathbf{q}}_t\left(\mathbf{B}_{Q,t-1} - \eta_Q\nabla\mathbf{B}_{Q,t-1}\right) - \mathbf{q}_t\right\|_{\mathrm{F}}^2. \tag{18}$$

This is a convex quadratic optimization problem with respect to the learning rate η_Q. Therefore, the optimal η_Q is given by,

$$\left(\widehat{\mathbf{q}}_t\mathbf{B}_{Q,t-1} - \mathbf{q}_t - \eta_Q\widehat{\mathbf{q}}_t\nabla\mathbf{B}_{Q,t-1}\right)\left(\widehat{\mathbf{q}}_t\nabla\mathbf{B}_{Q,t-1}\right)^\top = 0. \tag{19}$$

And the optimal η_Q^* will be,

$$\eta_Q^* = \frac{\left(\widehat{\mathbf{q}}_t\mathbf{B}_{Q,t-1} - \mathbf{q}_t\right)\left(\widehat{\mathbf{q}}_t\nabla\mathbf{B}_{Q,t-1}\right)^\top}{\left(\widehat{\mathbf{q}}_t\nabla\mathbf{B}_{Q,t-1}\right)\left(\widehat{\mathbf{q}}_t\nabla\mathbf{B}_{Q,t-1}\right)^\top}. \tag{20}$$

Similarly, for $\mathbf{B}_{K,t-1}$, the optimal η_K^* can be computed as,

$$\eta_K^* = \frac{\left(\widehat{\mathbf{k}}_t\mathbf{B}_{K,t-1} - \mathbf{k}_t\right)\left(\widehat{\mathbf{k}}_t\nabla\mathbf{B}_{K,t-1}\right)^\top}{\left(\widehat{\mathbf{k}}_t\nabla\mathbf{B}_{K,t-1}\right)\left(\widehat{\mathbf{k}}_t\nabla\mathbf{B}_{K,t-1}\right)^\top}. \tag{21}$$

B Additional Results

B.1 Results on RULER 128K And LongBench

Since the inverse computation in PyTorch operates in `float32`, the parameters are temporarily cast to `float32` before executing Algorithm 1 and Algorithm 2. The models are otherwise run with `bfloat16` for inference.

The CPU used is an AMD EPYC 7742 64-Core Processor, featuring 64 cores with 2 threads per core. For the RULER 128K experiment, the model runs on a single NVIDIA A100 GPU with 80 GB of memory. For the LongBench experiment, the model is executed on a single A100 GPU with 40 GB of memory. The batch size is set to one. Further results are provided in Table 5.

Table 5: Comparison of different models and methods on RULER 128K (left; top-2048, lite tokens=64) and LongBench (right; rank $r = 16$, lite tokens=64).

Methods	S1	S2	MK1	MK2	MQ	MV	QA-1	QA-2	VT	FWE		NQA	MQA	GRep	SAM	PRetr	LCC
Llama-3-8B-1M	100.00	100.00	98.96	98.96	98.96	95.57	75.00	48.96	78.54	71.85		18.98	41.84	34.18	35.96	81.50	56.07
Loki	18.75	1.04	2.08	0.00	1.56	0.78	4.17	13.54	26.04	25.35		2.26	10.19	28.97	7.84	40.52	31.44
InfiniGen	100.00	98.96	84.38	53.13	63.28	54.95	65.63	48.96	**81.67**	50.35		14.39	31.46	27.38	21.97	74.30	38.58
Quest	100.00	100.00	**98.96**	77.08	97.65	93.49	60.42	50.00	77.08	65.63		**20.13**	36.63	27.11	35.63	79.00	53.64
ShadowKV	100.00	100.00	97.92	**98.96**	96.88	95.83	72.92	52.08	**81.67**	**72.57**		17.17	39.73	31.62	35.87	80.00	63.93
LRQK $r = 32$	81.00	100.00	97.00	42.00	**99.25**	**98.00**	**75.00**	**53.00**	80.20	69.67	top1024	16.52	**40.48**	20.40	26.35	89.00	66.13
LRQK $r = 16$	63.67	69.00	50.00	90.00	80.25	81.50	70.00	56.00	64.00	63.67	top2048	15.86	40.15	**34.15**	**36.64**	**91.50**	**66.53**
Llama-3.1-8B	100.00	100.00	98.96	91.67	98.96	95.31	82.29	47.92	68.96	71.18		31.56	55.10	34.45	29.84	100.00	67.31
Loki	68.75	32.29	32.29	20.83	42.71	28.65	41.67	33.33	24.79	29.86		2.31	18.89	31.16	15.91	94.88	44.60
InfiniGen	100.00	77.08	78.13	13.54	58.07	47.40	65.63	41.67	60.83	50.35		27.23	52.72	29.61	24.42	98.93	56.89
Quest	100.00	98.96	97.92	34.38	93.49	88.54	70.83	44.79	65.63	**68.40**		29.70	49.04	30.43	29.85	98.50	57.35
ShadowKV	100.00	100.00	100.00	**83.33**	97.92	92.19	**81.25**	**48.96**	67.08	64.93		30.93	55.20	**32.79**	**30.40**	**99.50**	**66.03**
LRQK $r = 32$	87.00	100.00	98.00	80.00	**98.25**	**96.25**	81.00	41.00	**70.00**	44.67	top1024	29.01	**55.85**	19.50	12.02	**99.50**	24.60

