CoCoKV: Consensus- and Completeness-Aware KV Cache Compression for LLM Inference

Anonymous authors
Paper under double-blind review

Abstract

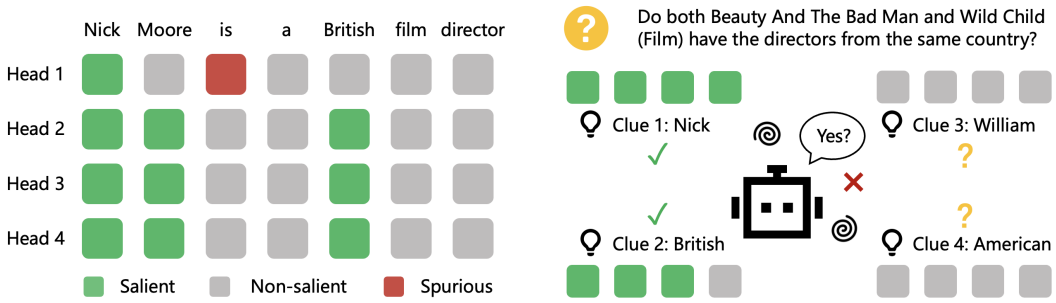
Efficient long-context inference in Large Language Models (LLMs) is severely constrained by the memory footprint of the Key-Value (KV) cache. Recent prompt-stage methods reduce this overhead by estimating historical-token importance from observation-window attention and retaining only a compact subset of the KV cache. However, their reliance on head-specific token importance gives rise to two potential limitations: i) Head-wise Attention Partiality, where individual attention heads can capture specialized yet partial views of token importance, and ii) Token-wise Information Omission, where selecting highly attended tokens leads to semantic omission while overlooking other crucial information. To address these limitations, we present CoCoKV, a novel KV cache compression framework designed to retain both reliable and comprehensive evidence under constrained cache budgets. Specifically, CoCoKV introduces a soft cross-head consensus mechanism that uses collective signals to refine head-specific attention while preserving head distinctiveness. Building on the calibrated importance scores, a cross-token completeness selection strategy is applied to promote broader semantic coverage. Evaluations across Needle-in-a-Haystack and 16 LongBench datasets on three LLMs demonstrate that CoCoKV outperforms competitive baselines, validating its effectiveness and robustness.

1 Introduction

Large Language Models (LLMs) have shown remarkable capabilities in long-context understanding, enabling a wide range of applications such as question answering, dialogue, retrieval, and code completion Singh et al. (2025); Comanici et al. (2025); Yang et al. (2025); Xu et al. (2026); Anthropic (2025). To accelerate inference, LLMs store the key and value states of previous tokens as the KV cache during autoregressive decoding, thereby avoiding repeatedly recomputing historical representations. Nevertheless, the memory footprint of the KV cache grows linearly with the sequence length. As modern LLMs increasingly support longer contexts, the KV cache has become a major bottleneck for efficient inference, particularly under limited GPU memory Xiao et al. (2024). Consequently, the maximum supported context length and serving throughput of LLMs are substantially restricted in practical deployments.

The increasing memory burden has motivated extensive research on KV cache compression, which aims to reduce memory consumption while maintaining the contextual information essential for inference. The prevailing paradigm estimates token importance from attention scores, based on the intuition that highly attended tokens contribute more to future generation. A straightforward realization is to retain tokens with the highest accumulated attention scores during decoding and evict less attended ones Zhang et al. (2023); Liu et al. (2023). However, such a direct strategy incurs considerable overhead, since token importance computation and eviction must be continuously performed at every generation step. To further improve efficiency, the community has shifted the KV cache compression to the prompt stage, leveraging an observation window to identify important tokens and complete the KV cache compression before decoding Li et al. (2024); Liu et al. (2025a); Chen et al. (2026); Liu et al. (2025b). Building on this framework, subsequent methods have enhanced KV cache compression from various perspectives, including layer-wise Cai et al. (2025) or head-wise Feng et al. (2025a); Fu et al. (2025) budget allocation, and improved attention aggregation Guo et al. (2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Head-wise Attention Partiality (b) Token-wise Information Omission

Figure 1: Illustration of two potential limitations in existing KV cache compression methods. (a) Head-wise Attention Partiality: Head-specific attention captures specialized information but may provide a partial view of token importance. (b) Token-wise Information Omission: Attention-based selection may leave essential semantic clues uncovered under a limited cache budget.

Despite these advances, most existing methods still make cache retention decisions based on head-specific token attention scores. We identify two potential limitations of this paradigm that can lead to sub-optimal retention decisions. To be specific, **i) Head-wise Attention Partiality:** Different attention heads naturally specialize in different aspects of the input, making head-specific attention valuable Fu et al. (2025); Xiao et al. (2025). However, under a highly constrained cache budget, relying on each head independently may provide only a partial view of token importance and overlook evidence that receives consistent support across multiple heads. As exhibited in Fig. 1a, while most heads emphasize informative tokens such as "Nick", "Moore", and "British", one head assigns spurious attention to meaningless evidence "is". **ii) Token-wise Information Omission:** Selecting tokens solely according to their attention scores could concentrate on only a subset of semantic evidence, while leaving other essential information uncovered. As illustrated in Fig. 1b, answering the question requires combining evidence about both directors, namely "Nick Moore–British" and "William–American", whereas attention-based selection concentrates on the former and discards the latter, leading to an incorrect conclusion.

To address the above two limitations, we propose Consensus- and completeness-aware KV cache compression (CoCoKV), which aims to preserve reliable and comprehensive contextual evidence under a limited cache budget. Specifically, CoCoKV first introduces a soft cross-head consensus mechanism that calibrates head-specific attention with collective evidence, thereby improving the reliability of token importance estimation while preserving head distinctiveness. Based on the calibrated importance scores, CoCoKV further performs cross-token completeness-aware selection, jointly considering token importance and semantic coverage to improve the coverage of essential semantic regions and reduce information omission. These two components respectively mitigate head-wise attention partiality and token-wise information omission. By allocating the limited cache to reliable and comprehensive contextual evidence, CoCoKV constructs a compact yet informative cache with broad semantic coverage.

Overall, our contributions can be summarized as follows:

- We introduce a cross-head consensus module to calibrate head-specific attention with soft collective evidence, improving the reliability of token importance estimation while preserving head distinctiveness.
- We develop a cross-token completeness module to jointly consider token importance and semantic coverage, reducing information omission and retaining more comprehensive contextual evidence.
- Extensive experiments on LongBench across three representative LLMs, together with evaluations on Needle-in-a-Haystack, demonstrate the robustness and effectiveness of our proposed CoCoKV.

2 Related Work

KV cache compression methods aim to reduce the memory overhead while keeping the essential information for inference. Early efforts mainly rely on heuristic eviction strategies, such as the most recent tokens Xiao et al. (2024), predefined retention policies Ge et al. (2024), and heavy-hitter tokens with high accumulated attention scores during decoding Zhang et al. (2023); Liu et al. (2023). Although these methods alleviate the GPU memory overhead, their reliance on heuristic cache retention rules can limit their efficiency and adaptivity in long-context settings.

