Reformulating KV Cache Eviction Problem for Long-Context LLM Inference

Tho Mai
KAIST
Daejeon, South Korea
thomh1511@kaist.ac.kr

Joo-Young Kim*
KAIST
Daejeon, South Korea
jooyoung1203@kaist.ac.kr

Abstract

Large language models (LLMs) support long-context inference but suffer from substantial memory and runtime overhead due to Key-Value (KV) Cache growth. Existing KV Cache eviction methods primarily rely on local attention weights, neglecting the influence of value representations, output projection, and inter-head interactions. In this work, we reformulate KV Cache eviction from a conventional head-wise, weight-averaging approach into an output-aware, layer-wise matrix multiplication approximation problem. We introduce LaProx, a novel eviction strategy that explicitly models the multiplicative interaction between attention maps and projected value states to accurately quantify token contributions while accounting for inter-head dependencies. Building on this metric, we propose the first unified eviction strategy that assigns globally comparable importance scores to tokens, enabling model-wide selection instead of local, head-wise decisions. Experimental results across 19 datasets on long-context benchmarks LongBench and Needle-In-A-Haystack demonstrate that our approach maintains model performance with only 5% of the KV cache and consistently outperforms prior works across all configurations. Notably, our method achieves up to $2\times$ accuracy loss reduction under extreme compression scenarios compared to existing state-of-the-art baselines with minimal overhead. Our code is available at https://github.com/huuthomai/LaProx.

1 Introduction

Recent advances in large language models (LLMs) have significantly extended their ability to process long contexts, enabling strong performance in applications such as multi-turn dialogue [40], question answering [25], code generation [18], and document understanding [50]. To accelerate autoregressive inference, transformers cache key and value states from previous tokens, avoiding repeated attention computation. While this Key-Value (KV) cache is essential for efficient decoding, its size grows linearly with context length, quickly becomes a major bottleneck for memory usage and decoding latency in long-context settings. While techniques such as head merging or architectural modifications [2] can partially alleviate these costs during training, they are often incompatible with fixed, pretrained models commonly used in deployment. Consequently, managing KV cache efficiently at inference time—without retraining or altering model parameters—becomes a critical challenge for scalable and cost-effective long-context LLM deployment under realistic memory and hardware constraints [53].

To operate large language models under constrained memory budgets, a common strategy is to dynamically reduce the size of the key-value (KV) cache by evicting entries deemed less influential during inference. Prior work has shown that, in practice, only a small subset of cached tokens meaningfully contributes to the attention output [52, 43], motivating a class of eviction-based methods that selectively retain critical entries while discarding the rest. Early approaches exploit

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

empirical observations that attention weights are highly concentrated, whereby a minority of tokens consistently receives the majority of attention mass. Building on this phenomenon, several methods identify important cache entries by averaging attention scores over time, with later refinements introducing observation windows, pooling mechanisms [30] or adaptive budget allocation [34] to better preserve salient information. However, such strategies are often heuristics and lack a principled formulation of what constitutes cache entry criticality. Consequently, the precise relationship between attention behavior, value representations, and their joint impact on the final model output remains insufficiently characterized.

In this paper, we reformulate KV cache eviction as an optimization problem that preserves layers' attention outputs under a fixed budget. By explicitly modeling the output as a product of attention, value, and output matrix, we move beyond conventional attention-only heuristics, allowing us to rank cache entries by their actual contribution to the multiplicative interactions that form the final layer output. Critically, this formulation reveals that token importance is fundamentally coupled to the aggregate representation formed within each layer and, consequently, to the model's eventual output. This observation suggests that eviction is most effectively managed at the model level rather than through isolated, head-wise decisions. Based on this insight, we propose a novel eviction strategy enabling more effective global cache selection.

Our contributions are summarized as follows:

1. We demonstrate that attention weights alone provide an incomplete measure of token importance, and accurate selection must account for output information and attention layer's structure itself.

2. We reveal that existing independent head-wise eviction is suboptimal because it neglects inter-head and inter-layer interactions, and show that eviction should be done at the model-level.

3. We introduce **La**yer **Approx**imated Cache (**LaProx**), a new eviction strategy that approximates layer's output by evaluating tokens across heads and layers simultaneously without any calibration.

4. Extensive evaluations on long-context benchmarks demonstrate that the proposed method consistently outperforms attention-based eviction strategies, confirming the effectiveness of our proposal.

2 Background and Related Works

2.1 Basic of Attention and KV Cache Operations

For clarity, we describe the mechanism using Multi-Head Attention (MHA) and omit the layer index, noting that the formulation applies identically to all transformer attention layers. Let $\mathbf{X} \in \mathbb{R}^{S \times D}$ be the token embeddings of a sequence of length S, where D is the model hidden dimension. Each attention head operates on a subspace of dimension d_h, with $D = H \cdot d_h$ for H heads. The projection matrices $\mathbf{W}_Q^{(h)}, \mathbf{W}_K^{(h)}, \mathbf{W}_V^{(h)} \in \mathbb{R}^{D \times d_h}$ map the shared hidden representations into head-specific query, key, and value states. During prompt processing, each head computes

$$\mathbf{Q^{(h)}} = \mathbf{X}\mathbf{W}_Q^{(h)}, \mathbf{K^{(h)}} = \mathbf{X}\mathbf{W}_K^{(h)}, \mathbf{V^{(h)}} = \mathbf{X}\mathbf{W}_V^{(h)} \tag{1}$$

with attention weights

$$\mathbf{A^{(h)}} = \mathrm{Softmax}\left(\frac{Q^{(h)}{K^{(h)}}^\top}{\sqrt{d_h}}\right) \tag{2}$$

The per-head attention outputs are then concatenated,

$$\mathbf{AV} = \mathrm{Concat}(\mathbf{A}^{(1)}\mathbf{V}^{(1)}, \ldots, \mathbf{A}^{(h)}\mathbf{V}^{(h)}) \tag{3}$$

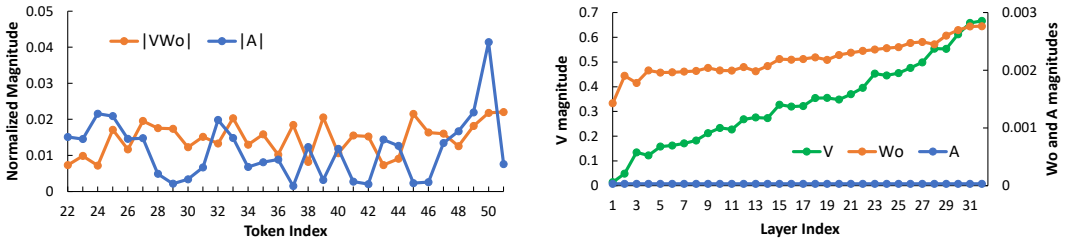
and projected to produce the final attention output,

$$\mathbf{O} = \mathbf{AV}\mathbf{W}_O \tag{4}$$

Following the projection $\mathbf{W}_O$, the final layer output is integrated via a residual connection:

$$\mathbf{Y} = \mathrm{Norm}(\mathbf{O} + \mathbf{X}) \tag{5}$$

where $\mathbf{X}$ is the input identity and Norm denotes a normalization function.



(a) A and VW_O patterns.

(b) Average strength.

Figure 1: Pattern and magnitude of A and VW_O.

During autoregressive decoding, at each decoding step i, only the newly generated token embedding $\mathbf{x_i} \in \mathbb{R}^{1\times D}$ is projected to obtain its head-wise query, key, and value states. To avoid recomputation of past tokens, the new key-value pairs are appended to the cache

$$\mathbf{K}^{(h)} \leftarrow \mathrm{Concat}(\mathbf{K}^{(h)}, \mathbf{x}_i\mathbf{W}_K^{(h)}), \qquad \mathbf{V}^{(h)} \leftarrow \mathrm{Concat}(\mathbf{V}^{(h)}, \mathbf{x}_i\mathbf{W}_V^{(h)}) \tag{6}$$

and the query $\mathbf{q_i^{(h)}} = \mathbf{x_i}\mathbf{W}_Q^{(h)}$ attends over the cached keys using equation 2.

While KV caching significantly reduces computation during decoding, the cache grows linearly with sequence length, leading to substantial memory and attention overhead in long-context inference.

2.2 KV Cache Eviction

KV cache eviction during inference reduces memory and computational overhead without modifying the attention mechanism. Its objective is to retain important tokens while removing low-impact ones. Early methods, such as StreamingLLM [45], adopt window-based strategies that preserve attention sinks and recent tokens while LongFormer [4] uses two types of sliding windows cooperating with some pre-selected input locations. While efficient, these approaches may discard informative tokens in the middle of long sequences, degrading long-context performance. Other works, including H2O [52] and Scissorhands [32], rank KV entries using accumulated attention scores to better capture token importance. Building on this line of work, SnapKV [30] and CAKE [34] further improve performance by averaging attention within an observation window and applying a pooling operation, achieving state-of-the-art (SOTA) results.

Beyond token selection, several studies explore non-uniform cache budget allocation. Layer-wise approaches such as PyramidInfer [47] and PyramidKV [5] assign budgets based on network depth, while D2O [42] and CAKE [34] adjust cache sizes using layer-specific attention variance. At the head level, AdaKV [11] applies top-k selection across head-scores with an empirical safeguard, whereas HeadKV [13] uses calibration procedures to determine fixed per-head budgets prior to inference.

A few works go beyond attention scores. For example, LAVa [36] and CAOTE [15] leverage value representations in their eviction indicators but omit the output projection; meanwhile, CriticalKV [12] relies on two empirical safeguards to rescale the mean attention scores with output information, disregarding the actual formulation of the attention layer.

Despite their competitive results, existing methods rely primarily on attention weights for both eviction and budget allocation, or heuristically leverage output information **without considering the actual layer's formulation.** Furthermore, these approaches are limited to performing eviction on a per-head basis, **neglecting cross-head and cross-layer interactions.** In contrast, this work proposes a principled eviction criterion that incorporates both attention probabilities and VW_O contributions, and the cross-heads interaction, providing a more accurate measure of token importance.

3 Motivation

In this section, we investigate the relationship between attention weight (A) and the value–output projection (VW_O). Specifically, we examine whether the average of A alone can serve as a faithful proxy for the attention layer output, i.e., whether attention weights A are sufficient to characterize the

whole product AVW_O. This approach assumes two key conditions are satisfied: (1) the patterns of A and VW_O are well aligned, and (2) the magnitude of A is not dominated by VW_O.

Experiment setup. Our analysis is conducted using the Mistral-7B-Instruct-v0.3 model. For visualization clarity, we display only a contiguous subset of tokens.

Observation. Figure 1a reports the normalized per-token magnitudes of $|A|$ and $|VW_O|$. While the two quantities share some high-score tokens (such as #25 or #49-50), their overall patterns differ significantly. Many steps even show opposite peaks; for instance, tokens #37 and #39 have high VW_O values but low A values. This indicates that A and VW_O assess token importance differently, and one cannot be used in place of the other.

Furthermore, Figure 1b reveals that the value range of A is much smaller than VW_O. As attention weights are normalized probabilities, their values are confined to a narrow range, whereas VW_O has a much wider range of values, which expands in deeper layers.

These observations demonstrate that attention weights alone are insufficient to represent the attention layer output, motivating the incorporation of value and output projection in cache eviction decisions.

4 Methodology

4.1 Eviction Indicator

Equations 3 and 4 show that the standard MHA is defined as the concatenation of all head outputs followed by a linear projection. Although the output projection mixes attention information from all heads, the computation can be exactly decomposed into a sum of independent head-wise contributions.