B.2 Results on Llama And Qwen

More results of the proposed methods on two different models, Llama-3-8B-1M and Qwen2.5-7B, are shown in Table 6. The model is running on a single A100 GPU with 40G memory. The configuration for LRQK are: rank=16, top-256 active tokens, and 16 lite tokens.

Table 6: More results on subset of RULER-4K/8K/16K of two models.

RULER 4K	QA-1	QA-2	VT	RULER 4K	QA-1	QA-2	VT
LLaMA-3-8B-1M	82.00	58.00	99.00	Qwen2.5-7B	90.00	64.00	99.20
+ LRQK	84.00	57.00	98.80	+ LRQK	91.00	65.00	96.60
RULER 8K	QA-1	QA-2	VT	RULER 8K	QA-1	QA-2	VT
LLaMA-3-8B-1M	82.00	57.00	99.60	Qwen2.5-7B	82.00	57.00	97.40
+ LRQK	79.00	47.00	94.00	+ LRQK	78.00	56.00	70.00
RULER 16K	QA-1	QA-2	VT	RULER 16K	QA-1	QA-2	VT
LLaMA-3-8B-1M	80.00	57.00	99.80	Qwen2.5-7B	76.00	63.00	98.80
+ LRQK	77.00	52.00	57.20	+ LRQK	71.00	62.00	64.20

B.3 Results on Mistral and Phi-3

Two additional models, 'mistralai/Mistral-7B-Instruct-v0.3' (Mistral) and 'microsoft/Phi-3-mini-128k-instruct' (Phi-3), are evaluated under the default LRQK hyperparameters (rank=32, top-2048, and 64 lite tokens). Due to the 48 GB memory constraints of the NVIDIA A6000 GPU, Mistral is evaluated on RULER 32K and Phi-3-mini is evaluated on RULER 16K. Table 7 summarizes the results, demonstrating that LRQK generalizes effectively to architecturally diverse models.

B.4 Results on Larger Models

To evaluate the scalability of LRQK to larger models, additional experiments are conducted on 'Qwen/Qwen2.5-14B-Instruct'(Qwen 14B) with 64K context length and 'Qwen/Qwen2.5-32B-Instruct' (Qwen 32B) with 16K context length under A100 80G GPU. Following the official guidance from Qwen2.5, YaRN [29] is applied for positional encoding extension in the 64K context setting. All experiments use the default LRQK hyperparameters (rank=32, top-2048, and 64 lite tokens).

Table 8 presents the results on the RULER benchmark. For Qwen 14B at 64K context, LRQK yields substantial improvements on retrieval tasks, demonstrating its effectiveness at extended context lengths. For Qwen 32B at 16K context, the baseline model already achieves near-perfect performance on most tasks, with LRQK maintaining comparable results, indicating that the method does not degrade performance on models with strong native long-context capabilities.

Table 7: Evaluation results of Mistral and Phi-3 on RULER benchmarks with varying context lengths.