Different from these heuristic methods, recent studies have adopted an effective prompt-stage compression paradigm based on the observation window attention Li et al. (2024). Specifically, an observation window refers to a short token segment at the end of the prompt, whose attention to preceding tokens is used to estimate the importance of KV tokens before decoding. Building on this framework, some researchers have investigated how cache budgets should be allocated across different layers Cai et al. (2025) or heads Fu et al. (2025); Feng et al. (2025a), enabling more flexible token retention. Beyond the attention of the key tokens, some works further incorporate the value states to better estimate the contribution of cached tokens to generation Feng et al. (2025b); An et al. (2026). Besides, extended observation window strategies have been designed to better capture the global structural semantics of KV tokens Zweiger et al. (2026); Chen et al. (2026). More recently, the aggregation of attention scores within the observation window has also been studied to make improvements Guo et al. (2026).

Despite their progress, most existing methods derive KV cache retention decisions independently from head-specific token attention. While such head-wise specialization is valuable, under a highly constrained budget, it can lead to unreliable importance estimation and insufficient semantic coverage. Therefore, CoCoKV introduces a soft cross-head consensus signal to calibrate head-specific scores while maintaining head distinctiveness. Moreover, CoCoKV explicitly models cross-token completeness during selection, encouraging broader semantic coverage under the limited cache budget. These two designs enable CoCoKV to preserve reliable and comprehensive contextual evidence.

3 Method

In this section, we detail CoCoKV, which preserves reliable and comprehensive contextual evidence, as illustrated in Fig. 2. To be specific, we first formulate the KV cache compression problem, and then introduce the Cross-Head Consensus module to calibrate head-specific attention with collective evidence. Based on the calibrated scores, the Cross-Token Completeness module further retains informative tokens with broader semantic coverage.

3.1 Problem Formulation

Given an input sequence $\mathbf{x} = \{x_i\}_{i=1}^{N}$, the LLM computes the key and value vectors for each input token at every attention head of every transformer layer. Mathematically, the corresponding key and value at the h-th head of the l-th layer are denoted as $\mathbf{K}^{h,l} = [\mathbf{k}_1^{h,l}, \ldots, \mathbf{k}_N^{h,l}]^\top$ and $\mathbf{V}^{h,l} = [\mathbf{v}_1^{h,l}, \ldots, \mathbf{v}_N^{h,l}]^\top \in \mathbb{R}^{N\times d}$, respectively, where d is the head dimension.

During autoregressive decoding, predicting the next token only requires the query vector of the most recent token, which attends to the cached key and value states of all preceding tokens. Accordingly, let $\mathbf{q}^{h,l}$ denote the last query token, then the attention score based on the full cache is

$$\widetilde{\mathbf{A}}^{h,l} = \mathrm{Softmax}[\frac{\mathbf{q}^{h,l}(\mathbf{K}^{h,l})^\top}{\sqrt{d}}]. \tag{1}$$

Notably, the memory cost of full KV cache increases linearly with the sequence length N. To tackle this prohibitive expense, KV cache compression is proposed to select a compact subset of historical tokens while maintaining the predictive behavior of the full cache model. Let $\mathcal{U} = \{1, 2, \ldots, N\}$ be the complete token set and $\mathbf{S}^{h,l} \subseteq \mathcal{U}$ be the subset retained by the h-th head in the l-th layer. The attention score based on the compressed cache is

$$\mathbf{A}_{\mathbf{S}^{h,l}}^{h,l} = \mathrm{Softmax}[\frac{\mathbf{q}^{h,l}(\mathbf{K}_{\mathbf{S}^{h,l}}^{h,l})^\top}{\sqrt{d}}], \tag{2}$$

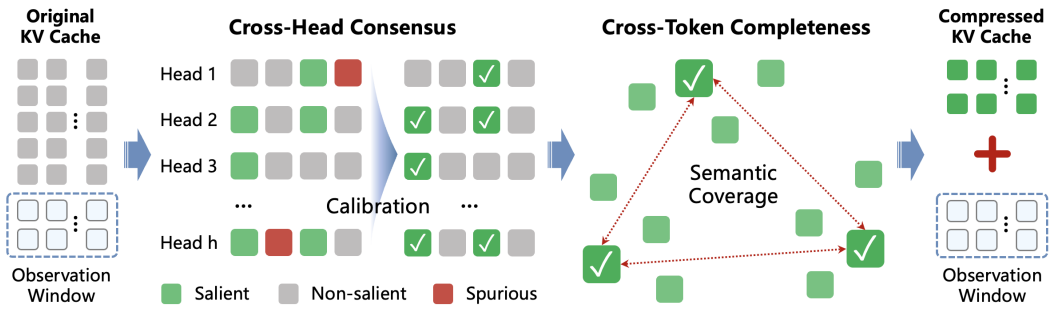


Figure 2: Overview of the proposed CoCoKV. Given the original KV cache, CoCoKV utilizes two successive modules to determine the retained tokens. First, the cross-head consensus module calibrates head-specific attention with collective evidence to improve importance reliability. Then, the cross-token completeness module selects informative tokens with broader semantic coverage, which are finally combined with the observation window to form the compressed KV cache.

where $\mathbf{K}^{h,l}_{\mathbf{S}^{h,l}}$ contains only the key states indexed by $\mathbf{S}^{h,l}$.

Let $\widetilde{\mathbf{A}}$ and $\mathbf{A}_{\mathbf{S}}$ denote the full cache and compressed attention scores, respectively. The objective of KV cache compression is to minimize the resulting performance degradation $\mathcal{D}$ under a limited cache budget C:

$$\begin{aligned} \mathbf{S}^* &= \arg\min_{\mathbf{S}} \mathcal{D}(\widetilde{\mathbf{A}}\mathbf{V}, \mathbf{A}_{\mathbf{S}}\mathbf{V}_{\mathbf{S}}) \\ \text{s.t.} \quad & |\mathbf{S}^{h,l}| \le C, \quad \forall h, l \end{aligned}. \tag{3}$$

To approximate this objective, CoCoKV identifies reliable and comprehensive contextual evidence through cross-head consensus and cross-token completeness.

3.2 Cross-Head Consensus

Following the observation-window paradigm Li et al. (2024), we adopt the last w tokens as queries to estimate the importance of the preceding $N - w$ tokens. For the h-th head in the l-th layer, its attention score is computed as

$$\mathbf{A}^{h,l} = \text{MaxPool}[\text{Softmax}(\frac{\mathbf{Q}^{h,l}_{-w:}(\mathbf{K}^{h,l}_{:-w})^\top}{\sqrt{d}})]. \tag{4}$$

It is well established that different attention heads generally capture distinct aspects of the input context. Consequently, simply retaining an identical cache set across all heads would suppress these specialized attention patterns. However, when the cache budget is severely limited, an individual head may provide a partial view of token importance while discarding critical evidence that is consistently recognized by most heads. Therefore, we introduce a soft cross-head consensus module that incorporates collective evidence while preserving head-specific distinctiveness.

Specifically, we aggregate the attention scores of all H KV heads within the same layer to obtain the layer-level consensus:

$$\overline{\mathbf{A}^l} = \frac{1}{H}\sum_{h=1}^{H} \mathbf{A}^{h,l}. \tag{5}$$

We then calibrate each head-specific importance vector using the soft collective signal:

$$\mathbf{Cons}^{h,l} = \mathbf{A}^{h,l} + \alpha\overline{\mathbf{A}^l}, \tag{6}$$

where α is the consensus ratio. The first term serves as the primary signal and preserves the specialized preference of the current head, while the second term provides an auxiliary signal favoring tokens that receive consistent support across heads. In this way, cross-head consensus supplements head-specific attention, preserving distinct token selections across heads while benefiting from consensual evidence.