Remark 4.1. Let $W_O = [W_O^1, W_O^2, \ldots, W_O^h] \in \mathbb{R}^{hd_h \times d}$ be the output projection matrix partitioned along the head dimension, and $H^i = A^i V^i \in \mathbb{R}^{S \times d_h}$ denote the output of the i-th attention head. Then, the output of a standard MHA layer can be expressed as

$$\text{Output} = \sum_{i=1}^{h} H^i W_O^i = \sum_{i=1}^{h} A^i V^i W_O^i. \tag{7}$$

Remark 4.1 shows that **by integrating VW_O at the head level, we can evaluate token importance across the entire layer**. To quantify token importance, we leverage matrix multiplication associativity and compute VW_O first to preserve the key-value alignment between the attention matrix A and the projected values VW_O. A naive but limited approach of independently **scaling the average attention scores by the magnitude of VW_O will overlook the fundamental nature of the attention output,** which is formed through a matrix multiplication. Since our objective is to preserve this layer attention output under a constrained KV cache budget, we instead treat cache eviction as a matrix multiplication approximation problem: selecting a subset of tokens that best approximates the full product $A \times (VW_O)$. From this perspective, we draw on Monte Carlo analyses of matrix multiplication to provide a rigorous mathematical basis for our eviction criteria. This theory states that the approximation error of a matrix product is minimized when the selection follows the product of the Euclidean norms of the corre-

Algorithm 1 Eviction Score Computation

Input: Query $\mathbf{Q}$, KV Cache $(\mathbf{K}, \mathbf{V})$, Projection W_O, Budget B_{total}, Observation Window w
Output: Compressed KV cache $(\tilde{\mathbf{K}}, \tilde{\mathbf{V}})$

// Compute attention weight and projected values
$\mathbf{A} \leftarrow \text{Softmax}\left(\frac{\mathbf{Q}[-\mathbf{w}:,]\mathbf{K}^\top}{\sqrt{d_k}}\right)$
$\mathbf{H} \leftarrow VW_O$

// Score tokens
$T \leftarrow$ number of cached tokens
for $i = 0$ **to** T **do**
  if $i < T - w$ **then**
    $\mathbf{p}[\mathbf{i}] \leftarrow \|\mathbf{A}[:, \mathbf{i}]\|_2 \cdot \|\mathbf{H}[\mathbf{i}, :]\|_2$
  else
    $\mathbf{p}[\mathbf{i}] \leftarrow \infty$
  end if
end for

// Evict tokens
$\mathcal{S} \leftarrow \text{TopK}(\mathbf{p}, B_{total})$
$(\tilde{\mathbf{K}}, \tilde{\mathbf{V}}) \leftarrow (\mathbf{K}[\mathcal{S}], \mathbf{V}[\mathcal{S}])$

return $(\tilde{\mathbf{K}}, \tilde{\mathbf{V}})$

sponding column-row pairs [9, 1].

$$p_i = \frac{\|\boldsymbol{A}[:,i]\|_2 \, \|\boldsymbol{V}\boldsymbol{W}_{\boldsymbol{O}}[i,:]\|_2}{\sum_j^S \|\boldsymbol{A}[:,j]\|_2 \, \|\boldsymbol{V}\boldsymbol{W}_{\boldsymbol{O}}[j,:]\|_2} \tag{8}$$

More generally, the score can be represented by:

$$p_i \propto \|\boldsymbol{A}[:,i]\|_2 \, \|\boldsymbol{V}\boldsymbol{W}\boldsymbol{o}[i,:]\|_2 \tag{9}$$

Equation 9 shows the eviction score $\mathbf{p_i}$ per token i per head. Intuitively, this criterion favors indices that simultaneously carry significant mass in both matrices, which are the terms that dominate the output. In this work, we select top indices with the highest scores given by equation 9 to ensure that the most influential tokens are preserved. By grounding our eviction indicator in this matrix approximation principle, we obtain a token importance measure that directly **aligns with the structure of the attention computation** and more faithfully preserves the layer output compared to heuristics based solely on attention weights. The detailed algorithm is shown in Algorithm 1.

4.2 Eviction Action

The additive structure in equation 7 implies that, although we estimate token impact independently within each head, the resulting contribution of a token is not confined to that head alone; rather, it directly participates in the formation of the shared layer output after the output projection. Consequently, token importance is fundamentally a layer-level concept: tokens with large estimated contributions impact the layer output equivalently, regardless of which head produces them.

In contrast, head-wise selection relies on head local rankings, which can be misleading. For example, a token may rank highly in its head, but its projected contribution after VW_O can be **numerically insignificant compared to tokens from other heads and provides minimal difference in the aggregation step.** Retaining such "local winners" may waste memory on signals that have little impact on the final layer output. Conversely, some tokens may not be top-ranked in their own heads, yet contribute highly across multiple heads. These tokens are naturally captured by a layer-level criterion but missed by head-wise selection.

Remark 4.2. For a multi-head attention layer l, the contribution of token j at query position i is

$$\Delta \mathbf{o}^l(i,j) = \sum_{h'=1}^{h} A^{l,h'}(i,j) \, V^{l,h'}(j) \, W_O^{l,h'} \tag{10}$$

and the layer output satisfies

$$\mathbf{o}^l(i) = \sum_j \Delta \mathbf{o}^l(i,j). \tag{11}$$

Therefore, the contribution of a token from any head enters the layer output directly through linear addition. Token importance must therefore be evaluated at the layer level.

With the presence of $\boldsymbol{VW_O}$, remark 4.2 motivates formulating KV cache eviction as a unified selection problem across an entire layer, rather than a series of independent head-wise decisions. Unlike existing weight-based methods [11] which rely on eviction scores that are strictly local to each head, our metric (Section 4.1) integrates W_O that maps head contributions onto a unified scale, making token importance comparable across the entire layer, and providing a theoretically grounded basis for global selection. This allows us to safely flatten eviction scores and perform a joint selection under a fixed layer budget, and naturally enables an adaptive allocation of cache capacity: heads containing influential tokens retain more entries, while those with lower-impact tokens are pruned more aggressively. In Table 6, we also prove that a greedy selection without our global score can even hurt the performance.

Similar to how a layer output is formed by accumulating contributions from tokens across all heads, **the final model output is also an accumulation across layers.** As shown in Equation 5, a Transformer consists of stacked blocks connected through a shared residual path, where each layer's output is added to this path. As a result, the final output reflects the aggregated contributions of all layers. Therefore, if a token has a strong impact on its layer's output, this impact is directly propagated through the residual path and influences the final output. In this sense, a token's importance can be understood by how much it affects/changes this accumulated representation.

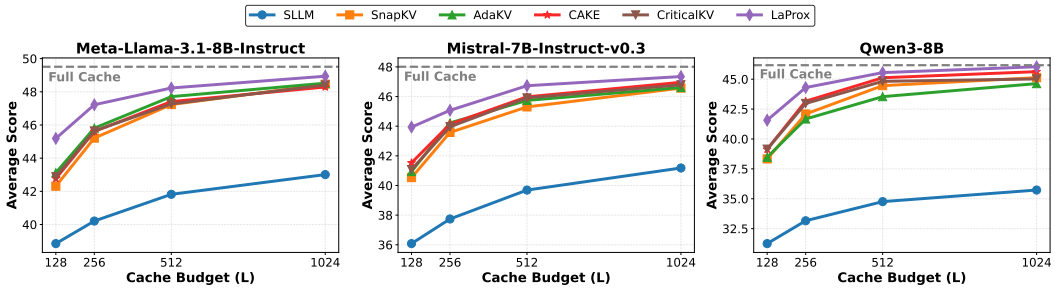


Figure 2: Average scores among 16 datasets of LongBench under different cache budgets.

However, while our VW_O-based metric accurately captures a token's contribution to the layer output, these raw scores are not directly comparable across different layers due to the inherent magnitude variations shown in Figure 1b. To resolve this scale disparity, we employ a simple layer-wise score normalization scheme that transforms flattened scores into a relative importance distribution:

$$s_{l,j} = \frac{p_{l,j}}{\sum_k p_{l,k}}, \tag{12}$$

where $p_{l,j}$ denotes the flattened eviction score of token j in layer l. This normalization neutralizes inter-layer scale differences for inter-layer comparison while strictly preserving the relative token rankings within each layer. The effectiveness of this simple normalization and its necessity is further demonstrated empirically in our evaluation and ablation study in Section 5 and Appendix D.1.

Finally, tokens across the model are jointly selected using $s_{l,j}$ (Algorithm 2). **This unified rule allows the model to focus on the most critical tokens model-wide.** Notably, this approach is *method-specific hyperparameter-free* and requires no calibration or training/finetuning, offering a straightforward implementation without complexity.

Algorithm 2 Eviction Action

Input: Eviction scores $\{p_{l,h,j}\}$, global budget K
Output: Selected token set $\mathcal{S}$

// Flatten head-wise scores
for each layer l **do**
  for each head h **do**
    for each token j **do**
      $p_{l,k} \leftarrow p_{l,h,j}$
    end for
  end for
end for

// Layer-wise normalization
for each layer l **do**
  for each token j **do**
    $s_{l,j} \leftarrow p_{l,j} / \sum_k p_{l,k}$
  end for
end for

// Global selection
$\mathcal{S} \leftarrow \mathrm{TopK}(\{s_{l,j}\}_{l,j}, K)$

5 Experiments

Models. We conduct experiments using three widely-used open-sourced LLMs: Llama-3.1-8B-Instruct [16], Mistral-7B-Instruct-v0.3 [23], and Qwen3-8B [46]. These models provide maximum context lengths of 128K, 32K, and 32K tokens, respectively. Additionally, we also perform experiments on larger models in Appendix G.3 to illustrate our approach's scalability.

Baselines. Our method is compared against the full-cache configuration (FullKV) and four SOTA baselines: StreamingLLM (SLLM) [45], SnapKV [30], AdaKV [11], CriticalKV [12], and CAKE [34]. A summary of these baselines is provided in Table 2.

Evaluation Scenarios. Our approach is compared across various memory budgets (average capacity per head). We use fixed absolute cache sizes rather than ratios relative to the full KV cache to prevent cache size increase with context length, better reflecting real-world hardware constraints. For implementation details, see Appendix A.

Evaluation Benchmarks. Following the standard practice of the prior works, our main experiments include two long-context benchmarks: LongBench [3] and RULER [21]. Additionally, we also evaluate on InfiniteBench [51], a very-long-context benchmark in G.1.