RULER 32K	Mistral	+LRQK	RULER 16K	Phi-3	+LRQK
FWE	67.00	93.00	FWE	92.00	91.00
S1	98.00	97.00	S1	100.00	100.00
S2	91.00	100.00	S2	100.00	100.00
MK1	83.00	97.00	MK1	96.00	96.00
MK2	63.00	51.00	MK2	100.00	100.00
MV	85.75	95.00	MV	90.00	90.25
MQ	83.75	93.00	MQ	90.50	87.50
QA-1	59.00	63.00	QA-1	80.00	81.00
QA-2	45.00	47.00	QA-2	51.00	51.00
VT	98.40	96.40	VT	99.60	99.60

Table 8: Performance comparison on Qwen2.5-14B-Instruct (64K context) and Qwen2.5-32B-Instruct (16K context) with and without LRQK on the RULER benchmark. Columns '+LRQK' indicate the performance of the model with LRQK.

RULER 64K	Qwen 14B	+LRQK	RULER 16K	Qwen 32B	+LRQK
FWE	90.33	78.33	FWE	94.67	94.00
S1	57.00	99.00	S1	100.00	100.00
S2	55.00	96.00	S2	100.00	100.00
MK1	46.00	80.00	MK1	100.00	100.00
MK2	24.00	22.00	MK2	99.00	99.00
MV	54.00	82.25	MV	100.00	100.00
MQ	49.75	88.50	MQ	100.00	100.00
QA-1	64.00	50.00	QA-1	86.00	86.00
QA-2	41.00	36.00	QA-2	66.00	66.00
VT	57.80	97.20	VT	86.40	82.60

B.5 Results With KVQuant

To investigate whether LRQK is compatible with quantization based compression methods, the combination of LRQK with KVQuant [6], a representative KV cache quantization approach, is evaluated. Experiments are conducted on 'meta-llama/Llama-3.1-8B-Instruct' (Llama-3.1-8B) and 'Qwen/Qwen2.5-7B-Instruct' (Qwen 2.5-7B-Instruct) using the RULER 32K benchmark. The LRQK hyperparameters are set to $r = 32$, top-$k = 2048$, and 64 lite tokens.

Table 9 presents the results comparing models with KVQuant alone versus the combination of KVQuant and LRQK. For Llama-3.1-8B, the combined approach maintains competitive performance across most tasks. For Qwen2.5-7B, combining both methods yields mixed results: while some tasks show improvements (e.g., MK1, MV, MQ), others exhibit slight degradation (e.g., MK2). Overall, the results demonstrate that LRQK can be effectively combined with quantization methods without catastrophic performance loss, suggesting that the proposed method and quantization operate on complementary aspects of KV cache compression.

B.6 Impact of Initialization Strategies

Three initialization strategies are investigated for the low rank factors $\mathbf{A}_Q \in \mathbb{R}^{l\times r}$ and $\mathbf{A}_K \in \mathbb{R}^{l\times r}$, where l is the sequence length, and r is the chosen rank, and the batch size or the number of attention heads are ignored as discussed in Section 4. The design rationale for these strategies stems from the goal of finding effective initialization that either minimize reconstruction error or provide sufficient representational capacity for the low rank approximation.

Random Gaussian Initialization (randn). The first strategy initializes both factors with random Gaussian noise:

$$\mathbf{A}_Q, \mathbf{A}_K \sim \mathcal{N}(0, 1). \tag{22}$$

Table 9: Performance comparison on RULER 32K with KVQuant applied alone and in combination with LRQK. Columns '+KVQuant' indicate KVQuant only, and columns '++LRQK' indicate both KVQuant and LRQK applied.

	Llama-3.1-8B		Qwen2.5-7B-Instruct	
RULER 32K	+KVQuant	++LRQK	+KVQuant	++LRQK
FWE	89.58	85.00	94.00	90.33
S1	100.00	100.00	100.00	100.00
S2	100.00	100.00	100.00	99.00
MK1	100.00	100.00	96.00	100.00
MK2	95.00	96.00	76.00	56.00
MV	95.31	96.75	70.50	93.50
MQ	89.00	89.00	97.00	99.25
QA-1	83.00	82.00	73.00	74.00
QA-2	56.25	51.00	52.00	53.00
VT	87.50	95.40	95.40	90.60

This approach provides a simple, parameter-free initialization that avoids any dependence on the input matrices. While it does not leverage information from $\mathbf{Q}$ and $\mathbf{K}$, it offers computational efficiency and serves as a neutral baseline.