Since the most recent w tokens are preserved as the window, given the cache capacity C, the number of cache tokens to be selected is $C' = C - w$. Rather than directly retaining the top-C' tokens, we construct an oversampled candidate set

$$\mathrm{Cand}^{h,l} = \mathrm{TopK}(\mathbf{Cons}^{h,l}, \gamma C'), \tag{7}$$

where TopK returns the indices of the largest entries, γ is the oversampling factor. The oversampled candidate pool retains high-importance tokens while providing sufficient alternatives for subsequent completeness-aware selection.

3.3 Cross-Token completeness

Although the cross-head consensus score provides a reliable estimate of token importance, directly retaining the highest-scoring tokens could concentrate the limited cache budget on only a subset of semantic regions, leaving other essential evidence uncovered. Such information omission hinders the semantic coverage of the compressed cache. Therefore, we then apply a cross-token completeness module to jointly consider the token importance and semantic coverage.

To be specific, we initialize the selected set with the candidate having the highest consensus score:

$$\mathbf{S}^{h,l} = \{\arg\max_{i \in \mathrm{Cand}^{h,l}} \mathbf{Cons}_i^{h,l}\}. \tag{8}$$

Subsequently, for each unselected candidate $j \in \mathrm{Cand}^{h,l} \setminus \mathbf{S}^{h,l}$, its completeness score is measured by its minimum cosine distance to the currently selected tokens (all tokens are normalized):

$$\mathbf{Comp}_j = \min_{g \in \mathbf{S}^{h,l}} [1 - (\mathbf{K}_j^{h,l})^\top \mathbf{K}_g^{h,l}]. \tag{9}$$

By this operation, a candidate receives a high $\mathbf{Comp}$ score when it lies far from the semantic regions already represented by the selected tokens, indicating its potential to expand the semantic coverage of the compressed cache, which could be further supported by the following Lemma 1.

Lemma 1 (Coverage Guarantee of Completeness). Define the covering radius R of a selected set $S^{h,l}$ as the minimum radius such that balls centered at all selected tokens can cover every candidate token in $\mathrm{Cand}^{h,l}$. Let R_{OPT} denote the theoretically optimal covering radius achievable by selecting C' tokens. Then, the set obtained by greedy completeness-based selection satisfies

$$R \leq 2R_{\mathrm{OPT}}. \tag{10}$$

A smaller coverage radius indicates that each candidate is closer to its nearest selected token, implying better semantic coverage. Thus, Lemma 1 shows that completeness-aware selection provides a bounded approximation to the optimal semantic coverage over the candidate set.

The proof of Lemma 1 can be found in Appendix A.4.1.

Overall, we select the candidate that maximizes the joint objective at each iteration

$$j^* = \arg\max_{j \in \mathrm{Cand}^{h,l} \setminus \mathbf{S}^{h,l}} (\mathbf{Cons}_j^{h,l} + \lambda \mathbf{Comp}_j), \tag{11}$$

where λ is a trade-off parameter. The consensus score $\mathbf{Cons}$ indicates which information is important to retain, whereas the completeness score $\mathbf{Comp}$ indicates where information remains uncovered.

Then we update the selected set as $\mathbf{S}^{h,l} \leftarrow \mathbf{S}^{h,l} \cup \{j^*\}$ until $|\mathbf{S}^{h,l}| = C'$.

To characterize the behavior of this selection process, we establish the following Lemma 2. It shows that candidates with higher informativeness and completeness are prioritized in earlier iterations.

Lemma 2 (Monotonicity of the Selection Process). The joint score of the selected token is monotonically non-increasing as the selection proceeds, $i.e.$,

$$\phi_{j_t}^{(t)} \geq \phi_{j_{t+1}}^{(t+1)}, \qquad \forall\, t \in \{1, \ldots, C' - 1\}, \tag{12}$$

where j_t denotes the token selected at the t-th iteration, and

$$\phi_j^{(t)} = \mathbf{Cons}_j^{h,l} + \lambda \mathbf{Comp}_j^{(t)} \tag{13}$$

denotes the joint score of candidate token j at the t-th iteration. Here, $\mathbf{Comp}_j^{(t)}$ is computed with respect to the $(t-1)$-th selected set $(\mathbf{S}^{h,l})^{(t-1)}$.

The proof of Lemma 2 can be found in Appendix A.4.2.

Finally, the selected historical indices are restored to their original temporal order and concatenated with the observation window, forming the compressed KV cache

$$\begin{aligned}(\mathbf{K}^{h,l})' &= \mathrm{Concat}(\mathbf{K}^{h,l}_{\mathbf{S}^{h,l}}, \mathbf{K}^{h,l}_{-w:}) \\ (\mathbf{V}^{h,l})' &= \mathrm{Concat}(\mathbf{V}^{h,l}_{\mathbf{S}^{h,l}}, \mathbf{V}^{h,l}_{-w:})\end{aligned}. \tag{14}$$

In this manner, CoCoKV preserves a compact KV cache with both reliable and comprehensive contextual information. The overall procedure is summarized in Appendix A.1.

4 Experiments

To evaluate the proposed CoCoKV, we apply it to two representative long-context benchmarks, $i.e.$, LongBench and Needle-in-a-Haystack, under multiple model architectures and cache budgets. Subsequently, we conduct component ablations and parameter sensitivity analyses to validate its effectiveness and robustness.

4.1 Experiments on LongBench

4.1.1 Datasets and Evaluation Metrics

We evaluate the comprehensive long-context understanding performance of our CoCoKV on 16 datasets spanning six task categories from LongBench Bai et al. (2024): single-document question answering Kočiskỳ et al. (2018); Dasigi et al. (2021), multi-document question answering Yang et al. (2018); Ho et al. (2020); Trivedi et al. (2022), summarization Huang et al. (2021); Zhong et al. (2021); Fabbri et al. (2019), few-shot learning Li & Roth (2002); Joshi et al. (2017); Gliwa et al. (2019), synthetic tasks Bai et al. (2024), and code completion Guo et al. (2023); Liu et al. (2024b). These datasets contain $3,750$ test instances in total, with average input lengths ranging from approximately $1k$ to $18k$ words. Following the standard LongBench protocol, we use F1 score for question answering tasks, ROUGE-L for summarization tasks, Accuracy for synthetic tasks, and Edit Similarity for code completion tasks. For the few-shot learning category, TREC is evaluated by Accuracy, TriviaQA by F1 score, and SAMSum by ROUGE-L.

4.1.2 Implementation Details

Our implementation is built upon the codebase of PyramidKV Cai et al. (2025). Unless otherwise specified, the completeness weight λ, oversampling factor γ, consensus ratio α, and observation window size w are set to 0.5, 3, 0.4, and 8, respectively. Besides, for a fair comparison, we adopt the StreamLLM Xiao et al. (2024) and H2O Zhang et al. (2023) implementations provided by PyramidKV Cai et al. (2025), where both methods compress only the prompt KV cache before decoding. All experiments can be conducted on a single NVIDIA RTX 4090 GPU.

4.1.3 Experimental Results

We compare CoCoKV with five well-established KV cache compression methods, including StreamLLM Xiao et al. (2024), H2O Zhang et al. (2023), SnapKV Li et al. (2024), PyramidKV Cai et al. (2025), and DefensiveKV Guo et al. (2026), while Full Cache is included as the uncompressed reference. Experiments are conducted on three widely used instruction-tuned LLMs, namely Llama-3-8B-Instruct Grattafiori et al. (2024), Mistral-7B-Instruct-v0.2 Jiang et al. (2023), and Qwen2.5-7B-Instruct An Yang et al. (2024).