Table 1: Comparison across 16 LongBench datasets, with **best** and second best results highlighted.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Meta-Llama-3.1-8B-Instruct, $B_{\text{total}} = 128L$																	
FullKV	30.28	45.46	54.94	56.02	46.5	31.28	35.11	25.28	27.19	72.5	91.65	43.86	8.88	99.5	65.13	58.7	49.51
SLLM	22.45	20.83	31.74	44.86	39.31	23.88	18.75	20.78	18.57	40.5	85.57	38.24	8.16	**99.5**	58.89	49.7	38.86
SnapKV	25.53	23.92	48.15	52.94	43.5	26.67	19.48	22.47	19.68	47	89.56	39.51	8.41	98	59.43	52.84	42.31
AdaKV	25.74	24.8	49.38	52.73	43.75	27.8	20.37	23.02	20.38	49	**90.19**	40.12	8.38	99	60.92	54.4	43.12
CAKE	**25.93**	25.5	48.33	51.95	43.42	28.94	20.91	22.93	19.96	47.5	88.84	**41.26**	8.38	99	59.56	51.97	42.77
CriticalKV	24.56	25.89	49.86	52.42	43.89	27.96	20.71	22.69	20.06	48.5	89.61	39.94	8.38	99	60.08	52.56	42.88
LaProx	25.78	**31.38**	**52.97**	**55.17**	**44.95**	**29.46**	**21.66**	**23.66**	**21.47**	**59**	89.94	41.09	**8.89**	**99.5**	**61.91**	**56.16**	**45.19**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 128L$																	
FullKV	29.07	41.58	52.88	49.37	39.01	28.58	34.81	25.66	27.82	76	88.59	47.4	5.5	98	61.4	62.53	48.01
SLLM	21.42	22.28	26.82	37.28	33.31	17.61	16.73	19.77	17.86	45.5	85.64	40.47	5.5	80	55.13	52.07	36.08
SnapKV	23.81	25.05	46.92	44.98	35.06	23.11	20.49	21.55	19.32	43.5	**89.78**	42.87	6	93.5	55.95	54.6	40.41
AdaKV	24.08	25.52	47.08	47.16	35.16	23.83	20.23	22.28	20.7	45	88.92	**43.77**	6.5	92.5	56.63	55.81	40.95
CAKE	25.04	27.32	48.49	47.48	35.24	24.63	**21.32**	22.54	20.78	45.5	89.41	43.37	**7**	95	56.47	54.99	41.54
CriticalKV	25.42	25.95	46.43	46.44	35.92	23.93	21.12	22.35	19.26	47	89.23	43.62	6.5	93.5	55.99	54.51	41.07
LaProx	**28**	**30.26**	**54.02**	**47.86**	**37.54**	**25.54**	21.26	**23.31**	**22.1**	**63.5**	89.7	43.46	5	**96**	**58.85**	**57.66**	**44.00**
Qwen3-8B, $B_{\text{total}} = 128L$																	
FullKV	32.26	46.59	52.73	59.35	51.21	33.2	32.44	24.13	25.68	71	65.82	40	1	100	53.06	50.17	46.17
SLLM	12.18	28.05	21.79	36.34	39.07	6.88	15.78	18.54	15.95	43	62.56	33.3	3	73	45.16	45.43	31.25
SnapKV	16.34	31.35	42.37	52.67	42.33	18.26	16.03	19.68	16.49	53	75.69	34.44	1	**99**	46.86	47.86	38.33
AdaKV	**21.66**	32.58	43.96	50.81	42.95	20.9	16.45	19.33	16.02	51	73.92	34.24	1	98	45.77	46.92	38.47
CAKE	19.69	32.67	45.07	57.5	**48.02**	20.5	17.82	20.87	16.73	47	69.51	35.39	**5**	**99**	45.55	45.94	39.14
CriticalKV	18.47	35.83	45.78	48.59	42.84	19.09	16.67	19.94	**17.61**	56	75.31	35.3	0	**99**	47.79	48.28	39.15
LaProx	17.51	**36.91**	**51.22**	**58.64**	46.31	**23.85**	**18.68**	**21.21**	17.32	**61**	**76.49**	**35.86**	1	**99**	**48.86**	**51.21**	**41.57**

5.1 Evaluations on LongBench Dataset

We evaluate LaProx against SOTA KV cache eviction techniques across the 16 datasets in the LongBench benchmark, using cache budgets ranging from 128 to 1024 tokens. Table 1 details the performance across three models at a budget of 128 tokens, while Figure 2 illustrates the average performance across the full range of budget constraints.

As shown in Table 1, LaProx consistently outperforms previous works in nearly every LongBench's dataset, leading to significant improvements in total performance. Figure 2 further demonstrates our superior results across all budget sizes and models. Notably, the performance gap between LaProx and the baselines widens as the memory budget becomes more constrained, highlighting the robustness of LaProx under extreme hardware limitations.

We can also see that SLLM consistently exhibits the weakest performance, which is expected given its aggressive removal of intermediate tokens. By contrast, other approaches improve the performance by actively selecting important tokens. However, these baselines introduce only heuristic refinements to SnapKV, lacking a principled foundation or consideration of the underlying model structure, thereby limiting their gains over the vanilla SnapKV. Nevertheless, they remain strong baselines due to their improvements across many settings. In some cases, however, **such heuristic strategies can even degrade performance,** as observed for Qwen3 at budgets of 256-1024 by AdaKV and CriticalKV. Meanwhile, LaProx achieves superior performance across the majority of datasets and memory configurations. These results underscore the advantages of our unified global eviction strategy.

5.2 Evaluations on Needle-in-A-Haystack

To evaluate retrieval performance, we employ Needle-in-A-Haystack (NIAH) test, where a target sentence is embedded within a long-context distractor. Following RULER [21], we examine three representative configurations: (1) Single-Needle with 1 needle and 1 target (1N-1T): There is a single needle that the model must retrieve from the context; (2) Multi-Needle with 4 needles and 1 target (4N-1T): 4 needles (1 target and 3 distractors) are inserted to the context and the model must isolate a specific target from three distracting needles; and (3) Multi-Needle with 4 needles and 4 targets (4N-4T): 4 needles are inserted and the model must retrieve all of them. To match the Mistral and Qwen models' context windows and to balance the evaluation cost, each configuration is evaluated at a 32K context length across 100 samples per task, using cache budgets ranging from 256 to 2048 tokens. Further evaluations on RULER tasks are provided in Appendix G.2 and G.3.

Consistent with LongBench results, LaProx demonstrates significant performance gains on NIAH tests across all evaluated models and cache budgets. Notably, on Mistral-7B-Instruct-v0.3 with a highly constrained budget of 256 tokens, LaProx achieves a 1.5 to 3$\times$ improvement in retrieval accuracy over CriticalKV and other prior methods, respectively, across all three NIAH variants.

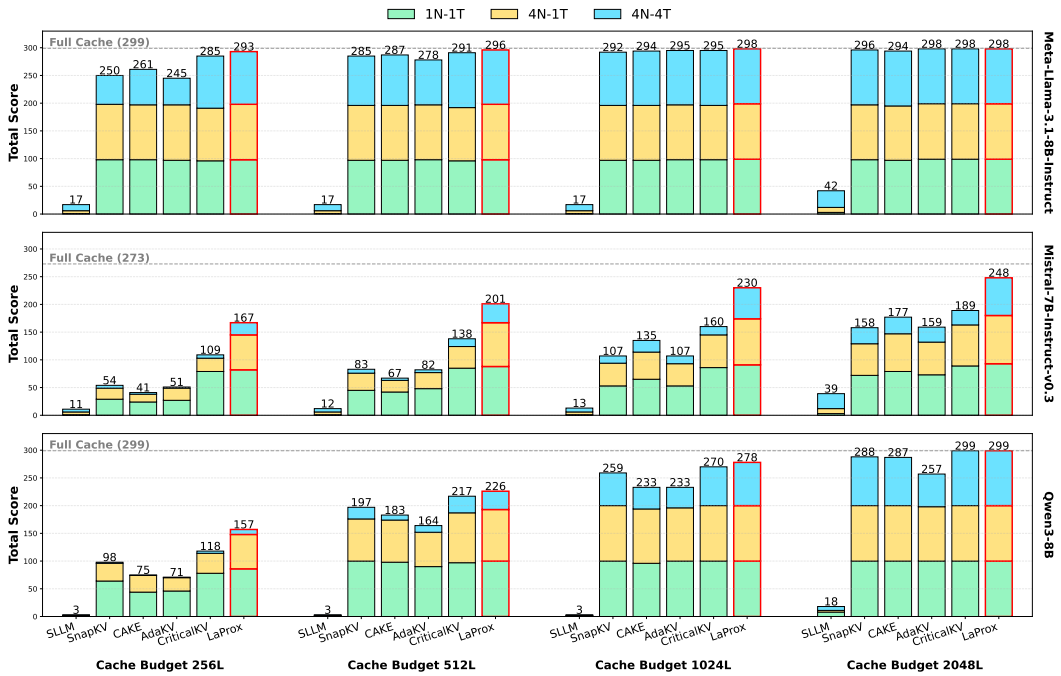


Figure 3: Comparison across 3 NIAH variants at 32K context length.

Meanwhile, although the baselines perform more competitively on Llama-3.1-8B-Instruct and Qwen3-8B, LaProx consistently maintains the highest scores and reaches FullKV performance sooner than other approaches. The performance gap becomes particularly pronounced in resource-constrained settings (256 tokens) or complex tasks (4 needles).

Furthermore, similar to the LongBench observations, **heuristic refinement methods continue to exhibit unstable behavior** and, in many cases, even degrade performance relative to the vanilla SnapKV baseline (such as CAKE and AdaKV).

5.3 Analysis of Matrix-Approximation-based Eviction Criterion

Beyond benchmark accuracy, we further investigate whether our matrix-based eviction criterion improves the similarity between the full and approximated attention outputs. Specifically, for each layer, we measure the cosine similarity between the full and compressed attention outputs for the first decoding token in Mistral-7B-Instruct-v0.3. The evaluation is conducted on the TREC dataset, with the KV cache of 128 tokens. As shown in Figure 4, our method consistently achieves higher cosine similarity across all layers compared to the vanilla approach based solely on local attention weights, indicating a more faithful approximation of the attention output.

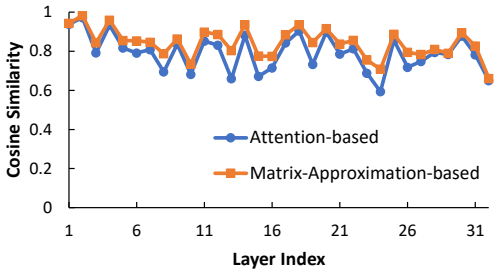


Figure 4: Similarity score between the full and approximated attention layer outputs

5.4 Evaluations on Efficiency

To evaluate efficiency, we report both prefill latency, which includes initial prompt processing and eviction overhead, per-token decoding latency, and peak memory usage. All measurements are conducted on a single NVIDIA H100 (80GB) GPU using the Meta-Llama-3.1-8B-Instruct model

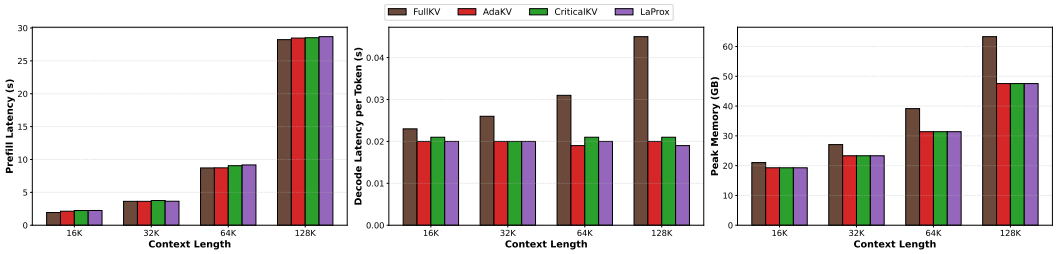


Figure 5: Efficiency Analysis.

with a fixed cache budget of 128 tokens. In this section, we compare our approach with AdaKV and CriticalKV, the strongest baselines supporting budget-allocation-based and output-aware eviction, respectively. Further efficiency analyses, including multibatch settings and latency breakdowns, can also be found in Appendix D.3

As shown in Figure 5, our method introduces only marginal overhead during prefilling across all context lengths. During decoding, all eviction methods achieve comparable efficiency and consistently outperform the FullKV setting. In contrast to FullKV, whose decoding latency grows rapidly with sequence length, our approach maintains a stable per-token latency by enforcing a strict KV cache budget. Consequently, at 128K context length, our method achieves a 2.3$\times$ decoding speedup over FullKV.

In addition, all eviction methods substantially reduce peak memory usage compared to FullKV. For instance, at 128K context length, eviction-based approaches require only around 47.5GB of memory, whereas FullKV consumes 63.3GB.

5.5 Analysis on the Number of Retained Tokens

Implicitly, our model-wide eviction strategy (Algorithm 2) retains different numbers of tokens across layers and heads according to their estimated importance. To analyze this behavior, we conduct experiments on the NIAH dataset using two representative models, Meta-Llama-3.1-8B-Instruct (Llama) and Mistral-7B-Instruct-v0.3 (Mistral), and record the number of retained tokens for each head. The total cache size is fixed to 128 tokens, corresponding to an average of 96 retained entries per head (with an additional 32-entry history window).