Independent Top-r Selection (top). The second strategy aims to minimize reconstruction error by selecting the most salient dimensions from $\mathbf{Q}$ and $\mathbf{K}$ independently. The L_1 norm is computed of each dimension across the sequence:

$$\mathbf{s}_Q = \sum_{i=1}^{l} |\mathbf{Q}[i,:]| \in \mathbb{R}^{1\times d}, \qquad \mathbf{s}_K = \sum_{i=1}^{l} |\mathbf{K}[i,:]| \in \mathbb{R}^{1\times d}, \tag{23}$$

where d is the head dimension. With these importance scores, the initialized low rank factors are constructed as:

$$\mathbf{A}_Q \leftarrow \mathbf{Q}_{\text{top-}r(\mathbf{s}_Q)}, \qquad \mathbf{A}_K \leftarrow \mathbf{K}_{\text{top-}r(\mathbf{s}_K)}. \tag{24}$$

This strategy independently optimizes each factor to capture the most significant features of its respective matrix.

Joint Top-r Selection (topcol). The third strategy selects dimensions based on the combined importance in both $\mathbf{Q}$ and $\mathbf{K}$, ensuring that $\mathbf{A}_Q$ and $\mathbf{A}_K$ share the same column indices:

$$\mathbf{s}_{QK} = \mathbf{s}_Q + \mathbf{s}_K, \qquad \Omega = \text{top-}r(\mathbf{s}_{QK}), \tag{25}$$

$$\mathbf{A}_Q \leftarrow \mathbf{Q}_\Omega, \qquad \mathbf{A}_K \leftarrow \mathbf{K}_\Omega. \tag{26}$$

This joint selection promotes alignment between the query and key subspaces, potentially facilitating more coherent attention patterns in the low rank approximation.

Experimental Setup and Results. These three initialization strategies are evaluated on 'meta-llama/Llama-3.1-8B-Instruct-1M' and 'Qwen/Qwen2.5-7B-Instruct' using the RULER 16K benchmark with default LRQK hyperparameters ($r = 32$, top-$k = 2048$, 64 lite tokens).

Table 10 presents the results. Remarkably, all three initialization strategies achieve nearly identical performance across most tasks, with differences typically within 1-2 percentage points. This robustness suggests that the specific choice of initialization has minimal impact on the final performance, indicating that the Algorithm 1 is resilient to the initialization of the low rank factors. The consistency across strategies, from uninformed random initialization to carefully selected subspaces, demonstrates the method's inherent stability.

Given this empirical equivalence, the **randn** initialization is recommenced as a default due to its computational efficiency: it avoids the overhead of computing importance scores and performing top-k selection, making it particularly suitable for large-scale deployments.

Table 10: Performance comparison of initialization strategies for $\mathbf{A}_Q$ and $\mathbf{A}_K$ on RULER 16K. Despite their different design rationales, all three methods achieve comparable results.

	Llama-3.1-8B-1M			Qwen2.5-7B-Instruct		
RULER 16K	randn	top	topcol	randn	top	topcol
FWE	86.67	86.67	87.00	85.67	86.33	86.33
S1	100.00	100.00	100.00	100.00	100.00	100.00
S2	100.00	100.00	100.00	100.00	100.00	100.00
MK1	100.00	100.00	100.00	99.00	99.00	99.00
MK2	99.00	99.00	99.00	91.00	91.00	91.00
MV	98.75	98.50	98.50	97.75	98.00	97.75
MQ	94.75	94.50	94.75	99.75	99.75	99.75
QA-1	89.00	89.00	89.00	76.00	76.00	76.00
QA-2	61.00	61.00	61.00	61.00	62.00	62.00
VT	83.80	83.00	85.40	98.80	98.80	98.40

C Runtime Performance Analysis

C.1 Throughput Results

Runtime performances of LRQK are evaluated on 'meta-llama/Llama-3.1-8B-Instruct-1M', comparing it against standard GPU-only and CPU offloading approaches. Experiments are conducted using the Hugging Face `transformers` library[2] on a single NVIDIA A100 GPU (40 GB memory) with batch size 1. We utilize the RULER QA-2 (HotpotQA) dataset with context lengths ranging from 4K to 64K tokens. The parameters of LRQK are set as $r = 16$, top-$k = 1024$ active tokens, and 16 lite tokens.