As shown in Table 1, under the limited cache size of 64, CoCoKV achieves the highest average performance across all three models. Compared with the strongest compression baseline for each model, CoCoKV achieves relative improvements of 3.9%, 3.8%, and 2.9% in the average score, respectively. Notably, H2O has a strong tendency to favor salient and repeatedly attended information, which benefits tasks like summarization, but systematically suppresses sparse yet critical evidence,

Table 1: Performance on 16 LongBench datasets spanning six task categories under a KV cache budget of 64 on three mainstream LLMs. Among the KV cache compression methods, the best and second-best results are highlighted in bold and underlined, respectively. Full Cache is included as the uncompressed reference. "Avg." denotes the average score across all datasets, and "Gain." denotes the relative improvement over the strongest compression baseline.

	single-document qa			multi-document qa			summarization			few-shot learning			synthetic		code		
Method	nrtvqa	qasper	mf-en	hotpot	2wikiqa	musique	govrep	qmsum	multinews	trec	triviaqa	samsum	pcount	pr-en	lcc	rb-p	Avg. (Gain.)
Llama-3-8B-Instruct																	
Full Cache	24.56	31.83	40.35	43.56	35.29	21.18	28.74	23.24	26.66	73.50	90.48	42.49	4.80	69.25	59.23	54.19	41.83
StreamLLM	19.53	7.85	19.92	30.94	25.90	15.15	15.32	20.39	15.71	39.00	71.75	32.21	**6.00**	69.33	52.99	51.22	30.83
H2O	19.46	11.84	20.99	25.78	17.61	14.48	**20.32**	21.33	**22.98**	41.50	84.87	34.23	5.38	66.84	51.70	46.47	31.61
SnapKV	20.76	12.46	31.90	37.21	26.03	16.32	18.59	21.83	18.46	51.00	89.32	36.43	4.95	69.50	54.96	52.22	35.12
PyramidKV	20.79	13.83	29.59	34.78	23.52	15.92	18.91	21.54	19.13	**58.50**	87.88	36.85	5.29	69.50	53.89	49.57	34.97
DefensiveKV	21.47	12.77	31.35	38.95	27.67	**19.31**	18.93	**22.14**	18.98	43.50	**89.52**	**37.41**	4.23	**70.00**	55.24	53.56	35.31
CoCoKV	**22.24**	**18.32**	**33.28**	**39.92**	**28.31**	17.11	18.60	21.83	19.08	57.50	88.88	36.41	5.12	**70.00**	**56.32**	**53.97**	**36.68 (+3.9%)**
Mistral-7B-Instruct-v0.2																	
Full Cache	26.91	33.20	49.45	43.02	27.35	18.78	32.93	24.27	27.14	71.00	86.23	42.80	2.75	86.98	56.97	54.43	42.76
StreamLLM	14.97	15.65	26.70	30.47	21.51	12.36	14.36	19.37	15.53	39.50	78.84	35.49	3.58	25.50	46.75	42.48	27.69
H2O	19.32	17.06	31.10	33.10	16.20	12.91	**22.37**	20.61	**23.80**	**62.50**	83.66	32.63	**4.75**	30.90	40.77	36.41	30.51
SnapKV	19.80	18.81	38.56	31.12	21.42	14.39	18.69	**21.77**	19.46	53.50	82.48	37.73	2.61	59.55	48.40	44.26	33.28
PyramidKV	20.18	19.07	38.08	32.26	21.34	15.02	18.86	21.37	19.77	55.00	82.61	37.30	2.21	61.57	48.38	43.06	33.51
DefensiveKV	17.91	16.52	38.92	29.04	20.39	13.34	17.06	21.04	17.62	37.00	81.24	**39.31**	3.07	50.29	47.93	**45.08**	30.99
CoCoKV	**20.77**	**19.46**	**40.04**	**35.42**	**22.19**	**15.97**	19.09	21.58	20.28	60.00	**83.99**	38.23	3.66	**61.75**	**49.40**	44.75	**34.79 (+3.8%)**
Qwen2.5-7B-Instruct																	
Full Cache	29.49	40.86	50.78	58.04	46.24	29.00	24.40	22.69	22.99	53.00	89.61	42.63	8.00	98.50	46.63	60.69	45.22
StreamLLM	15.86	19.23	20.45	39.22	33.89	17.55	9.95	16.87	10.38	20.00	69.33	27.20	**8.50**	29.50	**41.31**	44.89	26.51
H2O	22.59	24.40	28.59	44.02	27.49	21.11	**17.92**	**19.19**	**19.51**	**38.00**	80.73	**30.50**	7.00	10.00	36.11	35.40	28.91
SnapKV	25.50	26.21	33.83	45.83	35.86	22.08	12.21	17.52	13.61	26.50	81.49	29.17	7.50	**65.00**	34.96	39.09	32.25
PyramidKV	20.06	22.46	25.38	36.72	18.57	18.15	10.39	14.66	11.69	20.00	75.30	21.73	7.50	35.50	29.63	30.27	24.88
DefensiveKV	21.15	20.81	27.91	42.11	20.54	19.70	9.50	15.19	10.82	19.00	81.05	24.01	7.50	44.81	40.87	**45.12**	28.13
CoCoKV	**25.94**	**27.46**	**36.42**	**48.20**	**37.18**	**22.28**	12.48	17.19	13.93	26.75	**84.89**	29.04	**8.50**	**65.00**	36.17	39.44	**33.18 (+2.9%)**

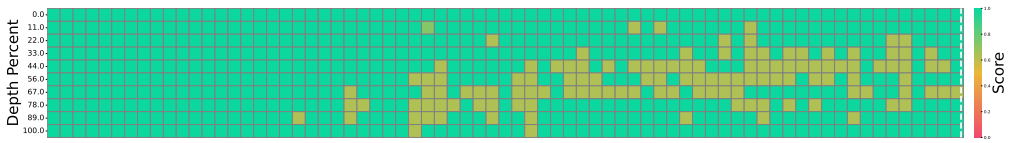
leading to substantial degradation in overall performance. These results establish CoCoKV as an effective and generalizable compression strategy across different model architectures.

To further investigate the performance of CoCoKV under different cache constraints, we evaluate all methods across varying cache budgets. As exhibited in Fig. 4a, CoCoKV consistently outperforms the competing compression methods across all evaluated cache budgets on Llama-3-8B-Instruct. These results demonstrate the effectiveness and robustness of our proposed CoCoKV.

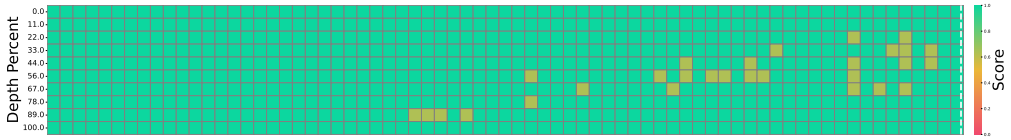
4.1.4 Computational Overhead

We further examine the computational overhead of CoCoKV. CoCoKV and all baseline methods are evaluated on a single NVIDIA RTX 4090 GPU under the same PyramidKV codebase. Under this experimental setting, the complete evaluation takes approximately 4 GPU hours for CoCoKV, SnapKV, PyramidKV, and DefensiveKV. This indicates that CoCoKV introduces little additional time latency compared with existing KV cache compression methods.