As shown in Figure 6, the number of retained tokens varies substantially across layers and heads. For example, in Llama, layers 13-14 retain significantly more tokens than layers 3 and 24. Similarly, within layer 27, a single head retains the majority of tokens.

We also observe distinct patterns across models. Llama concentrates retained tokens in the middle layers, whereas Mistral progressively shifts importance toward later layers. Moreover, these patterns are not stable and can vary by up to 500 tokens per head across inputs.

These results show that token contributions are highly non-uniform across layers and heads, vary across models, and are strongly input-dependent. **This underscores the need of our model-wide eviction strategy,** which dynamically selects tokens without offline calibration, and highlights a key limitation of calibration-based prior methods [34, 13, 5], which assume stable optimal budgets across diverse inputs.

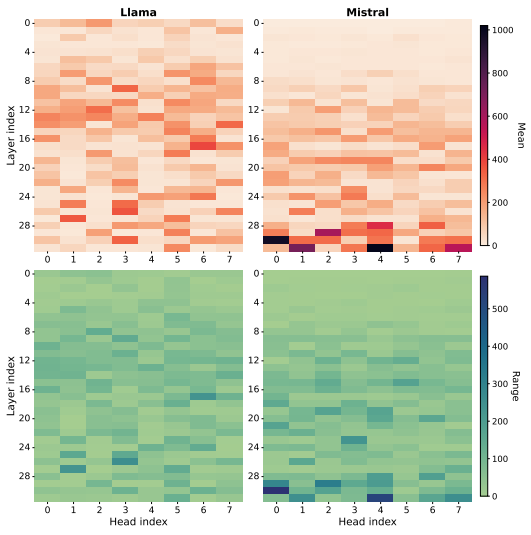


Figure 6: Retained tokens and variation per head

6 Conclusion

This paper revisits KV cache eviction for long-context LLM inference and addresses two fundamental oversights in current methods. First, we identify that current heuristic approaches neglect the structure of the attention layers; by considering the attention layer formulation, we provide a more complete measure of token importance. Second, we demonstrate that traditional head-level eviction is inherently suboptimal, as token importance is more accurately captured at the model-level. Based on these insights, we introduce LaProx that reformulates cache eviction as a global matrix approximation problem rather than a head-wise weight-averaging task. **Our work is the first to explore a unified model-level cache eviction strategy, exposing the weakness of the long-standing traditional head-based heuristic schemes.** Extensive evaluations across 19 datasets from LongBench and NIAH benchmarks confirm that this approach consistently outperforms prior strategies. Furthermore, our analysis validates that our method achieves higher fidelity to full-cache attention outputs than standard heuristics. Ultimately, this work introduces a different, global point of view to the study of KV cache, offering a principled foundation for efficient long-context inference.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (IITP-2026-RS-2020-II201847, Information Technology Research Center; IITP-2026-RS-2023-00256472, Graduate School of Artificial Intelligence Semiconductor).

References

[1] Menachem Adelman et al. "Faster neural network training with approximate tensor operations". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27877–27889.

[2] Joshua Ainslie et al. "Gqa: Training generalized multi-query transformer models from multi-head checkpoints". In: *arXiv preprint arXiv:2305.13245* (2023).

[3] Yushi Bai et al. "Longbench: A bilingual, multitask benchmark for long context understanding". In: *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*. 2024, pp. 3119–3137.

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. "Longformer: The long-document transformer". In: *arXiv preprint arXiv:2004.05150* (2020).

[5] Zefan Cai et al. "Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling". In: *arXiv preprint arXiv:2406.02069* (2024).

[6] Tri Dao. "Flashattention-2: Faster attention with better parallelism and work partitioning". In: *arXiv preprint arXiv:2307.08691* (2023).

[7] Pradeep Dasigi et al. "A dataset of information-seeking questions and answers anchored in research papers". In: *arXiv preprint arXiv:2105.03011* (2021).

[8] Alessio Devoto et al. "A Simple and Effective L_2 Norm-Based Strategy for KV Cache Compression". In: *arXiv preprint arXiv:2406.11430* (2024).

[9] Petros Drineas, Ravi Kannan, and Michael W Mahoney. "Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication". In: *SIAM Journal on Computing* 36.1 (2006), pp. 132–157.

[10] Alexander Richard Fabbri et al. "Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model". In: *Proceedings of the 57th annual meeting of the association for computational linguistics*. 2019, pp. 1074–1084.

[11] Yuan Feng et al. "Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference". In: *arXiv preprint arXiv:2407.11550* (2024).

[12] Yuan Feng et al. "Identify critical kv cache in llm inference from an output perturbation perspective". In: *arXiv preprint arXiv:2502.03805* (2025).

[13] Yu Fu et al. "Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning". In: *arXiv preprint arXiv:2410.19258* (2024).

[14] Bogdan Gliwa et al. "SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization". In: *arXiv preprint arXiv:1911.12237* (2019).

[15] Raghavv Goel et al. “CAOTE: KV Cache Selection for LLMs via Attention Output Error-Based Token Eviction”. In: *arXiv preprint arXiv:2504.14051* (2025).

[16] Aaron Grattafiori et al. “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (2024).

[17] Yifeng Gu et al. “AhaKV: Adaptive Holistic Attention-Driven KV Cache Eviction for Efficient Inference of Large Language Models”. In: *arXiv preprint arXiv:2506.03762* (2025).

[18] Daya Guo et al. “Longcoder: A long-range pre-trained language model for code completion”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 12098–12107.

[19] Xanh Ho et al. “Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps”. In: *arXiv preprint arXiv:2011.01060* (2020).

[20] Coleman Hooper et al. “Kvquant: Towards 10 million context length llm inference with kv cache quantization”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 1270–1303.

[21] Cheng-Ping Hsieh et al. “RULER: What’s the Real Context Size of Your Long-Context Language Models?” In: *arXiv preprint arXiv:2404.06654* (2024).

[22] Luyang Huang et al. “Efficient attentions for long document summarization”. In: *arXiv preprint arXiv:2104.02112* (2021).

[23] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

[24] Mandar Joshi et al. “Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension”. In: *arXiv preprint arXiv:1705.03551* (2017).

[25] Ehsan Kamalloo et al. “Evaluating open-domain question answering in the era of large language models”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2023, pp. 5591–5606.

[26] Gregory Kamradt. *Needle In A Haystack - pressure testing LLMs*. 2023. URL: https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

[27] Tomáš Kočiskỳ et al. “The narrativeqa reading comprehension challenge”. In: *Transactions of the Association for Computational Linguistics* 6 (2018), pp. 317–328.

[28] Wonbeom Lee et al. “{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)*. 2024, pp. 155–172.

[29] Xin Li and Dan Roth. “Learning question classifiers”. In: *COLING 2002: The 19th International Conference on Computational Linguistics*. 2002.

[30] Yuhong Li et al. “Snapkv: Llm knows what you are looking for before generation”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 22947–22970.

[31] Tianyang Liu, Canwen Xu, and Julian McAuley. “Repobench: Benchmarking repository-level code auto-completion systems”. In: *arXiv preprint arXiv:2306.03091* (2023).

[32] Zichang Liu et al. “Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 52342–52364.

[33] Zirui Liu et al. “Kivi: A tuning-free asymmetric 2bit quantization for kv cache”. In: *arXiv preprint arXiv:2402.02750* (2024).

[34] Ziran Qin et al. “Cake: Cascading and adaptive kv cache eviction with layer preferences”. In: *arXiv preprint arXiv:2503.12491* (2025).

[35] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *Journal of machine learning research* 21.140 (2020), pp. 1–67.

[36] Yiqun Shen et al. *LAVa: Layer-wise KV Cache Eviction with Dynamic Budget Allocation*. 2025. arXiv: 2509.09754 [cs.LG]. URL: https://arxiv.org/abs/2509.09754.

[37] Ying Sheng et al. “Flexgen: High-throughput generative inference of large language models with a single gpu”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 31094–31116.

[38] Hanlin Tang et al. “Razorattention: Efficient kv cache compression through retrieval heads”. In: *arXiv preprint arXiv:2407.15891* (2024).

[39] Jiaming Tang et al. “Quest: Query-aware sparsity for efficient long-context llm inference”. In: *arXiv preprint arXiv:2406.10774* (2024).

[40] Vicuna Team. "Vicuna: An open-source chatbot impressing gpt-4 with 90% chatgpt quality". In: *Vicuna: An open-source chatbot impressing gpt-4 with* 90 (2023).

[41] Harsh Trivedi et al. "MuSiQue: Multihop Questions via Single-hop Question Composition". In: *Transactions of the Association for Computational Linguistics* 10 (2022), pp. 539–554.

[42] Zhongwei Wan et al. "D2o: Dynamic discriminative operations for efficient long-context inference of large language models". In: *arXiv preprint arXiv:2406.13035* (2024).

[43] Hanrui Wang, Zhekai Zhang, and Song Han. "Spatten: Efficient sparse attention architecture with cascade token and head pruning". In: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE. 2021, pp. 97–110.

[44] Guangxuan Xiao et al. "Duoattention: Efficient long-context llm inference with retrieval and streaming heads". In: *arXiv preprint arXiv:2410.10819* (2024).

[45] Guangxuan Xiao et al. "Efficient streaming language models with attention sinks, 2024". In: *URL https://arxiv. org/abs/2309.17453* 1 (2024).

[46] An Yang et al. "Qwen3 technical report". In: *arXiv preprint arXiv:2505.09388* (2025).

[47] Dongjie Yang et al. "Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference". In: *arXiv preprint arXiv:2405.12532* (2024).

[48] Zhilin Yang et al. *HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering*. 2018. arXiv: 1809.09600 [cs.CL]. URL: https://arxiv.org/abs/1809.09600.

[49] Hailin Zhang et al. "Pqcache: Product quantization-based kvcache for long context llm inference". In: *Proceedings of the ACM on Management of Data* 3.3 (2025), pp. 1–30.

[50] Tianyi Zhang et al. "Benchmarking large language models for news summarization". In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 39–57.

[51] Xinrong Zhang et al. ∞*Bench: Extending Long Context Evaluation Beyond 100K Tokens*. 2024. arXiv: 2402.13718 [cs.CL]. URL: https://arxiv.org/abs/2402.13718.

[52] Zhenyu Zhang et al. "H2o: Heavy-hitter oracle for efficient generative inference of large language models". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 34661–34710.

[53] Youpeng Zhao, Di Wu, and Jun Wang. "Alisa: Accelerating large language model inference via sparsity-aware kv caching". In: *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE. 2024, pp. 1005–1017.

[54] Ming Zhong et al. "QMSum: A new benchmark for query-based multi-domain meeting summarization". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 5905–5921.

A Implementation details

Methods with uniform allocation assign equal cache capacity to each attention head, whereas non-uniform strategies, including ours, adaptively distribute cache capacity across heads or layers while keeping the total memory budget fixed. All baseline hyperparameters follow their default official implementations. For SLLM [45], we retain four sink tokens and allocate the remaining budget to a sliding recent window. All other methods employ an observation window of 32 tokens to limit overhead and apply average pooling with a kernel size of 7 to mitigate information fragmentation. The summary of the baselines is shown in Table 2.

To ensure compatibility with Grouped Query Attention (GQA) [2], we follow standard practice in prior works by using the mean attention weight within each query group as the selection criterion. All experiments are accelerated using FlashAttention-2 [6]. And consistent with prior works [30, 11], cache eviction is applied in the prefilling phase per layer.

Table 2: Summary of evaluated methods on eviction score.