Experimental Configurations:

- **GPU only**: All KV caches stored in GPU memory (baseline).
- **CPU offload**: KV caches offloaded to CPU memory and transferred to GPU during decoding.
- **LRQK default**: Full LRQK implementation with hit/miss buffer enabled for optimized cache management.
- **LRQK no hit/miss**: LRQK without hit/miss buffer to isolate the impact of buffer optimization.

Table 11 presents the tokens processed per second during prefill (P) and decode (D) stages. The GPU-only method achieves the highest throughput for shorter contexts (4K to 32K) but encounters out-of-memory (OOM) errors at 64K tokens due to the growth of KV cache size.

CPU offload enables processing of longer contexts but suffers severe throughput degradation, particularly during decoding at 32K to 64K tokens. This performance drop stems from the need to transfer the entire KV cache between CPU and GPU for each decoding step, creating a bottleneck that scales linearly with context length.

In contrast, LRQK default maintains relatively stable decoding throughput across all context lengths, demonstrating consistent performance even at 64K tokens. This stability arises from LRQK's selective transfer mechanism: instead of moving the entire KV cache, only the top-k active tokens and lite tokens are transferred from CPU to GPU, significantly reducing data movement overhead. At 64K context, LRQK achieves faster decoding than CPU offload while avoiding the OOM issues of GPU-only execution.

The LRQK no hit/miss configuration shows reduced decoding performance compared to LRQK default, highlighting the importance of the hit/miss buffer optimization. Without this buffer, the method incurs additional overhead from repeated top-k selection operations, which are not well-optimized for CPU cache locality. Notably, prefill performance remains comparable across LRQK variants, as this stage is less sensitive to cache management strategies.

[2] https://huggingface.co/docs/transformers/v4.52.2/en/main_classes/text_generation

Table 11: Throughput comparison (tokens/s) for ‘meta-llama/Llama-3.1-8B-Instruct-1M’ on RULER QA-2 across context lengths. P: prefill stage; D: decode stage. LRQK default maintains stable decode performance while avoiding OOM at 64K tokens.

	4K		8K		16K		32K		64K	
	P	D	P	D	P	D	P	D	P	D
GPU only	37500.29	35.40	37734.19	35.64	33649.79	35.36	32111.45	35.77	OOM	OOM
CPU offload	6945.10	4.31	6984.35	4.32	7073.53	2.40	4849.19	0.98	4131.00	0.50
LRQK default	7120.49	5.68	7280.39	5.72	6379.91	5.49	5436.66	5.36	4103.05	6.80
LRQK no hit/miss	7180.64	2.50	6747.19	2.53	5317.06	2.31	4920.02	2.48	4163.86	1.95

C.2 Comparison with Baseline Methods

To provide a more comprehensive performance analysis, LRQK is compared against vanilla attention and ShadowKV[16] on a text summarization task using 20 samples from LongBench with 32K context and up to 128 output tokens. Method ShadowKV is configured with its default parameters (sparse budget = 2048, rank=160, chunk size=8), and the definition of ‘rank’ is not the same as this paper, please refer to the original paper for more details. Experiments are conducted on NVIDIA GeForce RTX 3090 (24 GB) GPUs. Due to memory constraints, vanilla attention requires 2 GPUs, while ShadowKV and LRQK operate on a single GPU.

Table 12 presents detailed performance metrics. Vanilla attention achieves the highest throughput but requires dual-GPU deployment with large memory usage across both two devices. ShadowKV reduces throughput compared to vanilla. LRQK variants achieve 489 to 646 tokens/s depending on hyperparameters, representing a trade-off between memory efficiency and speed. Basically, LRQK with less rank and smaller k can achieve larger tokens/s. The GPU power of LRQK is smaller than Vanilla and ShadowKV, which means the proposed method is not fully utilizing the GPU resources. This is one limitation of the proposed method. It requires more time to wait data indexing in CPU.

Table 12: Performance comparison on NVIDIA GeForce RTX 3090 (24 GB, 250W) for text summarization (LongBench, 32K context). Vanilla attention uses 2 GPUs; other methods use 1 GPU. Tokens/s is computed via (number of all tokens / total time).