The limited overhead mainly stems from the restricted search space of the completeness-aware selection. Specifically, as shown in Eq. (7), CoCoKV first constructs a small candidate set containing only $\gamma C'$ tokens, and performs selection within this set rather than over the entire context. Then, the overall complexity across H attention heads and L layers is $O(LH\gamma {C'}^2 d)$, where d is the head dimension. The resulting cost therefore depends primarily on the cache budget, which is much smaller than the input sequence length. Moreover, this selection is performed only once during the prompt stage and introduces no additional computation during autoregressive decoding. Consequently, its overhead remains small relative to the Transformer forward computation, which is also reflected by the comparable overall runtime across different compression methods.



(a) SnapKV, Cache Budget=64, Acc=93.6%



(b) CoCoKV, Cache Budget=64, Acc=98.5%

Figure 3: Performance visualization of Needle-in-a-Haystack. The horizontal axis denotes the context length, and the vertical axis denotes the depth percentage of the needle.

Table 2: Performance comparison on the Needle-in-a-Haystack benchmark. The experiments are conducted on Llama-3-8B-Instruct with up to $8K$ context size under a cache size of 64.

Method	StreamLLM	H2O	SnapKV	PyramidKV	DefensiveKV	CoCoKV
Accuracy (%)	22.1	21.5	93.6	95.6	<u>97.1</u>	**98.5**

4.2 Experiments on Needle-in-a-Haystack

4.2.1 Experimental Setup

To test the long-context retrieval robustness of CoCoKV, we conduct experiments on the Needle-in-a-Haystack benchmark Liu et al. (2024a). Specifically, a target piece of information is inserted at varying depths within context lengths ranging from $1K$ to $8K$ tokens, and the model is required to retrieve it in response to the given query. Experiments are conducted on Llama-3-8B-Instruct with a cache size of 64, and retrieval accuracy is adopted as the evaluation metric.

4.2.2 Experimental Results

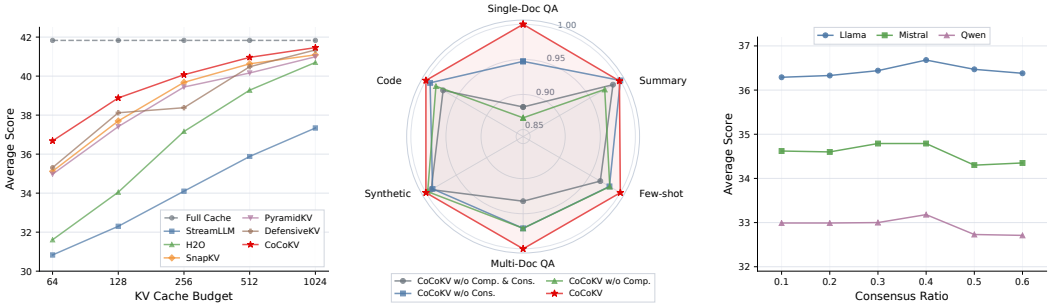
As shown in Table 2 and Fig. 3, CoCoKV achieves the best retrieval performance among all KV cache compression methods, outperforming the strongest baseline by 1.4%. We place other visualizations in Appendix A.2.2. This advantage arises from jointly promoting cross-head consensus and cross-token completeness, thereby enhancing the ability of CoCoKV to preserve critical evidence across different context lengths and positions. In contrast, StreamLLM struggles with performance because it only retains attention sinks, $i.e.$, the first few tokens, together with the most recent tokens. Under a highly constrained cache budget, this rigid retention policy covers only narrow regions at the beginning and end of the context, causing needles inserted at most depths to be systematically discarded. Besides, H2O also exhibits poor performance, as its heavy-hitter strategy prioritizes tokens that receive high accumulated attention scores. Since most tokens in the Needle-in-a-Haystack context are irrelevant distractors, their accumulated attention can easily overwhelm the isolated needle, making it particularly vulnerable to eviction.

4.3 Ablation Study and Parameter Analysis

To further demonstrate the effectiveness and robustness of the proposed CoCoKV, we conduct an ablation study of its two key design modules, namely cross-head consensus and cross-token completeness. Besides, for each module, we analyze the sensitivity to its associated parameters.

4.3.1 Effectiveness of the Selection Strategy

To investigate the contributions of cross-head consensus and cross-token completeness, we compare CoCoKV with three variants: removing consensus, completeness, and both modules (selecting tokens solely according to attention scores). As exhibited in Fig. 4b, removing either module



(a) Average performance across 16 LongBench datasets on Llama-3-8B-Instruct under varying KV cache budgets. The dashed line denotes Full Cache as the reference.

(b) Ablation study of the proposed KV cache compression strategy on LongBench datasets. The radar values are normalized by the performance of the complete CoCoKV.

(c) Average performance across 16 LongBench datasets under different consensus ratios on three different LLMs with the KV cache budget fixed at 64.

Figure 4: Performance analysis of CoCoKV in terms of KV cache budget, component effectiveness, and consensus ratio.

leads to performance degradation across all six task categories. The decline after removing consensus demonstrates that cross-head evidence can provide a beneficial complementary signal to head-specific attention, while the degradation without completeness indicates that attention-based selection alone tends to provide insufficient semantic coverage and omit essential information. By jointly promoting cross-head consensus and cross-token completeness, the complete CoCoKV achieves the best performance across all task categories.

4.3.2 Sensitivity of the Consensus Ratio

We investigate the sensitivity of CoCoKV to the consensus ratio in the cross-head consensus module across the three evaluated models. The clear performance gap between the complete CoCoKV and its consensus-free variant in Fig. 4b confirms the contribution of cross-head consensus. Moreover, as illustrated in Fig. 4c, the average performance remains stable as the consensus ratio varies from 0.1 to 0.6, with standard deviations of only 0.13, 0.19, and 0.16 on Llama-3-8B-Instruct, Mistral-7B-Instruct-v0.2, and Qwen2.5-7B-Instruct, respectively. These results demonstrate that CoCoKV is robust to the choice of the consensus ratio. Moreover, the best performance is achieved at 0.4, where head-specific attention and collective cross-head evidence are well balanced, preserving both head distinctiveness and consensus reliability.

For the other two parameters in the cross-token completeness module, $i.e.$, the oversampling factor and completeness weight, CoCoKV also exhibits stable performance across different settings of both parameters, further demonstrating its robustness, as shown in Appendix A.3.

5 Conclusion

In this work, we point out two potential limitations of existing KV cache compression methods, namely head-wise attention partiality and token-wise information omission, which arise from the reliance on head-specific token importance estimation. To tackle these two limitations, we propose CoCoKV to preserve both reliable and comprehensive contextual evidence. To be specific, CoCoKV first calibrates head-specific attention with soft cross-head evidence to improve the reliability of token importance estimates while preserving head distinctiveness. Based on the calibrated importance scores, CoCoKV further utilizes the cross-token completeness module to promote broader semantic coverage. Empirical evaluations across LongBench and Needle-in-a-Haystack demonstrate the effectiveness and robustness of CoCoKV, highlighting its long-context understanding capability. Nevertheless, the current evaluation of CoCoKV mainly focuses on text-based long-context scenarios, and the applicability to other challenging modalities remains to be further explored. Therefore, one promising future direction is extending our CoCoKV to multimodal large language models for effective multimodal KV cache compression.