METHODS	EVICT. SCORE	HEAD ALLOC.	LAYER ALLOC.	HYPER PARAM.
FullKV	-	-	-	-
SLLM	N/A (Fixed-indices eviction)	-	-	1 (sink tokens)
SnapKV	Mean($\boldsymbol{A}[:,i]$)	-	-	0
CAKE	Mean($\boldsymbol{A}[:,i]$) + γ Var($\boldsymbol{A}[:,i]$)	-	Entropy$(\boldsymbol{A})^{1/\tau_1} \cdot$ Var$(\boldsymbol{A})^{1/\tau_2}$	3 (γ, τ_1, τ_2)
AdaKV	Mean($\boldsymbol{A}[:,i]$)	Top-K(raw head scores)	-	1 (safeguard)
CriticalKV	(Mean($\boldsymbol{A}[:,i]$) + ϵ) $\|\boldsymbol{VWo}[i,:]\|$	-	-	2 (safeguard, ϵ)
LaProx (ours)	$\|\boldsymbol{A}[:,i]\|_2 \, \|\boldsymbol{VWo}[i,:]\|_2$	Top-K(normalized model scores)	Top-K(normalized model scores)	0

B Additional Related Works on KV Cache Management

KV Cache Eviction Metrics. Beyond standard attention weights, several works explore alternative definitions of token importance. The work in [8] utilizes the $L2$ norm of key states to reduce computational overhead at the expense of accuracy. Other approaches, such as [17], enhance attention-based metrics by incorporating value representations. However, these methods rely solely on value states, which only partially capture a token's contribution to the final representation. In practice, the output is further modulated by the output projection matrix (W_O)—a factor our ablation study (Section D.1) identifies as a critical.

CriticalKV [12] is the most closely related to our work, as it also incorporates both value states and output projections. However, a fundamental conceptual gap remains: CriticalKV acts as a helper for other works and it treats output information merely as a scaling factor applied to the average-based eviction score. In contrast, **we reformulate KV cache eviction as a principled approximation of the attention matrix product** and **provide a complete solution**. By explicitly modeling the interaction between attention weights and projected value states as a unified operation, our method provides a more accurate estimation of token influence, as validated in Section 5.

Furthermore, because CriticalKV lacks this rigorous matrix-product formulation, **it must rely on 2 hyper-parameters to maintain model performance** (an empirical safeguard and an ϵ to mitigate information loss). Its scoring also remains strictly head-centric, failing to address eviction as a layer-level optimization problem. Our approach moves beyond simple "scaling helpers" by providing a unified, theoretical-bound layer-wise solution that naturally handles budget allocation across heads without requiring empirical tuning.

KV Cache Selection. While KV cache eviction methods reduce memory usage by retaining only a small subset of critical key–value entries, sparse attention approaches such as Quest [39] and InfiniGen [28] preserve the entire KV cache during inference but restrict computation to a selected subset of entries at each attention step. Or, some works [44, 38] combine the Dynamic Selection with Eviction approach by classifying attention heads into Streaming Heads with a limited Cache size and Retrieval Heads with a full Cache size. By limiting attention computation rather than storage, sparse attention methods can significantly accelerate inference and often achieve high output quality. However, because all KV entries are still stored, these methods **do not reduce the memory footprint of the KV cache**.

Table 3: Details of each dataset in LongBench.

Task	Dataset	Source	Avg len	Eval Metric	Language	#data
Single-Doc	NarrativeQA	Literature, Film	18,409	F1	English	200
	Qasper	Science	3,619	F1	English	200
	MultiFieldQA-en	Multi-field	6,701	F1	English	150
Multi-Doc	HotpotQA	Wikipedia	9,151	F1	English	200
	2WikiMultihopQA	Wikipedia	4,887	F1	English	200
	MuSiQue	Wikipedia	11,214	F1	English	200
Summarization	GovReport	Government report	8,734	Rouge-L	English	200
	QMSum	Meeting	10,614	Rouge-L	English	200
	MultiNews	News	2,113	Rouge-L	English	200
Few-shot	TREC	Web question	5,177	Accuracy	English	200
	TriviaQA	Wikipedia, Web	8,209	F1	English	200
	SAMSum	Dialogue	6,258	Rouge-L	English	200
Synthetic	PassageCount	Wikipedia	11,141	Accuracy	English	200
	PassageRetrieval-en	Wikipedia	9,289	Accuracy	English	200
Code	LCC	Github	1,235	Edit Sim	Python/C#/Java	500
	RepoBench-P	Github	4,206	Edit Sim	Python/Java	500

KV Cache Quantization. KV cache quantization methods reduce memory and computation costs by representing cached values using lower-precision formats. These approaches can be broadly divided into fixed-precision quantization, where all tokens share the same bit-width [49, 37], and mixed-precision schemes, which allocate different bit-widths to different tokens [20, 33]. However, while quantization effectively lowers per-token storage cost, the total **KV cache size continues to grow linearly with context length**, limiting its ability to address memory bottlenecks in very long-context settings.

C Details of Evaluation Benchmarks

LongBench Benchmark. LongBench [3] is a widely used long-context benchmark, serving as a standardized evaluation protocol commonly adopted by prior works and baselines [30, 34, 11]. It consists of 16 datasets across 6 task domains: Single-Doc QA [27, 7], Multi-Doc QA [48, 19, 41], Summarization [22, 54, 10], Few-shot Learning [29, 24, 14], Synthetic Task [35], and Code Completion [18, 31]. The average token length across all 16 datasets is 6,711. More details can be found in Table 3.

Ruler Benchmark. Ruler [21] is a diagnostic benchmark designed to evaluate long-context capabilities beyond simple retrieval. It consists of 13 datasets across 4 task domains:

- Retrieval: An extension of NIAH [26] that tests retrieval robustness using diverse needle types and varying quantities of hidden information.
- Multi-hop Tracing: Evaluates the model's ability to track variable assignments and identify co-occurrence patterns that require connecting multiple pieces of information across the sequence.
- Aggregation: Tests the ability to identify the most frequent or common words distributed throughout the text.
- Question Answering: Tests the capability to answer questions where the answer is deeply embedded within extensive distracting or irrelevant content.

Further evaluations on more tasks from Ruler are provided in Appendix G.2 and G.3.

D Ablation Study

D.1 Eviction Criteria Analysis

In this section, we present a series of ablation studies to evaluate the effectiveness of our proposed eviction strategy. We use Mistral-7B-Instruct-v0.3 with cache budget $B_{total} = 128L$ on LongBench as the default settings.

Table 4: Comparison with different Eviction Indicators.

EVICTION INDICATOR		
VALUE STATE	OUTPUT WEIGHT	AVG.
		40.53
✓		41.69
✓	✓	41.84

Table 5: Comparison with different Allocation Strategies.

ALLOCATION STRATEGY			
HEAD-FLATTEN	LAYER-FLATTEN	NORMALIZE	AVG.
			41.84
✓			42.96
✓	✓		41.14
✓	✓	✓	44.00

Effectiveness of Eviction Indicator

To validate the necessity of our metric, we isolate the influence of V from VW_O then compare them against the baseline. As shown in Table 4, incorporating V improves performance over attention-only baselines, but remains suboptimal as it still provides an incomplete computation of the attention output. The highest accuracy is achieved by our full AVW_O indicator, demonstrating that the output projection is essential for quantifying a token's contribution to the layer's output.

Effectiveness of Allocation Strategy

We further analyze the necessity of each component in our allocation method, specifically flattening (head-wise and layer-wise) and normalization (Table 5). While head-wise allocation provides immediate performance gains, layer-wise allocation degrades performance to below the baseline when the scores are used in their raw form. This decline stems from raw score scale differences across layers; without normalization, the budget focuses only on high-magnitude layers while starving others. Despite its simplicity, our normalization scheme resolves this bias, allowing layer-wise allocation to effectively complement head-wise settings and further improve performance.

While the allocation gain shown in Table 5 is more noticeable, it is worth noting that our method still outperforms prior approaches without it. Furthermore, **projection and allocation strategy are not independent; rather, the former is the mathematical prerequisite for the latter.** This is because without our projection, attention-based score is limited in its own head and not comparable globally.

To validate this, we experiment with 4 different settings: Uniform budget and Global allocation with Attention score (A_L and A_G), Uniform budget and Global allocation with LaProx score (L_L and L_G). The experiments were conducted using LongBench benchmark under the budget of 128 tokens. The results are shown in Table 6.

Table 6: Comparison across 16 LongBench datasets for the cache budget of 128L. The best result is highlighted in **bold**.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Meta-Llama-3.1-8B-Instruct, $B_{\text{total}} = 128L$																	
A_L	25.53	23.92	48.15	52.94	43.5	26.67	19.48	22.47	19.68	47	89.56	39.51	8.41	98	59.43	52.84	42.31
A_G	24.98	27.16	49.96	52.1	43.38	27.73	21.21	23.21	19.93	48.5	**89.95**	40.1	7.92	99	60.45	53.24	43.05
L_L	25.49	26.72	49.91	54.7	42.81	29.18	20.67	**23.67**	19.72	48	89.36	**41.23**	8.88	99	59.18	53.49	43.26
L_G	**25.78**	**31.38**	**52.97**	**55.17**	**44.95**	**29.46**	**21.66**	23.66	**21.47**	**59**	89.94	41.09	**8.89**	**99.5**	**61.91**	**56.16**	**45.19**

As seen from the results, **without VW_O, global allocation offers no improvement compared to the baseline.** Particularly, the performance of A_G is similar to A_L, and is lower than L_L, which is our method without any allocation.

Moving from sampling to deterministic

The Monte Carlo matrix multiplication framework derives the optimal importance measure for each row/column by showing that the sampling probability

$$p_i \propto \|A[:, i]\|_2 \, \|B[i, :]\|_2$$

minimizes the expected Frobenius-norm approximation error. Our work adopts this theoretically grounded importance measure but replaces stochastic sampling with deterministic Top-K selection.

The Monte Carlo framework minimizes the expected (i.e., average over repeated trials) approximation error over the sampling distribution. However, **LLM inference executes a single-shot**

forward pass. Consequently, stochastic sampling may discard high-contribution KV pairs due to sampling randomness, introducing unnecessary approximation error in that particular inference. Deterministic Top-K instead retains the highest-scoring KV pairs according to the same theoretically derived importance measure, **eliminating sampling variance** while preserving the intended ranking.

To validate this design choice, we additionally compare stochastic sampling ("Sample") and deterministic Top-K ("Deter.") using the identical norm-product score on the LongBench benchmark with the Mistral-7B-Instruct-v0.3 model under a 128-token budget.

Table 7: Comparison between deterministic Top-K selection and stochastic sampling using the same norm-product importance score on LongBench.

Method	NrtvQA	Qasper	MF	Hotpot	2WikiQA	MuSiQue	GovRep	QMSum	MultNews	TREC	TriviaQA	SAMSum	PCount	PR	LCC	RB-P	Avg.
Deter.	**28.00**	**30.26**	**54.02**	**47.86**	**37.54**	**25.54**	**21.26**	**23.31**	**22.10**	**63.50**	89.70	**43.46**	5.00	**96.00**	**58.85**	**57.66**	**44.00**
SnapKV	23.81	25.05	46.92	44.98	35.06	23.11	20.49	21.55	19.32	43.50	**89.78**	42.87	**6.00**	93.50	55.95	54.60	40.41
Sample	22.73	24.61	41.54	39.73	30.66	16.38	14.99	21.94	18.58	55.00	83.10	39.59	5.00	94.67	37.16	38.45	36.48

As shown in Table 7, **stochastic sampling significantly degrades performance**, even below the SnapKV baseline, despite using the same theoretically derived importance score. In contrast, deterministic Top-K (LaProx) consistently obtains the highest scores across the majority of datasets and achieves the best overall performance.