	Vanilla		ShadowKV	LRQK			
				$k = 2048$		$k = 1024$	
	GPU0	GPU1		$r = 32$	$r = 16$	$r = 32$	$r = 16$
Average GPU Util (%)	38.61	48.38	58.00	63.64	65.29	56.86	51.15
Max GPU Util (%)	99.95	100.00	100.00	100.00	100.00	100.00	100.00
GPU Memory (GB)	16.94	17.56	17.93	19.82	18.85	19.51	19.33
GPU Power (W)	187.24	233.21	243.85	203.98	211.96	199.59	199.85
Tokens/s	1775.70		1083.55	489.39	556.22	603.05	646.33
Time (s)	17.87		29.29	64.84	57.04	52.62	57.04

D Hyperparameter Selection Guidelines

LRQK introduces several hyperparameters that require configuration: rank r for low rank approximation, top-k for active token selection, number of lite tokens, and convergence parameters (iterations and tolerance). While optimal values vary across model architectures and tasks, practical guidelines are provided based on our extensive experiments to minimize tuning effort.

Default Configuration. It is recommended starting with the following default configuration, which achieves strong performance across diverse settings: rank $r = 32$, active tokens top-$k = 2048$, lite tokens 64, iterations 2, and tolerance 10^{-2}. This configuration provides a balanced trade-off between approximation quality, memory efficiency, and computational overhead, and serves as an effective starting point for task-specific tuning.

Rank r. The rank r controls the dimensionality of the low rank factors $\mathbf{A}_Q, \mathbf{A}_K \in \mathbb{R}^{\cdots l \times r}$ for each attention head independently. Since typical attention heads have dimension $d_{\text{head}} = 128$, the theoretical range for r is $[1, 128]$. In practice, it is found that $r \in \{8, 16, 32, 48\}$ can provide a favorable balance between approximation quality and computational cost. Lower ranks (e.g., $r = 8$) reduce memory usage but may sacrifice accuracy, while higher ranks (e.g., $r = 48$) improve approximation at the cost of increased computation. For most applications, $r = 32$ offers an effective compromise.

Top-k Active Tokens. The top-k parameter determines how many high-attention tokens are retained in GPU memory and directly influences the memory-context length trade-off. The optimal k scales with context length: short contexts ($\leq$ 4K) $k = 256$ suffices for most tasks, medium contexts (8K to 16K): $k = 512$ to 1024 may provide good coverage, long contexts ($\geq$ 32K) $k = 2048$ is recommended. Smaller k values reduce memory footprint but may miss important context, while larger k improves recall at the cost of increased GPU memory usage and data transfer overhead. The relationship between k and context length should be considered when deploying LRQK for production use cases.

Lite Tokens. The number of lite tokens specifies how many recent tokens are always retained in GPU memory to capture local context. Recommend choices are $\{16, 32, 64\}$, with minimal impact on overall computational cost. This parameter is relatively insensitive; it is recommended 64 lite tokens as a conservative default that ensures sufficient local context coverage.

Convergence Parameters. The iterative updates in Algorithms 1 and 2 require two convergence parameters:

- **Iterations**: Number of update cycles for low rank factors. Typically, 2 or 3 iterations may suffice, as the approximation would converge.
- **Tolerance**: Convergence threshold defined as the mean squared error between successive low rank matrices. Values of 10^{-2} or 10^{-3} work well in practice. Tighter tolerance (e.g., 10^{-3}) improves accuracy slightly but increases computation.

Tuning Strategy. The search space for LRQK hyperparameters is relatively compact compared to other optimization based methods. It is recommended the following tuning protocol:

1. Start with the default configuration ($r = 32, k = 2048$, 64 lite tokens, 2 iterations, tolerance 10^{-2}).
2. If memory is limited, reduce k proportionally to the context length or decrease r.
3. If accuracy is insufficient, increase k (up to hardware limits) or r.

In our experiments, this strategy typically requires evaluating fewer configurations to identify near-optimal settings for a given task and model architecture.

Implementation and Integration Overhead. LRQK introduces modest computational overhead relative to standard attention:

- **Prefill stage**: Additional cost for computing low rank projections, partially offset by reduced attention computation over selected tokens.
- **Decode stage**: Incremental updates to low rank matrices add computation, but attention over fewer tokens (k instead of full sequence) reduces overall cost.
- **Cache management**: CPU-GPU transfers for selected tokens and cache lookup operations add latency, though this is mitigated by transferring only k tokens rather than the entire KV cache.