AI USE STATEMENT

In this work, we used generative AI tools only for proofreading, including identifying typographical and grammatical errors and improving the clarity of a small number of sentences. Generative AI tools were not used for research ideation, methodology or experimental design, implementation, data analysis, or interpretation of results. All AI-assisted edits were reviewed and verified by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work does not involve human subjects, sensitive or private data, or other ethical concerns covered by the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We provide a detailed description of the proposed method in the methodology section, while the experimental section reports the implementation details, hyperparameter settings, datasets, and evaluation protocols used in our experiments. All datasets and pretrained models used in this work are publicly available. The source code will be made publicly available upon acceptance of the paper.

REFERENCES

Yongqi An, Chang Lu, Kuan Zhu, Tao Yu, Chaoyang Zhao, Hong Wu, Ming Tang, and Jinqiao Wang. Rest-kv: Robust kv cache eviction with layer-wise output reconstruction and spatial-temporal smoothing. *arXiv preprint arXiv:2605.08840*, 2026.

Baosong Yang An Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Anthropic. Claude code: An agentic coding tool. `https://www.anthropic.com/claude-code`, 2025. Accessed: 2026-07-18.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 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)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL `https://aclanthology.org/2024.acl-long.172`.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=ayi7qezU87`.

Zhirui Chen, Peiyang Liu, and Ling Shao. Structkv: Preserving the structural skeleton for scalable long-context inference. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12784–12797, 2026.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the*

2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4599–4610, 2021.

Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 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*, pp. 1074–1084, 2019.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Advances in Neural Information Processing Systems*, 38:113152–113188, 2025a.

Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S Kevin Zhou, and Xike Xie. Criticalkv: Optimizing kv cache eviction from an output perturbation perspective. *arXiv preprint arXiv:2502.03805*, 2025b.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 99269–99290, 2025.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. In *International Conference on Learning Representations*, volume 2024, pp. 22975–22988, 2024.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70–79, 2019.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*, pp. 12098–12107. PMLR, 2023.

Haoyu Guo, Junlin Lv, S Kevin Zhou, Xike Xie, et al. Defensivekv: Taming the fragility of kv cache eviction in llm inference. In *International Conference on Learning Representations*, volume 2026, pp. 43407–43432, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 conference of the north American chapter of the association for computational linguistics: Human language technologies*, pp. 1419–1436, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

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

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

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

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024a.

Tianyang Liu, Canwen Xu, and Julian McAuley. Repobench: Benchmarking repository-level code auto-completion systems. In *International Conference on Learning Representations*, volume 2024, pp. 47832–47850, 2024b.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Bo Li, Xuming Hu, and Xiaowen Chu. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference. *Advances in Neural Information Processing Systems*, 38:28728–28778, 2025a.

Xuyang Liu, Xiyan Gui, Yuchao Zhang, and Linfeng Zhang. Mixing importance with diversity: Joint optimization for kv cache compression in large vision-language models. *arXiv preprint arXiv:2510.20707*, 2025b.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *International Conference on Learning Representations*, volume 2025, pp. 37228–37253, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, 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*, pp. 5905–5921, 2021.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast kv compaction via attention matching. *arXiv preprint arXiv:2602.16284*, 2026.

A Appendix

in Appendix, we provide additional details and analyses to further support our proposed CoCoKV. Specifically, we first present the algorithm of CoCoKV. We then provide additional visualizations of the LongBench and Needle-in-a-Haystack results for a more intuitive comparison, followed by further parameter analyses that complement those in the main text. Finally, we present detailed proofs of Lemma 1 and Lemma 2 to provide theoretical support for the proposed method.

A.1 Algorithm

Algorithm 1 CoCoKV

Require: Query, key, and value states $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, head dimension d, cache capacity C, window size w, oversampling factor γ, completeness weight λ, consensus ratio α
Ensure: Compressed cache $\mathbf{K}', \mathbf{V}'$

if $|\mathbf{K}| \leq C$ **then**
  return $\mathbf{K}, \mathbf{V}$
end if
$C' \leftarrow C - w$
// Compute attention scores
$\mathbf{A} \leftarrow \mathrm{MaxPool}\left(\mathrm{Softmax}\left(\mathbf{Q}_{-w:}\mathbf{K}_{:-w}^{\top}/\sqrt{d}\right)\right)$
for each KV head and layer (h, l) **do**
  // Cross-Head Consensus
  $\overline{\mathbf{A}^{l}} \leftarrow \mathrm{Mean}(\mathbf{A})$
  $\mathbf{Cons} \leftarrow \mathbf{A}^{h,l} + \alpha\overline{\mathbf{A}^{l}}$
  // Oversample to obtain candidates
  $\mathrm{Cand} \leftarrow \mathrm{TopK}(\mathbf{Cons}, \gamma C')$
  $\mathbf{S} \leftarrow \{\arg\max_{i \in \mathrm{Cand}} \mathbf{Cons}_i\}$
  while $|\mathbf{S}| < C'$ **do**
    // Cross-Token Completeness
    $\mathbf{Comp}_j \leftarrow \min_{g \in \mathbf{S}} \left(1 - \mathbf{K}_j^{\top}\mathbf{K}_g\right)$
    $j^* \leftarrow \arg\max_{j \in \mathrm{Cand} \setminus \mathbf{S}} \left(\mathbf{Cons}_j + \lambda\mathbf{Comp}_j\right)$
    $\mathbf{S} \leftarrow \mathbf{S} \cup \{j^*\}$
  end while
end for
// Combine window and selected tokens
$\mathbf{K}' \leftarrow \mathrm{Concat}\left(\mathbf{K}_{\mathbf{S}}, \mathbf{K}_{-w:}\right)$
$\mathbf{V}' \leftarrow \mathrm{Concat}\left(\mathbf{V}_{\mathbf{S}}, \mathbf{V}_{-w:}\right)$
return $\mathbf{K}', \mathbf{V}'$

A.2 VISUALIZATIONS

A.2.1 VISUALIZATION OF OVERALL PERFORMANCE

To provide a more intuitive comparison of the overall performance, we visualize the average LongBench results reported in Table 1. As shown in Fig. 5, CoCoKV achieves the highest average score on all three evaluated models and therefore obtains the largest aggregate score among the compression methods. These results demonstrate its effectiveness and robustness across different LLMs.

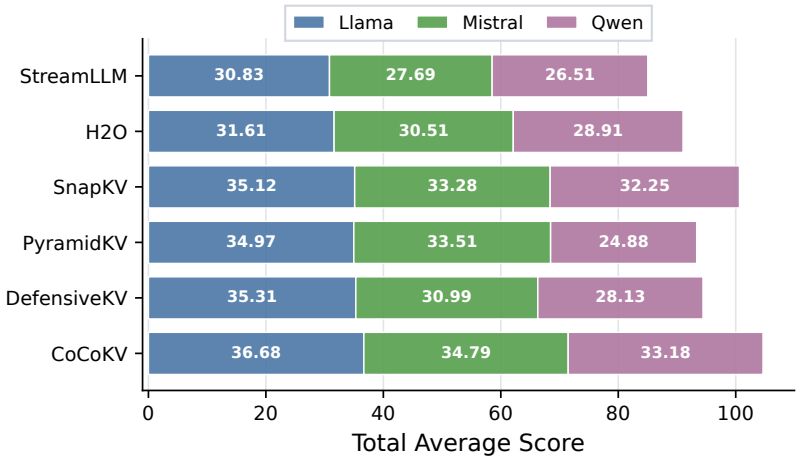


Figure 5: Graphical comparison of the average LongBench performance under a KV cache budget of 64. Each horizontal stacked bar reports the average scores on Llama-3-8B-Instruct, Mistral-7B-Instruct-v0.2, and Qwen2.5-7B-Instruct. The values correspond to the "Avg." results in Table 1 of the main text, and the length of each segment reflects the performance on the corresponding model.