D.2 Sensitivity Analysis

Window size robustness. Following the configurations of the baselines [30, 34, 11], we utilized a default historical window size of 32 for our primary experiments. We note that the *hyperparameter-free* property of LaProx refers specifically to its eviction scoring and budget allocation mechanisms, which require no method-specific tunable coefficients. The historical window, by contrast, is a common structural component of KV-cache compression frameworks used to preserve local context and reduce attention-computation overhead.

To assess the sensitivity of our method to this hyperparameter, we conducted an ablation study on the Mistral-7B-Instruct-v0.3 model (with a cache budget of 128) across four window sizes: 8, 16, 32, and 64. The results are summarized in Table 8. LaProx maintains stable performance across window sizes 8 through 32, with scores ranging between 43.79 and 44.10. However, performance degrades at a window size of 64; this occurs because the large observation window reduces the candidate space for eviction, thereby limiting the flexibility of our eviction strategy. Overall, these results demonstrate that LaProx is highly robust to variations in the historical window size.

Table 8: Analysis on different observation window sizes.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = Full$																	
FullKV	29.07	41.58	52.88	49.37	39.01	28.58	34.81	25.66	27.82	76	88.59	47.4	5.5	98	61.4	62.53	48.01
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 128L$																	
w=8	27.87	30.51	53.85	48.2	35.95	25.98	22.58	23.2	22.24	72	88.66	43.68	5.5	94	56.99	54.4	44.10
w=16	25.75	31.43	52.3	47.46	35.57	25.55	22.93	22.98	22.26	69.5	88.44	44.17	5.5	94	57.43	55.34	43.79
w=32	28	30.26	54.02	47.86	37.54	25.54	21.26	23.31	22.1	63.5	89.7	43.46	5	96	58.85	57.66	44.00
w=64	23.24	24.99	45.7	48.43	35.28	24.29	19.13	21.2	19.9	52	88.74	43.28	3.5	92	55.53	55.08	40.77

D.3 Efficiency Analysis

Latency Breakdown

Since our method differs from the baselines primarily in the eviction score computation and allocation stages (while all other components, such as attention computation and linear projections, remain identical), we focus our analysis on these components.

To better understand the source of latency overhead, we focus on the attention layer and decompose its forward pass into four stages: attention computation (with a historical window), eviction score computation, allocation, and final output projection. We then profile each stage separately. All measurements are taken at the last layer at the end of the prefill stage, where KV cache eviction is performed. For other layers, eviction is not triggered and the remaining computations are identical. The results are summarized in the table below.

Table 9: Latency breakdown (s).

	Attn Comp	Evict. Comp	Allocation	Proj	Total
AdaKV	0.03898	0.00042	0.10230	0.62226	0.76396
LaProx	0.03889	0.04174	0.09684	0.6162	0.79368

Table 10: Prefilling latency (seconds).

Batch size	FullKV	AdaKV	CriticalKV	LaProx
1	8.586	9.854	10.471	10.619
2	16.769	19.803	20.263	20.728
3	OOM	28.398	29.187	30.025

Table 11: Decoding latency (seconds).

Batch size	FullKV	AdaKV	CriticalKV	LaProx
1	0.031	0.027	0.026	0.025
2	0.043	0.030	0.025	0.031
3	OOM	0.026	0.027	0.027

The results show that the latency of attention computation and output projection remains unchanged, as these components share the same implementation across methods. The allocation stage is slightly faster in LaProx, since it does not require additional safeguard logic (for example, threshold checks and budget adjustments to enforce constraints). In contrast, the eviction score computation in LaProx introduces higher latency compared to AdaKV. However, **when placed in the context of the full pipeline, this overhead is minimal,** as other components (e.g., linear projections) dominate the overall latency.

Scalability analysis

In Section 5.4, we have evaluated the efficiency of LaProx on context lengths up to 128K using Meta-Llama-3.1-8B-Instruct, which employs 32 attention heads with GQA. These experiments demonstrate that the proposed scoring mechanism maintains low overhead even under extremely long contexts and modern multi-head architectures.

To further strengthen this point, we additionally evaluate scalability with respect to batch size. The experiments are conducted on Meta-Llama-3.1-8B-Instruct with a 64K context length and a cache budget of 128 tokens, while comparing against the two strongest baselines, AdaKV and CriticalKV.

As shown in Tables 10 and 11, all KV eviction methods scale similarly with increasing batch size. Compared with AdaKV and CriticalKV, LaProx introduces only a marginal increase in prefilling latency while maintaining comparable decoding latency. More importantly, all eviction-based methods can support the configuration where the FullKV baseline runs out of memory (batch size $= 3$). These results indicate that the additional Frobenius-norm computations do not become the dominant runtime bottleneck, even for long contexts, multiple attention heads, or large batch sizes.

E Limitations

Our work is the first to introduce a unified eviction framework that assigns globally comparable importance scores to tokens, enabling model-wide joint selection of KV cache entries. To achieve practical efficiency, our current formulation relies on relatively simple approximation and normalization schemes, which we view as an initial foundation for future model-wide KV cache research. Future work may explore more sophisticated matrix multiplication approximation schemes or different normalization methods for global token comparison to further improve cache eviction performance.

F Proofs

F.1 Proof of Remark 4.1

Let:

- h be the number of attention heads.
- d_v be the dimension of each head.
- d_{model} be the model dimension.

- $H^i \in \mathbb{R}^{n \times d_v}$ be the output of the i-th head ($H^i = \text{Attention}(Q^i, K^i, V^i)$).
- $W_O \in \mathbb{R}^{(h \cdot d_v) \times d_{model}}$ be the output weight matrix.

Then, the standard MHA is defined as the concatenation of all head outputs followed by a linear projection:

$$\text{Output} = \text{Concat}(H^1, H^2, \ldots, H^h) W_O \tag{13}$$

We can represent the concatenation as a partitioned row matrix:

$$H = \begin{bmatrix} H^1 & H^2 & \ldots & H^h \end{bmatrix} \tag{14}$$

We can similarly partition the weight matrix W_O vertically into h blocks, where each $W_O^i \in \mathbb{R}^{d_v \times d_{model}}$:

$$W_O = \begin{bmatrix} W_O^1 \\ W_O^2 \\ \vdots \\ W_O^h \end{bmatrix} \tag{15}$$

By applying the rules of block matrix multiplication:

$$\begin{aligned} \text{Output} &= HW_O \\ &= \begin{bmatrix} H^1 & H^2 & \ldots & H^h \end{bmatrix} \begin{bmatrix} W_O^1 \\ W_O^2 \\ \vdots \\ W_O^h \end{bmatrix} \\ &= (H^1 W_O^1) + (H^2 W_O^2) + \cdots + (H^h W_O^h) \\ &= \sum_{i=1}^{h} H^i W_O^i \end{aligned} \tag{16}$$

□

F.2 Proof of Remark 4.2

Consider a transformer layer l with H attention heads. Let i denote the current query position and j a cached token position. The output of multi-head attention at layer l is

$$\mathbf{o}^l(i) = \sum_{h=1}^{H} \mathbf{o}^{l,h}(i)\, W_O^{l,h}, \tag{17}$$

where the output of head h is

$$\mathbf{o}^{l,h}(i) = \sum_j A^{l,h}(i,j)\, V^{l,h}(j). \tag{18}$$

Substituting and reordering terms yields

$$\mathbf{o}^l(i) = \sum_j \sum_{h=1}^{H} A^{l,h}(i,j)\, V^{l,h}(j)\, W_O^{l,h}. \tag{19}$$

Equation (19) admits a natural decomposition over cached tokens. In particular, the contribution of token j to the layer output can be written as

$$\Delta\mathbf{o}^l(i,j) = \sum_{h=1}^{H} A^{l,h}(i,j)\, V^{l,h}(j)\, W_O^{l,h}. \tag{20}$$

This expression shows that a token's effect on the layer output is not localized to any single head, but instead accumulates additively across all heads without any nonlinear operations. □

G Extended Experimental Results

G.1 Extended Evaluation on InfiniteBench Benchmark

To further assess the performance of our proposal under extreme long-context settings, we conducted additional evaluations on the InfiniteBench benchmark [51] and compared against 2 representative baselines, AdaKV and CriticalKV.

Since InfiniteBench involves extremely long contexts (including some datasets with an average input length of upto 190K+ tokens), we evaluate on Meta-Llama-3.1-8B-Instruct, which supports 128K context lengths. To balance evaluation cost, we set the cache budget to 1024 tokens and the dataset limit to 100 samples per dataset. The results are summarized in Table 12.

Table 12: Comparison of different methods on InfiniteBench. Best result is highlighted in **bold**.

Method	En.Sum	En.QA	En.MC	En.Dia	Code.Debug	Code.Run	Math.Find	Ret.Pass	Ret.Num	Ret.KV	Avg.
Meta-Llama-3.1-8B-Instruct, $B_{\text{total}} = 1024L$											
AdaKV	23.32	8.66	**71**	9	**22**	1	**53**	**100**	94	2	38.40
CriticalKV	**23.34**	9.31	70	8	**22**	**2**	**53**	**100**	93	0	38.07
LaProx	21.64	**9.83**	**71**	**10**	**22**	**2**	**53**	**100**	**96**	**4**	**38.95**

The results show that, on this highly challenging long-context benchmark, LaProx consistently achieves stronger performance than our strongest baseline across most datasets for both models. These results demonstrate the robustness of our method under extreme long-context settings, as it maintains strong performance across different models and evaluation tasks.

G.2 Evaluations on Question-Agnostic Setting

In our main experiments, we follow the standard practice of the prior works [30, 5, 34], where the question is compressed together with the context, to provide a unified evaluation scenario. In this section, we adopt a question-agnostic setting, where the context is compressed without any knowledge of the question, providing a more challenging evaluation. The results are summarized in Tables 13-14.

Evaluating RULER [21] is particularly computationally expensive, as it consists of 13 synthetic tasks and requires approximately 17 GPU hours for a single 32K-context evaluation per configuration. As a result, conducting a comprehensive sweep across all settings becomes prohibitively costly. Therefore, in this Appendix, we restrict comparisons to 2 representative baselines, AdaKV and CriticalKV.

Table 13: Comparison across 16 LongBench datasets. The best result is highlighted in **bold**.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Meta-Llama-3.1-8B-Instruct, 20% Cache																	
FullKV	29.21	44.6	55.73	58.14	49.26	32.61	34.65	25.24	26.9	73	92.45	43.42	7.62	99.5	63.58	52.73	49.29
AdaKV	27.02	29.01	33.12	49.19	29.45	21.24	26.86	21.84	22.83	54	91.33	43.84	6.12	80	66.59	55.38	41.11
CriticalKV	29.56	30.02	33.25	51.37	33.98	25.5	28.84	22.53	22.92	54	91.75	43.85	**7.61**	95.5	66.73	54.68	43.25
LaProx	**31.34**	**41.26**	**51.99**	**55.04**	**39.25**	**26.36**	**29.95**	**24.04**	**24.05**	**68**	**92.2**	**43.83**	6.91	**99.5**	**68.02**	**56.09**	**47.36**

Table 14: Comparison across 13 Ruler datasets. The best result is highlighted in **bold**.