A.2.2 VISUALIZATION OF NEEDLE-IN-A-HAYSTACK

In this subsection, we provide performance visualization of the Needle-in-a-Haystack results for all baseline methods in Fig. 6, enabling a more intuitive comparison of their retrieval performance under different context lengths and needle depths.

A.3 ADDITIONAL PARAMETER ANALYSES

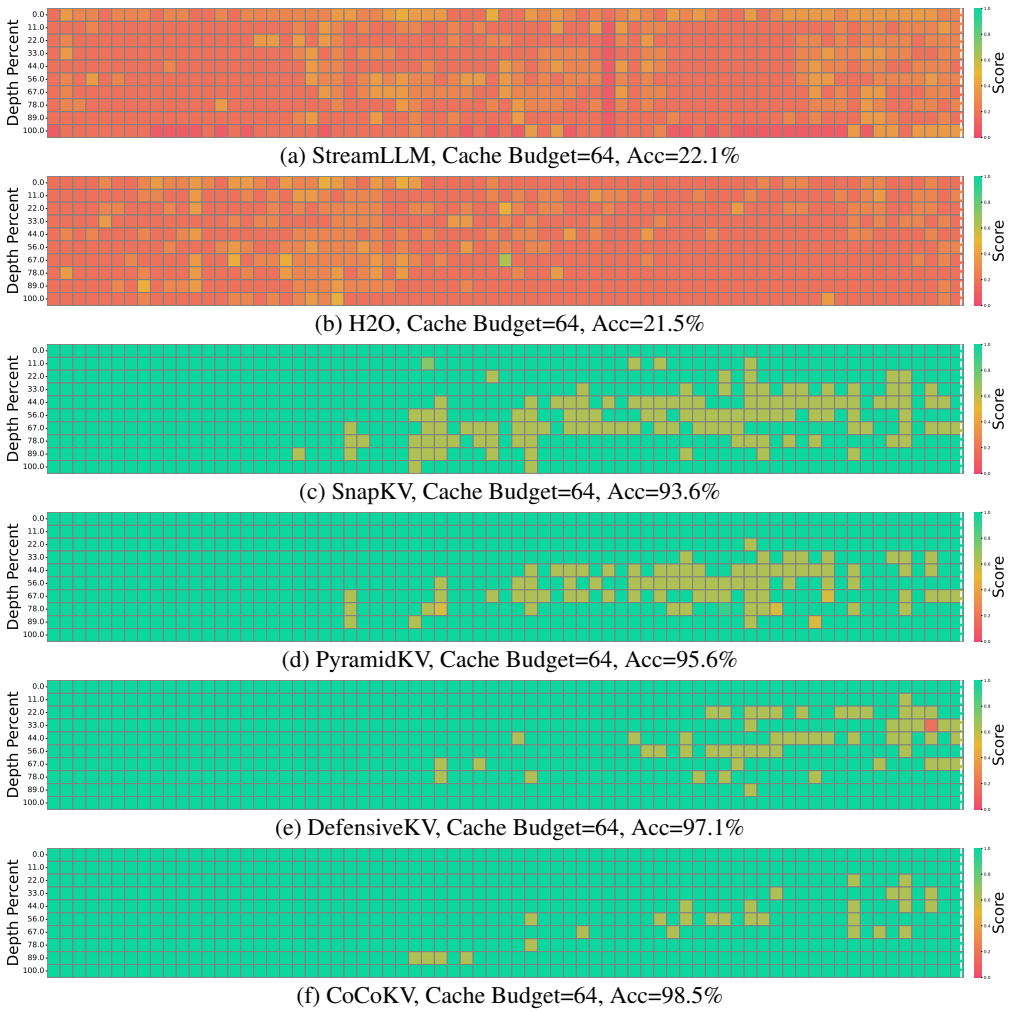
In addition to the consensus ratio analyzed in the main text, we further investigate the effects of the oversampling factor γ and the completeness weight λ. The former determines the number of tokens in the candidate set for completeness-aware selection, while the latter controls the contribution of cross-token completeness to the joint score. The corresponding analyses are presented below.

A.3.1 EFFECT OF THE OVERSAMPLING FACTOR

We investigate the sensitivity of CoCoKV to the oversampling factor γ across the three evaluated models used in the main text. As exhibited in the left panel of Fig. 7, the average performance remains stable as γ varies from 2.0 to 4.0, demonstrating that CoCoKV is robust to its choice. The best performance is consistently achieved at $\gamma = 3.0$, which provides a sufficiently broad pool of high-importance tokens for achieving broader semantic coverage while avoiding the introduction of excessive low-importance candidates.

A.3.2 EFFECT OF THE COMPLETENESS WEIGHT

We further investigate the sensitivity of CoCoKV to the completeness weight λ across the three evaluated models. As shown in the right panel of Fig. 7, the average performance exhibits only minor variations as λ ranges from 0.3 to 0.7, validating the parameter robustness of CoCoKV. The best performance is consistently achieved at $\lambda = 0.5$, where token importance and completeness are well balanced, allowing informative tokens to be retained while reducing information omission.



(a) StreamLLM, Cache Budget=64, Acc=22.1%

(b) H2O, Cache Budget=64, Acc=21.5%

(c) SnapKV, Cache Budget=64, Acc=93.6%

(d) PyramidKV, Cache Budget=64, Acc=95.6%

(e) DefensiveKV, Cache Budget=64, Acc=97.1%

(f) CoCoKV, Cache Budget=64, Acc=98.5%

Figure 6: Performance visualization of Needle-in-a-Haystack on Llama-3-8B-Instruct with up to $8K$ context size under a cache size of 64.

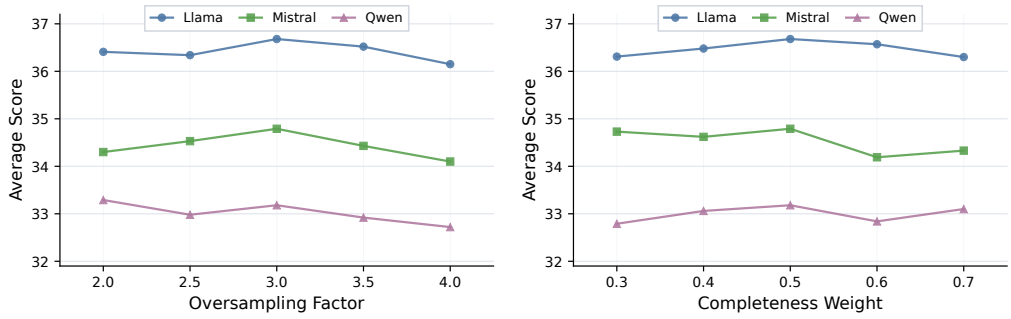


Figure 7: Average performance across 16 LongBench datasets under different oversampling factors (left) and completeness weights (right) on three LLMs with the KV cache budget fixed at 64.