Method	CWE	FWE	NIAH_MK1	NIAH_MK2	NIAH_MK3	NIAH_MQ	NIAH_MV	NIAH_S1	NIAH_S2	NIAH_S3	QA1	QA2	VT	Avg.
Meta-Llama-3.1-8B-Instruct, 20% Cache														
FullKV	51	94.67	100	100	100	97.5	99.5	100	100	100	84	54	100	90.82
AdaKV	10	85.33	82	21	25	74	62.5	98	94	21	34	47	92.2	57.38
CriticalKV	**12.6**	75.33	88	14	5	86.25	77.25	99	**100**	18	44	48	90	58.26
LaProx	12	**90.33**	**100**	**99**	**97**	**97.5**	**96.75**	**100**	**100**	**97**	**65**	**54**	**99.2**	**85.21**

The results in Table 13 and 14 reveal that, under this challenging setting, methods that do not leverage VW_O information suffer significant performance degradation. In contrast, LaProx consistently maintains performance close to the full KV cache and outperforms all competing approaches. Furthermore, on the more challenging NIAH variants (multi-needle), most prior methods fail to reliably retrieve the needles, while LaProx maintains performance comparable to the FullKV setting.

G.3 Evaluations on Mixture-of-Experts

To demonstrate that our approach easily adapts across architectures, we evaluate it on Mixture-of-Experts (MoE) models. Although MoE introduces sparsity in the Feed-Forward Networks (FFN), the attention layers—where the KV cache resides—remain unchanged. Since cache eviction operates within attention, our method transfers directly to MoE without any architectural modifications. In this experiment, we use Qwen3-30B-A3B-Instruct-2507 and perform the same tasks and settings with Appendix G.2. The results are presented in Tables 15 and 16.

Table 15: Comparison across 16 LongBench datasets. The best result is highlighted in **bold**.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Qwen3-30B-A3B-Instruct-2507, 20% Cache																	
FullKV	27.71	40.09	53.83	66.99	62.1	32.69	30.56	22.24	24.25	76	88.82	48.17	14	100	74.62	70.8	52.05
AdaKV	27.54	31.19	30.39	48	39.67	18.38	25.81	18.58	19.2	45	75.58	31.11	11.09	53	36.81	41.78	34.57
CriticalKV	32.06	30.79	33.19	54.06	**47.49**	**26.13**	**29.08**	19.88	21.04	70	**88.24**	47.02	**12**	98.5	75.34	69.79	47.16
LaProx	**32.94**	**37.41**	**46.1**	**59.49**	47.43	25.38	28.82	**19.97**	**22.07**	**78**	87.29	**48.54**	**12**	**100**	**75.78**	**71.74**	**49.56**

Table 16: Comparison across 13 Ruler datasets. The best result is highlighted in **bold**.

Method	CWE	FWE	NIAH_MK1	NIAH_MK2	NIAH_MK3	NIAH_MQ	NIAH_MV	NIAH_S1	NIAH_S2	NIAH_S3	QA1	QA2	VT	Avg.
Qwen3-30B-A3B-Instruct-2507, 20% Cache														
FullKV	83.2	99.33	100	100	100	100	98.5	100	100	100	86	64	100	94.69
AdaKV	57.8	92	8	10	2	16	0	16	0	4	32	46	28.4	24.02
CriticalKV	60.6	96	**100**	14	2	100	**98**	100	100	4	34	48	97.2	65.67
LaProx	**64.5**	**97**	**100**	**100**	**99**	**100**	92.75	**100**	**100**	**100**	**71**	**63**	**99.8**	**91.31**

The results in Tables 15 and 16 further demonstrate the effectiveness of our method in the MoE setting, where it limits performance degradation to only around 3

G.4 Scalability to Larger Models

Section G.3 already includes experiments on a larger model, Qwen3-30B-A3B-Instruct-2507 MoE, where LaProx consistently outperforms the strongest baselines. These results demonstrate that the proposed scoring mechanism generalizes beyond the 7B–8B models evaluated in the main paper.

To further strengthen this evidence, we additionally evaluate LaProx on the Qwen3-32B dense model, comparing against the same baselines, AdaKV and CriticalKV, under the same setting. The results are shown below.

Table 17: LongBench results on Qwen3-32B with a 20% cache size.

Methods	Single-Doc QA	Multi-Doc QA	Summarization	Few-shot Learning	Synthetic	Code	Average
AdaKV	28.57	41.00	23.83	61.57	50.50	20.52	37.93
CriticalKV	32.23	47.75	25.00	61.26	**57.50**	**21.83**	41.08
LaProx	**39.84**	**50.37**	**26.54**	**67.39**	**57.50**	20.76	**44.29**

As shown in Tables 17 and 18, **LaProx consistently maintains its advantage on both dense and MoE models at the 30B+ scale.** On LongBench, it achieves the highest scores on the majority of tasks. The improvement is even more pronounced on RULER, particularly on challenging tasks such as Needle-in-A-Haystack, where LaProx substantially outperforms both AdaKV and CriticalKV.

These experiments demonstrate that the proposed method scales well beyond the 7B–8B models presented in the main paper and remains effective for larger models.

G.5 Detailed Performance Across Cache Budgets on LongBench Benchmark

Tables 19, 20, and 21 report detailed LongBench results shown in Figure 2 for LaProx and competing baselines under cache budgets ranging from 128 to 1024 tokens on Meta-Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen3-8B, respectively.

Overall, the results show that LaProx consistently outperforms prior methods across all LongBench task categories on all evaluated models.

Table 18: RULER results on Qwen3-32B with a 20% cache size.

Methods	Word Extraction	Needle-in-A-Haystack	Multi-Hop Tracing	Question Answering	Average
AdaKV	79.57	20.38	79.20	36	36.41
CriticalKV	71.80	58.25	99.60	45	61.48
LaProx	**86.47**	**84.57**	**100**	**65**	**84.23**

Table 19: Comparison across 16 LongBench datasets on Meta-Llama-3.1-8B-Instruct for cache budgets from 128L to 1024L. The best result is highlighted in **bold** and the second best in underline.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Meta-Llama-3.1-8B-Instruct, $B_{\text{total}} = Full$																	
FullKV	30.28	45.46	54.94	56.02	46.5	31.28	35.11	25.28	27.19	72.5	91.65	43.86	8.88	99.5	65.13	58.7	49.51
Meta-Llama-3.1-8B-Instruct, $B_{\text{total}} = 128L$																	
SLLM	22.45	20.83	31.74	44.86	39.31	23.88	18.75	20.78	18.57	40.5	85.57	38.24	8.16	**99.5**	58.89	49.7	38.86
SnapKV	25.53	23.92	48.15	52.94	43.5	26.67	19.48	22.47	19.68	47	89.56	39.51	8.41	98	59.43	52.84	42.31
AdaKV	25.74	24.8	49.38	52.73	43.75	27.8	20.37	23.02	20.38	49	**90.19**	40.12	8.38	99	60.92	54.4	43.12
CAKE	**25.93**	25.5	48.33	51.95	43.42	28.94	20.91	22.93	19.96	47.5	88.84	**41.26**	8.38	99	59.56	51.97	42.77
CriticalKV	24.56	25.89	49.86	52.42	43.89	27.96	20.71	22.69	20.06	48.5	89.61	39.94	8.38	99	60.08	52.56	42.88
LaProx	25.78	**31.38**	**52.97**	**55.17**	**44.95**	**29.46**	**21.66**	**23.66**	**21.47**	**59**	89.94	41.09	**8.89**	**99.5**	**61.91**	**56.16**	**45.19**
Meta-Llama-3.1-8B-Instruct, $B_{\text{total}} = 256L$																	
SLLM	22.52	23.45	32.21	45.69	37.93	24.61	21.12	20.86	20.91	46	87.5	41	7.61	**99.5**	61.11	51.44	40.22
SnapKV	27.6	33.56	51.53	54.95	44.8	29.27	22.56	23.44	22.28	55	91.38	40.69	**8.88**	**99.5**	62.48	55.5	45.21
AdaKV	26.55	34.38	52.96	55.21	44.23	29.24	23.12	24	22.85	60.5	91.57	41.04	7.84	**99.5**	63.32	56.72	45.81
CAKE	27.28	34.41	52.34	54.14	45.57	30.06	23.66	24	22.63	59	91.25	41.48	7.97	**99.5**	62.1	54.37	45.61
CriticalKV	**28.31**	33.76	51.49	54.96	44.09	29.73	23.39	**24.14**	22.6	57.5	91.49	41.75	8.44	**99.5**	62.72	56.49	45.64
LaProx	28.02	**38.56**	**53.32**	**55.23**	**45.72**	**30.83**	**24.84**	23.78	**23.92**	**67.5**	**92.12**	**42.4**	8.56	**99.5**	**63.57**	**57.61**	**47.22**
Meta-Llama-3.1-8B-Instruct, $B_{\text{total}} = 512L$																	
SLLM	25.65	26.05	33.92	46.14	36.98	24.69	23.87	21.2	23.62	57.5	87.7	41.92	7.38	96.5	62.37	53.72	41.83
SnapKV	29.7	39.31	53.78	55.09	45.95	30.44	25.64	24.16	24.14	65.5	91.66	41.18	8.12	**99.5**	64.7	56.9	47.23
AdaKV	28.54	41.77	54.18	**55.68**	45.44	30.48	25.77	24.34	25	69	**92.04**	41.66	7.66	**99.5**	**64.75**	57.66	47.71
CAKE	29.59	41.8	**54.39**	54.94	45.79	30.91	26.16	24.44	24.83	63.5	91.93	41.8	8.17	**99.5**	63.99	56.82	47.41
CriticalKV	**30.33**	39.69	53.81	54.91	45.89	**31.05**	25.59	**24.74**	24.44	62	91.94	**42.66**	8.12	**99.5**	64.53	57.24	47.27
LaProx	29.82	**43.46**	54.33	55.44	**46.16**	30.83	**27.08**	24.1	**25.3**	**70.5**	91.89	42.48	**8.4**	**99.5**	64.1	**58.16**	**48.23**
Meta-Llama-3.1-8B-Instruct, $B_{\text{total}} = 1024L$																	
SLLM	25.34	29.79	37.08	47.79	39.83	24.44	26.28	21.43	25.85	63.5	88.83	42.56	7.91	88.5	63.17	55.96	43.02
SnapKV	30.54	43.93	53.96	55.8	46.3	31.19	28.41	**24.75**	25.99	69.5	91.78	42.22	8.16	**99.5**	64.62	58.85	48.46
AdaKV	**31.09**	43.71	54.44	**55.82**	46.15	30.65	28.54	24.41	25.82	71	91.87	42.32	7.96	**99.5**	64.57	58.56	48.52
CAKE	30.32	43.25	53.98	55.69	46.53	30.96	28.81	24.21	25.74	68	**91.9**	42.87	7.79	**99.5**	**65.01**	58.11	48.29
CriticalKV	30.14	43.8	54.11	55.63	**46.6**	30.88	28.76	24.66	26.1	68.5	91.89	42.42	7.93	**99.5**	64.83	**58.9**	48.41
LaProx	31.08	**44.45**	**55.64**	**55.82**	45.98	**31.96**	**29.22**	24.42	**26.36**	**71.5**	91.75	**43.32**	**8.23**	**99.5**	**65.01**	58.69	**48.94**

H Design Choice Discussion

The goal of KV-cache eviction is to identify which cached key-value (KV) pairs can be removed while minimizing their impact on model outputs. Since cached KV pairs are directly consumed only by the attention mechanism, their eviction primarily affects the attention output AVW_O, which is subsequently incorporated into the residual stream. Therefore, we formulate our optimization objective around the perturbation of AVW_O, as it **directly captures the impact of KV eviction** before its effects propagate through subsequent computations.

Although this perturbation propagates through residual connections, MLPs, and subsequent Transformer blocks, these components operate on the modified residual stream rather than individual KV pairs. Once the attention output is computed, the contributions of individual KV pairs have already been aggregated and can no longer be explicitly isolated. Consequently, directly estimating the effect of each KV pair on the final model output would require tracing its influence through all subsequent transformations. Such an approach would introduce substantial computational complexity, making it impractical for **efficient online KV-cache eviction.**

Furthermore, assigning importance scores to individual KV pairs based on their contributions to the attention computation is a fundamental **characteristic of efficient KV-cache eviction frameworks.** Within this formulation, LaProx provides a more principled theoretical framework for estimating the impact of KV eviction, which consistently translates into superior empirical performance.