A.4 PROOF

A.4.1 PROOF OF LEMMA 1

Lemma 1 (Coverage Guarantee of Completeness). Define the covering radius R of a selected set $S^{h,l}$ as the minimum radius such that balls centered at all selected tokens can cover every candidate token in $\mathrm{Cand}^{h,l}$. Let R_{OPT} denote the theoretically optimal covering radius achievable by selecting C' tokens. Then, the set obtained by greedy completeness-based selection satisfies

$$R \leq 2R_{\mathrm{OPT}}. \tag{15}$$

Proof. Since the key vectors are normalized, for any candidate token j, we have

$$\mathbf{Comp}_j = \min_{g \in \mathbf{S}^{h,l}} [1 - (\mathbf{K}_j^{h,l})^\top \mathbf{K}_g^{h,l}] = \frac{1}{2} \min_{g \in \mathbf{S}^{h,l}} \left\| \mathbf{K}_j^{h,l} - \mathbf{K}_g^{h,l} \right\|_2^2. \tag{16}$$

Therefore, selecting the token with the maximum completeness score is equivalent to selecting the candidate with the largest Euclidean distance to its nearest token among the already selected tokens, $i.e.$, the farthest-first greedy procedure.

Let the C' tokens selected by the completeness-based greedy procedure be $\{j_1, j_2, \ldots, j_{C'}\}$, and let R denote the covering radius of the final selected set. At the t-th farthest-first iteration, the newly selected token j_t is the candidate whose nearest distance to the current selected set is the largest. Therefore, this nearest distance is exactly the covering radius before the t-th selection. Since adding more selected tokens can only decrease the covering radius, the covering radius at any intermediate iteration is no smaller than the final radius R. Hence, when j_t is selected, its distance to every previously selected token is at least R. Consequently, the distances between any two selected tokens are no smaller than R, $i.e.$,

$$\left\| \mathbf{K}_{j_a}^{h,l} - \mathbf{K}_{j_b}^{h,l} \right\|_2 \geq R, \quad \forall\, a \neq b, \;\; a, b \in \{1, 2, \ldots, C'\}. \tag{17}$$

After selecting C' tokens, by the definition of the covering radius, there exists a candidate token $j_{C'+1} \in \mathrm{Cand}^{h,l} \setminus \mathbf{S}^{h,l}$ whose distance to its nearest selected token is R. Thus,

$$\left\| \mathbf{K}_{j_{C'+1}}^{h,l} - \mathbf{K}_{j_t}^{h,l} \right\|_2 \geq R, \quad \forall\, t \in \{1, 2, \ldots, C'\}. \tag{18}$$

Consequently, the $C'+1$ tokens $\{j_1, \ldots, j_{C'}, j_{C'+1}\}$ are pairwise separated by a distance of at least R.

Now consider the theoretically optimal subset of C' tokens achieving the covering radius R_{OPT}, such that each of the above $C'+1$ tokens $\{j_1, \ldots, j_{C'}, j_{C'+1}\}$ lies within a distance of R_{OPT} from at least one token in this optimal subset. By the pigeonhole principle, at least two of them, denoted by j_a and j_b, must be covered by the same optimal selected token m. Therefore,

$$\left\| \mathbf{K}_{j_a}^{h,l} - \mathbf{K}_m^{h,l} \right\|_2 \leq R_{\mathrm{OPT}}, \qquad \left\| \mathbf{K}_{j_b}^{h,l} - \mathbf{K}_m^{h,l} \right\|_2 \leq R_{\mathrm{OPT}}. \tag{19}$$

By the triangle inequality,

$$\left\| \mathbf{K}_{j_a}^{h,l} - \mathbf{K}_{j_b}^{h,l} \right\|_2 \leq \left\| \mathbf{K}_{j_a}^{h,l} - \mathbf{K}_m^{h,l} \right\|_2 + \left\| \mathbf{K}_m^{h,l} - \mathbf{K}_{j_b}^{h,l} \right\|_2 \leq 2R_{\mathrm{OPT}}. \tag{20}$$

Meanwhile, the pairwise separation established above gives

$$\left\| \mathbf{K}_{j_a}^{h,l} - \mathbf{K}_{j_b}^{h,l} \right\|_2 \geq R. \tag{21}$$

Combining the two inequalities yields

$$R \leq 2R_{\mathrm{OPT}}, \tag{22}$$

This completes the proof of Lemma 1. □

A.4.2 PROOF OF LEMMA 2

In this section, we provide the proof of Lemma 2 in the main text. For completeness, we first restate the lemma below, which shows that tokens with high consensus and completeness scores will be prioritized earlier during the selection.

Lemma 2 (Monotonicity of the Selection Process). The joint score of the selected token is monotonically non-increasing as the selection proceeds, $i.e.$,

$$\phi_{j_t}^{(t)} \geq \phi_{j_{t+1}}^{(t+1)}, \qquad \forall\, t \in \{1, \ldots, C' - 1\}, \tag{23}$$

where j_t denotes the token selected at the t-th iteration, and

$$\phi_j^{(t)} = \mathbf{Cons}_j^{h,l} + \lambda \mathbf{Comp}_j^{(t)} \tag{24}$$

denotes the joint score of candidate token j at the t-th iteration. Here, $\mathbf{Comp}_j^{(t)}$ is computed with respect to the $(t-1)$-th selected set $(\mathbf{S}^{h,l})^{(t-1)}$.

Proof. During the selection, one token is added to the selected set at each iteration, $i.e.$,

$$(\mathbf{S}^{h,l})^{(t)} = (\mathbf{S}^{h,l})^{(t-1)} \cup \{j_t\}. \tag{25}$$

Therefore,

$$(\mathbf{S}^{h,l})^{(t-1)} \subseteq (\mathbf{S}^{h,l})^{(t)}. \tag{26}$$

For any token j that remains unselected after the t-th iteration, its completeness score at the next iteration is computed over the enlarged selected set. Thus,

$$\begin{aligned} \mathbf{Comp}j^{(t+1)} &= \min_{r \in (\mathbf{S}^{h,l})^{(t)}} \left(1 - \cos\left(\mathbf{K}_j^{h,l}, \mathbf{K}r^{h,l}\right)\right) \\ &\leq \min_{r \in (\mathbf{S}^{h,l})^{(t-1)}} \left(1 - \cos\left(\mathbf{K}_j^{h,l}, \mathbf{K}_r^{h,l}\right)\right) \\ &= \mathbf{Comp}_j^{(t)}. \end{aligned} \tag{27}$$

Consequently, given that $\lambda \geq 0$ and the consensus score of a token is fixed across iterations, it follows that

$$\begin{aligned} \phi_j^{(t+1)} &= \mathbf{Cons}_j^{h,l} + \lambda \mathbf{Comp}_j^{(t+1)} \\ &\leq \mathbf{Cons}_j^{h,l} + \lambda \mathbf{Comp}_j^{(t)} \\ &= \phi_j^{(t)}. \end{aligned} \tag{28}$$

This shows that the joint score of every unselected token is non-increasing as the selection proceeds.

Note that at the t-th iteration, j_t is selected as the token with the highest joint score among all currently unselected tokens. Since j_{t+1} is unselected at the t-th iteration, we have

$$\phi_{j_t}^{(t)} \geq \phi_{j_{t+1}}^{(t)}. \tag{29}$$

Meanwhile, applying Eq. (28) to token j_{t+1} gives

$$\phi_{j_{t+1}}^{(t)} \geq \phi_{j_{t+1}}^{(t+1)}. \tag{30}$$

Combining Eq. (29) and Eq. (30) yields

$$\phi_{j_t}^{(t)} \geq \phi_{j_{t+1}}^{(t+1)}, \qquad \forall\, t \in \{1, \ldots, C' - 1\}. \tag{31}$$

This completes the proof of Lemma 2. □