Table 20: Comparison across 16 LongBench datasets on Mistral-7B-Instruct-v0.3 for cache budgets from 128L to 1024L. The best result is highlighted in **bold** and the second best in underline.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = Full$																	
FullKV	29.07	41.58	52.88	49.37	39.01	28.58	34.81	25.66	27.82	76	88.59	47.4	5.5	98	61.4	62.53	48.01
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 128L$																	
SLLM	21.42	22.28	26.82	37.28	33.31	17.61	16.73	19.77	17.86	45.5	85.64	40.47	5.5	80	55.13	52.07	36.08
SnapKV	23.81	25.05	46.92	44.98	35.06	23.11	20.49	21.55	19.32	43.5	**89.78**	42.87	6	93.5	55.95	54.6	40.41
AdaKV	24.08	25.52	47.08	47.16	35.16	23.83	20.23	22.28	20.7	45	88.92	**43.77**	6.5	92.5	56.63	55.81	40.95
CAKE	25.04	27.32	48.49	47.48	35.24	24.63	**21.32**	22.54	20.78	45.5	89.41	43.37	**7**	95	56.47	54.99	41.54
CriticalKV	25.42	25.95	46.43	46.44	35.92	23.93	21.12	22.35	19.26	47	89.23	43.62	6.5	93.5	55.99	54.51	41.07
LaProx	**28**	**30.26**	**54.02**	**47.86**	**37.54**	**25.54**	21.26	**23.31**	**22.1**	**63.5**	89.7	43.46	5	**96**	**58.85**	**57.66**	**44.00**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 256L$																	
SLLM	22.49	23.21	29.75	39.62	32.51	16.96	19.14	19.29	20.12	54.5	85.12	42.98	5.5	80	57.71	55.05	37.75
SnapKV	27.33	31.02	51.54	48.95	36.54	27.04	22.16	22.94	22.06	55.5	89.4	44.52	5	**96.5**	58.87	58.03	43.58
AdaKV	27.01	31.11	52.32	47.14	37.06	28.01	22.45	23.05	22.65	**62.5**	**89.44**	44.31	**6**	**96.5**	58.65	59.32	44.22
CAKE	27.1	31.42	53.3	**49.38**	37.56	27.63	22.87	23.17	23.08	57	89.24	**44.57**	4.5	**96.5**	**59.5**	59.47	44.14
CriticalKV	26.8	31.32	52.35	48.85	36.85	27.33	22.55	23.1	22.95	58.5	89.23	43.96	4.5	**96.5**	59.18	59.38	43.96
LaProx	**28.53**	**34.71**	**54.87**	48.88	**37.62**	**28.99**	**23.53**	**24.07**	**23.64**	**62.5**	89.03	44.29	5.5	96	59.15	**59.95**	**45.08**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 512L$																	
SLLM	24.19	25.89	30.45	40.6	32.36	17.35	22.02	20.2	23.28	65.5	86.95	43.75	**6**	81	59.29	56.34	39.70
SnapKV	28.08	33.93	**53.91**	48.96	37.63	27.39	24	23.77	24.07	67	89.23	45.26	5	96.5	59.95	60.22	45.31
AdaKV	29.04	34.64	53.35	48.99	36.72	27.66	24.41	23.99	24.47	71	89.03	45.69	5	97	60.03	60.96	45.74
CAKE	28.68	35.54	53.68	48.64	38.6	28.32	25.07	23.76	**25.05**	69.5	89.44	45.64	5.5	**97.5**	59.98	60.87	45.98
CriticalKV	28.09	37.07	53.71	49.52	37.34	28.08	24.48	24.25	24.84	71	89.33	44.28	3.5	**97.5**	60.33	61.33	45.91
LaProx	**29.7**	**37.29**	52.9	**49.57**	**38.73**	**28.47**	**25.41**	**25.09**	24.77	**74.5**	**89.66**	**45.94**	**6**	97	**60.83**	**61.86**	**46.74**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 1024L$																	
SLLM	24.79	27.88	30.99	42.91	32.65	18.03	24.64	20.7	25.45	68.5	88.71	45.37	5.5	82.5	61.1	59.24	41.19
SnapKV	28.35	37.47	53.41	49.12	**39.46**	**28.38**	26.29	24.67	25.94	71	88.89	46.84	5	97.5	61.08	61.77	46.57
AdaKV	28.14	37.35	52.86	48.97	38.59	28.24	26.6	24.82	25.81	72.5	89.19	46.12	5	98.5	60.82	62.36	46.61
CAKE	**29.93**	37.6	53.12	50.3	38.63	28.36	27.43	24.49	**26.73**	73	89.19	45.95	**6**	97.5	61.01	62.13	46.96
CriticalKV	29.06	37.28	**53.9**	49.92	38.24	28.21	26.62	24.14	26.56	73.5	89.19	45.27	5	97.5	**61.69**	**62.67**	46.79
LaProx	28.99	**39.16**	52.1	**50.39**	38.87	28.05	**27.77**	**25.12**	26.36	**76**	**89.61**	**47**	5.5	**99**	61.47	62.31	**47.36**

Table 21: Comparison across 16 LongBench datasets on Qwen3-8B for cache budgets from 128L to 1024L. The best result is highlighted in **bold** and the second best in underline.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Qwen3-8B, $B_{\text{total}} = Full$																	
FullKV	32.26	46.59	52.73	59.35	51.21	33.2	32.44	24.13	25.68	71	65.82	40	1	100	53.06	50.17	46.17
Qwen3-8B, $B_{\text{total}} = 128L$																	
SLLM	12.18	28.05	21.79	36.34	39.07	6.88	15.78	18.54	15.95	43	62.56	33.3	3	73	45.16	45.43	31.25
SnapKV	16.34	31.35	42.37	52.67	42.33	18.26	16.03	19.68	16.49	53	75.69	34.44	1	**99**	46.86	47.86	38.33
AdaKV	**21.66**	32.58	43.96	50.81	42.95	20.9	16.45	19.33	16.02	51	73.92	34.24	1	98	45.77	46.92	38.47
CAKE	19.69	32.67	45.07	57.5	**48.02**	20.5	17.82	20.87	16.73	47	69.51	35.39	**5**	**99**	45.55	45.94	39.14
CriticalKV	18.47	35.83	45.78	48.59	42.84	19.09	16.67	19.94	**17.61**	56	75.31	35.3	0	**99**	47.79	48.28	39.15
LaProx	17.51	**36.91**	**51.22**	**58.64**	46.31	**23.85**	**18.68**	**21.21**	17.32	**61**	**76.49**	**35.86**	1	**99**	**48.86**	**51.21**	**41.57**
Qwen3-8B, $B_{\text{total}} = 256L$																	
SLLM	14.01	28.83	22.05	36.99	38.97	7.83	18.6	18.7	18.73	52	70.4	34.79	5	67	49.86	46.88	33.16
SnapKV	18.2	37.54	49.44	63.13	**48.93**	24.45	20.4	21.7	19.2	59	70.62	37.04	0	**100**	50.64	**53.17**	42.09
AdaKV	22.31	36.05	47.75	57.07	44.62	22.33	20.47	20.46	18.83	64	**74.07**	37.82	1	**100**	50.12	49.91	41.67
CAKE	20.37	38.78	50.31	**64.01**	48.27	25.47	22.02	21.56	19.81	61	73.17	38.12	**9**	**100**	49.14	49.66	43.16
CriticalKV	22.15	40.41	48.89	58.01	46.83	27.89	21.62	21.72	20.1	68	70.52	37.64	0	**100**	51.35	52.14	42.97
LaProx	**23.5**	**41.26**	**50.96**	62.98	46.55	**30.44**	**22.55**	**22.29**	**21.63**	**70**	72.97	**38.17**	0	**100**	**52.51**	53	**44.30**
Qwen3-8B, $B_{\text{total}} = 512L$																	
SLLM	15.13	31.56	24.92	39.07	41.29	8.16	22.21	18.87	22.25	62	74.21	35.3	**7**	51	52.22	50.92	34.76
SnapKV	22.11	42.59	**53.05**	61.28	49.68	30.65	23.94	23.26	21.88	67	69.27	**39.44**	0	**100**	54.12	53.16	44.46
AdaKV	22.56	40.55	48.46	60.76	47.57	30.57	23.93	21.88	21.47	67	68.97	38.13	0	**100**	52.28	52.52	43.54
CAKE	23.06	43.77	51.53	60.53	49.84	**31.84**	25.25	23.11	22.76	64	**75.43**	38.42	6	**100**	53.02	53.45	45.12
CriticalKV	**25.47**	43.98	51.37	60.7	48.81	28.44	**25.5**	22.75	22.67	69	71.47	38.47	0	**100**	**54.89**	53.75	44.82
LaProx	24.76	**45.84**	52.68	**61.96**	**49.98**	28.87	25.37	**23.3**	**24.33**	**71**	74.77	38.3	0	**100**	53.8	**53.76**	**45.55**
Qwen3-8B, $B_{\text{total}} = 1024L$																	
SLLM	19.18	33.34	27.03	44.7	40.99	9.59	25.6	20.15	21.02	63	80.02	37.23	**7**	39	53.29	50.53	35.73
SnapKV	26.38	43.49	**53.2**	**62.25**	48.92	29.77	27.31	**23.49**	20.87	69	71.97	38.09	1	**100**	54.51	51.95	45.13
AdaKV	24.88	42.59	51.68	60.18	46.78	30.56	26.68	21.96	20.88	**70**	74.77	38.43	0	**100**	53.28	51.42	44.63
CAKE	24.09	44.97	52.02	60.58	**50.62**	**31.54**	27.83	23.25	**24.33**	**70**	73.99	37.71	1	**100**	54.67	53.56	45.63
CriticalKV	24.97	43.61	51.95	60.02	48.73	28.8	**28.23**	23	21.28	**70**	73.7	37.59	0	**100**	**54.71**	**53.8**	45.02
LaProx	**27.71**	**46.12**	53.04	61.83	50.48	30.72	27.58	22.9	21.07	**70**	**76.27**	**41.35**	1	**100**	54.18	52.37	**46.04**

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 statements presented in the abstract and introduction are completely consistent with the paper's primary contributions.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 limitations of our work are discussed in Appendix E.

   Guidelines:

   - The answer [N/A] 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 and statements are accompanied by complete proofs. Detailed proofs are provided in Appendix F.1 and F.2.

Guidelines:

- The answer [N/A] 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: All experiment settings and implementation details are described in section 5 and A.

   Guidelines:

   - The answer [N/A] 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: [No]

Justification: We provided pseudo codes for our implementations (Algorithm 1 and 2) and the detailed experimental settings (Section 5 and Appendix A). We will open-source our code upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

Answer: [Yes]

Justification: Our method is a plug-and-play method that doesn't include any training process. Detailed settings are described in Section 5 and Appendix A.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: We employ deterministic approaches without randomness, making error bars unnecessary. We also validate our approach through an exceptionally broad range of tests on LongBench, RULER, and InfiniteBench, which exceeds typical benchmarks and confirms the effectiveness of our method."

Guidelines:

- The answer [N/A] 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: We detail the computation resources and efficiency in Section 5.4.

   Guidelines:

   - The answer [N/A] 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 complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: This paper improves LLMs inference performance and efficiency via a new KV cache management. The proposed method introduces no new ethical concerns beyond those already associated with LLMs. We do not identify any broader societal impacts requiring specific discussion.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new models or data.

    Guidelines:

    - The answer [N/A] 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: All third-party assets used in our work are properly credited in the paper with appropriate citations.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We do not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.